International Journal of
Applied Data Science in Engineering and Health
https://ijadseh.com

A PSO-based Neural Network for Multiple-response Optimization

Hossein Abbasi[1], Rassoul Noorossana[2], and Reza Tavakkoli-Moghaddam[3]

[1] *Islamic Azad University, Industrial Engineering department, South Tehran Branch, Tehran, Iran.*

[2] *Department of Industrial Engineering University of Science and Technology, Tehran, Iran.*

[3] *Department of Industrial Engineering College of Engineering, University of Tehran, Tehran, Iran.*

Received date: August 2, 2024; Accepted date: September 5, 2024

Abstract

An important problem in manufacturing or product and process design is the optimization of several responses simultaneously. Common approaches for multiple response optimization problems often begin with estimating the relationship between responses as outputs and control factors as inputs. Among these methods, response surface methodology (RSM), has attracted more attention in recent years, but in certain cases, the relationship between responses and control factors is far too complex to be efficiently estimated by regression models and the RSM method especially when we want to optimize several responses simultaneously. An alternative approach proposed in this paper is to use an artificial neural network (ANN) to estimate the response functions, Because of the high mean square error (MSE) in the neural network training step we use heuristic algorithms instead of Descent Gradient-based algorithms. In the optimization phase, a particle swarm optimization (PSO) and desirability function are considered to determine the optimal settings for the control factors. Two case studies from the literature are prepared to illustrate the strength of the proposed approach in optimizing multiple response problems.

Keywords*:* Design of experiments; Multiple response optimization; Artificial neural network; Particle swarm optimization; Multiple layer perception; Genetic Algorithm.

Introduction

Nowadays companies for remaining in competitive markets need to improve the quality of their products. Design of Experiments (DOE) is a strength method for manufacture process optimizing. In manufacturing process, control factors are some factors that we can control the value of that during process like temperature, noise factors are those that their values cannot be constant during the process like air humidity, and responses are outputs or characteristics of final product. In the real world, most customer consider more than one quality response problem, while selecting industrial products. In addition, the goals of the multiple response systems often conflict with each other. Experimental design methods describe the effects of control factors and noise factors on one or more responses. DOE determine the optimum setting for control factors such that the product quality characteristics achieve desirable values. In real DOE determine amount of inputs value and control factors to manufacturer obtain desirable responses. In most cases, any change in each input variable can affect on one or more responses simultaneously, this problem is referred as multiple response optimization (MRO) problem. Montgomery [1] discussed more about DOE and MRO problem. Figure1 illustrate simple multiple response system, suppose that there are r output responses $Y = (y_1, y_2, \dots\ , y_r)$ that are determined by a set of control factors and inputs. Note that because of input values are controllable by experimenter, inputs are also control factors, X.

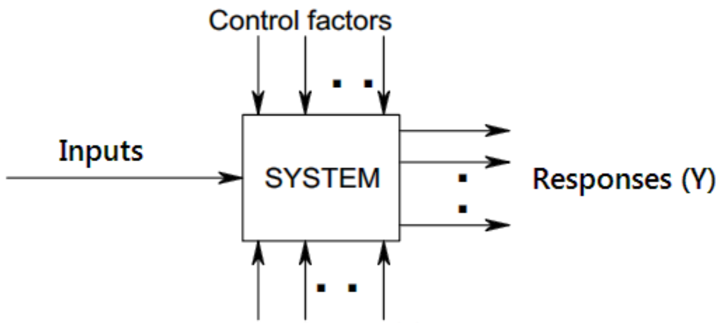

Figure 1 simple multiple response system

Typically, there are three stages in the solution of such problems: Experimental design and data collection, model building, and optimization. In the second stage we use Artificial Neural Networks (ANN) to build the model, ANN training is improved by a metaheuristic algorithm in comparison with traditional gradient descent algorithms. In the third stage, we would optimize the built model by particle swarm optimization (PSO) as a well-known optimization algorithm. After 1st and 2nd stage we can write the model as follows:

$$y_j = f_j(x) + \varepsilon_j\ , \qquad j = 1,2, \dots\ , m. \tag{1}$$

Where y_j is jth of m responses, $f_j(x)$ is a function relating to the jth response to the control factors and ε_j is a random error. This paper presents an approach for simultaneous optimization of all responses in MRO problems by use of artificial neural networks and metaheuristic algorithms. Also, neural network training is improved by PSO and GA in comparison with traditional Gradient descent algorithms.

Related Works

There are many methods for solving MRO problems that we can classify to four categories as Carlos A. [2]: In the first category, we simplify the problem by selecting the most important response and ignoring the other responses. Hartmann and Beaumont [3] used a linear programming approach to model the MRO problem. Biles et al. [4] combined Box's complex method [5] with the gradient method. Ortiz et al [6] modeled the problem to a constrained optimization problem form. The disadvantage of this method is that there is not enough consideration to all of the responses, simultaneously. In the second category, the problem is modeled as a single objective function, and after that single function would be optimized. Clayton et al [7] used this approach by goal programming and the weighted sum method which consists of adding the entire objective together by using different weighted coefficients. Baesler and sepulveda [8] integrated goal programming and genetic algorithm (GA) methods to solve the MRO problem. In this category, the most used approaches for modeling the problem to a single function are:

- Priority-based approach, Myers [9]
- Desirability functions, Derringer and Suich [10]
- Loss function, Pignatiello [11]

In the priority-based approach, the decision maker selects the most important of the responses as an objective function and uses the desired values of the other responses as constraints; there is no simultaneous optimization of all responses. In the desirability function approach, all responses are transformed to a scale-free value between 0 and 1 using the desirability function d_j for the jth response. The computed desirability for each response is combined to construct an overall desirability, which is then optimized. Derringer [12] proposed a weighted geometric mean for the overall desirability function. Kim and Lin [13,14] suggested maximizing the lowest d_j, as the overall desirability value of the responses. The loss function approach (equation 2) attempts to minimize the costs associated with the distances of the responses from their targets namely:

$$L\big(y(x)\big) = (y(x) - T)'C(y(x) - T), \tag{2}$$

Here y(x) is the vector of responses, x is the vector of control factors, T is the target vector of the responses and C is the cost matrix containing the relative importance of each response. In the third category, some multi-attribute value function is used. Mollaghasemi et al [15] used this method and then they used the gradient search technique to find the optimum value of the assessed function. Boyle [16] applied a Pair-wise Comparison Stochastic Cutting Plane (PCSCP), which combines features from interactive multi-objective mathematical programming and response surface methodology (RSM). Response surface methodology (RSM) is a collection of mathematical and statistical techniques used in the empirical study of the response and control factors. Detailed descriptions of various response surface techniques can be found in Myers and Montgomery [17] they used RSM to optimize multiple response processes and products using design experiments. Su et al [18] proposed a new circuit design optimization method where genetic algorithm is combined with the Taguchi method. Lo and Tsao [19] modified an analytical linkage-spring model based on neural network analysis and the Taguchi method to determine the design rules for reducing the loop height and the sagging altitude of the gold wire-bonding process of the integrated circuit (IC) package. The main disadvantage of the RSM and Taguchi is that these methods can only be used for single-response problems. They cannot be used to optimize multiple response optimization problems. But in most industries, we have dealt with more than one response variable, and improving them simultaneously is important and the main purpose of this paper. The fourth category used heuristic algorithms. Because of traditional methods weakness in MRO problems when number of responses are increased. Specially researchers work on genetic algorithms (GA) and improve it, relative to their problem.

Methods

In this article, we utilize artificial neural networks to estimate the relationship between control factors and responses. At first, we detect significant control factors of each response and train a network for that response with their significant control factors. The most important step in neural networks is training step that applies optimization algorithms to determine relation between control factors and responses precisely. Most of the algorithms used in training artificial neural networks employ some form of gradient descent. This is done by simply taking the derivative of the cost function with respect to the network parameters and then changing those parameters in a gradient-related direction. Supervised neural networks use a Mean Square Error (MSE) cost function that uses formal statistical methods to determine the confidence of the trained model. Networks that have been trained by gradient descent-based algorithms have high MSE (lower is better). Here we use metaheuristic algorithms to determine the optimum weights of neural networks. Particle swarm optimization and genetic algorithm as the most famous and strongest methods in optimization are applied to compare efficiency of them in MRO problems. Then we use the desirability function proposed by Ortiz et al [6] to model the neural network output to a single function. At the end of the optimization step, we search problem space by PSO to find optimization settings. Here we use neural networks to utilize function approximation and use metaheuristic algorithms because of their strength in optimizing of complex functions. In the next sections, we explain desirability function, neural networks, particle swarm optimization and genetic algorithm. At the end, before conclusion, usage of the proposed method on two case study from literature is illustrated.

The Modeling of All Responses to a Single Function by Desirability Function

The desirability function approach is one of the most used methods in industry for optimization of multiple-response problems which transforming multi objective problem into a single objective function. In this method if the quality of the product was influenced by multiple quality characteristics when only one of the characteristics lies outside the desired limits, quality of product is completely unacceptable. The desirability function assigns a score to a set of all responses and chooses factor settings that maximize that score. The desirability function transforms each response into its desirability according to its target value. Depending on whether a particular response,y_j, is to be maximized, minimized, or assigned a target value, different desirability functions can be used. Derringer and suich [10] proposed a useful class of desirability functions. Special desirability, $d_j(y_j)$, $j = 1,2,\dots,m$ transforms a response into a value in the range $0 < d_j(y_j) < 1$. The higher $d_j(y_j)$ is better. At the end of this phase, all desirability functions combined to a single total desirability, $D(x)$.

Defining Individual Desirability Function

Derringer and suich [10] define the individual desirability function as:

$$d_j\left(\hat{y}_j(x)\right) = \begin{cases} 0 & if\ \hat{y}_j(x) \le y_{\min j} \\ \left(\frac{\hat{y}_j - y_{\min j}}{y_{\max j} - y_{\min j}}\right)^r & if\ y_{\min j} \le \hat{y}_j(x) \le y_{\max j} \\ 1 & if\ \hat{y}_j(x) \ge y_{\max j} \end{cases} \tag{3}$$

For the responses that would be maximized (one-sided), and:

$$d_j\left(\hat{y}_j(x)\right) = \begin{cases} 0 & if\ \hat{y}_j(x) \le y_{\min j} \\ \left(\frac{\hat{y}_j - y_{\max j}}{y_{\min j} - y_{\max j}}\right)^r & if\ y_{\min j} \le \hat{y}_j(x) \le y_{\max j} \\ 1 & if\ \hat{y}_j(x) \ge y_{\max j} \end{cases} \tag{4}$$

For the responses that would be minimized (one-sided), and:

$$d_j\left(\hat{y}_j(x)\right) = \begin{cases} \left(\frac{\hat{y}_j - y_{\min j}}{T_j - y_{\min j}}\right)^s & if\ y_{\min j} \le \hat{y}_j(x) \le T_j \\ \left(\frac{\hat{y}_j - y_{\max j}}{T_j - y_{\max j}}\right)^t & if\ T_j \le \hat{y}_j(x) \le y_{\max j} \\ 0 & otherwise \end{cases} \tag{5}$$

For those responses that should be reached to a bounded value (two-sided). Where $\hat{y}_j, j = 1,2,\dots,m$ is the output of the jth network, $y_{\min j}$ is minimum value, $y_{\max j}$ is maximum value and T_j is target values for the jth response. r, s, t are the weights that allow for linear $(s = t = 1)$ or nonlinear behavior between a bounded ($y_{\min j}$ or $y_{\max j}$) and the target(T_j). In this paper r,s,t are equal 1. Ortiz et al [6] added a penalty term to the model proposed by Derringer and suich [10] which helps the optimization algorithm to maintain an infeasible solution while not allowing it to have a total desirability higher than a feasible solution:

$$p_j(\hat{y}_j) = \begin{cases} c + \left|\frac{\hat{y}_j - y_{\min j}}{T_j - y_{\min j}}\right| & -\infty \leq \hat{y}_j \leq y_{\min j} \\ c & y_{\min j} \leq \hat{y}_j \leq y_{\max j} \\ c + \left|\frac{\hat{y}_j - y_{\max j}}{T_j - y_{\max j}}\right| & y_{\max j} \leq \hat{y}_j \leq +\infty \end{cases} \quad (6)$$

c is a positive and very small number near zero like 0.00001 which forces $p_j(\hat{y}_j)$ to be greater than zero.

Design of Experiments (DOE)

At first, for optimization of multiple response systems, operation system's data is needed. Design of experiments help us to collect these data by giving specified inputs as control factors and record output values responses. RSM designs such as central composite design or Box-Behnken designs, due to their ability to provide the required information by covering the experimental space more thoroughly, are usually considered as effective designs for collecting the required data. These data help the neural networks to approximate the process function more precisely. As illustrated in figure 2, proposed method starts with designing an experiment.

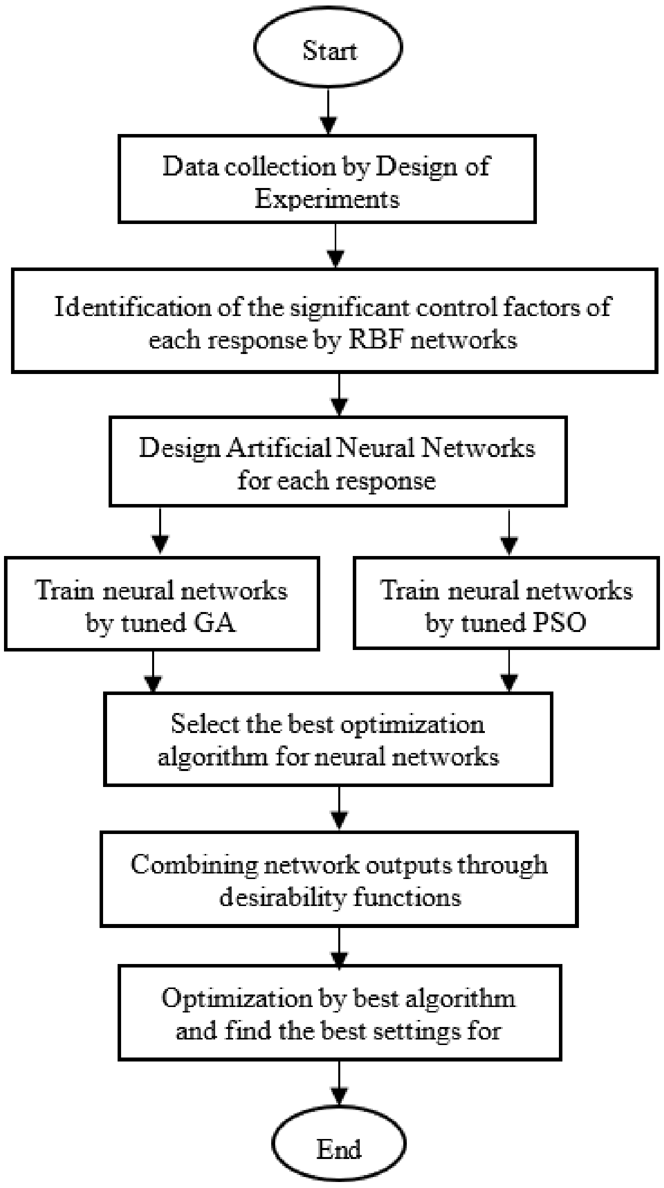
Start
Data collection by Design of Experiments
Identification of the significant control factors of each response by RBF networks
Design Artificial Neural Networks for each response
Train neural networks by tuned GA
Train neural networks by tuned PSO
Select the best optimization algorithm for neural networks
Combining network outputs through desirability functions
Optimization by best algorithm and find the best settings for
End

Figure 2 Flowchart of the proposed approach

Artificial Neural Networks

Machine learning is a field of Artificial Intelligence that allows computers to learn from data and make decisions without explicit programming. It powers innovations in areas like healthcare [21-24], financial systems [25-26], and different fields in engineering [27-31], enabling systems to predict outcomes, automate tasks, and improve over time through pattern recognition and data analysis. Neural networks are a key component of machine learning, inspired by the structure of the human brain. They consist of layers of interconnected nodes that process data by adjusting weights and biases to learn patterns. Neural networks are especially effective for complex tasks like image recognition, natural language processing, and deep learning applications, enabling

systems to handle vast amounts of data with high accuracy.
ANN is a set of layers of parallel processors that are called neurons. Each neuron can process information separately and has a connection with other neurons in the next and previous layers. ANN would be trained for each response to approximate its relation with control factors. Training the neural network is an important operation for accurate results, and less training makes the ANNs inefficient and may provide inaccurate predictions. In the training phase subset of data (control factors and responses) is used to learn the network by means of a suitable algorithm for each response, so for each response, a network should be trained. A number of neurons in the input layer is equal to the control factors and in the output layer, there is a single neuron. Figure 3 illustrates the topology of neural networks. Also, it is shown that demonstrated that a single hidden layer, given enough neurons, can form any mapping needed. In practice, two hidden layers are used to speed up convergence. The additional hidden layers are not necessary. The important thing to remember is that the network learns best when the mapping is simplest, so two hidden layers is enough.

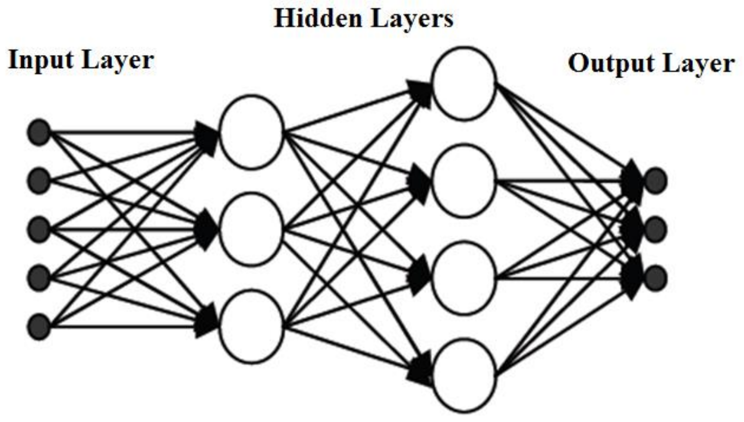

Figure 3 Topology of neural networks

The neural network uses a set of control factors (as inputs) and responses (as outputs) parameter values usually written as row vectors. A couple of correspondent input and output row vectors are joined into a new vector called “example curve” or simply “curve”. All the curves so obtained are grouped in a “training matrix”. Data in each curve are values of Y_j and X_i parameters which can be either experimental or calculated. We indicate Y_j as output and X_i as input parameters respectively. The input layer contains neurons that receive input data values from the rows of the training matrix. This information is transmitted from the $i - ththe\ the\ the$ neuron of a layer to $j\ the$ neuron of the subsequent one after weighing with a weight w_{ij}. In each neuron of a hidden layer weighed inputs coming from the previous ones are summed each other and added to a bias. The result is then transformed by means of a suitable mathematical function to obtain an output called “activation” of the neuron. The activation is transferred to neurons in the next layer after another weighing step. In the last layer, output parameter values are estimated using a suitable transformation function. The described process is called “learning” and it is repeated iteratively. After each epoch the estimated values Y_N (network output) of output, parameters are compared with those Y_R (real output from DOE step) of the corresponding curve in the training matrix and the value of MSE is calculated as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_{i(N)} - Y_{i(R)})^2 \quad (7)$$

Where n is the number of rows of the training matrix. During learning, weight values are changed according to suitable algorithms in order to decrease the value of MSE. Different methods of training algorithms were developed to increase the training speed and performance. [32]

In our study, the back-propagation algorithm was used. it is a convenient and simple iterative algorithm that locates the optimal weights by minimization of an error function of the training set. The learning is complete when the lowest MSE is reached. For a given system studied, the network architecture must be optimized. While the number of input and output neurons is given by data used, the optimal number of hidden layers and that of their neurons is found using the criteria of the lowest MSE. If hidden layers are comprised of a small number of neurons our training and total MSE will both be high caused by bias and underfitting. Then increase the number of nodes in the hidden layer, one at a time, until the generalization error begins to increase, this time due to overfitting. For each hidden layer, a graph of MSE value versus the number of neurons depicted that the optimal number of neurons in that hidden layer is given by the point of intersection of two branches of the graph.
Here we select 80 percent of data randomly as training data and the rest of them as test data. Neural network training needs a large amount of data, but sometimes in industries because of expensive or time-consuming tasks, we cannot prepare a large amount of data. We applied DOE to achieve that points of the problem space that have a large information and represent the problem space truly. Here we use metaheuristic algorithms in optimization MSE in neural networks to predominate this problem. Network training is the most important phase in multiple-response optimization, because if we simulate the multi-response system precisely, then we have a precise problem space that describes the real multiple-response system. In the neural network training phase, descent gradient-based algorithms are most used. In multiple response optimization, we have a low amount of data derived from the design of experiments (DOE), and the descent gradient algorithm with a low amount of data gives high MSE in simulation, this means function approximation is not very accurate. Because of that, the optimization algorithm in neural network training must be very strong to achieve the lowest MSE. In this paper metaheuristic optimization algorithms (PSO and GA) because of their strength in optimization are utilized in the network training step.

Problem Modeling

In this step by designing MLP or RBF network for each response and then training and testing the network will estimate the relation between control factors and responses to approve the acceptable training for each network, the network output for the test and training data must be plotted and compared with the real data that derived from DOE step.
The total desirability function, D(x) computed as follows:

$$D_{DS}(x) = [d_1(Y_1(x))d_2(Y_2(x))d_3(Y_3(x)) \ldots\ d_m(Y_m(x))\ ]^{1/m} \quad (8)$$

$$P(x) = [\sqrt[m]{(p_1(y_1)(p_2(y_2) \ldots (p_m(y_m)} - c]^2 \quad (9)$$

$$D(x) = D_{DS}(x) - P(x), \quad (10)$$

$D_{DS}(x)$ is the geometric mean of the individual desirability $(d_j(y_j))$ Derringer and Suich [10]. $P(x)$ is the overall penalty function. $P(x)$ will be zero for any feasible solution. $D(x)$ is the total desirability function that we want to maximize it. $D(x)$ can be used as a criterion for comparing different solutions.

Train Neural Networks by Particle Swarm Optimization (PSO) and Genetic Algorithm (GA)

In this step, we use PSO and GA in neural network training. These algorithms are selected because versus unlike other optimization algorithms are powerful search methods for optimizing highly nonlinear and complex functions. In the next subsections PSO and GA are explained, also in examples, we compare both of them in the neural network training phase.

Particle Swarm Optimization

Particle swarm optimization (PSO) is a population-based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995 and is a search strategy inspired by the social behavior of bird flocking or fish schooling and attempts to find the global optimal solution based on movement of organisms [33]. PSO starts with a population of candidate solutions and moves these particles around in the search space such that each particle's movement is influenced by its local best-known (*l_best*) position that is explored by that particle and is also guided towards the best-known position (*g_best*) that explored by the swarm. The particle swarm optimization at each time step, updates the velocity of each particle by a random term, toward its g*_best* and *l_best* locations. At the end, all particles move step by step to a location that has a best fitness in the problem search space. Equation (11) shows that the velocity vector is updated by the global best position, personal best position and current position of each particle, equation (12) shows that each particle moves with its own velocity.

$$v_i(t+1) = w.v_{i(t)} + c_1.rand\big(l_{best} - x_i(t)\big) + c_2.rand(g_{best} - x_i(t)) \tag{11}$$

$$x_i(t+1) = x_i(t) + v_i(t) \tag{12}$$

Where i indexes the particle, t is the index of iteration, v_i is the vector of velocity, x_i is the position of a particle, w is the weight of the current velocity, c_1 is the weight of the difference between local best position and current position, c_2 is the weight of the difference between global best and current position and $rand$ is used for randomization. Here, we use some methods to optimize the weights which are illustrated in Table 1. Six networks from example 1 of this study were trained by these algorithms. Results in all networks were approximate.

***Table 1** Six types of PSO that designed to optimize neural network training*

NO	Method	Author (Year)	*w*	C1	C2	*Train MSE*	*Test MSE*
1	Dual Layered PSO (DLPSO)	Subrarnanyam, et. al. [34]	1	C1+(n/$Iter_{max}$)	C2-(n/$Iter_{max}$)	0.0291	0.4734
2	Neural PSO (NPSO)	Dou, et. al. [35]	0.5	1.4	1.4	0.0132	0.1063
3	Hybrid rescursive PSO (HRPSO)	Chen et. al. [36]	0.75	1.5	1.5	0.0336	0.1509
4	Interactive PSO (IPSO)	Madar et. al. [37]	$w = 0.4 + \frac{Iter_{max} - i}{2 * Iter_{max}}$	2	2	0.0077	0.0972
5	Self-adaptive velocity PSO(SAVPSO)	Lu and Chen [38]	1/2	1	1	0.0317	0.1834
6	Adaptive Dissipative PSO(ADPSO)	Shen et al. [39]	0.4 ≤w≤0.9	2	2	0.0163	0.1725

$* \, Iter_{max}$ *is maximum iteration*

Figure 4 shows the performance of Table 1, PSO algorithms in third neural network training with 200 iterations and 25 particles. As seen in Table 1, in all experiments the best algorithm with the lowest MSE in the test and training phase is Interactive PSO (IPSO), so in this study we select the IPSO algorithm as the best to train neural networks in the MRO problem.

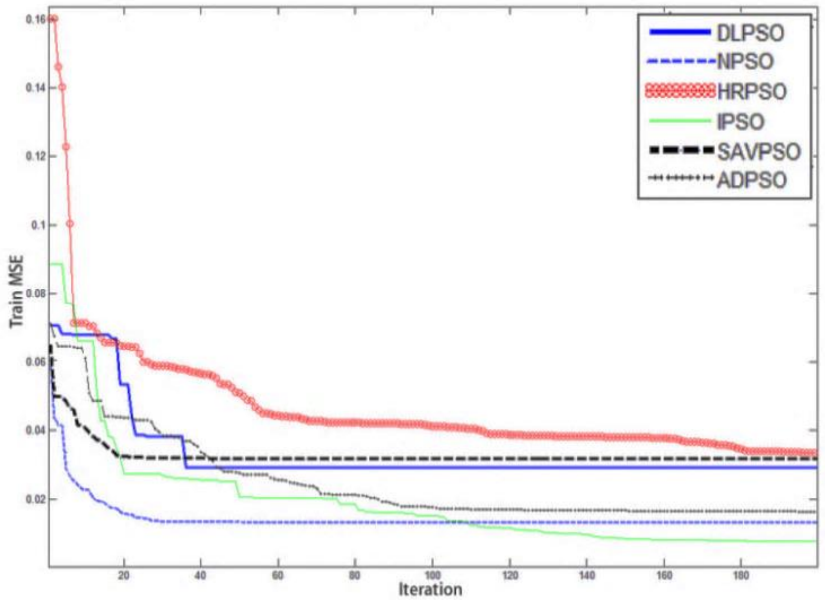

Figure 4 The MSE of Train in PSO types

In the next step as you see in table 2, we train one network with various number of particles:

Table 2. MSE of the train and test with PSO swarm size

swarm size	n=5	n=10	n=15	n=20	n=25	n=30	n=35
MSE Train	0.3128	0.0929	0.2105	0.0749	0.0355	0.0708	0.0603
MSE Test	1.0634	0.7083	0.1814	0.6331	0.1878	0.6101	0.3621

In this paper we use 25 particles as swarm size and algorithm iteration is 1000.

Genetic Algorithm

Nils Aall Barricelli introduced GA in 1954. GAs were a computational analogy of adaptive systems. They are modeled loosely on the principles of evolution via natural selection, employing a population of individuals (a population of candidate solutions called individuals) that undergo selection in the presence of variation-inducing operators such as mutation and crossover. A fitness function is used to evaluate individuals, and reproductive success varies with fitness. The steps of GA are:

a) Randomly generate an initial population M(0).
b) Compute and save the fitness u(m) for each individual (m) in the current population M(t).
c) Define selection probabilities p(m) for each individual (m) in M(t) so that p(m) is proportional to u(m).
d) Generate M(t+1) by probabilistically selecting individuals from M(t) to produce offspring via genetic operators
e) Repeat step b until a satisfying solution is obtained.

GA has various parameters whose values need to be determined before the optimization phase begins. Different authors, including Ortiz et al [6], have proposed the use of a robustly designed experiment to determine the best settings for a GA's parameters. Therefore, we have incorporated a robustly designed experiment to find the best settings for the parameters. The GA's control and noise variables and their levels are shown in Table 3.

Table 3. Levels of the control and noise variables used in the robustly designed experiment

Variable	Low level	High level	Type
Parent population	20	50	Control
Parent/offspring ratio	1:1	1:7	Control
Selection type	Rank	Tournament	Control
Number of elites	2	6	Control
Crossover rate	0.5	0.85	Control
Mutation type	Uniform	Gaussian	Control
Number of factors	4	8	Noise
Number of responses	4	16	Noise
Constraint width (% of target)	5	15	Noise

The GA performance measures are the same as in Ortiz et al [6]. For more details refer to those papers. The final parameter settings for the robust GA are shown in Table 4.

Table 4. *Final parameter settings for the genetic algorithm (GA)*

Parameter	Value
Parent population	20
Parent/offspring ratio	1:7
Selection type	Tournament
Number of elites	2
Crossover rate	0.85
Mutation type	Gaussian

Finally, the tuned GA is run for 1000 iterations with 20 population size and the most desirable solution with the highest total desirability would be the final solution.

Optimizing via Particle Swarm Optimization (PSO) algorithm

In previous steps, our MRO problem is simulated, in this step an optimization algorithm is applied to search the problem space and find the optimum settings to achieve desirable responses. Here we use PSO to perform the optimization. gradient-based optimization methods such as GRG, cannot be used because they require surfaces to compute the gradient and direction of improvement. However, when neural networks are used, no response surface would occur. PSO is known as a potent heuristic search method for complex function optimization.

Results

We provide a results section with two known and important examples and compare our method with others:

First Comparison (the wire-bounding process in the semiconductor industry)

This example discussed by Del Castillo et al [40] is about the semiconductor industry's wire-bounding process. This example is chosen because Ortiz et al [6], Dell Castillo et al [40] solve it by another MRO approach and it’s possible to compare the final results. During this process, the manufacturer must assemble a hybrid module in a pre-modeled package by bonding wires between the leads (position A, Fig 5) and the silicon chips (position B, Fig 5). The control factors that influence the temperature at the wire bond are the N_2 flow rate (x_1), the N_2 temperature (x_2) and the heater block temperature (x_3). The responses for the experiment are:
$Y_1 = maximum\ temperature$ at position A,
$Y_2 = beginning\ bond\ temperature$ at position A,
$Y_3 = finish\ bond\ temperature$ at position A,
$Y_4 = maximum\ temperature$ at position B,
$Y_5 = beginning\ bond\ temperature$ at position B,
$Y_6 = finish\ bond\ temperature$ at position B.

***Table 5.** Factors, levels for Box-Behnken experimental design*

Factor	Name	Units	Low level	High level
A	Flow rate	SCFM	40	120
B	Flow temp	℃	200	450
C	Block temp	℃	150	350

To investigate the three control factors, a Box-Behnken [5] design was used. Box–Behnken experimental designs do not suggest factor values that are all at their maximum or minimum level in the same experiment (run). In this way, experiments performed under extreme conditions that can lead to unsatisfactory results are bypassed by Siljic Tomic [41]. The control factors, along with their levels used in the design, is shown in table 5. The experimental results are shown in Table 6.

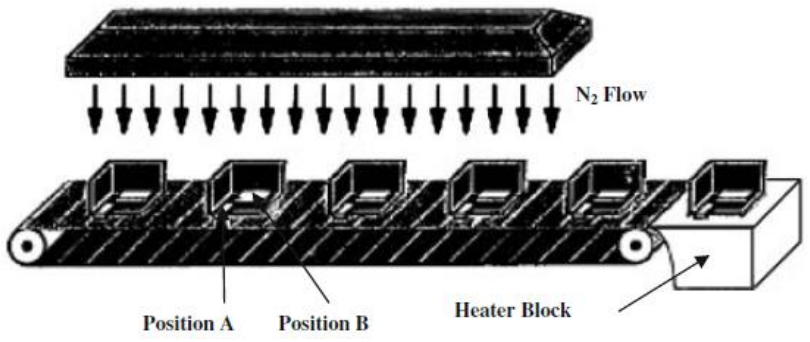

Figure 5 Wire bond heating system, Del Castillo et al [36]

***Table 6.** Experimental runs*

Flow rate	Flow temp	Block temp	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
40	200	250	139	103	110	110	113	126
120	200	250	140	125	126	117	114	131
40	450	250	184	151	133	147	140	147
120	450	250	210	176	169	199	169	171
40	325	150	182	130	122	134	118	115
120	325	150	170	130	122	134	118	115
40	325	350	175	151	153	143	146	164

120	325	350	180	152	154	152	150	171
80	200	150	132	108	103	111	101	101
80	450	150	206	143	138	176	141	135
80	200	350	183	141	157	131	139	160
80	450	350	181	180	184	192	175	190
80	325	250	172	135	133	155	138	145
80	325	250	190	149	145	161	141	149
80	325	250	180	141	139	158	140	148

In this design, three replicates are available at the center of the point. Hence, one of the runs is chosen randomly. The RBF networks were first designed to identify the significant control factors. The MSE for the test data corresponding to the full model and models including two control factors are shown in Table 7.

Table 7. *Experimental runs*

Factors	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
A,B,C	463	392	400.5	157.3	60.2	22.6
B,C	12.7	26.8	103.4	994.1	187.2	163
A,C	3809	1706	390.5	2922	1440	810.7
A,B	246.4	3367	1207.2	311.3	237.1	1070.6

Significant control factors for each of the six responses are shown in Table 8. As you see factor A for responses Y1, Y2 and Y3 is not significant. For responses Y4, Y5, and Y6 there is significant difference between the full model (A, B, C) and the subset control factor models, because of that we select the full model. This means that all control factors are significant and couldn't be discarded each of in calculations for these responses.

Table 8. *Significant control factors for responses*

Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
B,C	B,C	B,C	A,B,C	A,B,C	A,B,C

By comparing the significant control factors achieved by our approach and those that were used in the regression models of Del Castillo et al. [40], it can be seen that, except for Y_2 and Y_3, all of the model shares the same factors. For these two responses, their model includes factor A, which is excluded by our approach. However, it should be noted that the regression multiplier calculated for factor A in both of their models is similar than the rest of the multipliers. Next, considering the control factors mentioned in Table 8 for each response, different MLP and RBF networks with different parameters are trained. In a neural network when an iterative training algorithm is run, two problems may occur. First over learning (overfitting) and second over-parameterization.

Overfitting occurs when the algorithm is run for too long and the network is too complex for the problem or the available quantity of data. To prevent overfitting must increase the number of neurons in hidden layers, but increasing the neurons in hidden layers causes to occur over-parameterization. In this study, we plotted the MSE of the train and test data in all likelihood the number of neurons for each network, so the optimum number of neurons in each layer is obtained. For example, the first hidden layer of network five has five neurons, figure 6 illustrates that the optimum number of neurons in the second hidden layer in this network is three. The most appropriate networks with MSE are presented in Table 9.

Table 9. *Properties of the six neural networks*

Net	Output	Type	NO. of neurons in the hidden layers		MSE	
					Test	Training
1	Y_1	MLP	6	4	0.0299	0.0138
2	Y_2	MLP	5	4	0.0200	0.0017
3	Y_3	MLP	6	4	0.0142	0.0178
4	Y_4	MLP	5	3	0.0021	0.0065
5	Y_5	MLP	5	3	0.0115	0.0248
6	Y_6	MLP	7	4	0.0360	0.0086

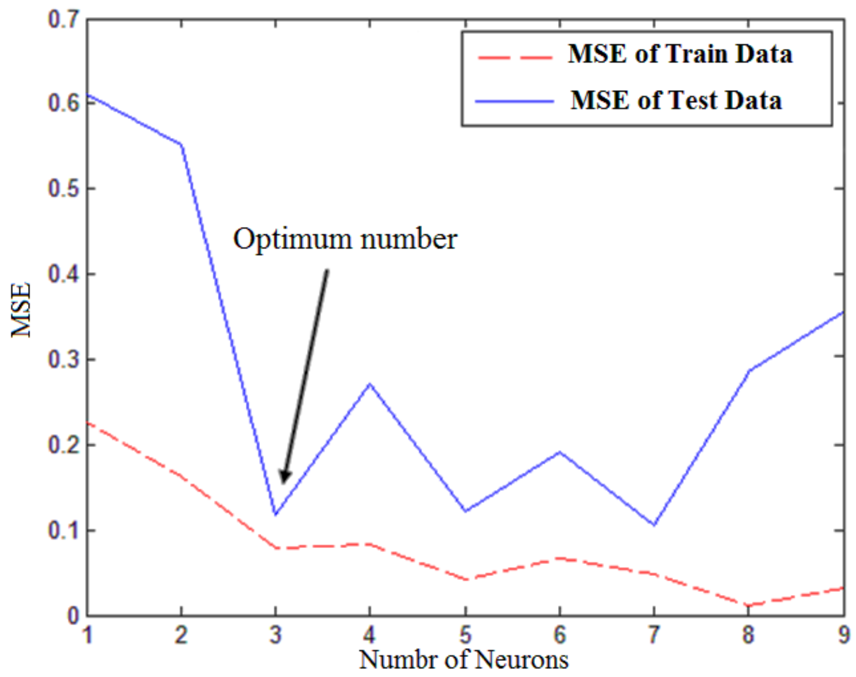

Figure 6 Optimum number of neurons in second hidden layer of network five (first hidden layer has 5 neurons)

As can be seen, MLP is selected as the best network for all of the responses. All of the MLP networks have two hidden layers with hyperbolic tangent activation functions. The output neurons have linear activation functions. We use first GA as the training algorithm in all of the six networks and repeat the training phase with PSO. Here we use GA and PSO because some algorithms such as Levenberg-Marquardt and descent gradient algorithms do not have a low MSE in train and test data. The final results are shown in Table 10, network training by Levenberg-Marquardt is done by Noorossana et al [20], and Table 10 use that's the result. This table shows that the performance of PSO in neural network training is better than others. So, in this article, we use PSO (in real, IPSO) as the best optimization algorithm in the ANN training step in the MRO problem. Figure 7 shows the trend of GA and PSO algorithms versus MSE of the train for the fourth neural network.

Table 10. *Comparison of Levenberg-Marquardt, GA, and PSO algorithms in neural network training*

MSE versus training algorithms		**Levenberg-Marquardt**	**GA**	**PSO**
Y_1	Train MSE	0.08	0.0552	0.0138
	Test MSE	0.55	0.1885	0.0299
Y_2	Train MSE	0.11	0.0326	0.0017
	Test MSE	0.55	0.1028	0.0200
Y_3	Train MSE	0.00	0.0150	0.0178
	Test MSE	1.98	0.1833	0.0142
Y_4	Train MSE	0.20	0.0267	0.0065
	Test MSE	1.60	0.0995	0.0021
Y_5	Train MSE	0.03	0.0205	0.0248
	Test MSE	14.00	0.0174	0.0115
Y_6	Train MSE	0.28	0.0122	0.0086
	Test MSE	1.14	0.7364	0.0360

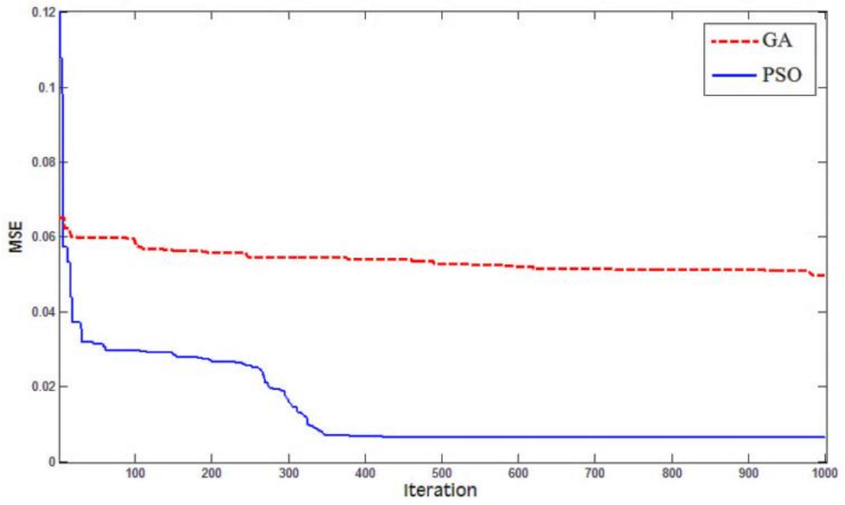

Figure 7 Comparison of GA and PSO in fourth network training

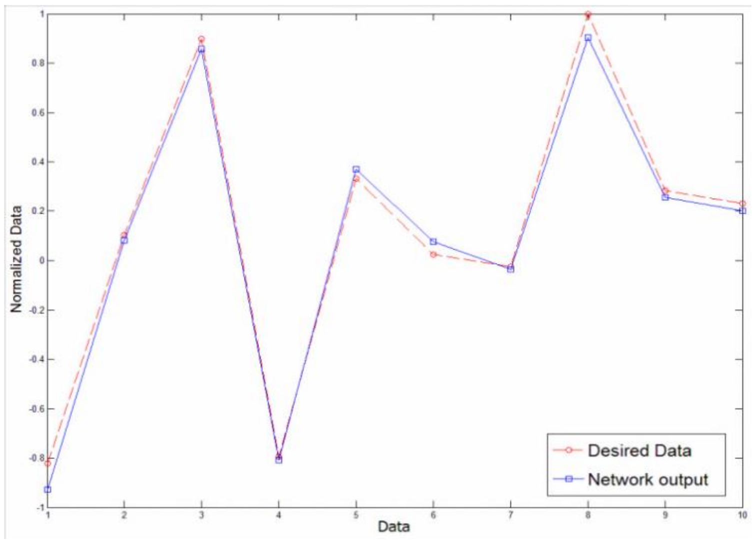

Figure 8 Desired data and network output in the training of the fourth neural network by PSO

The MSE of the six regression models presented in Del Castillo et al. [40] are presented in Table 11. As can be seen, the computed MSE from the regression models is high, and this shows a poor fitness of the models. However, the six neural networks produce absolutely lower MSE, hence, they can approximate the process function more accurately.

Table 11. *MSE from the regression models*

Factors	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
MSE	78.31	38.64	63.33	33.25	8.38	7.58

To use the desirability approach, the process engineer selects the lower, upper and target values for individual desirability, as shown in Table 12. Individual desirability are supposedly linear, with $s = t = 1$ and c equal to 0.0001.

Table 12. *Desirable values for six responses*

Factors	$y_{\min j}$	T_j	$y_{\max j}$	$d_j(y_{\min j})$	$D_j(T_j)$	$D_j(y_{\max j})$
Y_1	185	190	195	0.0	1.0	0.0
Y_2	170	185	195	0.0	1.0	0.0
Y_3	170	185	195	0.0	1.0	0.0
Y_4	185	190	195	0.0	1.0	0.0
Y_5	170	185	195	0.0	1.0	0.0
Y_6	170	185	195	0.0	1.0	0.0

Now our MRO system is simulated and by using another PSO algorithm, we can find optimum settings. PSO selects some number as inputs and follows outputs of the simulated problem, then changes inputs intelligently to find the best input combination finally. Because of using a potent PSO algorithm in the training step, optimization of neural network output is not very hard. To prove this claim, problem space must be plotted but the problem space of this example is 4-dimensional (three control factors and one total desirability based on control factors) so we cannot draw that. In Figure 9, it is assumed that X_3 is a fi number then $X_1, ,X_2$ and total desirability is plotted, after that X_3 change to another fixed number and another plot is drawn, finally, all plots are merged together. Of course, due to image cluttering, all plots are stored in a matrix form, and at the end, all matrixes are plotted. As you see in Figure 9, this MRO search space is not very complicated and by simple optimization algorithm, we can find optimum settings. Here simple PSO is applied to optimize the output of the neural network. If you take more samples in the design of the experiments stage, you will have a smoother surface.

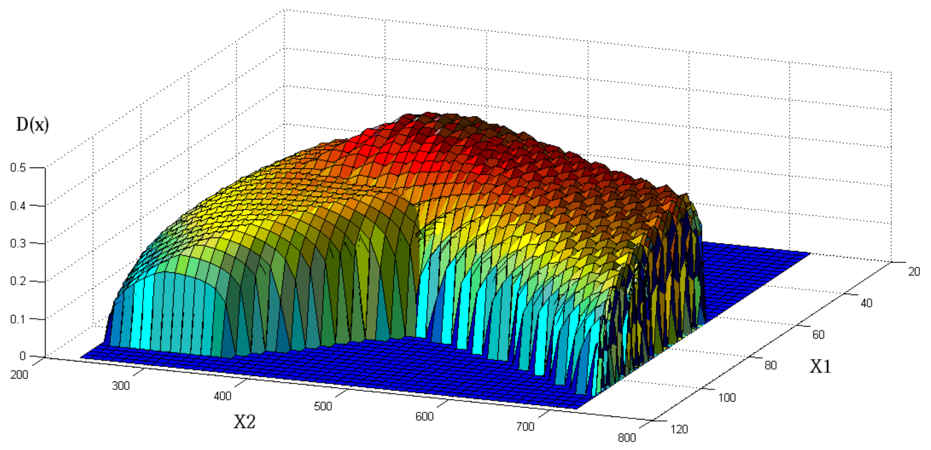

Figure 9 MRO problem search space in example 1

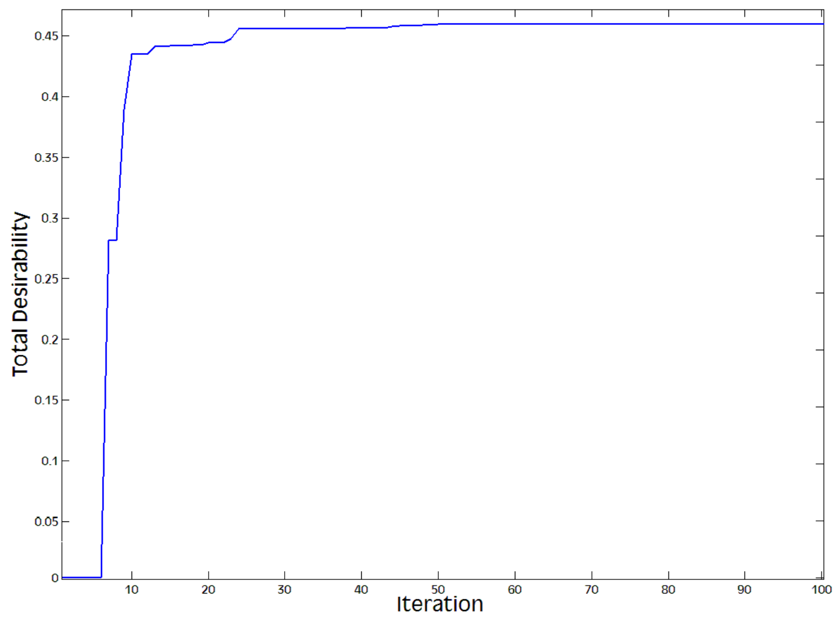

Figure 10 Final optimization by PSO in example 1

Table 13 compares the solution found using the revised desirability approach of Del Castillo et al. [36], the unconstrained desirability approach of Ortiz et al. [6], Noorossana et al [20] and our proposed approach. As can be seen in Table 13, the total desirability achieved by the proposed approach shows its ability to optimize multiple-response problems in comparison with other approaches.

Table 13. *Comparison of the final solution*

Approach	X	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	$D(x)$
Proposed	(66.80,468.60,357.80)	189.9987	173.7826	175.1454	190.3942	172.6982	179.6304	0.4578
Noorossana et al [20]	(68.97,370.00,286.06)	192.1	184.1	181.0	193.7	180.3	171.2	0.4168
Ortiz et al [6]	(74.55,472.90,332.75)	187.0	176.7	173.8	192.9	174.2	186.2	0.4081
Dell Castillo et al [40]	(84.16,450.00,329.87)	186.0	174.5	172.0	192.6	173.0	185.0	0.3061

Second Comparison

In this section, a simulation study is carried out on the example given by Tong et al. [42]. He uses the principal component analysis (PCA) technique to solve this example. Another one, he uses the VIKOR method by Tong et al. [43] to solve it again. In this example, there are five control factors and two responses. It is assumed that both responses have the same relative importance and are smaller-the-better and larger-the-better, respectively. Five control factors, each with three levels, are allocated sequentially to an L_{18} orthogonal array. The experiments are conducted randomly. Table 14 shows the experimental observations.

Table 14. *Experimental data*

NO.	**Control factors**					**Responses**	
	X_1	X_2	X_3	X_4	X_5	Y_1	Y_2
1	-1	-1	-1	-1	-1	14.3	4.0
2	-1	0	0	0	0	15.7	4.3
3	-1	1	1	1	1	23.2	5.6
4	0	-1	0	0	0	12.1	3.7
5	0	0	1	1	1	8.70	4.9
6	0	1	-1	-1	-1	6.50	6.1
7	1	-1	0	-1	1	8.99	4.2
8	1	0	1	0	-1	11.8	4.3
9	1	1	-1	1	0	12.4	5.3
10	-1	-1	1	1	0	16.2	4.6
11	-1	0	-1	-1	1	26.9	4.1
12	-1	1	0	0	-1	10.5	5.3
13	0	-1	0	1	-1	16.9	3.9
14	0	0	1	-1	0	5.06	4.7
15	0	1	-1	0	1	7.08	5.4
16	1	-1	1	0	1	8.76	5.2
17	1	0	-1	1	-1	15.1	4.6
18	1	1	0	-1	0	5.00	5.8

Here, based on our proposed method, to identify the significant control factors two radial bias function (RBF) networks were designed to determine which control factors are significant for each response. The MSE for the test data corresponding to the full model and models including four control factors are shown in Table 15.

Table 15. *Experimental runs*

Factors	Y_1	Y_2
A,B,C,D,E	1.2160	1.3252
A,B,C,D	0.6938	8.8083
A,B,C,E	7.2435	1.1648
A,B,D,E	5.4616	2.3622
A,C,D,E	3.2952	2.5940
B,C,D,E	2.2240	3.3579

A, B, C, D and E are control factors. Significant control factors for each of the two responses are shown in Table 16. As you see factor E for responses Y1 and factor D for response Y2 is not significant.

Table 16. *Significant control factors for responses*

Y_1	Y_2
A,B,C,D	A,B,C,E

Now we design two MLP neural networks then train them by GA and PSO algorithms. Test and train data is selected randomly from DOE data with a ratio of 20 to 80 percent. Table 17 shows the performance of the GA and PSO in neural network training. As you see same as example **I**, PSO works better than GA. Also, performance of PSO versus GA in second neural network is shown in figure 11. Also figure 12 shows the proceeding of second neural network that follows the real data precisely. Optimize responses are Y_1=5.3427, Y_2=5.9626

Table 17. *The MSE of the test and train data in neural network training by using GA and PSO.*

MSE versus training algorithms		**GA**	**PSO**
Y_1	Train MSE	0.0886	0.0128
	Test MSE	0.3273	0.2404
Y_2	Train MSE	0.0235	0.0029
	Test MSE	0.0650	0.0106

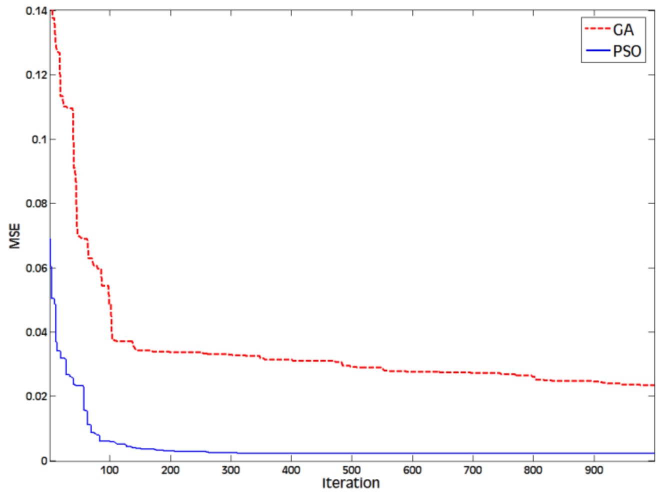

Figure 11 training of the second neural network by GA and PSO.

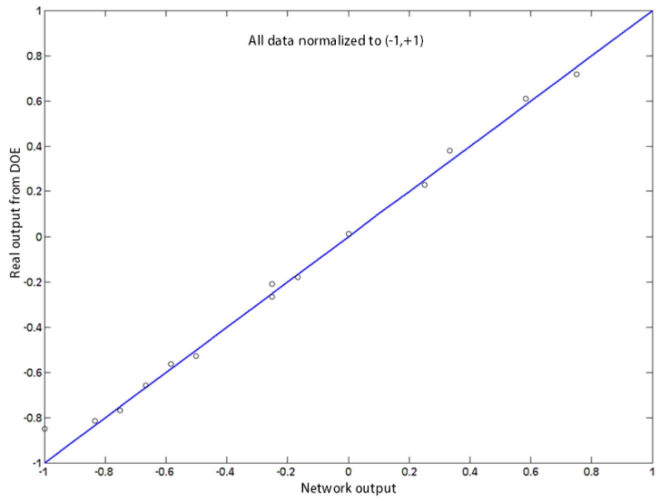

Figure 12 Desired data and network output in the training of the second neural network by PSO

Now by utilizing the PSO algorithm, we can find optimum settings this PSO has 50 iterations and the particle size is 10. See Figure 13. Orouskhani et al [44] proposed to weight each Node because central nodes are more important than others, because of having low data in this paper we can't use this method, but we propose this weighting method if our case study has more training data. Table 18 compares the proposed method with Lin et al. [45] and Tong et al [42,43]. As can be seen in Table 18, the total desirability achieved by the proposed approach shows its ability to optimize multiple-response problems in comparison with other approaches.

Table 18. *Comparison of the final solution*

Approach	X_1	X_2	X_3	X_4	X_5	D(x)
Proposed method	0.332	1	-0.081	-0.998	1	0.963
Lin et al [41]	1	1	0	-1	-1	0.940
tong et al [38]	1	1	1	-1	1	0.884
tong et al [39]	1	1	0	0	-1	0.819

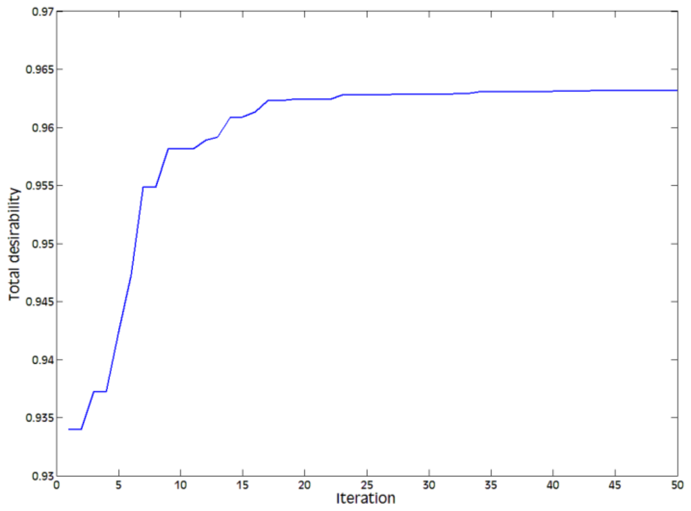

Figure 13 Final optimization by PSO in example 2

Conclusion

We proposed an approach for solving multiple response optimization problems using neural networks and metaheuristic algorithms. A radial basis function (RBF) neural network identifies significant control factors, and for each response, a multi-layer perceptron (MLP) is trained using two metaheuristic algorithms: genetic algorithm (GA) and particle swarm optimization (PSO). The performance of both algorithms is compared using mean square error (MSE), with PSO yielding better results. The optimized outputs are combined into a single function using a desirability function, and the best inputs are found via PSO. Two examples, including a semiconductor manufacturing case, demonstrate lower MSE compared to other methods, confirming the reliability of our approach. Our method addresses key issues: 1) it improves upon polynomial regression models for complex processes, 2) it considers the relative importance of multiple responses, 3) it avoids mathematical complexities, 4) it doesn't require statistical assumptions, and 5) it helps reduce problem complexity by filtering out insignificant factors. The method efficiently utilizes DOE data for accurate results. Future work could explore other metaheuristic algorithms, such as CSO, ICA, and IWD, for further comparisons.

References

[1] Montgomery DC. Design and analysis of experiments. 3rd edition. Wiley, New York. 1991.

[2] Carlos A, *Coello Coello. An updated survey* of GA-based multiobjective optimization techniques. ACM Computing Surveys. 2000; 32: *109-143*.

[3] Hartmann N E, Beaumont R A. Optimum compounding by computer. Journal of the Institute of the Rubber Industry. 1968; 2: *272-275*.

[4] Biles W E, Swain J J. Optimization and Industrial Experimentation. *Wiley-Interscience,* New York, USA. 1980.

[5] Box G E P, Liu P Y T. *Empirical Model-Building* and Response Surfaces. John *Wiley & sons*, New York, USA. 1987.

[6] Ortiz F, Simpson J, Pignatiello J, Heredia-Langner A. A Genetic Algorithm Approach to Multiple-Response Optimization. Journal of Quality Technology. 2004; 36: 432-450.

[7] Clayton E, Weber W, *Taylor* B W. A goal *programming approach* to the optimization of multi-response *simulation models*. IIE Transactions. 1982; 14: *282-287.*

[8] Baesler FF, Sepulveda J A. Multi-response simulation optimization using stochastic genetic search within a goal-programming framework. proceedings of the 2000 Winter Simulation Conference. 2000.

[9] Myers R, Carter W. Response surface techniques for dual response systems. Technometrics. 1973; Vol.15: 301-317.

[10] Derringer G, Suich R. Simultaneous optimization of several response variables. Journal of Quality Technology. 1980; 12: 214-219.

[11] Pignatiello JJ. Strategies for robust multiresponse quality engineering. IIE Transactions. 1993; 25: 5-15.

[12] Derringer G. A balancing act: optimizing a product’s properties. Quality Progress. 1994; 27: 51-57.

[13] Kim K, Lin D. Dual response surface optimization: a fuzzy modeling approach. Journal of Quality Technology. 1998; Vol. 30: PP. 1-10.

[14] Kim K, Lin D. Optimization of multiple responses considering both location and dispersion effects. Working paper, POSTECH. 2002.

[15] Mollaghasemi M, *Evans* G, *Biles* W. An *approach* for optimizing multiple response simulation models. In *Proceedings of the 13th Annual* Conference on Computers in *Industrial* Engineering.1991: pp *201-203.*

[16] Boyle CR. An *interactive* multiple response simulation *optimization method. IIE Transactions.1996; 14:453-463.*

[17] Myers R, Montgomery DC. Response Surface Methodology: Process and Product Optimization Using Designed Experiments. 2nd ed. John Wiley & Sons Inc.,New York. 2002.

[18] Su Y, Bao Z, Wang F, Watanabe T. Efficient GA Approach Combined with Taguchi Method for Mixed Constrained Circuit Design. International Conference on Computational Science and Its Applications (ICCSA). 2011: 290-293.

[19] Lo Y L, Tsao C C. Integrated Taguchi method and neural network analysis of physical profiling in the wirebonding process. IEEE Transactions on Components and Packaging Technologies. 2002; Vol.25. No. 2: P.P. 270-277.

[20] Noorossana R, et al. An artificial neural network approach for multiple-response optimization. he International Journal of Advanced Manufacturing Technology. 2009; vol. 40, no. 11: pp. 1227-1238.

[21] Akhoondinasab M, Shafaei Y, Rahmani A, Keshavarz H. A Machine Learning-Based Model for Breast Volume Prediction Using Preoperative Anthropometric Measurements. Aesthetic Plastic Surgery. 2024 Feb;48(3):243-9.

[22] Mahmoudiandehkordi S, Yeganegi M, Shomalzadeh M, Ghasemi Y, Kalatehjari M. Enhancing IVF Success: Deep Learning for Accurate Day 3 and Day 5 Embryo Detection from Microscopic Images. International Journal of Applied Data Science in Engineering and Health. 2024 Aug 14;1(1):18-25.

[23] Maydanchi M, Ziaei M, Mohammadi M, Ziaei A, Basiri M, Haji F, Gharibi K. A Comparative Analysis of the Machine Learning Methods for Predicting Diabetes. Journal of Operations Intelligence. 2024 May 18;2(1):230-51.

[24] Ashrafi N, Liu Y, Xu X, Wang Y, Zhao Z, Pishgar M. Deep learning model utilization for mortality prediction in mechanically ventilated ICU patients. Informatics in Medicine Unlocked. 2024 Jan 1;49:101562.

[25] Tashakkori A, Talebzadeh M, Salboukh F, Deshmukh L. Forecasting Gold Prices with MLP Neural Networks: A Machine Learning Approach. International Journal of Science and Engineering Applications (IJSEA). 2024; 13:13-20.

[26] Tashakkori A, Erfanibehrouz N, Mirshekari S, Sodagartojgi A, Gupta V. Enhancing stock market prediction

accuracy with recurrent deep learning models: A case study on the CAC40 index. World Journal of Advanced Research and Reviews. 2024;23(1):2309-21.

[27] Mahdavimanshadi M, Fan N. Multistage Stochastic Optimization for Semi-arid Farm Crop Rotation and Water Irrigation Scheduling Under Drought Scenarios. Journal of Agricultural, Biological and Environmental Statistics. 2024 Sep 5:1-24.

[28] Zadeh SS, Khorshidi M, Kooban F. Concrete Surface Crack Detection with Convolutional-based Deep Learning Models. arXiv preprint arXiv:2401.07124. 2024 Jan 13.

[29] Zarreh M, Khandan M, Goli A, Aazami A, Kummer S. Integrating Perishables into Closed-Loop Supply Chains: A Comprehensive Review. Sustainability. 2024 Aug 5;16(15):6705.

[30] Khameneh RT. Metaheuristic approaches for maximum blood collection problem (Doctoral dissertation, Ozyegin University).

[31] Khameneh RT, Elyasi M, Özener OÖ, Ekici A. A non-clustered approach to platelet collection routing problem. Computers & Operations Research. 2023 Dec 1;160:106366.

[32] Karkehabadi A, Homayoun H, Sasan A. FFCL: Forward-Forward Net with Cortical Loops, Training and Inference on Edge Without Backpropogation. InProceedings of the Great Lakes Symposium on VLSI 2024 2024 Jun 12 (pp. 626-632).

[33] Aghakhani S, Pourmand P, Zarreh M. A Mathematical Optimization Model for the Pharmaceutical Waste Location-Routing Problem Using Genetic Algorithm and Particle Swarm Optimization. Mathematical Problems in Engineering. 2023;2023(1):6165495.

[34] Subrarnanyam V, Srinivasan D, Oniganti R. Dual layered PSO Algorithm for evolving an Artificial Neural Network controller. IEEE/CEC. 2007: 2350-2357.

[35] Dou Q, Zhou C, Pan G, Luo H, Liu Q., Neural Particle Swarm Optimization for Casing Damage Prediction. Lecture Notes in Computer Science (LNCS). 2005; 3498:903-907.

[36] Chen C, Feng H, Ye F. Hybrid Recursive Particle Swarm Optimization Learning algorithm in the Design of Radial Basis Function Networks, J. of Marine Science and Technology.2007; Vol 15, No 1: 31-40.

[37] Madar J, Abonyi J, Szeifert F. Interactive particle swarm optimization. Proc. Int. IEEE conference on Intelligent Systems Design and Applications (IEEE/ ISDA). 2005; Vol.8, No.10: pp. 314–319.

[38] Lu H, Chen W. Self-adaptive velocity particle swarm optimization for solving constrained optimization problems. Journal of Global Optimization. 2008; Vol.41, No.3: pp. 427-445.

[39] Shen X, Wei K, Wu D, Tong Y, Li Y. A Dynamic Adaptive Dissipative Particle Swarm Optimization with Mutation Operation, Proc. IEEE/ ICCA. 2007: 586-589.

[40] Del Castillo E, Montgomery DC, McCarville DR. Modified desirability functions for multiple response optimization. Journal of Quality Technology. 1996; 3. 28: 337-345.

[41] Siljic Tomic A, Antanasijevic D, Ristic M, et al. Application of experimental design for the optimization of artificial neural network-based water quality model: a case study of dissolved oxygen prediction. Environ Sci Pollut Res. 2018; 25:9360–9370.

[42] Tong LI, Wang CH, Chen HC. Optimization of multiple responses using principal component analysis and technique for order preference by similarity to ideal solution. Int J Adv Manuf Technol.2005; 27, 407–414

[43] Tong LI, Chen CC, Wang CH. Optimization of multi-response processes using the VIKOR method. International Journal of Advanced Manufacturing Technology. 2007 Feb 1;31(11-12):1049-1057. doi: 10.1007/s00170-005-0284-6

[44] Orouskhani M, Daming S, Orouskhani Y. Multi-objective evolutionary clustering with complex networks. Expert Systems with Applications. 2021;165: 113916.

[45] Lin J L, Wang K S, Yan B H, Tarng Y S. Optimization of the electrical discharge machining process based on the Taguchi method with fuzzy logics. Journal of Materials Processing Technology. 2000; Vol. 102, no. 1: pp. 48–55.