

Accepted Manuscript

Artificial neural networks used in optimization problems

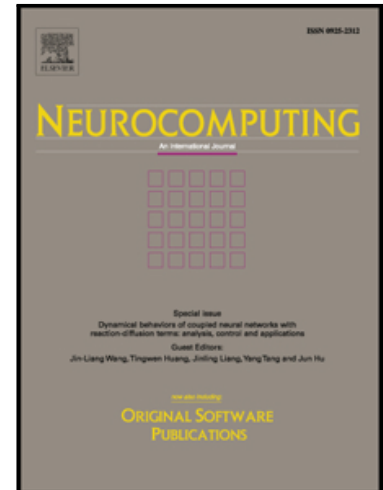
Gabriel Villarrubia , Juan F. De Paz , Pablo Chamoso ,
Fernando De la Prieta

PII: S0925-2312(17)31111-6
DOI: [10.1016/j.neucom.2017.04.075](https://doi.org/10.1016/j.neucom.2017.04.075)
Reference: NEUCOM 18612

To appear in: *Neurocomputing*

Received date: 28 October 2016
Revised date: 11 March 2017
Accepted date: 3 April 2017

Please cite this article as: Gabriel Villarrubia , Juan F. De Paz , Pablo Chamoso ,
Fernando De la Prieta , Artificial neural networks used in optimization problems , *Neurocomputing* (2017), doi: [10.1016/j.neucom.2017.04.075](https://doi.org/10.1016/j.neucom.2017.04.075)



This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

Artificial neural networks used in optimization problems

Gabriel Villarrubia, Juan F. De Paz, Pablo Chamoso, Fernando De la Prieta.

¹University of Salamanca, BISITE Research Group, Edificio I+D+I, 37007, Salamanca, Spain

{gvg, fcofds, chamoso, fer}@usal.es

Abstract: Optimization problems often require the use of optimization methods that permit the minimization or maximization of certain objective functions. Occasionally, the problems that must be optimized are not linear or polynomial; they cannot be precisely resolved, and they must be approximated. In these cases, it is necessary to apply heuristics, which are able to resolve these kinds of problems. Some algorithms linearize the restrictions and objective functions at a specific point of the space by applying derivatives and partial derivatives for some cases, while in other cases evolutionary algorithms are used to approximate the solution. This work proposes the use of artificial neural networks to approximate the objective function in optimization problems to make it possible to apply other techniques to resolve the problem. The objective function is approximated by a non-linear regression that can be used to resolve an optimization problem. The derivative of the new objective function should be polynomial so that the solution of the optimization problem can be calculated.

Keywords: neural networks, optimization problems, non-linear optimization.

1 Introduction

Optimization problems are an important part of soft computing, and have been applied to different fields such as smart grids [1], logistics [2] [3], resources [4] or sensor networks [5]. Such problems are characterized by the presence of one or more objective maximizing or minimizing

functions [5] and various restrictions that must be met so that the solution is valid. The problems are easy to resolve when we are working with linear restrictions and objective functions because there are methods to obtain the optimal solution. However, in the case of non-linear restrictions or objective functions, it may be necessary to use heuristics [2] [5] to obtain a pseudo-optimal solution. The management of heuristic solutions is continually evolving, which is precisely why we are looking for alternatives to problems in which it is not feasible to find an optimal solution. When working with linear restrictions and objective functions, optimization problems can be resolved with algorithms such as the Simplex [6], which limits the study of this type of problem. Certain non-linear problems can be optimally resolved by using algorithms such as Lagrange multipliers or Kuhn–Tucker conditions [7]. In many cases, it is not possible to resolve a problem with Lagrange multipliers because the generated system of equations cannot be resolved without resorting to numerical methods, which would prevent a direct approach to resolving the problem. In other cases, the Kuhn-Tucker conditions are not met. There is a broad range of opportunities to study optimization problems that cannot be solved with an exact algorithm. These problems are usually solved by applying a heuristics and metaheuristics solution such as genetic algorithms [8], particle swarm optimization [9], Simulated annealing [10], ant colony optimization [12] etc.

This work proposes the use of neural networks such as heuristics to resolve optimization problems in those cases where the use of linear programming or Lagrange multipliers is not feasible. To resolve these problems a multilayer perceptron is applied to approximate the objective functions; the same process could be followed in the restrictions. The proposal establishes the activation function to be used and the criteria to conduct the training using a dataset according to the

defined domain of the variables. This process makes it possible to transform objective functions into other functions, which can then be applied to resolve optimization problems that can be resolved without metaheuristics. The objective function is approximated with a non-linear regression with the objective to obtain a new function that facilitates the solution of the optimization problem. The activation function of the neural network must be selected so that the derivate of the transformed objective functions should be polynomial. Once the new objective functions has been calculated the problem can be resolved with other techniques. The same process can be applied to non-equality restrictions, but it is necessary to introduce gaps to satisfy the restrictions.

This paper is organized as follows: Section 2 revises related works, Section 3 describes the proposal, and finally Section 4 shows the results and conclusions obtained.

2 Heuristics applied to optimization

On certain occasions, optimization problems cannot be solved by applying methods such as Simplex or Lagrange. Methods such as Simplex are applicable only when problems are linear, so the algorithm cannot be properly applied when the objective function or constraints are nonlinear. Lagrange makes it possible to resolve optimization problems even when problems are not linear, but it is not always possible to resolve the equations after applying Lagrange. When exact algorithms do not allow obtaining an optimal solution, it is necessary to apply heuristics and metaheuristics algorithms. Some heuristics, such as ant colony optimization, are oriented to resolve optimization problems in graphs [13], although they can be applied in other optimization fields

such as control processes [14]. The authors in this study [14] applied fuzzy logic in a nonlinear process to improve the efficiency in the learning process with regard to execution time. Other alternatives such as Simulated annealing or PSO (Particle Swarm Optimization) are commonly applied in optimization functions. In general, several evolutionary algorithms can be applied to resolve optimization problems, as seen in various studies [9][14].

In mathematics, there are heuristics methods that work with approximation functions. Approximation functions are usually defined around a point, which would make it possible to use polynomials to approximate functions by applying the Taylor theorem. Based on this idea, it would be possible to solve non-linear optimization problems by applying Taylor nonlinear functions. This idea has been applied in algorithms such as Frank-Wolfe[15], which allows linearizing objective functions by applying derivatives in a point to calculate the straight line, plane or hyperplane crosses through that point. The solutions are calculated iteratively with a new hyperplane for each iteration. MAP (Method of Approximation Programming) is a generalization of the Frank-Wolfe algorithm, which permits linearizing the restrictions.

This work proposes carrying out this approximation in a more generic manner, making it possible to solve the problem without needing to calculate a new approximation for each tentative solution. We propose to do so by applying neural networks.

3 Proposal

Komogorov's theorem says that a multilayer perceptron with 3 layers makes it possible to precisely define any continuous function. However, for the approximation to be exact, it is neces-

sary to define an activation function and parameters for which there are no calculation procedures. It is not possible to apply just any activation function, because we must take into account the objective of the functions to simplify the problem.

The proposal to solve an optimization problem is explained in figure 1. The system generates a dataset in the domain of the variables to train a neural network. The objective function of the optimization problem is redefined with the multilayer perceptron that transforms the function, making it possible to generate a polynomial equation to resolve the optimization problem. Finally, when the new objective function is calculated another solution can be applied to resolve the problem.

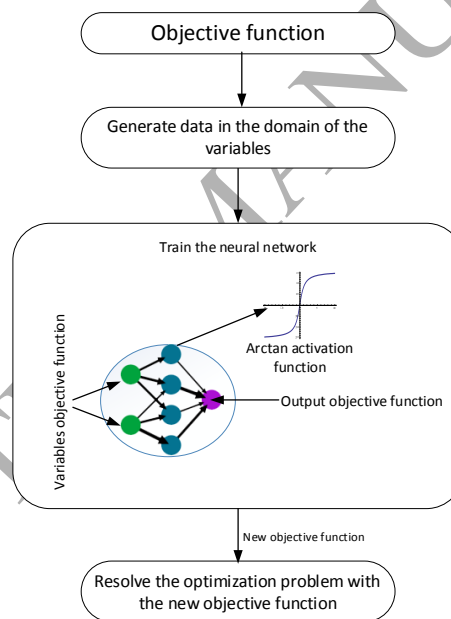


Fig. 1 Workflow optimization problem

To define a neural network, it is necessary to establish parameters, such as the connections, number of layers, activation functions, propagation rules etc. In the case of the multilayer per-

ceptron, we need to consider its two different stages: the learning stage, and the prediction process. In both stages, the number of layers and activation functions have to be the same. In the prediction stage, other parameters such as the learning rate or the momentum are not relevant.

In the case of the multilayer perceptron, the propagation rule is the weighted sum, and it is defined according to (1).

$$\sum_{i=1}^n w_{ij} x_i(t) \quad (1)$$

Where w_{ij} is the weight that connects neuron i in the input layer with neuron j in the hidden layer, x_i is the output from neuron i in the input layer, n is the number of neurons in the input layers, and t is the pattern.

In case of having bias in the neuron, the result would be what is shown in (2).

$$\sum_{i=1}^n w_{ij} x_i(t) + \theta_j \quad (2)$$

After calculating the propagation rules, we should apply the activation function. If the activation function is linear, we would have an output of neuron j that would be a linear combination of the neurons in the input layer and, consequently, y_j would be a linear function. Therefore, if the activation function is the identity, the net output would correspond to the output of the neuron.

So, if neuron k in the output layer also has the activation function f , the output would be defined as (3).

$$y_j(t) = f\left(\sum_{i=1}^n w_{ij} x_i(t) + \theta_j\right) \quad (3)$$

Bearing in mind that the multilayer perceptron has three layers, it is necessary to apply the propagation rule on two occasions in order to transmit the value in the input layer to the neurons in the output layer (4).

$$y_k(t) = \sum_{j=1}^m w_{jk} y_j(t) + \theta_k \quad (4)$$

Where k represents neuron k in the output layer, and m is the number of neurons in the hidden layer.

Replacing (4) with (3) we would have the output in neuron k defined according to the equation represented in (5).

$$y_k(t) = \sum_{j=1}^m w_{jk} f\left(\sum_{i=1}^n w_{ij} x_i(t) + \theta_j\right) + \theta_k \quad (5)$$

In the function defined in (5), if f is the identity, the output in neuron k from the output layer is calculated as a linear combination of the inputs, so the function is linear.

As a result, if we train the multilayer perceptron with an identity activation function, it would be possible to make an approximation of the trained function. Given an optimization problem in which the objective function is not linear, it would then be possible to redefine the function according to the expression (5) so that it would be linear. Although we have defined approximation functions, we were not able to prevent them from becoming hyperplane, so the activation function f cannot be linear. In this case, we have selected the arctan activation function because its derivative is simple and makes it possible to simplify functions. For example, a trigonometric or exponential objective function to polynomial objective function can be solved with Lagrange.

The arctan activation function is shown in figure 2a. We can see that it is similar to the sigmoidal function in figure 2b; as such, the training of the neuronal network should be similar with both activation functions.

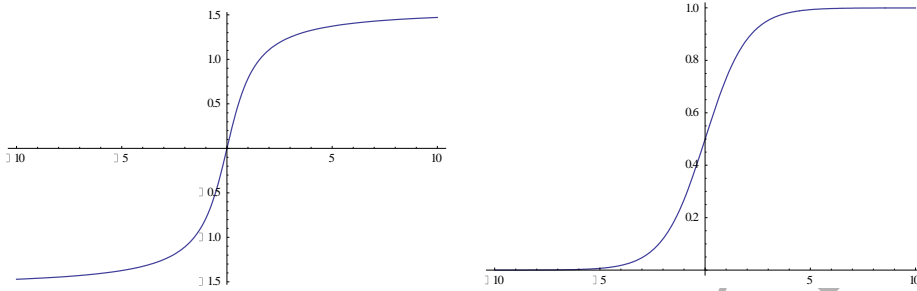


Fig. 2 a) arctan activation function, b) sigmoidal

The sigmoidal function is commonly used in neural networks; however, we used arctan because the equation of the derivative is simpler than the expression of the sigmoidal. Although the functions are similar, as we can see in figure 3a and 3b, the equation of the derivative of arctan is shown in equation (6), and the equation of the derivative of sigmoidal is shown in (7).

$$\text{ArcTan}(f) = \frac{f'}{1+f^2} \quad (6)$$

$$\text{sigmoidal}(f) = -\frac{f'e^f}{(1+e^f)^2} \quad (7)$$

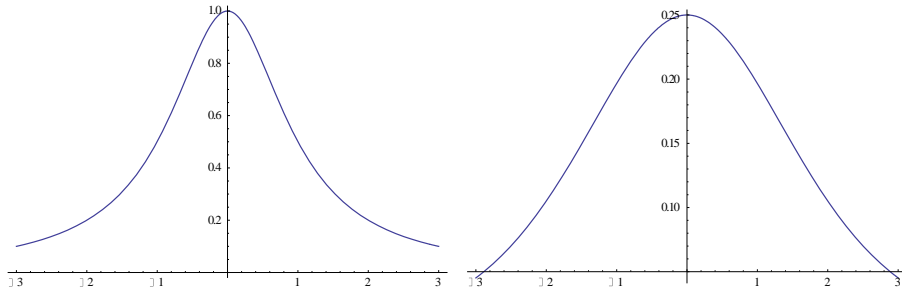


Fig. 3 Derivate of a) arctan activation function, b) sigmoidal

Then, if we have to solve an optimization problem defined (in) (8)

$$\begin{aligned}
 & f(x_1, \dots, x_n) \\
 & \textit{st} \\
 & r_1(x_1, \dots, x_n) \leq 0 \\
 & \dots \\
 & r_m(x_1, \dots, x_n) \leq 0
 \end{aligned} \tag{8}$$

Where r_i is the constraint and f the objective function.

We could approximate the objective function with the neural network defined in (5)

$$\begin{aligned}
 f(x_1, \dots, x_n) &= \sum_{j=1}^m w_{jk} f\left(\sum_{i=1}^n w_{ij} x_i(t) + \theta_j\right) + \theta_k \\
 & \textit{st} \\
 & r_1(x_1, \dots, x_n) \leq 0 \\
 & \dots \\
 & r_m(x_1, \dots, x_n) \leq 0
 \end{aligned} \tag{9}$$

Therefore, the following optimization problem could be solved with Kuhn-Tucker, applying the equation

$$\begin{aligned}
\frac{\partial f(x_1, \dots, x_n)}{\partial x_j} \pm \sum_{i=1}^m \lambda_i \frac{\partial r_i(x_1, \dots, x_n)}{\partial x_j} &\leq 0 & j = 1..m \\
\lambda_1 * r_1(x_1, \dots, x_n) &= 0 \\
\lambda_2 * r_2(x_1, \dots, x_n) &= 0 \\
&\dots \\
\lambda_m * r_m(x_1, \dots, x_n) &= 0
\end{aligned} \tag{10}$$

With Lagrange the idea would be similar, given the optimization problem defined by (11)

$$\begin{aligned}
&f(x_1, \dots, x_n) \\
&st \\
&r_1(x_1, \dots, x_n) = 0 \\
&\dots \\
&r_m(x_1, \dots, x_n) = 0
\end{aligned} \tag{11}$$

The optimization problem would be solved according to

$$\begin{aligned}
L(x_1, \dots, x_n, \lambda_1, \dots, \lambda_m) &= \sum_{j=1}^m w_{jk} f\left(\sum_{i=1}^n w_{ij} x_i(t) + \theta_j\right) + \theta_k + \sum_{i=1}^m \lambda_i \cdot r_i \\
\frac{\partial L}{\partial x_i} &= 0 \quad \forall i \in \{1, \dots, n\} \\
\frac{\partial L}{\partial \lambda_j} &= 0 \quad \forall j \in \{1, \dots, m\}
\end{aligned} \tag{12}$$

According to expression (10) and (12), we can see that it is very important to use an activation function to simplify its derivative and calculate the solution more easily.

Likewise, we could continue with the constraints; however, when working with equal constraints, it would be more complicated to apply this heuristic. For restrictions with inequality, it is possible to introduce a threshold based on the training carried out in the neural network. For the threshold to be lower, it would be best to have previously trained the neural network with

values in the variables around their definition. In other words, in a problem with restrictions similar to (13) we should generate a dataset for the training phase that matches the restrictions for the variable x_i . The lower the difference among consecutives values, the lower the error to define the threshold.

$$\begin{aligned} a_i &\leq x_i \leq b_i \\ x_i &\geq a_i \\ x_i &\leq b_i \end{aligned} \quad (13)$$

Therefore the restrictions defined according to (14) through the inclusion of threshold and based on the expression (5), will be defined as (15)

$$r_s(x_1, \dots, x_n) \leq 0 \quad (14)$$

$$\sum_{j=1}^m w_{jk} f\left(\sum_{i=1}^n w_{ij} x_i(t) + \theta_j\right) + \theta_k + \mu_s = 0 \quad (15)$$

Finally, the optimization problem defined as (8) would be defined according the expression (16)

$$\begin{aligned} f(x_1, \dots, x_n) &= \sum_{j=1}^{m^1} w_{jk}^1 f\left(\sum_{i=1}^{n^1} w_{ij}^1 x_i^1(t) + \theta_j^1\right) + \theta_k^1 \\ &st \\ \sum_{j=1}^{m^1} w_{jk}^1 f\left(\sum_{i=1}^{n^1} w_{ij}^1 x_i^1(t) + \theta_j^1\right) + \theta_k^1 + \mu_s^1 &= 0 \\ &\dots \\ \sum_{j=1}^{m^n} w_{jk}^n f\left(\sum_{i=1}^{n^n} w_{ij}^n x_i^n(t) + \theta_j^n\right) + \theta_k^n + \mu_s^n &= 0 \end{aligned} \quad (16)$$

As an alternative to the use of neural networks, it would also be possible to use other techniques to approximate functions, such as applying Support Vector Regression (SVR). SVR can approximate functions as a linear combination in a space with higher dimensions than the original.

5. Results and conclusions

In order to analyze the performance of the proposal, we analyzed different optimization problems and compared the predicted and optimal values in the system. The tests were made with a neural networks tool developed by our research group and the Mathematica program. Mathematica was used to solve the equations after defining the approximation with a multilayer perceptron. The dataset used to train the neural network is generated according to the domain of the variables. It contains the input variables in the objective function and the output in the objective function obtained for these values. The domain of the variables is defined in the constraints of the optimization problem.

The first test was to analyze the performance of the system with a simple optimization problem. It was a linear function that was approximated with a multilayer perceptron, which activates functions in the hidden and output linear layers.

$$\begin{aligned}
 & \text{Max } 15x + 10y \\
 & \text{st} \\
 & \frac{1}{3000}x + \frac{1}{6000}y \leq \frac{2}{25} \\
 & \frac{1}{3000}x + \frac{1}{2000}y \leq \frac{1}{100} \\
 & x \geq 0, y \geq 0
 \end{aligned} \tag{17}$$

The result is $x=30$, $y=0$ and the value of the objective function is 450 when we replace the objective function $15x+10y$ by the approximated function with the neural network shown in figure 4a. The result of the objective function is 444.406 and the value of $x=30$ and $y=0$. This result was obtained in a quick training of the neural network; therefore, it could be easily improved. If we applied PSO to resolve the optimization problem the results for the variable $x=29,98$ and $y=0$. The objective function was 449.84, a better result than that provided by the proposal; however, as we said, it was only to test the system with a fast training of the neural network.

The second test was to analyze the prediction with two variables and with non-linear objective functions.

$$\begin{aligned}
 & \text{Max } \frac{3x_1}{50} + \frac{3x_2}{100} - \frac{x_1}{25} \frac{x_2}{100} - \frac{2x_1^2}{100} + \frac{3x_2^2}{100} \\
 & \text{st} \\
 & \quad x_1 + x_2 \leq 1 \\
 & \quad 2x_1 + 3x_2 \leq 4 \\
 & \quad 3x_1 + 2x_2 \geq 2 \\
 & \quad x_1 \geq 0 \\
 & \quad x_1 \leq 0.8 \\
 & \quad x_2 \geq 0 \\
 & \quad x_2 \leq 0.3
 \end{aligned} \tag{18}$$

The optimal value of the objective function with Nelder Mead was 0.039936, and the values of the variables are $x_1=0.8$, $x_2=0.200009$, the constraint 1 is not valid with this solutions. The optimal value of the objective function with Differential Evolution was 0.0015949, and the values of the variables are $x_1=0.64$, $x_2=0.200008$; the constraint 1 is not valid with this solutions. The solu-

tion obtained with the neural network shown in figure 4b is 0.039936, $x_1=0.8$, $x_2=0.2$. The neural network had two input layers, 11 neuros in the hidden layer an arctan activation function, and one output with the value of the objective function. The learning rate was defined as 0.01, and momentum as 0.001. The neural network was trained manually and the training was stopped when the error remained constant. The result obtained with PSO was 0.035, $x_1= 0.46$ and $x_2=0.3$. In this case, the better result was obtained by the proposal.

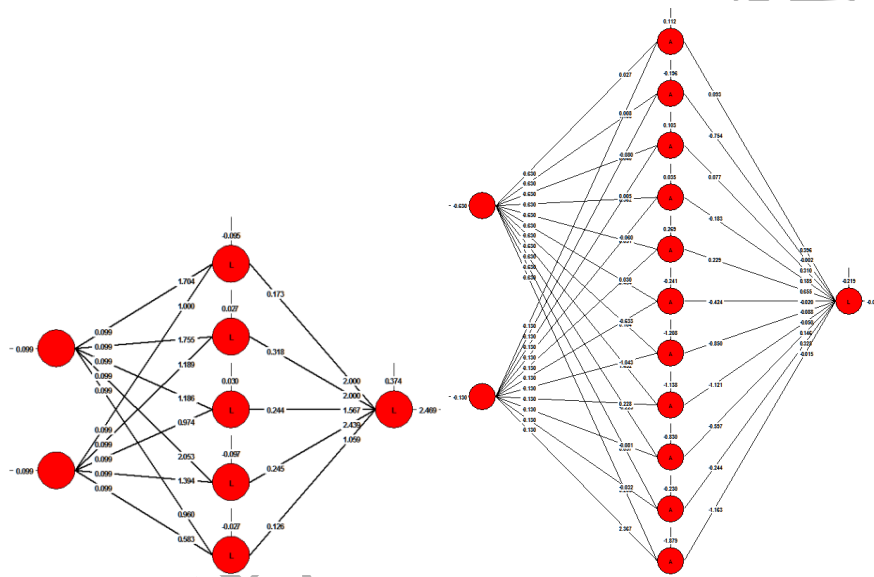


Fig. 4 Neural network a) linear approximation b) non-linear approximation

In this example, we selected a trigonometric objective function, similar to that in (19). The result with Nelder Mead was 0.998673, $x_1=0.249963$, $x_2=-0.250028$, the solution did not match with the first constraints. The result with Differential Evolution was 0.998673, $x_1=0.249963$, $x_2=-0.250028$, the solution did not match with the first constraints. We have obtained the RNA the result was 0.998569, $x_1= 0.255127$, $x_2=-0.244873$. We can see the original function in figure 5a

and the approximation in figure 5b. It is evident that the approximation is quite good. The neural network had two input layers, 17 neurons in the hidden layer, an arctan activation function, and one output with the value of the objective function. The learning rate was defined as 0.01, and momentum as 0.001. The neural network was trained manually and the training was stopped when the error remained constant. The result obtained with PSO was 1.00004 and the variables were $x_1=0.100$, $x_2=-0.100$. In this case the better result was obtained by the PSO algorithm.

$$\begin{aligned}
 & \text{Max } \cos(x_1 x_2) - x_1 x_2 / 100 - \sin(x_1 + x_2)(x_1 + x_2) \\
 & \text{st} \\
 & x_1 - x_2 \geq 0.5 \\
 & x_1 \cdot x_2 \leq 15 \\
 & x_1 \geq 0 \\
 & x_1 \leq 1.5 \\
 & x_2 \geq -1 \\
 & x_2 \leq 1
 \end{aligned} \tag{19}$$

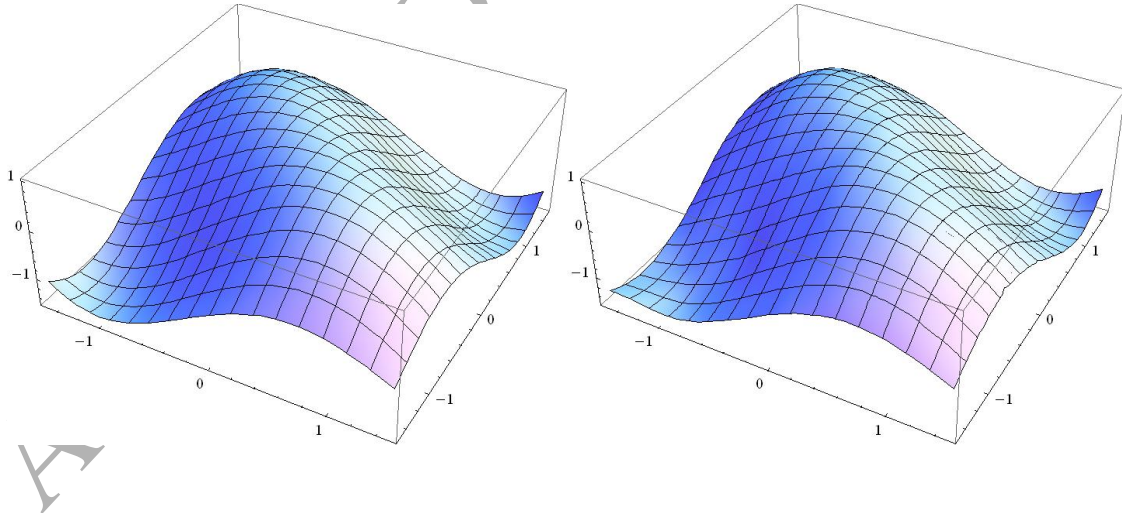


Fig. 5 a) Original function b) approximated function.

As we can see in the results, the system is able to approximate objective functions with a multi-layer perceptron and use these approximations to solve optimization problems. In some cases, metaheuristics are not able to provide a solution that matches the restriction, although the solution calculated with the new objective function did match. The main disadvantage of the proposal is that it is necessary to train the neural network and it is necessary to use Lagrange or Kuhn-Tucker with the neural network; this makes it impossible to use any activation function, such as a sigmoidal. The proposal uses an arctan activation function because its derivative is polynomial, which makes it possible to solve the generated equation system with Lagrange or Kuhn-Tucker. The main problem of the proposal is that when we approximate equality constraints we will have to deal with small errors and the solution will not be valid; in other kinds of restrictions we can introduce a threshold in order to obtain valid solutions.

Acknowledgements: This work has been supported by project Máquina social para la gestión sostenible de ciudades inteligentes: movilidad urbana, datos abiertos, sensores móviles SA70U16. Project co-financed with Junta Castilla y León funds.




References


- [1] Ríos-Mercado, R.Z., Conrado Borraz-Sánchez, Optimization problems in natural gas transportation systems: A state-of-the-art review, *Applied Energy*, 147, 2015, 536–555.
- [2] Wang, Y., Ma, X., Xu, M., Liu, Yong ., Wang, Y., Two-echelon logistics distribution region partitioning problem based on a hybrid particle swarm optimization–genetic algorithm, 42(12), 2015, 5019–5031.

- [3] Salari, M., Reihaneh, M., Sabbagh, M.S, Combining ant colony optimization algorithm and dynamic programming technique for solving the covering salesman problem, *Computers & Industrial Engineering*, 83, 2015, 244–251.
- [4] Zheng, X.L., Wang, L., A multi-agent optimization algorithm for resource constrained project scheduling problem, *Expert Systems with Applications*, In Press.
- [5] Lanza-Gutierrez, J.M., Gomez-Pulido, J.A., Assuming multiobjective metaheuristics to solve a three-objective optimisation problem for Relay Node deployment in Wireless Sensor Networks, *Applied Soft Computing*, 30, 2015, 675–687.
- [6] Ploskas, N., Samaras, N., Efficient GPU-based implementations of simplex type algorithms, *Applied Mathematics and Computation*, 250, 2015, 552–570,
- [7] Kuhn, H. W.; Tucker, A. W. *Nonlinear Programming*. Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 481--492, University of California Press, Berkeley, Calif., 1951.
- [8] Wang, Y., The hybrid genetic algorithm with two local optimization strategies for traveling salesman problem, *Computers & Industrial Engineering*, 70, 2014, 124–133.
- [9] Li, L., Yu, Z., Chen, Y., Evacuation dynamic and exit optimization of a supermarket based on particle swarm optimization, *Physica A: Statistical Mechanics and its Applications*, 416, , 157–172.
- [10] Kolonko, M., Some new results on simulated annealing applied to the job shop scheduling problem , *European Journal of Operational Research*, 113(1), 2009, 123–136
- [11] Hannah, L.A., *Stochastic Optimization*, International Encyclopedia of the Social & Behavioral Sciences (Second Edition), 2015, 473–481.
- [12] İnkaya, T., Kayaligil, S., Özdemirel, N.E., Ant Colony Optimization based clustering methodology, *Applied Soft Computing*, 28, 2015, 301–311,
- [13] Yang, J., Zhuang, Y., An improved ant colony optimization algorithm for solving a complex combinatorial optimization problem, *Appl. Soft Comput.*, 10 (2) (2010), 653–660.
- [14] Bououden, S., Chadli, M., Karimi, H.R., An ant colony optimization-based fuzzy predictive control approach for nonlinear processes, *Information Sciences*, 299, 2015, 143–158.

- [15] Ñanculef, R., Frandi, E., Sartori, C., Allendea, H., A novel Frank–Wolfe algorithm. Analysis and applications to large-scale SVM training, *Information Sciences*, 285, 2014, 66–99.

ACCEPTED MANUSCRIPT

	<p>Gabriel Villarrubia González. PhD Student on Computer Engineer. Currently, he is Assistant Professor at the Department of Computer Science at the University of Salamanca and member of the BISITE Research group.</p> <p>He obtained a Technical Engineering in Management of Computer in 2009 at the Pontifical University of Salamanca, then an Engineering in Computer Sciences degree in 2010 at the same university and postgraduate in Intelligent Systems at the University of Salamanca in 2011.</p> <p>He is co-author of more than 50 papers published in recognized journal, workshops and symposiums, most of them related to Intelligent Systems, Ambient Intelligence, and Distributed Systems. He has been member of the organizing and scientific committee of several international symposiums such as FUSION, PAAMS, MIS4TEL, ISAMI, PACBB.</p>
	<p>Juan Francisco De Paz (PhD.). Received a PhD in Computer Science from the University of Salamanca (Spain) in 2010. He is Assistant Professor at the University of Salamanca and researcher at the BISITE research group (http://bisite.usal.es). He obtained a Technical Engineering in Systems Computer Sciences degree in 2003, an Engineering in Computer Sciences degree in 2005 at the University of Salamanca and Statistic degree in 2007 in the same University. He has been co-author of published papers in several journals, workshops and symposiums.</p>
	<p>Pablo Chamoso Santos. PhD Student on Computer Engineer. He is member of the BISITE Research group.</p> <p>He obtained a Technical Engineering in Computing in 2010 at the University of Salamanca, then an Engineering in Computer Sciences degree in 2013 at the same university. Then, he obtained a postgraduate in Intelligent Systems at the University of Salamanca in 2014.</p> <p>He is co-author of more than 20 papers published in recognized journal, workshops and symposiums, most of them related to</p>

	<p>Artificial Intelligence, Visual Analysis and Distributed Systems. He has been member of the organizing and scientific committee of several international symposiums such as PAAMS, FIADI, SAC, ISAMI, PACBB.</p>
	<p>Fernando de la Prieta. PhD on Computer Engineer from the University of Salamanca. Currently, he is Assistant Professor at the Department of Computer Science of the same University and member of the BISITE Research group. He has been member of the organizing and scientific committee of several international symposiums such as ACM-SAC, FUSION, PAAMS, MIS4TEL, etc. and co-author of more than 60 papers published in recognized journal, workshops and symposiums, most of them related to Intelligent Systems, Ambient Intelligence, Distributed Systems, Technology Enhanced Learning and Information Security.</p>