algorithms

MDPI

Article

A Quantum-Behaved Neurodynamic Approach for Nonconvex Optimization with Constraints

Zheng Ji, Xu Cai and Xuyang Lou *

Key Laboratory of Advanced Process Control for Light Industry (Ministry of Education), Jiangnan University, Wuxi 214122, China

* Correspondence: Louxy@jiangnan.edu.cn

Received: 29 May 2019; Accepted: 3 July 2019; Published: 5 July 2019

Abstract: This paper presents a quantum-behaved neurodynamic swarm optimization approach to solve the nonconvex optimization problems with inequality constraints. Firstly, the general constrained optimization problem is addressed and a high-performance feedback neural network for solving convex nonlinear programming problems is introduced. The convergence of the proposed neural network is also proved. Then, combined with the quantum-behaved particle swarm method, a quantum-behaved neurodynamic swarm optimization (QNSO) approach is presented. Finally, the performance of the proposed QNSO algorithm is evaluated through two function tests and three applications including the hollow transmission shaft, heat exchangers and crank–rocker mechanism. Numerical simulations are also provided to verify the advantages of our method.

Keywords: nonconvex optimization; feedback neural network; collective neurodynamic optimization; quantum-behaved particle swarm optimization

1. Introduction

Constrained optimization problems arise in many scientific and engineering applications including robot control [1], regression analysis [2], economic forecasting [3], filter design [4] and so on. In many real-time applications, the optimization problems are often subject to complex and time-varying nature, which makes it difficult to compute global optimal solutions in real time using traditional numerical optimization techniques [4,5], such as Lagrange methods, descent methods and penalty function methods. A promising approach for handling these optimization problems is to employ neurodynamic optimization which is available for hardware implementation and possesses parallel and distributed computing ability. However, as pointed out in [6], the dynamic behaviors of a neural network could change drastically and become unpredictable, when applying neurodynamic optimization to deal with general nonconvex optimization problems. To compute global optimal solutions in such optimization problems, one solution is resorting to strategies in the meta-heuristics research field.

Recently, neurodynamic optimization based on recurrent neural networks has attracted much focus in solving various linear and nonlinear optimization problems. The essence of such neurodynamic optimization approaches is fundamentally different from those of iterative numerical algorithms such as the sequential programming method and interior point algorithms. The gradient method has been widely used in optimization problems [7,8]. Recurrent neural networks are also based on the gradient method and many other methods have been used to improve recurrent neural networks such as dual neural network [9,10], projection neural network [11], delayed neural network [12], weight-constrained neural networks [3,13] and so on. Some researchers focused on improving the performance of neural networks. Leung et al. [14] addressed a high-performance feedback neural network for solving convex nonlinear programming problems. Nazemi [15] explored

a high performance neural network model to solve chance constrained optimization problem. Some researchers give different forms of neural networks for specific problems. Mansoori et al. presented a kind of recurrent neural network to solve quadratic programming problems and nonlinear programming problems with fuzzy parameters in [16,17]. Xia and Kamel presented a cooperative projection neural network to solve the least absolute deviation problems with general linear constraints [18]. Che and Wang presented a two-timescale duplex neurodynamic system for constrained biconvex optimization in [19]. For some special nonconvex and nonsmooth problems, some achievements have been made in neurodynamic optimization. Based on projection method and differential inclusions, Yang et al. [20] proposed a generalized neural network. In [21], a one-layer recurrent projection neural network was utilized for solving pseudoconvex optimization problems with general convex constraints. Li et al. [22] provided a one-layer recurrent neural network based on an exact penalty function method for solving nonconvex optimization problems subject to general inequality constraints. Bian et al. [23] proposed a one-layer recurrent neural network for solving a class of nonsmooth, pseudoconvex optimization problems with general convex constraints.

Despite great progresses of neurodynamic optimization approaches in solving optimization problems with convex or some pseudoconvex functions with constraints, it is difficult to find global optimal solutions for the optimization problems with more nonconvex functions. In addition, many neural networks seems inadequate when dealing with constrained optimization problems containing multiple local minima. As one of population-based evolutionary computation approaches, the well-known particle swarm optimization (PSO) introduced by by Kennedy and Eberhart [24] is a popular meta-heuristic method for global optimization with multimodal objective functions. In the PSO algorithm, a simple velocity and displacement model to adjust the position according to the group optimal value and its own optimal value. It has strong global search ability and robustness. However, the PSO algorithm has at least three shortcomings. Firstly, it has been proved by Frans in [25] that, when the number of iterations tends to infinity, the traditional PSO algorithm can not converge to the global optimal solution with probability 100%. Secondly, in the PSO algorithm, the velocity of particles has an upper limit to ensure that particles can aggregate and avoid divergence, which limits the search space of particles and thus makes the algorithm ineffective to jump out of local optimal solutions. Thirdly, as the evolution equation of the PSO algorithm is based on a set of simple state equations of velocity and position, which makes the randomness and swarm intelligence relatively low. Due to its simple calculation, easy implementation, and few control parameters, many improved PSO algorithms have been to overcome the drawbacks (e.g., see [26–28]). Yan et al. [6] presented a collective neurodynamic optimization approach by combining the traditional PSO algorithm and a projection neural network to solve nonconvex optimization problems with box constraints. Later, Yan et al. [29] apply an adaptive PSO algorithm together with a one-layer recurrent neural network to solve nonconvex general constrained optimization problems.

Among the improved PSO algorithms, the quantum-behaved particle swarm optimization (QPSO) algorithm introduced in [30] is also one promising alternate, which describes particle behavior with probability. Compared with the traditional PSO algorithm, the QPSO algorithm has certain advantages in global optimization and speed of convergence. The convergence proof of QPSO is also given in [27]. Applying the QPSO algorithm for global search intermittently by incorporating a novel feedback neural network for local search in a hybrid mode can be a good choice.

In this paper, we propose a quantum-behaved neurodynamic swarm optimization (QNSO) approach combined with an efficient neural network and the QPSO algorithm to solve the nonconvex optimization problems with constraints. We improve an efficient feedback neural network proposed in [14] and prove the global convergence of the neural network for a class of nonconvex optimization problems. Inspired by [6,14,29], we employ the feedback neural network and present a quantum-behaved neurodynamic swarm approach for solving the nonconvex optimization problems with inequality constraints efficiently. The proposed QNSO approach combining the QPSO algorithm [30] and the feedback neural network is applied to deal with constrained optimization

problems with multiple global minima. Compared with the methods in [6,29], it shows a better convergence performance. Both numerical examples and practical applications are provided to demonstrate the effectiveness of the QNSO approach.

This paper is organized as follows. In Section 2, a constrained optimization problem is described and some theoretical analysis of an improved neural network is given. In Section 3, combined with the QPSO algorithm and the proposed neural network, the QNSO approach is developed for solving the constrained optimization problems. In Section 4, we perform experiments on two multimodal functions to demonstrate the performance of the QNSO algorithm. In Section 5, we apply the QNSO method to the optimization problems of the hollow transmission shaft, heat exchangers and crank–rocker mechanism. Finally, Section 6 concludes the paper.

2. Problem Statement and Model Description

Consider the following constrained optimization problem given by

$$\begin{aligned} &\min f(x), \\ &\text{subject to } g_i(x) \leq 0,\ i = 1, 2, \cdots, m, \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ is the decision vector, $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function, $g_i : \mathbb{R}^n \to \mathbb{R}$ $(i = 1, 2, \cdots, m)$ are continuously differentiable functions, denoting the inequality constraints. $f(x)$ and $g_i(x)$ are not necessarily convex. The feasible region $\Omega = \{x \in \mathbb{R}^n : g_i(x) \leq 0, i = 1, 2, \cdots, m\}$ is assumed to be a nonempty set.

Denote $g(x) = (g_1(x), g_2(x), \ldots, g_m(x))^\top$. Then, the problem (1) can be written as

$$\begin{aligned} &\min f(x), \\ &\text{subject to } g(x) \leq 0. \end{aligned} \tag{2}$$

Let M_1 be a lower bound of the optimal value of $f(x^*)$ in the problem (1), i.e., $M_1 \leq f(x^*)$, where x^* is an optimal solution of (1). Denote $d(x, M_1) = f(x) - M_1$ and $F(x, M_1) = 0.5d(x, M_1)(d(x, M_1) + |d(x, M_1)|)$. Consider an energy function

$$E(x) = F(x, M_1) + \gamma\big(g(x)^\top \mathrm{sgn}(g(x) + |g(x)|)\big), \tag{3}$$

where $\gamma > 0$ is a penalty parameter, and $\mathrm{sgn}(\cdot)$ is the sign function.

Then, inspired by [14,22], we construct the following feedback neural network for minimizing $E(x)$

$$\begin{aligned} \dot{x} &= -\nabla E(x) \\ &= -\nabla F(x, M_1) - \gamma\big(\nabla g(x)^\top \mathrm{sgn}(g(x) + |g(x)|)\big). \end{aligned} \tag{4}$$

Remark 1. *In [14], Leung et al. proposed the energy function* $G(x, M_1) = F(x, M_1) + \frac{1}{2}g(x)^\top(g(x) - |g(x)|) + \frac{1}{2} \parallel Ax - b \parallel^2$ *for the convex nonlinear programming problem*

$$\begin{aligned} &\textit{min} && f(x), \\ &\textit{subject to} && g(x) \geq 0, \\ & && Ax = b. \end{aligned}$$

In fact, for some equality constraint $h(x) = 0$*, one can transform the equality constraint into inequality constraints* $-h(x) \leq 0$ *and* $h(x) \leq 0$*. The energy function* $E(x)$ *in (3) contains a penalty parameter* γ*. Such a penalty* γ *can strengthen the constraints and make the results converge into the constraints more efficiently in optimization problems with inequality constraints. In addition, as we will show later, the penalty parameter* γ *makes the neural network feasible to fit in the network model for solving some nonconvex problems in [22].*

To derive the convergence results of the feedback neural network (4), we first introduce the following definitions.

Definition 1. *[22] Suppose that f is a differentiable function and defined on an open convex set $M \subset \mathbb{R}^n$. Then, f is quasiconvex if $a, b \in M, f(a) \geq f(b) \Rightarrow \nabla f(a)^\top (b-a) \leq 0$.*

Definition 2. *[22] Suppose that f is a differentiable function and defined on an open convex set $M \subset \mathbb{R}^n$. Then, f is pseudoconvex if $a, b \in M, f(a) > f(b) \Rightarrow \nabla f(a)^\top (b-a) < 0$.*

Theorem 1. *If $f(x)$ in (2) is a pseudoconvex function, then $F(x, M_1)$ in (3) is also a pseudoconvex function.*

Proof of Theorem 1. Let us rewrite $F(x, M_1) = 0.5d(x, M_1)(d(x, M_1) + |d(x, M_1)|)$ as

$$F(x, M_1) = \begin{cases} d(x, M_1)^2, & d(x, M_1) \geq 0, \\ 0, & d(x, M_1) < 0. \end{cases}$$

By definition of $d(x, M_1)$, i.e., $d(x, M_1) = f(x) - M_1$, we have

$$\nabla F(x, M_1) = \begin{cases} 2d(x, M_1)\nabla f(x), & d(x, M_1) \geq 0, \\ 0, & d(x, M_1) < 0, \end{cases} \tag{5}$$

and

$$F(x_1, M_1) - F(x_2, M_1) = \begin{cases} d(x_1, M_1)^2 - d(x_2, M_1)^2, & d(x_1, M_1) \geq 0, d(x_2, M_1) \geq 0, \\ d(x_1, M_1)^2, & d(x_1, M_1) \geq 0, d(x_2, M_1) < 0, \\ -d(x_2, M_1)^2, & d(x_1, M_1) < 0, d(x_2, M_1) \geq 0, \\ 0, & d(x_1, M_1) < 0, d(x_2, M_1) < 0. \end{cases}$$

Note that $F(x_1, M_1) - F(x_2, M_1) > 0$ holds when

$$F(x_1, M_1) - F(x_2, M_1) = \begin{cases} d(x_1, M_1)^2 - d(x_2, M_1)^2 > 0, & d(x_1, M_1) \geq 0, d(x_2, M_1) \geq 0, \\ d(x_1, M_1)^2 > 0, & d(x_1, M_1) \geq 0, d(x_2, M_1) < 0. \end{cases}$$

Using the definition of $d(x, M_1)$ again, it follows that $F(x_1, M_1) - F(x_2, M_1) > 0$ implies that both $d(x_1, M_1) > 0$ and $f(x_1) > f(x_2)$ holds. Therefore, using the fact that $f(x)$ is a pseudoconvex function, $F(x_1, M_1) - F(x_2, M_1) > 0$ also implies $\nabla f(x_1)(x_2 - x_1) < 0$. In addition, since $F(x_1, M_1) - F(x_2, M_1) > 0$ implies $d(x_1, M_1) > 0$, it follows from (5) that

$$\nabla F(x_1, M_1)^\top (x_2 - x_1) = 2d(x_1, M_1)\nabla f(x_1)(x_2 - x_1) < 0$$

if $F(x_1, M_1) - F(x_2, M_1) > 0$. The proof is completed. □

Remark 2. *From the proof of Theorem 1, it should be noted that the function F is constructed as a semi-positive definition function and the term $|d(x, M_1)|$ can be seen as an adaptive adjustment function when computing the gradient of $E(x)$. That is, when $f(x)$ approaches $f(x^*)$, $d(x, M_1)$ becomes smaller and $\nabla f(x)$ plays a minor role in computing the gradient of $E(x)$; when $f(x)$ is far away from $f(x^*)$, $d(x, M_1)$ becomes larger and $\nabla f(x)$ plays an important role in computing the gradient of $E(x)$.*

To establish a relationship between the feedback neural network (4) and the neural network in [22], we consider the following notion and lemmas.

Definition 3. *[29] (Clarke's generalized gradient) The generalized gradient of f at x, denoted by $\partial f(x)$, is the convex hull of the set of limits of the form $\lim \nabla f(x + l_i)$, where $l_i \to 0$ as $i \to \infty$.*

Lemma 1. *[22] [Proposition 6] Let $g(x)$ be continuously differentiable. Then, $\max\{0, g(x)\}$ is a regular function, its Clarke's generalized gradient is*

$$\partial \max\{0, g(x)\} = \begin{cases} \nabla g(x), & g(x) > 0, \\ \alpha_g \nabla g(x), & g(x) = 0, \\ 0, & g(x) < 0, \end{cases}$$

where $\alpha_g \in [0, 1]$.

Lemma 2. *[22] [Theorem 4] For the problem (1), if one of the following two conditions holds,*

(a) *$f(x)$ and $g_i(x)$, $i = 1, 2, \cdots, m$ are convex functions;*
(b) *$f(x)$ is a pseudoconvex function and $g_i(x)$, $i = 1, 2, \cdots, m$ are quasiconvex functions,*

then, for a sufficiently small penalty factor $\sigma > 0$, any state of the following recurrent neural network

$$\dot{x} \in -\nabla f(x) - \tfrac{1}{\sigma} \partial \textstyle\sum_{i=1}^{m} \max\{0, g_i(x)\} \tag{6}$$

converges to an optimal solution of problem (1).

Definition 4. *(KKT Point) For the problem (1), suppose that $f(x) : \mathbb{R}^n \to \mathbb{R}$ is differentiable and $g_i(x) : \mathbb{R}^n \to \mathbb{R}, i = 1, 2, \cdots, m$ are differentiable functions. If $(\bar{x}, \bar{\mu}) \in \mathbb{R}^n \times \mathbb{R}^m$ satisfies the following conditions:*

$$\nabla f(\bar{x}) + \bar{\mu}^\top \nabla g(\bar{x}) = 0, \bar{\mu} \geq 0, g(\bar{x}) \leq 0, \bar{\mu}^\top g(\bar{x}) = 0,$$

then $\bar{x}$ is said to be a KKT point of problem (1). The KKT conditions provide first-order necessary conditions for nonlinear programming problem and can be considered as local optimization points for programming problem.

Under some acceptable assumptions (see [22] for details), the following properties hold.

Proposition 1. *[22] [Theorem 1] When the penalty parameter σ is sufficiently small, then any state of (6) is guaranteed to be convergent to the feasible region in finite time and stay there thereafter.*

Proposition 2. *[22] [Corollary 1] When the penalty parameter σ is sufficiently small, any state of the neural network (6) will converge to an equilibrium point $\bar{x}$ and any equilibrium point $\bar{x}$ of the neural network (6) corresponds to a KKT twofold $(\bar{x}, \bar{\mu})$ of the problem (1).*

Next, we claim that for the problem (1), the neural network (4) is a special form of neural network (6). Firstly, note that, to solve the the problem $\min F(x, M_1)$ with the constraint $g(x) \leq 0$, it is equivalent to solving the problem $\min f(x)$ with the constraint $g(x) \leq 0$. Secondly, take $\alpha_g = 0$ in Lemma 1, the neural network (4) can be written as

$$\begin{aligned} \dot{x} &= -\nabla F(x, M_1) - \gamma(\nabla g(x)^\top \mathrm{sgn}(g(x) + |g(x)|)) \\ &= -\nabla F(x, M_1) - \gamma \partial \sum_{i=1}^{m} \max\{0, g_i(x)\}. \end{aligned} \tag{7}$$

Define $\sigma := \frac{1}{\gamma}$ in (7). It follows that

$$\dot{x} = -\nabla F(x, M_1) - \tfrac{1}{\sigma} \partial \sum_{i=1}^{m} \max\{0, g_i(x)\},$$

which is a special case of the neural network (6) in Lemma 2 if $F(x, M_1)$ has the same property of the function f in Lemma 2.

The following result shows the convergence property of the proposed neural network.

Theorem 2. *The neural network (4) converges to an optimal solution of problem (1) if there exists a large enough penalty parameter $\gamma > 0$ and one of the following two conditions holds*

(a) *$f(x)$ and $g_i(x)$, $i = 1, 2, \cdots, m$ are convex functions;*
(b) *$f(x)$ is a pseudoconvex function and $g_i(x)$, $i = 1, 2, \cdots, m$ are quasiconvex functions.*

Proof of Theorem 2. Firstly, if condition (a) holds, it follows from [14] [Theorem 1] that $F(x, M_1)$ in (3) is also a convex function. Then, condition (a) in Lemma 2 holds. Secondly, if condition (b) holds, it follows from Theorem 1 that $F(x, M_1)$ in (3) is also a pseudoconvex function. Then, condition (b) in Lemma 2 holds. In addition, it follows from the above discussion that the neural network (4) fits in the form of neural network (6). Therefore, by Lemma 2, the proof is completed. □

Remark 3. *Theorem 2 shows that the proposed neural network (4) will converge to an optimal solution for some nonconvex optimization problems and (4) satisfies the properties in Propositions 1 and 2, that is, the neural network (4) can converge to the feasible region of the optimization problem in finite time and find a KKT point which is a local optimal point of the problem (1).*

3. Quantum-Behaved Neurodynamic Swarm Approach

To solve the nonconvex programming problems with box constraints, a collective neurodynamic approach is proposed in [6], which can be seen as a combination of neurodynamic optimization and particle swarm optimization. In order to improve the convergence speed of the optimization algorithm, we explore a quantum-behaved neurodynamic approach combining the QPSO algorithm with the feedback neural network model (7) for nonconvex programming problems with general constraints. In the QPSO algorithm, each particle represents a position, which represents a potential optimal solution of optimization problem. Imitating the swarm intelligence behavior of animals, the renewal of particles is related to their optimal position and global optimal position. Unlike the basic PSO algorithm, the updating equation of particle position follows the quantum-behaved model. The QPSO algorithm has the aggregation of particles with the iteration, increases the intelligence of particle behavior, and makes the possible search range of particles wider. Therefore, the ability of global optimization increases.

Given q particles, define the position of the ith particle as $X_i = (X_{i1}, X_{i2}, \cdots, X_{in})$, the individual best position of the ith particle as $P_i = (P_{i1}, P_{i2}, \cdots, P_{in})$ and the best position of the swarm as $P_g = (G_1, G_2, \cdots, G_n)$. The particle renewal equation of the QPSO algorithm is given by

$$X_{ij}(k+1) = p_{ij}(k) \pm \beta |C_j(k) - X_{ij}(k)| \ \ln \frac{1}{\eta_{ij}(k)}, \tag{8}$$

where

$$\begin{aligned} p_{ij}(k) &= \alpha_j(k) P_{ij}(k) + [1 - \alpha_j(k)] G_j(k) \\ C(k) &= (C_1(k), C_2(k), \cdots, C_n(k)) = \frac{1}{q} \sum_{i=1}^{q} P_i(k) \\ &= \frac{1}{q} \Big(\sum_{i=1}^{q} P_{i1}(k), \sum_{i=1}^{q} P_{i2}(k), \cdots, \sum_{i=1}^{q} P_{in}(k) \Big) \end{aligned}$$

and $\alpha_j(k), \eta_{ij}(k)$ are random numbers between $(0, 1)$, $j \in \{1, 2, \cdots, n\}$, $i \in \{1, 2, \cdots, q\}$, β is an adjustable parameter, $p_{ij}(k)$ is the center of attraction potential field of $X_{ij}(k+1)$. $C(k)$ is the average optimal position of particle. The iteration stops when the maximum number of iterations is reached or the given conditions is achieved.

Note that, despite of its excellent global optimization ability, the QPSO algorithm lacks the ability for constraint processing and deep local searching. Inspired by neurodynamic optimization, during the local search process, we apply the feedback neural network to optimize each particle and improve

the search efficiency. Next, we explore the QNSO approach to deal with constrained optimization problems, where the QPSO algorithm is applied to adjust the initial conditions of the feedback neural network at each local search phrase.

The steps of the QNSO algorithm are summarized as follows:

Step 1: Initialize the position of particles.
Step 2: Initialize the individual best position P_i of the ith particle, $i = 1, 2, \cdots, q$ and the global best position P_g of the particle swarm.
Step 3: Each particle reaches a local optimal solution based on the feedback neural network (7).
Step 4: Update the individual best position P_i of the ith particle, $i = 1, 2, \cdots, q$ and the global best position P_g of the particle swarm.
Step 5: Update the position of each particles based on (8).
Step 6: Repeat *Step 3* to *Step 5* before reaching the given conditions.

Following the description in Section 2, we simply consider the energy function in (3) as the fitness value of the QPSO algorithm. Then, the individual best position P_i of the ith particle is updated according to

$$P_i(k) = \begin{cases} X_i(k), & \text{when } E(X_i(k)) < E(P_i(k-1)), \\ P_i(k-1), & \text{when } E(X_i(k)) \geq E(P_i(k-1)), \end{cases} \tag{9}$$

where $k-1$ and k represent the $k-1$th iteration and kth iteration, respectively. The global best position P_g of the particle swarm is updated by

$$P_g(k) = \min_{1 \leq i \leq q} \{E(P_i(k))\}. \tag{10}$$

Remark 4. *The fitness value of the QNSO algorithm could be different from the energy function of the neural network. A large γ may bring a better performance as it effectively restricts particles to diverge beyond the feasible region.*

QNSO can always find the accurate local minimum of each particle within one iteration. Compared with general global optimization algorithms (such as PSO), when the number of local minimum is small, QNSO is extremely insensitive to heuristic parameters; when the number of local minimum is large, the QNSO algorithm needs sufficient divergence ability of heuristic parameters to jump out of local minimum. In other words, appropriately larger β in QPSO can be more suitable for the QNSO algorithm.

Detailed steps of the QNSO algorithm are given in Algorithm 1. During local search process, the feedback neural network in the algorithm is applied to solve the local optimization problem with constraints. The QPSO algorithm is applied to perform global search process. Therefore, the advantages of the QPSO algorithm and the feedback neural network can complement each other. The flow chart of QNSO is shown in Figure 1.

Algorithm 1 QNSO Algorithm.

1: Set the swarm size q, parameter β, tolerance error precision ε, the maximum number $K_{\max}$ of iterations, the expected cost function value E_e, the lower bound of cost function M_1 and the initial step $k = 0$.
2: Initialize the position $X_i \in \mathbb{R}^n$ of particles, $i = 1, 2, \cdots, q$, using uniform random distribution, and get the initial individual best positions and global best position.
3: Carry out the following feedback neural network to obtain a local optimal solution for each particle:

$$\epsilon \dot{x} = -\nabla F(x, M_1) - \gamma \partial \sum_{i=1}^{q} \left(\max\{0, g_i(x)\} \right),$$

where $\epsilon > 0$ is a scaling parameter which is introduced to accelerate the convergence rate of the neural network.
4: Update the individual best position P_i of the ith particle, $i = 1, 2, \cdots, q$ and the global best position P_g of the particle swarm using

$$P_i(k) = \begin{cases} X_i(k), & \text{when } E(X_i(k)) < E(P_i(k-1)), \\ P_i(k-1), & \text{when } E(X_i(k)) \geq E(P_i(k-1)), \end{cases}$$

and

$$P_g(k) := (G_1, G_2, \cdots, G_n) = \min_{1 \leq i \leq q} \{E(P_i(k))\}.$$

5: Update the position $X_i(k+1)$ of each particle, that is, the initial conditions of the feedback neural network, using the following the renewal equations:

$$\begin{aligned} p_{ij}(k) &= \alpha_j(k) P_{ij}(k) + [1 - \alpha_j] G_j(k), \\ C(k) &= \frac{1}{q} \sum_{i=1}^{q} P_i(k), \\ X_{ij}(k+1) &= p_{ij}(k) \pm \beta |C_j(k) - X_{ij}(k)| \ \ln \frac{1}{\eta_{ij}(k)}, \end{aligned}$$

where $i = 1, 2, \cdots, q$, $j = 1, 2, \cdots, n$ and $\alpha_j(k) \sim U(0,1)$, $\eta_{ij}(k) \sim U(0,1)$.
6: The iteration stops if one of the following conditions is satisfied:

(a) The algorithm reaches the maximum iteration value $K_{\max}$.
(b) $|E(P_g) - E_e| < \varepsilon$.
(c) P_g stops updating for five consecutive iterations.

Otherwise, set $k := k + 1$ and go to step 3.

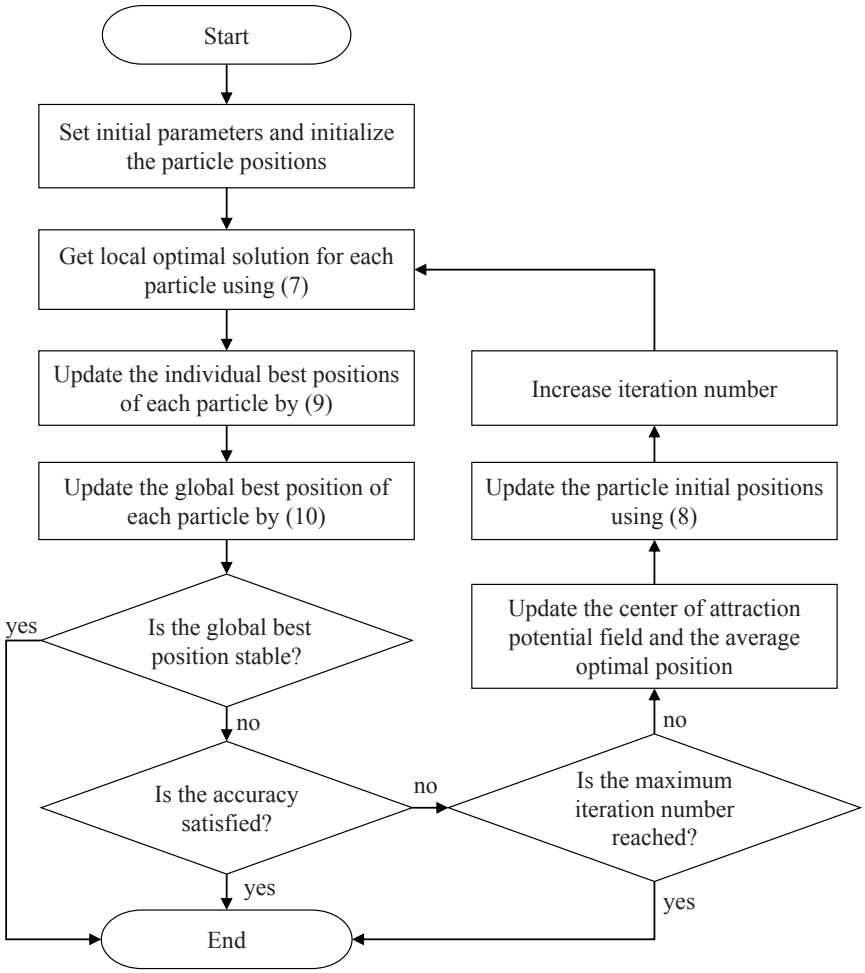

Figure 1. Flow chart of the QNSO algorithm.

4. Function Tests

In this section, we consider two multimodal benchmark problems, i.e., the constrained six-hump camel back function [6] and the constrained Rastrigrin function, to illustrate the proposed optimization approach. The first example is provided to verify the convergence performance of the proposed QNSO algorithm, while the second example is provided to demonstrate the constraint processing ability of the algorithm.

Example 1. *Consider the six-hump camel back function [6]*

$$\begin{aligned} \min \quad & f(x_1, x_2) = (4 - 2.1x_1^2 + \tfrac{x_1^4}{3})x_1^2 + x_1x_2 + 4(x_2^2 - 1)x_2^2, \\ \text{subject to} \quad & -2 \le x_i \le 2,\ i = 1, 2. \end{aligned}$$

Figure 2 shows the contour map of the six-hump camel back function with box constraints. It is seen that there are multiple local minima in the area. In fact, the optimization problem has six minima

$$\begin{cases} f(-1.6071, -0.5687) = 2.1043, \\ f(1.6071, 0.5687) = 2.1043, \\ f(-1.7036, 0.7961) = -0.2155, \\ f(1.7036, -0.7961) = -0.2155, \\ f(0.0898, -0.7127) = -1.0316, \\ f(-0.0898, 0.7127) = -1.0316. \end{cases}$$

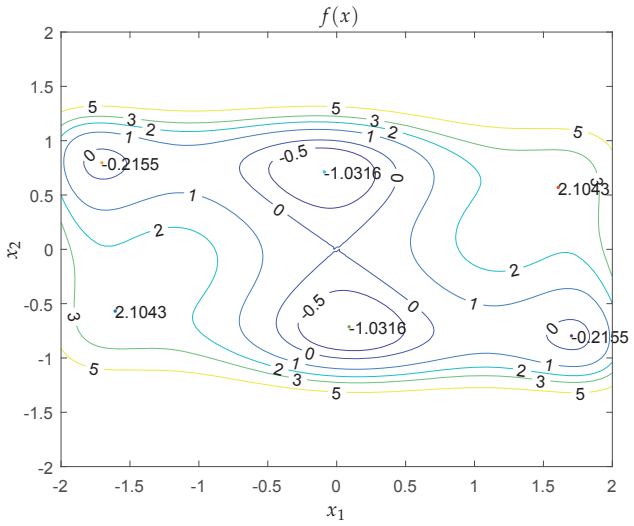

Figure 2. Contour map of f in Example 1, where different color lines represent contours corresponding to different values of f.

Two of them are the global minima, located at $x^* = (-0.0898, 0.7127)$ and $x^* = (0.0898, -0.7127)$, respectively. To carry out the QNSO algorithm, set swarm size $q = 3$, parameter $\beta = \frac{1}{2}$, tolerance error precision $\varepsilon = 10^{-6}$, the maximum iteration number $K_{\max} = 10$, $M_1 = -20$, and $\gamma = 1000$. Before performing the simulation, initialize the position $X_{ij} \in [-2, 2]$ of particles, $i = 1, 2, 3$, $j = 1, 2$, using uniform random distribution. After 50 experiments, we obtain the two global minima at every experiment. Except two experiments, the global minima are found at the first step iteration for most cases, which shows that the QNSO algorithm has a high efficiency and good accuracy to find the optimal solution for the problem in this example.

In order to verify the convergence efficiency of the proposed method, we compare the optimization results of the QNSO algorithm with the results derived by the approaches in [6,29]. For the neural networks in all the three methods, we take the scaling parameter $\epsilon = 10^{-3}$. Figures 3–5 show the convergence of the trajectory of the proposed neural network (7), the projection neural network in [6], and the recurrent neural network in [29] at the first iteration, respectively. As shown in the figures, each particle converges to its local optimum shortly. It can be seen that, among the three methods, the proposed neural network has the fastest convergence speed.

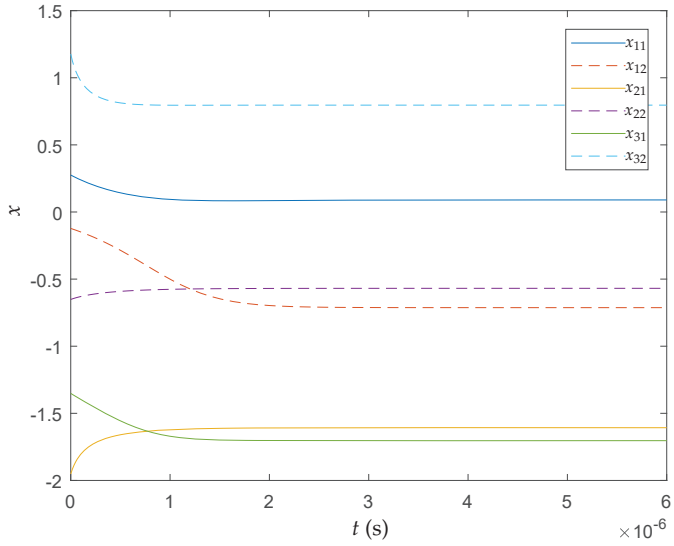

Figure 3. Convergence of the proposed neural network.

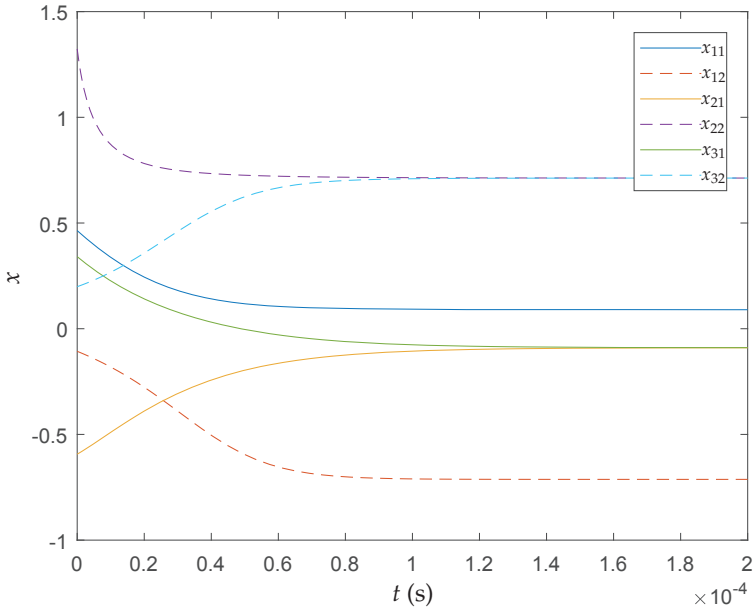

Figure 4. Convergence of the neural network in [6].

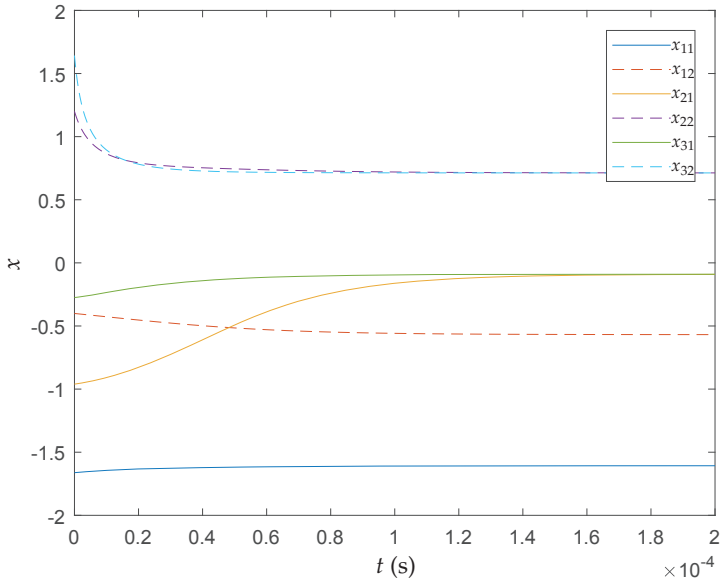

Figure 5. Convergence of the neural network in [29].

Figure 6 shows the behavior of particles in the search process by the proposed QNSO algorithm. The initial positions of particles are randomly selected. After one iteration, two particles reach the global minima along the direction perpendicular to the contour.

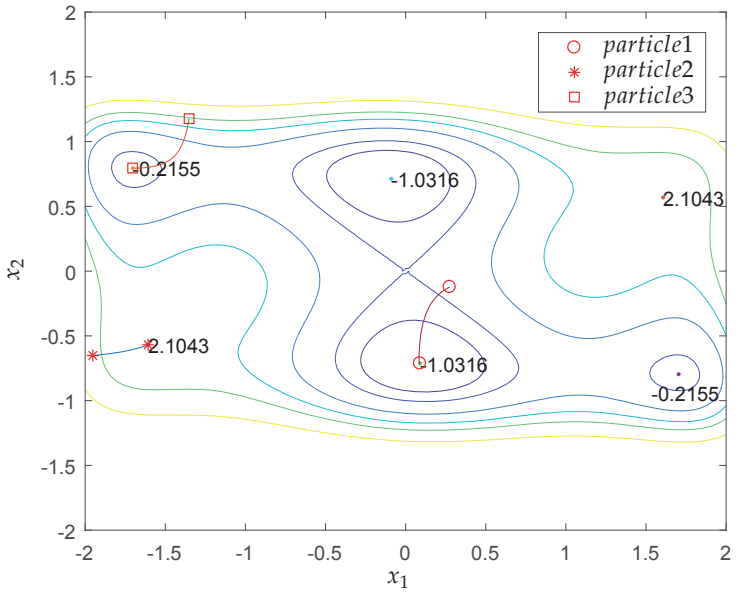

Figure 6. Moving trails of the three particles using the QNSO algorithm, where different color lines represent contours corresponding to different values of f and the corresponding values are shown in Figure 2.

Example 2. *Consider the optimization problem of the constrained Rastrigrin function described by*

$$\begin{aligned} \min \quad & f(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10), \\ \text{subject to} \quad & -6 \le x_i \le 6, \\ & \sum_{i=1}^{n} x_i^2 \ge 4.5, \end{aligned} \tag{11}$$

which has hundreds of local minima and many global minima for $n \ge 4$. Note that the origin, which is the global minimum point for unconstrained Rastrigrin function, is not within the feasible range. The complexity of this problem depends on the size of n and the number of minima increases exponentially with the increase of n.

To carry out the QNSO algorithm, set the initial parameters of QNSO algorithm: $M_1 = 0, \gamma = 1000$, β decreased from 0.9 to 0.3 on the course of the search, swarm size $q = 20$, tolerance error precision $\varepsilon = 10^{-5}$, the maximum iteration number $K_{\max} = 500$. To establish a reference for comparison with the QNSO algorithm in this paper, we performed experiments with Rastrigrin function in 2, 4 and 10 dimensions, respectively. Some other optimization methods were also examined for performance comparisons, including the latest standard particle swarm optimization (SPSO) [31], the adaptive particle swarm optimization (APSO) [32], the random drift particle swarm optimization (RDPSO) [33], the firefly algorithm (FA) [34], the repair genetic algorithm (RGA) [35], the QPSO [30], and the method in [29]. We set swarm size $q = 20$, tolerance error precision $\varepsilon = 10^{-5}$, the maximum iteration number $K_{\max} = 500$ for all the tested methods. Standard normal distribution was used in RGA as mutation operator. The parameters for QPSO were the same as those in this paper. Other parameters for SPSO, APSO, FA, RDPSO and RGA were the same as those recommended in [31–35] and these methods used the same fitness value (3) with a large $\gamma = 10^6$ to keep optimizations within the feasible range.

Tables 1–3 show the computational results, where the best result, the average result, the worst result, the standard deviation (Std.) result of $f(P_g)$, and the average number of iterations (No. of Iterations) over 50 experiments are provided. The optimum results are coarsened with boldface in these tables. Obviously, when the particles and the iterations are limited, the general optimization algorithm has obvious deficiencies with the increase of the complexity of the problem, but the method

in [29] and QNSO can obtain accurate optimal solutions with good robustness. QNSO requires the least iterations with a stable optimization efficiency as shown in Tables 1 and 2.

Table 1. Comparing computational efficiency ($n = 2$).

Methods	Best	Average	Worst	Std.	No. of Iterations
SPSO [31]	4.9751	4.9925	5.0594	0.0190	500
APSO [32]	4.9758	4.9978	5.0738	0.0211	500
FA [34]	**4.9748**	**4.9748**	4.9750	3.6985×10^{-5}	481.5
RGA [35]	**4.9748**	5.0164	6.5421	0.7317	414.9
RDPSO [33]	**4.9748**	4.9774	4.9956	5.7606×10^{-3}	450.56
QPSO [30]	**4.9748**	4.9749	4.9760	2.6078×10^{-4}	327.92
[29]	**4.9748**	**4.9748**	**4.9748**	4.39×10^{-8}	2.4
QNSO	**4.9748**	**4.9748**	**4.9748**	$\mathbf{3.151 \times 10^{-9}}$	**1.3**

Table 2. Comparing computational efficiency ($n = 4$).

Methods	Best	Average	Worst	Std.	No. of Iterations
SPSO [31]	5.6356	6.8906	8.2099	0.65794	500
APSO [32]	5.1884	6.7265	8.0925	0.87531	500
FA [34]	4.9793	7.0177	20.9417	3.1047	500
RGA [35]	**4.9748**	5.0627	7.4586	0.45259	500
RDPSO [33]	**4.9748**	5.3332	9.0534	0.88538	472.26
QPSO [30]	**4.9748**	5.2261	5.9917	0.4433	468.6
[29]	**4.9748**	**4.9748**	**4.9748**	5.8247×10^{-7}	5.34
QNSO	**4.9748**	**4.9748**	**4.9748**	$\mathbf{9.5127 \times 10^{-8}}$	**4.18**

Table 3. Comparing computational efficiency ($n = 10$).

Methods	Best	Average	Worst	Std.	No. of Iterations
SPSO [31]	27.5833	39.7451	51.0513	6.2649	500
APSO [32]	23.8351	39.3527	50.4733	7.1493	500
FA [34]	30.3567	56.3786	81.7154	12.5423	500
RGA [35]	4.9754	4.9872	5.0509	0.01809	500
RDPSO [33]	**4.9748**	6.9563	13.2190	2.3012	500
QPSO [30]	**4.9748**	5.7736	8.9549	0.9786	495.7
[29]	**4.9748**	**4.9748**	**4.9748**	1.5934×10^{-6}	**12.54**
QNSO	**4.9748**	**4.9748**	**4.9748**	$\mathbf{1.1478 \times 10^{-6}}$	14.3

5. Applications

In this section, three optimization problems in applications, including hollow transmission shaft [36], heat exchangers [37] and crank–rocker mechanism [36], are solved using the proposed QNSO algorithm.

Application 1. (Hollow transmission shaft) Consider the hollow transmission shaft optimization problem in [36], where D and d are the outer diameter and inner diameter, $d = 8$ mm, shaft length $L = 3.6$ m. The power transmitted by the shaft is $P = 7$ kW and the speed $n = 1500$ r/min. Shaft material density p = 7800 kg/m^2, shear modulus $S = 81$ GPa, and allowable shear stress $\bar{\tau} = 45$ MPa, allowable torsion angle per unit length $\bar{\varphi} = 1.5^\circ$ m. The aim is to minimize the mass of the shaft under the limitations of torsional strength and torsional stiffness:

(1) Decision variables and cost function

In this problem, there is only one decision variable $x := D$. According to the design requirements, the optimization goal is to minimize the mass of the shaft that is, $f(x) = \frac{\pi}{4}\rho L(D^2 - d^2)$.

(2) Restrictions

(a) Torsional strength condition is $\tau_{max} = \frac{T}{W_n} \leq \bar{\tau}$, where T is the torque received by the round shaft, $T = 9550\frac{P}{n}$, W_n is the torsional section modulus, $W_n = \frac{\pi(D^4-d^4)}{16D}$.

(b) Torsional stiffness condition is $\varphi = \frac{T}{SJ_p} \leq \bar{\varphi}$, where φ is unit length twist angle, G is shear modulus, J_p is polar moment of inertia, $J_p = \frac{\pi(D^4-d^4)}{32}$.

(c) Outer diameter minimum limit condition is $x \geq d$.

Following the above analysis, we summarize the optimization problem as follows:

$$\begin{aligned} \min f(x) &= \frac{\pi}{4}\rho L(x^2-d^2)\times 10^{-6}, \\ \text{subject to } g_1(x) &= d - x \leq 0, \\ g_2(x) &= \frac{16xT\times 10^9}{\pi(x^4-d^4)} - \bar{\tau}\times 10^6 \leq 0, \\ g_3(x) &= \frac{32T\times 10^3}{S\pi(x^4-d^4)} - \bar{\varphi}\times\frac{\pi}{180} \leq 0. \end{aligned} \tag{12}$$

For the above problem (12), one can use the *fmincon* function in Matlab to obtain a solution $x^* = 21.6121$ mm with $f(x^*) = 8.8896$ kg [36]. To carry out the QNSO algorithm, we set swarm size $q = 5$, parameter $\beta = \frac{1}{2}$, tolerance error precision $\varepsilon = 10^{-6}$, the maximum iteration number $K_{\max} = 15$, $M_1 = 0$, and $\gamma = 1000$. The initial values of the position $X_i \in [8, 100]$ of particles, $i = 1, 2, \cdots, 5$, are randomly chosen. We perform 50 realizations and obtain an optimal solution $x^* = 21.5965$ mm with $f(x^*) = 8.8747$ kg, which is slightly better than that obtained in [36].

Figure 7 shows the iteration results of the energy function E, the cost function f and the solution x of the best particle among five particles. As seen from the results, the proposed algorithm successfully solves the optimization problem with a fast convergence speed and finds a relative ideal solution within five iterations. Figure 8 depicts the iteration results of the inequality constraint functions and illustrates the ability of the algorithm to deal with constraints. Table 4 shows the best result, the average result, the worst result, and the standard deviation result of $f(P_g)$ over 50 realizations.

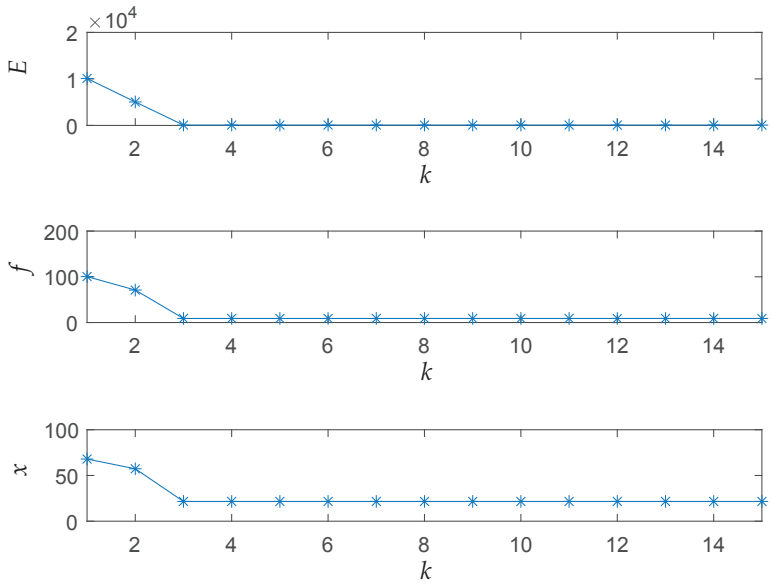

Figure 7. Convergence results of E,f and x in Application 1.

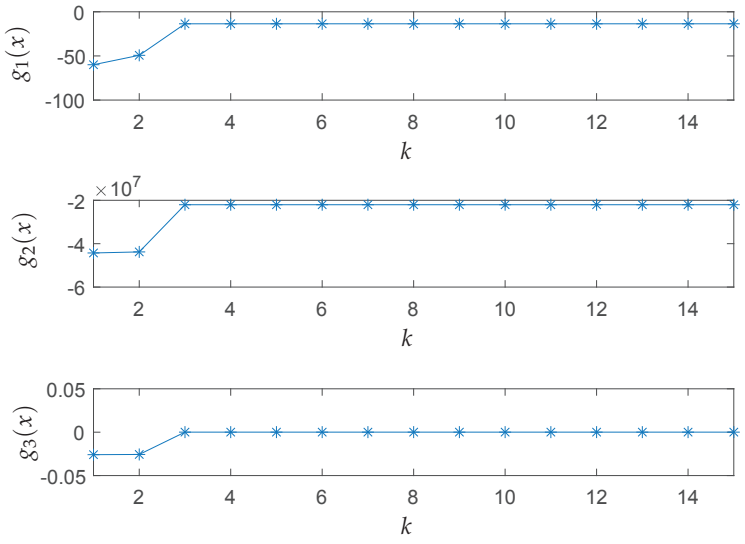

Figure 8. Iteration results of $g_i(x)$ in Application 1, $i = 1, 2, 3$.

Table 4. Statistical analysis of the results in Application 1.

Best	Average	Worst	Std.
8.8747	8.8759	8.8767	6.62×10^{-5}

Application 2. (Heat exchangers [37]) Consider three connected heat exchangers as shown Figure 9, which are used to heat a material temperature from 100° to 500°. Let $mc_p = 10^5$, where m is the mass flow rate of the fluid and c_p is the specific heat capacity of the fluid. The heat transfer coefficients of the three heat exchangers are $K_1 = 120$, $K_2 = 80$ and $K_3 = 40$, respectively. For simplicity, the temperature difference $\triangle t_m$ uses arithmetic mean value and the above values are obtained through dimensionless processing. The aim is to minimize the total heat transfer area by selecting the appropriate temperature.

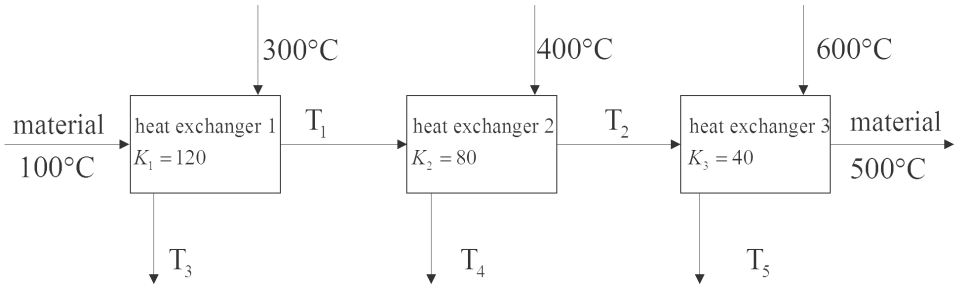

Figure 9. Flow chart of heat exchangers.

The decision variables are $x = [T_1, T_2]^\top$. The restrictions are given by $100 \leq T_1 \leq 300$ and $T_1 \leq T_2 \leq 400$. It follows from the law of conservation of heat that

$$
\begin{aligned}
&mc_p(T_1 - 100) = mc_p(300 - T_3) \Rightarrow T_3 = 400 - T_1,\\
&mc_p(T_2 - T_1) = mc_p(400 - T_4) \Rightarrow T_4 = 400 - T_2 + T_1,\\
&mc_p(500 - T_2) = mc_p(600 - T_5) \Rightarrow T_5 = 100 - T_2.
\end{aligned}
$$

The arithmetic mean values of temperature difference of heat exchangers are

$$\begin{aligned}\triangle t_{m1} &= \frac{300 - T_1 + T_3 - 100}{2} = 300 - T_1,\\ \triangle t_{m2} &= \frac{400 - T_2 + T_4 - T_1}{2} = 400 - T_2,\\ \triangle t_{m3} &= \frac{600 - 500 + T_5 - T_2}{2} = 100.\end{aligned}$$

Therefore, according to the design requirements, the cost function is obtained

$$f(x) = \frac{10^5(T_1 - 100)}{120(300 - T_1)} + \frac{10^5(T_2 - T_1)}{80(400 - T_2)} + \frac{10^5(500 - T_2)}{4000}.$$

To sum up, the optimization problem can be written as

$$\begin{aligned}\min f(x) = \frac{10^5(x_1 - 100)}{120(300 - x_1)} + \frac{10^5(x_2 - x_1)}{80(400 - x_2)} + \frac{10^5(500 - x_2)}{4000},\\ \text{subject to } g_1(x) = 100 - x_1 \le 0,\\ g_2(x) = x_1 - 300 \le 0,\\ g_3(x) = x_1 - x_2 \le 0,\\ g_4(x) = x_2 - 400 \le 0.\end{aligned} \tag{13}$$

In [37], the obtained minimum cost function value is 7049.4191 by using sequence quadratic programming. To carry out the QNSO algorithm, we set swarm size $q = 5$, parameter $\beta = \frac{1}{2}$, tolerance error precision $\varepsilon = 10^{-6}$, the maximum iteration number $K_{\max} = 6$, $M_1 = 0$, and $\gamma = 1000$. The initial values of the position $X_{ij} \in [0, 400]$ of particles, $i = 1, 2, \cdots, 5$, $j = 1, 2$, are randomly chosen. Since the problem is relatively simple, after performing the simulation, we can obtain the optimal solution at the first iteration almost every time. We perform 50 realizations and obtain the same results which are shown in Table 5. Figure 10 shows the behavior of particles in the search process in the phase plot by the proposed QNSO algorithm. It can be found that all particles move into the feasible region and reach some local optimal solutions.

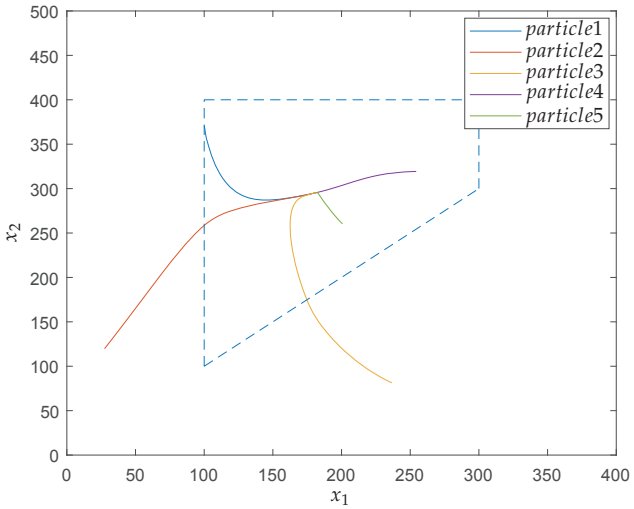

Figure 10. Moving trails of the particles using the QNSO algorithm in Application 2 (dotted line: feasible region).

Table 5. Optimization results in Application 2.

$f(x^*)$	x_1^*	x_2^*	$g_1(x^*)$	$g_2(x^*)$	$g_3(x^*)$	$g_4(x^*)$
7049.2493	182.0179	295.6012	−82.01787	−117.9821	−113.5834	−104.3988

Application 3. (Crank–rocker mechanism [36]) Consider a crank–rocker mechanism as shown in Figure 11. The anti-clockwise angle is calculated with the frame AD as the baseline. ψ_0 and φ_0 correspond to the position angles of the rocker and the crank respectively when the rocker is in the right limit position. The minimum transmission angle γ_{min} is greater than 45°. The crank–rocker mechanism is required to be designed when the crank is transferred from φ_0 to $\varphi_0 + \frac{\pi}{2}$ and the output angle ψ of the rocker meets the following predetermined motion rules ψ_s: $\psi_s = \psi_0 + \frac{1}{6}(\varphi - \varphi_0)^2$.

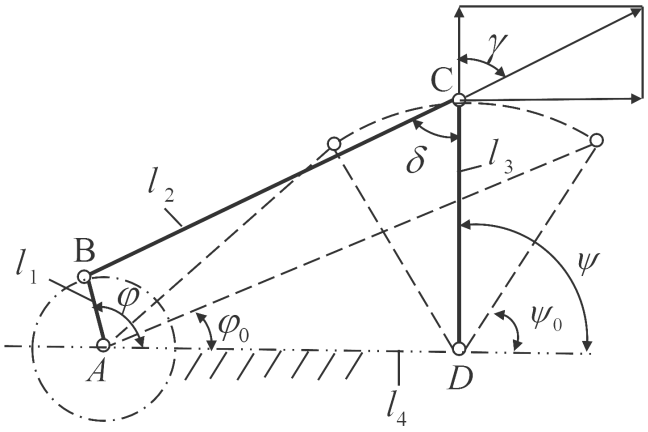

Figure 11. Crank–rocker mechanism.

It is shown in Figure 11 that the lengths of the four shafts are marked as l_1, l_2, l_3 and l_4, respectively. l_1 is the length of short shaft. Set l_1 be the unit length that is, $l_1 = 1$. Set the decision variable $x := [l_2, l_3, l_4]^\top$. It follows from the cosine theorem that

$$\psi_0 = \pi - \arccos \frac{l_4^2 + l_3^2 - (l_1 + l_2)^2}{2 l_4 l_3},$$
$$\varphi_0 = \arccos \frac{l_4^2 + (l_1 + l_2)^2 - l_3^2}{2 l_4 (l_1 + l_2)}.$$

According to the design requirements, the optimization goal is to minimize

$$f(x) = \int_{\varphi_0}^{\varphi_0 + \pi/2} (\psi - \psi_s)^2.$$

To numerically minimize $f(x)$, discretizing the cost function by dividing the interval $[\varphi_0, \varphi_0 + \pi/2]$ into 50 parts yields

$$f(x) = \sum_{p=0}^{p=50} (\psi_p - \psi_{sp})^2.$$

The actual output angle of the crank–rocker mechanism is ψ_p which can be calculated by

$$\psi_p = \begin{cases} \pi - \alpha_p - \beta_p, & 0 \le \varphi_p \le \pi, \\ \pi - \alpha_p + \beta_p, & \pi \le \varphi_p \le 2\pi, \end{cases}$$
$$\varphi_p = \varphi_0 + \tfrac{\pi}{2} \tfrac{p}{50},\ p = 1, 2, \cdots, 50,$$

where

$$r_p = \sqrt{l_1^2 + l_4^2 - 2l_1 l_4 \cos\varphi_p},$$
$$\alpha_p = \arccos\frac{r_p^2 + l_3^2 - l_2^2}{2l_3 r_p},$$
$$\beta_p = \arccos\frac{r_p^2 + l_4^2 - l_1^2}{2l_4 r_p}.$$

The minimum transmission angle constraints are given by

$$\cos\gamma = \begin{cases} \dfrac{l_2^2 + l_3^2 - (l_4 - l_1)^2}{2l_2 l_3} \le \cos 45^\circ, \delta \le 90^\circ, \\ \dfrac{(l_4 + l_1)^2 - l_2^2 - l_3^2}{2l_2 l_3} \le \cos 45^\circ, \delta > 90^\circ, \end{cases}$$

which are equivalent to

$$l_2^2 + l_3^2 - (l_4 - l_1)^2 - \sqrt{2} l_2 l_3 \le 0,$$
$$(l_4 + l_1)^2 - l_2^2 - l_3^2 - \sqrt{2} l_2 l_3 \le 0.$$

The length constraints are given by

$$l_2 \ge l_1, l_3 \ge l_1, l_4 \ge l_1,$$
$$l_1 + l_4 \le l_2 + l_3,$$
$$l_1 + l_2 \le l_4 + l_3,$$
$$l_1 + l_3 \le l_2 + l_4.$$

Following the above analysis, we summarize the optimization problem as follows:

$$\begin{aligned} \min f(x) &= \sum_{p=0}^{p=50} (\psi_p(x) - \psi_{sp}(x))^2, \\ \text{subject to } g_1(x) &= x_1^2 + x_2^2 - (x_3 - 1)^2 - \sqrt{2} x_1 x_2 \le 0, \\ g_2(x) &= (x_3 + 1)^2 - x_1^2 - x_2^2 - \sqrt{2} x_1 x_2 \le 0, \\ g_3(x) &= 1 - x_1 \le 0, \\ g_4(x) &= 1 - x_2 \le 0, \\ g_5(x) &= 1 - x_3 \le 0, \\ g_6(x) &= 1 + x_3 - x_1 - x_2 \le 0, \\ g_7(x) &= 1 + x_2 - x_1 - x_3 \le 0, \\ g_8(x) &= 1 + x_1 - x_3 - x_2 \le 0, \end{aligned} \tag{14}$$

where

$$\psi_{sp}(x) = \psi_0(x) + \frac{1}{6}(\varphi_p(x) - \varphi_0(x))^2,$$
$$\psi_p(x) = \begin{cases} \pi - \alpha_p(x) - \beta_p(x), & 0 \le \varphi_p \le \pi, \\ \pi - \alpha_p(x) + \beta_p(x), & \pi \le \varphi_p \le 2\pi, \end{cases}$$
$$\varphi_p(x) = \varphi_0(x) + \frac{\pi}{2}\frac{p}{50},\ p = 1, 2, \cdots, 50,$$
$$\varphi_0(x) = \arccos\frac{x_3^2 + (1 + x_1)^2 - x_2^2}{2x_3(1 + x_1)},$$
$$\psi_0(x) = \pi - \arccos\frac{x_3^2 + x_2^2 - (1 + x_1)^2}{2x_3 x_2},$$

$$\alpha_p(x) = \arccos \frac{r_p^2(x) + x_2^2 - x_1^2}{2x_2 r_p(x)},$$
$$\beta_p(x) = \arccos \frac{r_p^2(x) + x_3^2 - 1}{2x_3 r_p(x)},$$
$$r_p(x) = \sqrt{1 + x_3^2 - 2x_3 \cos \varphi_p}.$$

In this problem, the cost function and constraints are more complicated than those problems in Applications 1 and 2. In [36], the obtained cost function value is 0.0051 at the optimal solution $x^* = (5.6695, 2.9143, 7.0000)$ when the upper bound of x is limited to $(8, 8, 7)$. For comparison, we also impose the upper bound of x as $(8, 8, 7)$. To carry out the QNSO algorithm, we set swarm size $q = 20$, the parameter β decreased from 1 to 0.5, tolerance error precision $\varepsilon = 10^{-6}$, the maximum iteration number $K_{\max} = 500$, $M_1 = 0$, and $\gamma = 1000$. The initial values of the position $X_{ij} \in [1, 10]$ of particles, $i = 1, 2, \cdots, 20$, $j = 1, 2, 3$ are randomly chosen.

We perform 20 realizations and obtain $x^* = (5.6691, 2.9145, 7.0000)$, the minimum cost function value is 0.0050983, which is better than the result obtained in [36]. Figures 12–14 show the transient behaviors of x_1, x_2 and x_3 of the proposed neural network initialized from 20 random particles for each state, respectively. It can be found that all states converge instantaneously local optimal solutions since the cost function itself has many local optima and the searching space is divided into many nonconvex areas by the constraints. Therefore, more iterations have been required to make full use of interconnections of particles so that the QNSO algorithm is able to fully exploit the parallel nature of the computation and produce results that are more globally optimal. Table 6 shows the best result, the average result, the worst result, and the standard deviation result of $f(P_g)$ over 20 experiments.

Table 6. Statistical analysis of the results in Application 3.

Best	Average	Worst	Std.
5.0983×10^{-3}	5.1364×10^{-3}	5.4642×10^{-3}	1.1526×10^{-4}

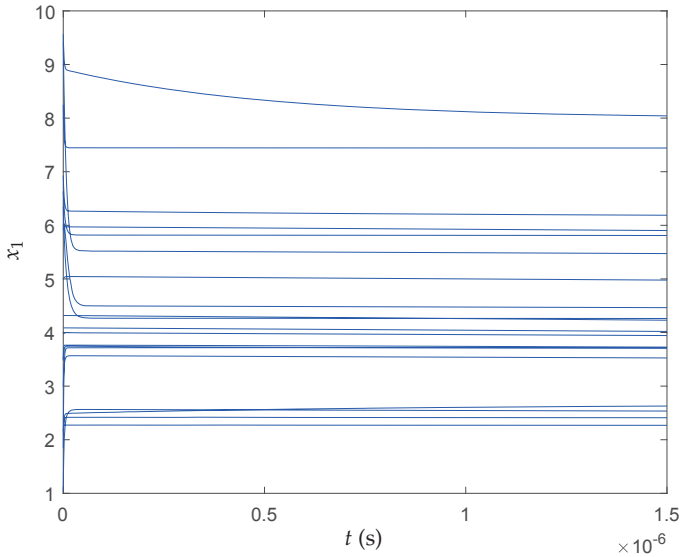

Figure 12. Transient behaviors of the proposed neural network state x_1.

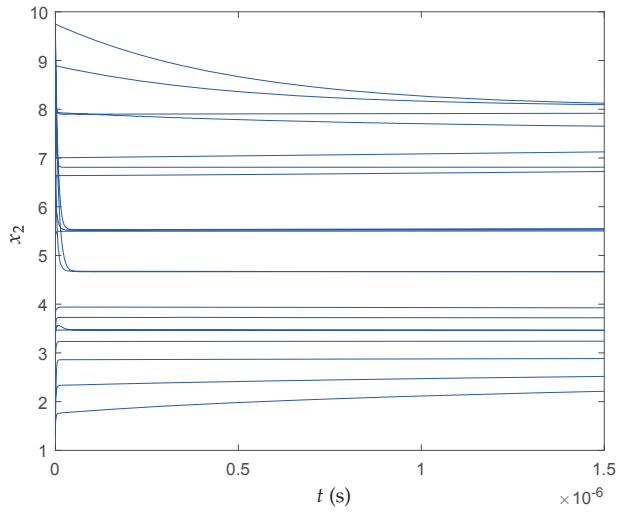

Figure 13. Transient behaviors of the proposed neural network state x_2.

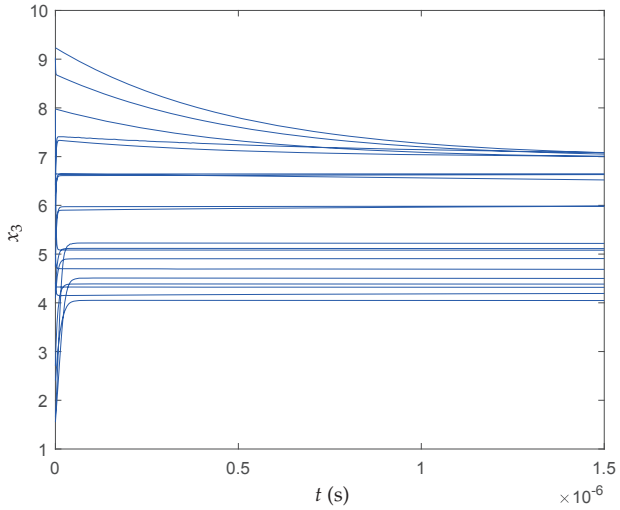

Figure 14. Transient behaviors of the proposed neural network state x_3.

6. Conclusions

We have presented the design of a new global optimization method called a quantum-behaved neurodynamic swarm optimization (QNSO) approach. This method is composed of an efficient neural network and the quantum-behaved swarm optimization. As the QPSO algorithm is used, the search process in the QNSO method has a high convergence rate. In addition, the global searching ability of particle swarm optimization and the local searching and constraint-processing abilities of neurodynamic optimization can complement each other very well. Meanwhile, the QNSO method and its implementation have been described extensively in the context. Then, the QNSO method has been illustrated with two function tests. Finally, the presented method has been applied to the optimization of three applications.

Author Contributions: Conceptualization, X.L.; Research and experiments, Z.J.; Writing—original draft preparation, Z.J. and X.C.; Writing—review and editing, X.L.

Funding: This research received no external funding.

Acknowledgments: This work is partially supported by the National Natural Science Foundation of China (61473136, 61807016), China Postdoctoral Science Foundation (2018M642160), and Jiangxi Natural Science Foundation Youth Project (20161BAB212032).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Al-Gallaf, E.; Mutib, K.A.; Hamdan, H. Artificial neural network dexterous robotics hand optimal control methodology: Grasping and manipulation forces optimization. *Artif. Life Robot.* **2010**, *15*, 408–412. [CrossRef]
2. Xia, Y.S.; Leung, H.; Bosse, E. Neural data fusion algorithms based on a linearly constrained least square method. *IEEE Trans. Neural Netw.* **2002**, *13*, 320–329. [PubMed]
3. Livieris, I.E. Forecasting economy-related data utilizing weight-constrained recurrent neural networks. *Algorithms* **2019**, *12*, 85. [CrossRef]
4. Boyd, S.; Vandenberghe, L. *Convex Optimization*; Cambridge University Press: New York, NY, USA, 2004.
5. Bazaraa, M.S.; Sherali, H.D.; Shetty, C.M. *Nonlinear Programming: Theory and Algorithms,* 3rd ed.; John Wiley and Sons: Hoboken, NJ, USA, 2006.
6. Yan, Z.; Wang, J.; Li, G. A collective neurodynamic optimization approach to bound-constrained nonconvex optimization. *Neural Netw.* **2014**, *55*, 20–29. [CrossRef] [PubMed]
7. Zheng, X.; Shi, J. A modified sufficient descent Polak-Ribiére-Polyak type conjugate gradient method for unconstrained optimization problems. *Algorithms* **2018**, *11*, 133. [CrossRef]
8. Yang, Y.; Cao, J. A feedback neural network for solving convex constraint optimization problems. *Appl. Math. Comput.* **2008**, *201*, 340–350. [CrossRef]
9. Xia, Y.S. A new neural network for solving linear and quadratic programming problems. *IEEE Trans. Neural Netw.* **1996**, *7*, 1544–1548. [PubMed]
10. Xia, Y.S.; Wang, J. A dual neural network for kinematic control of redundant robot manipulators. *IEEE Trans. Cybern.* **2001**, *31*, 147–154.
11. Effati, S.; Mansoori, A.; Eshaghnezhad, M. An efficient projection neural network for solving bilinear programming problems. *Neurocomputing* **2015**, *168*, 1188–1197. [CrossRef]
12. Liu, Q.; Cao, J.; Xia, Y. A delayed neural network for solving linear projection equations and its analysis. *IEEE Trans. Neural Netw.* **2005**, *16*, 834–843. [CrossRef]
13. Ioannis, E.L. Improving the classification efficiency of an ann utilizing a new training methodology. *Informatics* **2019**, *6*, 1.
14. Leung, Y.; Chen, K.Z.; Gao, X.B. A high-performance feedback neural network for solving convex nonlinear programming problems. *IEEE Trans. Neural Netw.* **2003**, *14*, 1469–1477. [CrossRef] [PubMed]
15. Nazemi, A.; Tahmasbi, N. A high performance neural network model for solving chance constrained optimization problems. *Neurocomputing* **2013**, *121*, 540–550. [CrossRef]
16. Mansoori, A.; Effati, S.; Eshaghnezhad, M. A neural network to solve quadratic programming problems with fuzzy parameters. *Fuzzy Optim. Decis. Mak.* **2016**, *17*, 75–101. [CrossRef]
17. Mansoori, A.; Effati, S. An efficient neurodynamic model to solve nonlinear programming problems with fuzzy parameters. *Neurocomputing* **2019**, *334*, 125–133. [CrossRef]
18. Xia, Y.S.; Kamel, M. Cooperative recurrent neural networks for solving L1 estimation problems with general linear constraints. *Neural Comput.* **2008**, *20*, 844–872. [CrossRef] [PubMed]
19. Che, H.; Wang, J. A two-timescale duplex neurodynamic approach to biconvex optimization. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**. [CrossRef] [PubMed]
20. Yang, Y.; Cao, J.; Xu, X.; Liu, J. A generalized neural network for solving a class of minimax optimization problems with linear constraints. *Appl. Math. Comput.* **2012**, *218*, 7528–7537. [CrossRef]
21. Li, Q.; Liu, Y.; Zhu, L. Neural network for nonsmooth pseudoconvex optimization with general constraints. *Neurocomputing* **2014**, *131*, 336–347. [CrossRef]
22. Li, G.; Yan, Z.; Wang, J. A one-layer recurrent neural network for constrained nonconvex optimization. *Neural Netw.* **2015**, *61*, 10–21. [CrossRef]

23. Bian, W.; Ma, L.; Qin, S.; Xue, X. Neural network for nonsmooth pseudoconvex optimization with general convex constraints. *Neural Netw.* **2018**, *101*, 1–14. [CrossRef]
24. Kennedy, J. Particle swarm optimization. In Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
25. Van den Bergh, F. An Analysis of Particle Swarm Optimizers. Ph.D. Thesis, Natural and Agricultural Science Department, University of Pretoria, Pretoria, South Africa, 2001.
26. Goudarzi, S.; Hassan, W.H.; Anisi, M.H.; Soleymani, A.; Sookhak, M.; Khurram Khan, M.; Hashim, A.A.; Zareei, M. ABC-PSO for vertical handover in heterogeneous wireless networks. *Neurocomputing* **2017**, *256*, 63–81. [CrossRef]
27. He, G.; Huang, N. A modified particle swarm optimization algorithm with applications. *Appl. Math. Comput.* **2012**, *219*, 1053–1060. [CrossRef]
28. Dai, H.P.; Chen, D.D.; Zheng, Z.S. Effects of random values for particle swarm optimization algorithm. *Algorithms* **2018**, *11*, 23. [CrossRef]
29. Yan, Z.; Fan, J.; Wang, J. A collective neurodynamic approach to constrained global optimization. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 1206–1215. [CrossRef]
30. Sun, J.; Feng, B.; Xu, W. Particle swarm optimization with particles having quantum behavior. In Proceedings of the 2004 Congress on Evolutionary Computation, Portland, OR, USA, 19–23 June 2004; Volume 1, pp. 325–331.
31. Zambrano-Bigiarini, M.; Clerc, M.; Rojas, R. Standard particle swarm optimisation 2011 at CEC-2013: A baseline for future PSO improvements. In Proceedings of the IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 2337–2344.
32. Zhan, Z.; Zhang, J. Adaptive particle swarm optimization. *IEEE Trans. Syst. Man Cybern.* **2009**, *39*, 1362–1381. [CrossRef]
33. Sun, J.; Palade, V.; Wu, X.J.; Fang, W.; Wang, Z.Y. Solving the power economic dispatch problem with generator constraints by random drift particle swarm optimization. *IEEE Trans. Ind. Inf.* **2014**, *10*, 222–232. [CrossRef]
34. Yang, X.S.; Hosseini, S.S.S.; Gandomi, A.H. Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect. *Appl. Soft Comput.* **2012**, *12*, 1180–1186. [CrossRef]
35. Dakuo, H.; Fuli, W.; Mingxing, J. An improved genetic algorithm for a type of nonlinear programming problems. In Proceedings of the IEEE International Conference on Automation and Logistics, Qingdao, China, 1–3 September 2008.
36. Zhang, Y. *Engineering Optimization Design and MATLAB Implementation*; Tsinghua University Press: Beijing, China, 2011.
37. Huang, H.J. *Computer Simulation of Practical Chemical Engineering: Application of Matlab in Chemical Engineering*; Chemical Industry Press: Beijing, China, 2004.