12

Application of Continuous ACO_R to Neural Network Training: Direction of Arrival Problem

Hamed Movahedipour
Department of Electrical Engineering, Tarbiat Modares University
Iran

1. Introduction

In this chapter, a hybrid ACO_R-based artificial neural network is investigated and applied to solve a Direction of Arrival (DoA) estimation problem. This approach is compared with Radial Basis Function Neural Network (RBFNN) that has been used broadly in the literature for DoA estimation.

The Ant Colony Optimization is a stochastic optimization technique that has attracted much attention towards numerous optimization problems during the past decade. ACO is a subset of swarm intelligence methods in which the collective intelligence emerges in decentralized and self-organized systems with simple individuals.

Social insects are distributed systems that carry out complex tasks, having individuals with very simple and rudimentary cognitive abilities. In many cases, these tasks exceed the capabilities of a single individual. In fact, social insects are self-organized systems and some simple principles and processes such as stigmergy can explain their social behaviour. Stigmergy is an indirect communication among individuals, in which different entities communicate by modifying the environment.

Ants possess very limited visual and vocal perceptive abilities and some types are totally blind. Hence, the only efficient communication channel in these species is various types of chemicals, which are called pheromones. One specific type of pheromone is the trail pheromone that is deposited for instance while searching for food and the other ants smell the pheromone and tend to follow the paths with high pheromone concentration. Therefore, by indirect communication via pheromone and the simple rule of following the higher density of pheromone, one complex colony-level behaviour is emerged which is finding the short paths to the food. This behaviour is quite above the capabilities of each ant. In fact this collective capability emerges out of microscopic simple processes of pheromone laying and pheromone following.

Ant colony optimization is an algorithm, which models foraging behaviour of ants to solve optimization problems and it has inspired many researchers to provide solutions to various combinatorial optimization problems such as travelling salesman problem (Dorigo et al., 1996), routing problem (Schoonderwoerd et al., 1997) and many other NP-hard problems in which the values for discrete variables are found to optimize an objective function. In fact ACO, models ant agents walking on a graph that implies typical discrete problems or structures. Since ACO was originally proposed for discrete optimization problems, its application to continuous domain was not straightforward. Among various adaptations of

ACO algorithm for continuous optimization problems, the approach of Socha tries to avoid conceptual changes to principles of ACO by introducing ACO_R algorithm (Socha, 2004; Socha & Blum, 2007; Socha & Dorigo, 2008). Socha's approach to adapt ACO to continuous domain is utilized in this work to optimize the numerical weights of a Multi-Layer Perceptron (MLP) network for interpolation of the nonlinear relation of antenna array outputs and Angle of Arrival (AoA). In other words, ACO_R is used to minimize Mean Square Error (MSE) of the output layer of the neural network by adjusting continuous variables of weights during training phase.

Direction of arrival estimation has turned out to be a substantial part of many applications like channel characterization, car tracking (Danneville et al., 2005), source localization in radar and sonar (Godara, 2002), receiver algorithm design and co-channel interference reduction (Christodoulou, 2001). Spectral-based algorithmic solutions like Multiple Signal Classification (MUSIC) and parametric methods such as Maximum Likelihood (ML) have been the major methods to tackle DoA problem for a long period. The foremost drawback of these techniques is that they are computationally expensive and do not perform quite efficiently in real time operation. Artificial neural networks have been also utilized to estimate DoA, often using RBF networks due to its acceptable estimation error, capability of inexpensive implementation and fast performance (Zooghby et al., 2000; Titus et al., 1994).

In this chapter, after a general introduction to ant colony optimization, ACO_R adaptation to continuous domain is explained. ACO_R has been applied to train a classifier neural network (Socha & Blum, 2007) and in this work, its application to an interpolator neural network is investigated. The possibilities to enhance the performance of ACO_R are studied as well.

2. Ant Colony Optimization

2.1 Combinatorial Optimization

Combinatorial optimization (CO) problem $P = (S,f)$ is an optimization problem in which $f:S\rightarrow R^+$ is an objective function that assigns a positive value, called cost to each solution $s \in S$ in the finite search space S which encompasses feasible solutions. The goal of this kind of problem is to find a solution in the search space with the lowest cost (Papadimitriou & Steiglitz, 1982). The search space in combinatorial problems includes variables X_i, $i=1, \ldots, n$ in discrete domains, therefore combinatorial problems are in fact discrete optimization problems. CO problems are arisen in industry and in applied sciences such as statistics, physics and chemistry. Some instances in industry are manufacturing and distribution, telecommunication network design and routing, airline crew scheduling, etc. Meanwhile this field is connected to various areas of mathematics such as algebra, analysis and continuous optimization, geometry, numerical analysis, topology, graph theory and enumerative combinations (Lee, 2004). Therefore, due to the wide range of important applications of CO problems, various algorithms and methods have been developed to attack them.

2.1 ACO algorithm

ACO is an algorithm which finds an approximate optimal solution in a reduced amount of time for NP-hard problems and it was introduced by Dorigo and colleagues (Dorrigo et al., 1991 and 1996) in the early 1990's. ACO simulates the foraging behavior of ants, which communicate using pheromone trails and manage to find the shortest path from their nest to feeding source. The significant features of this model are positive feedback, distributed computation and incremental construction of solutions.

Double bridge experiment was a practice designed by Deneubourg and colleagues (Deneubourg et al., 1990) in which the ability of real ant colonies in finding shorter paths to food was investigated. In this experiment a bridge with two branches l_i *(i*=1, 2*)* was used to connect a nest of real ants to a food source. In the first phase of the experiment equal branches were used and the behavior of ants regarding choosing branches was studied. The observations showed that at the beginning, ants had random choices but after a while they were converged to one of the branches. In fact at the start of the experiment, there was no pheromone on the bridge so the ants chose different branches randomly, but after awhile due to random fluctuations more pheromone was accumulated in one of the branches and after a short time all other ants converged to that branch. This positive feedback process is one of the main characteristics of the foraging ant behavior and consequently ACO algorithm. In the second phase of the experiment, branches with different lengths were used and it was observed that in most of the trials ants were converged to the shorter branch. In fact again at the start, ants chose branches randomly but the ants that opted the short branch reached the food sooner and also started their return to the nest sooner than the ants which selected the long branch. In this way, more pheromone was accumulated in the shorter path and consequently biased decision of the other ants in its favor. It is noteworthy that in the second phase of the experiment, the effect of the random fluctuations was overcame by the difference in the path length and the ants converged to the shorter path.

This experiment showed optimization capability of ant colonies and inspired researchers to design artificial ants and exploit the same principles utilized by ants to solve combinatorial optimization problems. Double bridge is modeled by a graph, passing the nodes constructs the solution and finally the length of the chosen path can be considered as the cost. By considering a static, connected graph and the described mechanism to solve a combinatorial optimization problem, the ants may generate loops while solution construction and this will prevent them from finding the short paths between source and destination. By extending the capabilities of artificial ants like a limited form of memory, getting trapped in the loops can be avoided. Besides if each arc in the graph has a cost, artificial ants can not deposit pheromone before reaching to the destination because first they need to construct the solution and then evaluate the solution according to the objective function.

In one of the double bridge experiments, ants were faced with one long branch for 30 minutes and then one short bridge was added. In this experiment ants were trapped in the suboptimal path, because once they were converged to the long branch and it was difficult for them to reinforce the pheromone concentration in the shorter path. In fact the evaporation rate of pheromone was too slow and ants couldn't forget the converged solution, or in other words the lifetime of pheromone was comparable to the duration of trial. Therefore enhanced evaporation mechanism is utilized in artificial ants to avoid being trapped in suboptimal solutions. In fact evaporation can favor exploration of more possible solutions and paths. Therefore to tackle the inefficiencies of real ant colonies, the capabilities of artificial ants are extended in a way that the main principles of real ants are not violated. The major differences of real ants and the algorithm of ACO are as follows:

- Ants deposit pheromone while they move but artificial ants just leave pheromone on their path back to the nest which is called backward path pheromone update.
- Real ants do not evaluate the quality of their solution explicitly. It means naturally shorter paths are traversed earlier and consequently pheromone is accumulated more quickly in the shorter paths. In this way without any need of explicit calculation, the

quality of solution is evaluated. In contrast, the artificial ants calculate the objective function and perform pheromone update with respect to the quality of constructed solution.
- Artificial ants hold a memory and store the partial paths they have traversed as well as the costs of the paths. Therefore they can avoid loops and also evaluate the solution according to the objective function.
- After pheromone update, a portion of the accumulated pheromone is evaporated to enhance exploration.

Considering above extensions, ant colony optimization algorithms encompasses following basic steps:
- Probabilistic solution construction in forward mode is biased by pheromone concentration, without pheromone updating.
- Deterministic backward path pheromone update is done with loop elimination.
- Evaluation of the constructed solution and pheromone update is carried out based on the quality of solution. Note that this behavior is also visited in some types of real ants that deposit higher densities of pheromone when they return from a rich food source.
- Evaporation mechanism is used to enable the artificial ants to forget the suboptimal solutions that they have found at the early stages of the search.

4. Continuous ACO

Ant colony optimization was initially developed for combinatorial optimization problems and due to the concept and structure of the algorithm, it was difficult to apply it to continuous optimization problems.

Some methods have been developed to apply ACO to continuous optimization problems but some basic principles of ACO have been deviated in these methods. Generic principles behind ACO can be categorized in following key characteristics:
- Data source is distributed.
- Incremental construction of solutions is performed by a population of individuals.
- No direct communication is present among ants and stigmergy is utilized as the indirect communication.

ACO is a constructive algorithm in which solutions are constructed incrementally. It means ants add solution components in each step until a complete solution is developed. Contrary to various proposed methods to adapt ACO to tackle continuous optimization problems, Socha (Socha, 2004) proposed an extension of ACO to continuous domain called ACO_R without any change in basic characteristics of original ACO method. ACO_R algorithm is described in this section.

In ACO algorithm for combinatorial problems, probabilistic decision is made among different choices for certain solution component and the probability is calculated according to the amount of pheromone in available choices. In combinatorial optimization problems, available choices are discrete; therefore, each probabilistic decision is made according to a discrete probability distribution. In case of continuous ACO, solution components may assume any real value in a defined range. Therefore, in ACO_R it is proposed to use Probability Density Function (PDF) instead of discrete probability distribution and PDF is sampled by ants to model pheromone effect in probabilistic selection of solution components. In regular ACO, while pheromone update, ants add a certain amount of

pheromone to each solution component with respect to the quality of solution and in this way, each possible value of solution components gets a pheromone value which reflects the search experience of the ants. In case of combinatorial optimization problems, this is possible since the set of possible values for solution components is finite. In case of continuous optimization problem, the set of possible values is not finite and to store the search experience of the algorithm, a limited number of recent found solutions are stored in an explicit memory called solution archive (Socha & Dorigo, 2008).

$$Solution\ Archive = \begin{bmatrix} s_1^1 & s_1^2 & \dots & s_1^n \\ s_2^1 & s_2^2 & \dots & s_2^n \\ . & . & & . \\ . & . & & . \\ s_k^1 & s_k^2 & \dots & s_k^n \end{bmatrix}_{k\times n} \tag{1}$$

where n is the number of variables or solution components, k is the number of solutions that are stored in the archive and s^i_j is the i-th solution component of j-th solution. It is noteworthy that in ACO for combinatorial optimization problems, the whole history of search is stored but in ACO_R the limited number of solutions k represent the history of search. For instance if k=30 it means in each stage of the algorithm the solutions of 30 ants are utilized to make probabilistic solutions. Each row in the solution archive corresponds to a found solution, and by calculation of the objective function, the quality of each solution is measured. In fact, guidance towards better solutions and promising areas of search space in ACO_R is accomplished by formation of w_j, which is associated with solution j and is attained with respect to the value of objective function.

$$w = \begin{bmatrix} w_1 \\ w_2 \\ . \\ . \\ w_k \end{bmatrix}_{k\times 1} \tag{2}$$

4.1 Pheromone initialization

Initialization of solution archive is performed in a way that a uniform distribution over the search domain *(a,b)* is generated. This is accomplished by defining k normal distributions with uniformly distributed means.

$$P^i(x^i) = \sum_{j=1}^{k} \frac{1}{k} . g\left(x^i, a + (2j-1)\frac{b-a}{2k}, \frac{b-a}{2k} \right) \tag{3}$$

4.2 Solution construction

In solution construction phase of ACO_R, each ant chooses probabilistically one of the solutions in the archive according to its corresponding weight.

$$p_j = \frac{w_j}{\sum_{i=1}^{k} w_i} \tag{4}$$

After the *j*-th solution is selected, an ant starts constructing the solution incrementally and assigning value to each solution component $i = 1, \ldots, n$ separately. Starting with the first variable, the ant generates a random number by sampling a Gaussian function centered by the value of first variable of the chosen solution s^1_j and assigns the sampled value to the first variable of the solution under construction. Same process is performed for each solution component until the whole solution is constructed. In the above process the Gaussian function is utilized to sample the neighborhood of s^i_j (Socha, 2004).

$$P(x) = g(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{5}$$

where μ is the mean of the normal distribution and equals the value of s^i_j for *i*-th variable of *j*-th solution. In this way, the value of s^i_j has the highest probability of being chosen while sampling. σ is the standard deviation of normal distribution.

After each solution construction, pheromone update is performed to modify the probability distributions and to provide guidance for ants to discover better solutions in the next iterations. Pheromone update is carried out in two forms of positive update and negative update.

4.3 Positive update

In positive update, the solutions with good quality are stored in the solution archive to bias the probabilistic decision of ants in favor of promising areas of the search space.

After solution construction phase, the performance of solutions is measured using objective function and the weight of each solution is determined. Then a number of high performing solutions are added to the solution archive:

$$k \leftarrow k + nos \tag{6}$$

nos is the number of solutions added to solution archive. After positive update, the size of stored solutions is increased to *k*+*nos*.

4.4 Negative update

In ACO algorithms, negative update is performed to forget the old solutions, avoid being stuck in local optimum solutions and in general enhance exploration. Pheromone evaporation is the traditional method to perform negative update in ACO for combinatorial optimization problems. In ACO_R, there are different ways to accomplish negative update (Socha, 2004).

- One of the methods for negative update in ACO_R is removing some old or low performing solutions from solution archive. To keep the size of solution archive constant, same number of solutions that were added in positive update phase can be removed.

$$k \leftarrow k - nos \tag{7}$$

- Another method for negative update is similar to pheromone evaporation in generic ACO. In case of ACO_R the vector of weight is evaporated:

$$w^i{}_j \leftarrow (1-\rho).w^i{}_j \tag{8}$$

- Last method utilizes normal distribution property to implement negative update. In this method after each iteration, standard deviation of normal distributions are increased:

$$\sigma_j^i \leftarrow \gamma.\sigma_j^i \tag{9}$$

where γ is the rate of dissolving. By increasing the standard deviation of normal distributions, the random numbers generated based on normal PDFs will be more spread and it implies the decrease of impact of old solutions. This type of negative update is called dissolving.
Depending to various applications, different methods for negative update may be used. In our application, all methods of negative update are combined.

5. Direction of arrival estimation

Rapid development of high-speed data services and increasing demand for coverage and capacity in new wireless systems has caused a substantial need for more effective solutions. For instance in cellular communications, frequency reuse is utilized to fulfill capacity requirements. In these networks, increasing the capacity is possible by allowing closer co-frequency cells that on the other hand will increase the risk of interference. Therefore, interference cancelation gives us the freedom of using additional frequency reuse. The most effective way of interference reduction is to make use of direction of arrival estimation to locate the users and then exploit an adaptive array antenna to steer its main lobe toward the subscribers of interest and nulls toward the co-frequency cells. On the other hand, as mobile satellite communication systems and global positioning systems (GPS) grow, smart antennas can enhance the performance of those systems. Therefore, direction estimation is a major concern of any spatial division multiple access (SDMA) system. Besides, due to various applications of DoA in car tracking, channel characterization and source localization, it has received substantial attention for several decades. Neural network is one of the methods that has been used to model DoA estimators and it has shown an efficient performance. Neural networks approach for handling computational task of DoA estimation has following advantages:

- Fast execution, which facilitates real-time operation
- Acceptable estimation error
- Possibility of inexpensive manufacturing

Radial basis function neural network has been presented as a successful candidate among various neural structures (Christodoulou, 2001; zooghby et al., 2000). In this chapter ACO_R method is used to train a feed forward neural network and it is compared to RBFNN method. The improvements in estimation error are discussed.

5.1 DoA problem formulation

In this section, the DoA is formulated and the training set of neural network that is applied to the input layer is attained.

For DoA estimation, the phase shift between the received signals in different array elements can be used to find the direction of signals.

$$e^{-jw_c t_d} = e^{-j\frac{2\pi d \sin(\theta)}{\lambda}} = e^{-jk} \tag{10}$$

where d is the element spacing of the antenna array and λ is the wavelength of incoming waves. Consider a linear array antenna with M elements exposed to K ($K<M$) narrowband plane waves impinging on the antenna from directions $\{\theta_1\ \theta_2\ ...\ \theta_K\}$. Using wave propagation relations and complex signal notation, the received signal of the i-th array element can be shown as

$$\begin{aligned} x_i(t) &= \sum_{m=1}^{K} s_m(t) e^{-j(i-1)k_m} + n_i(t) \\ i &= 1,2,...M \end{aligned} \tag{11}$$

$s_m(t)$ is the signal of m-th wave, $n_i(t)$ is the noise signal which i-th element receives and

$$k_m = \frac{w_0 d}{c} \sin(\theta_m) \tag{12}$$

where w_0 is the center frequency and c is the speed of light. We can convert the summation to vector notation and then express the output of the array in matrix form (Christodoulou, 2001).

$$X(t) = AS(t) + N(t) \tag{13}$$

where $X(t)$, $S(t)$, A and $N(t)$ are defined as follows

$$X(t) = [x_1(t)\ \ x_2(t)\ \ ...\ \ x_M(t)]^T \tag{14}$$

$$N(t) = [n_1(t)\ \ n_2(t)\ \ ...\ \ n_M(t)]^T \tag{15}$$

$$S(t) = [s_1(t)\ \ s_2(t)\ \ ...\ \ s_K(t)]^T \tag{16}$$

$$A = [a(\theta_1)\ \ ...\ \ a(\theta_m)\ \ ...\ \ a(\theta_K)] \tag{17}$$

$$a(\theta_m) = [1\ \ e^{-jk_m}\ \ e^{-j2k_m}\ \ ...\ \ e^{-j(M-1)k_m}]^T \tag{18}$$

Assuming noise signals as statistically independent white noise with zero mean and independent of $S(t)$, we can form the spatial correlation matrix R which is used to provide the input data for neural network (Christodoulou, 2001).

$$R = E\{X(T)X^H(t)\} = AE\{S(t)S^H(t)\}A^H + E\{N(t)N^H(t)\} \tag{19}$$

where H represents the conjugate transpose. The antenna array receives signals in different directions and forms the matrix $X(t)$ which is the output of array elements. According to the

fact that receiver noise is the dominating noise source, *n(t)* is assumed to be white and Gaussian distributed:

$$E\{n(t)\}=0,\ E\{n(t)n^H(s)\}=\sigma^2 I\delta_{ts},\ E\{n(t)n^T(s)\}=0 \tag{20}$$

The signal is also assumed to be Gaussian distributed.

$$E\{s(t)\}=0,\ E\{s(t)s^H(s)\}=P\delta_{ts},\ E\{s(t)s^T(s)\}=0 \tag{21}$$

$E\{S(t)S^H(t)\}$ can be considered as a diagonal matrix P for uncorrelated signals. Note that A has $M\times K$ dimension and P is a $K\times K$ matrix, consequently R is an $M\times M$ matrix. Hence, the spatial correlation matrix is presented as:

$$\begin{aligned} R &= E\{X(T)X^H(t)\} \\ &= AE\{S(t)S^H(t)\}A^H + E\{N(t)N^H(t)\} \\ &= APA^H + \sigma^2 I \end{aligned} \tag{22}$$

By exploiting the symmetry in R, either the upper or the lower triangular of the matrix is required (Christodoulou, 2001). As a result, the input data of neural network is decreased. Notice that R contains complex values and regular neural networks do not support complex values. Hence, the real and imaginary parts in the correlation matrix should be separated and all presented in a vector, termed b (Christodoulou, 2001; zooghby et al., 2000; Movahedipour et al., 2006). Matrix b with scaled values is applied to the neural network as a training set and known directions are considered as target values. Therefore, by considering just one triangular part of the matrix and real and image separation, b will include $M\ (M+1)$ elements (Christodoulou, 2001). Normalized b is applied to the input layer of neural network.

6. ACO_R-based neural network for DoA estimation

In the simulations, two $S=2$ uncorrelated signals impinge on array antenna with five $M=5$ elements. These assumptions are made based on the literature to facilitate the comparison (Titus et al., 1994; Zooghby, 1997).

The neural network contains five neurons in its hidden layer. Array elements receive two uncorrelated narrow band signals in training mode with angular separations of 15^o and 20^o with the assumption that DoA is uniformly distributed from -90^o to $+90^o$. Therefore the first source is set at $\theta = -90^o$ and the second source at $\theta = -75^o$ (15^o separation). Next, we set the first source at -89^o and the second source at -74^o and so on until the interval -90^o to 90^o is covered. Same method is used for 20^o angular separation. For validation phase, separation angles of 15^o, 16^o, 17^o, 18^o, 19^o and 20^o are used to investigate the generalization capability of the neural network. All of the results are compared with RBF, which has been found as a substantially powerful solution to DoA estimation problem (Zooghby, 1997).

6.1 Application of ACO_R to train feed forward neural network

In this work, ACO_R is utilized to train a feed forward neural network, therefore numerical weights of neuron connections represent the solution components of the optimization problem and minimizing the MSE of output neurons is the performance criterion or

objective function. Search domain is (-1, 1), i.e. solution components may assume any real value in this range.
The solution archive and its correspondent normal distributions are implemented in a three dimensional matrix.

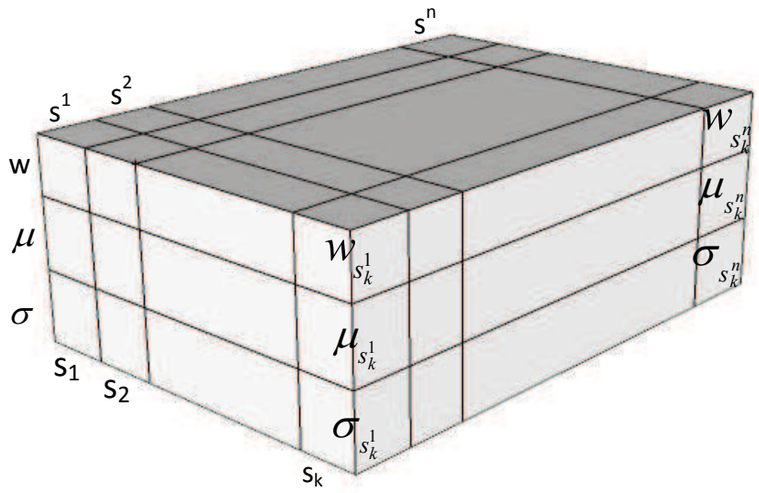


Fig. 1. The mixture $3\times k\times n$ matrix to store k solutions with their corresponding normal distributions.

Figure 1 demonstrates the mixture matrix and the way solutions and their corresponding normal distributions are stored. Mixture is a $3\times k\times n$ matrix in which the first row, represents the weight $w_{s_j^i}$ assigned to each variable of solutions. The second row holds the mean of the normal distribution which equals the value x^i of i-th variable of j-th solution s^i_j. The standard deviations of solutions are stored in the third row. In this way, for each variable of each solution, three values of w_j^i, μ_j^i and σ_j^i are assigned and ants can find probabilistic solutions referring to the mixture matrix. In fact, pheromone maintenance is performed by updating this matrix. In the simulations positive and negative update are carried out so that k is constant. n is the number of variables that is the number of weights of the neural network in this application.
In positive update phase, weights of the solutions stored in solution archive are calculated according to the MSE of the output layer. In this work, two approaches are utilized to calculate the weight of probability distributions. In the first attempt w_j is assigned in inverse relation to MSE as the objective function and it is termed *MSE-function* method.

$$w_j = \frac{t}{MSE_j} \qquad j = 1,\ldots,K \tag{23}$$

where t is a constant.
The second approach is sorting the solutions according to their performance and assigning them a "*rank*" value. Then the weight of each solution is calculated according to below *Rank-function* (Socha & Blum, 2007).

$$w_j = \frac{1}{\sqrt{2\pi}qk} e^{\frac{-(r_j-1)^2}{2q^2k^2}} \tag{24}$$

where q is a parameter and r represents the rank of corresponding solution. The mean of the above Gaussian function is set to 1, therefore the best solution with r=1 has the highest value.

The mean μ of normal PDFs are equal to the solution component values. The standard deviation of normal PDFs is calculated as

$$\sigma_j^{\,i} = \max\left\{\frac{\max(x^i{}_{1..m}) - \min(x^i{}_{1..m})}{\sqrt{c}}, \varepsilon\right\} \tag{25}$$

where c represents the iteration and m is the number of ants. Standard deviation is inversely proportional to the iteration number and it implies that solutions found in the first iterations are less useful than the outputs of the final iterations. Note that less standard deviation leads to the higher probability for selection of values around x^i. In this work, following equation for standard deviation is proposed to enhance the performance of algorithm.

$$\sigma_j^{\,i} = \max\left\{\frac{\max(x^i{}_{1..m}) - \min(x^i{}_{1..m})}{u.w_j.\sqrt{c}}, \varepsilon\right\} \tag{26}$$

where u is a parameter and standard deviation of each normal PDF is inversely proportional to its weight, therefore superior solutions with higher weights, lead to smaller standard deviations and higher probability of selection. Low rank and larger standard deviation is almost equivalent to uniform distribution and it results in decreasing the effect of bad solutions. The performance of proposed equation is evaluated in section 7.3.

6.2 Parameter setting for weight assignment

As shown in the equations (6) and (7), *nos* is the number of solutions used to update solution archive which is equivalent to pheromone update in regular ACO. After each iteration, all ants are sorted with respect to their solution quality and then the normal distribution correspondent to the *nos* best solutions are added to the mixture matrix. As shown in figure 1, the first row of the mixture matrix is the weight of each distribution and it is attained by *MSE-function* or *Rank-function* using equations (23) or (24). *Rank-function* is a normal distribution with qk as the standard deviation. Considering *nos*=10 and K=40, we need to calculate *Rank-function* for r=1, …, 10. r=10 gives the weight w_{10} to the solution of 10th ant. Depending on the value of qK as the σ of the *Rank-function*, the weight of last ants may get values quite close to zero. To avoid assignment of weights close to zero to the last solutions of the sorted ants, we can determine a limit C for the last r=*nos* solution and find its correspondent standard deviation:

$$\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(nos-1)^2}{2\sigma^2}} = C \tag{27}$$

$$nos = 1 + \sigma\sqrt{-2(Ln(C) + Ln(\sigma) + 0.9189)} \tag{28}$$

Assuming C=0.01, K=40 and *nos*=10, the standard deviation and consequently q is achieved ($\sigma = qK = 4.254$, $q = 0.1063$). By calculating q using above relation, no solution of *nos* ants

will get a weight under the limit *C*, and in fact, one parameter is reduced in parameter setting of the algorithm.

7. Simulation results

In the literature, RBFNN method is utilized for DoA problem and its performance in terms of its feasibility to handle real time computation is investigated (Titus et al., 1994; Zooghby et al., 1997i; Kuwahara & Matsumoto, 2005). In this section, result of applying ACO_R-based neural network to DoA problem is presented and estimation errors are compared with RBFNN method. Besides possible ways to enhance the performance of ACO_R are investigated. The achieved results using *Rank-function* and *MSE-function* are compared to RBFNN method in figure 2. The separation angles of 3^o to 7^o are used and the generalization capability of the neural network is verified with 4^o, 5^o and 6^o separation angles that were not part of the training set.

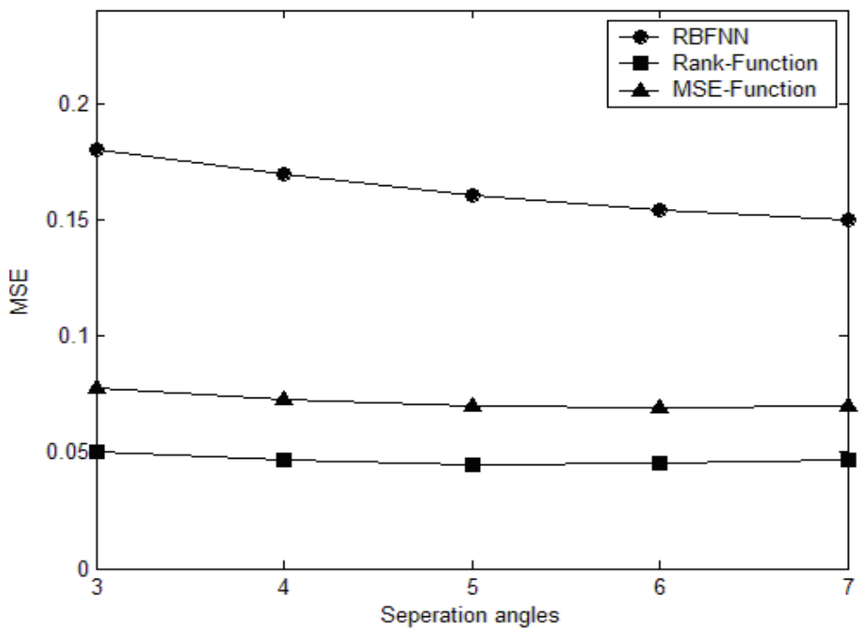


Fig. 2. The neural network is trained with 3^o and 7^o separation angles and its generalization capability is verified in 4^o, 5^o, 6^o separation angles. *Rank-function* and *MSE-function* based weight assignments are compared as well.

Figure 3 demonstrates MSE for one more set of separation angles of 15^o to 20^o and it indicates that *Rank-function* method attains 3.12 to 5.9 times less MSE in comparison with RBFNN. Therefore, in DoA applications that high resolution is needed, ACO_R can be utilized to achieve lower and more uniform estimation errors.

As it is shown in figures 2 and 3, *Rank-function* approach for pheromone update generates better solutions but it takes three times longer than *MSE-function* approach to converge. This is reasonable because ranking function normalizes weight assignments for achieved MSE and this leads to uniform effect of performance in different stages of the simulation. Hence, stagnation is avoided and consequently search time is extended (Movahedipour et al., 2008). The ACO_R parameters used in above simulations are presented in table 1.

To study the behaviour of ACO_R algorithm, we investigated the role of various parameters on the performance of ACO_R and presented the results in the following sections.

Parameter	*Description*	*Value*
k	Number of solutions in mixture matrix	15
nos	Number of solutions used for pheromone update	8
m	Number of ants	85
ρ	Evaporation rate	0.2
γ	Dissolving rate	1.1
q	Ranking function parameter	0.2

Table 1. ACO_R Parameters

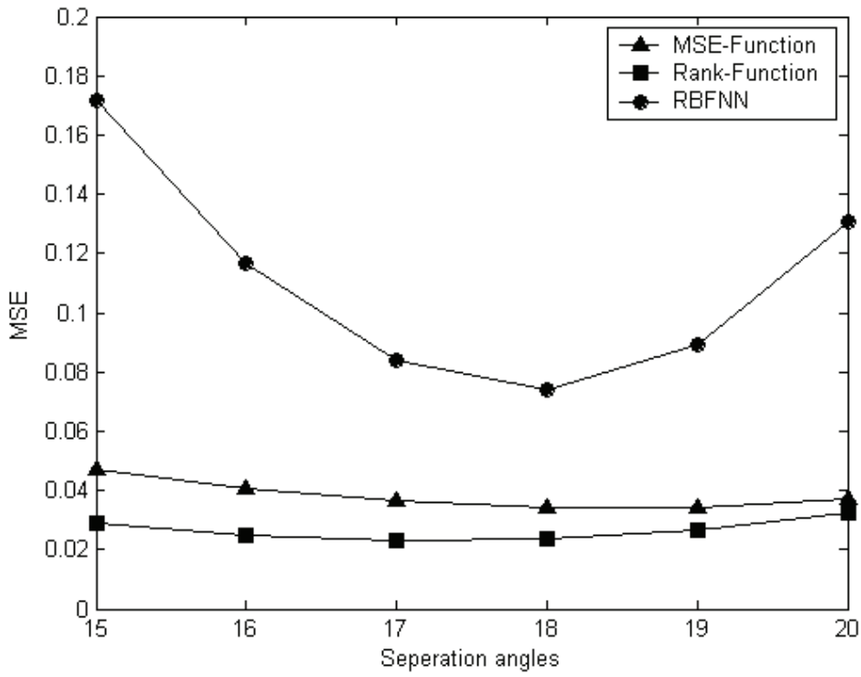


Fig. 3. The neural network is trained with 15^o to 20^o separation angles and its generalization capability is verified in 16^o, 17^o, 18^o and 19^o separation angles.

7.1 Number of solutions (*k*, *nos*, *q*)

We ran ACO_R using parameter settings of $k \in \{10, 15, 20, 25, 30, 35, 40, 45, 50\}$. We repeated the execution three times for each parameter setting, selected the best performing one for each k, and presented them in figure 4. *nos* was set to $k/3$ and $k/2$, q was achieved using equation (28) and all other choices were left the same. The parameter k represents the stored history of the algorithm and it guides ants to the promising areas of the search space. The algorithm does not show acceptable performance for k=10 because in this configuration very few number of found solutions are stored in the solution archive and consequently the experience of other ants cannot be utilized in a proper way to generate better solutions (see

figure 4). On the other hand assigning values larger than 40, increases simulation time and decreases the quality of found solutions because a big solution archive keeps a larger amount of solutions with less quality. The numbers from one to nine and letters from *a* to *i*, indicate various parameter settings presented in table 2.

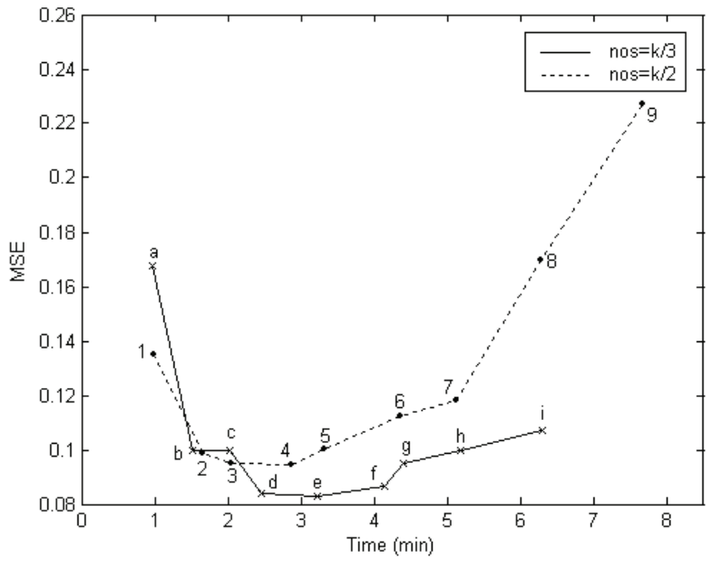


Fig. 4. ACO_R is executed for $k \in \{10, 15, 20, 25, 30, 35, 40, 45, 50\}$, *nos*={k/3, k/2} and *q* is achieved according to equation (28).

	k	*nos*	*q*
a	10	3	0.07
b	15	5	0.1
c	20	7	0.13
d	25	8	0.13
e	30	10	0.14
f	35	12	0.16
g	40	13	0.16
h	45	15	0.17
i	50	17	0.19
1	10	5	0.16
2	15	8	0.21
3	20	10	0.21
4	25	13	0.25
5	30	15	0.26
6	35	18	0.3
7	40	20	0.3
8	45	23	0.37
9	50	25	0.44

Table 2. ACO_R parameter settings for different *k*, *nos* and *q*. Other parameters are *m*=30, ρ =0.2 and γ =1.1.

7.2 Number of ants (*m*)

The results presented in figure 5 are based on the parameter settings of $m \in \{20, 35, 40, 50, 60, 70, 80, 90, 100\}$ leaving all other parameters the same. Increasing the number of ants improves the final solution but the computation time increases as well. We ran three times the ACO_R algorithm for each *m*, chose the best execution and plotted in figure 5.

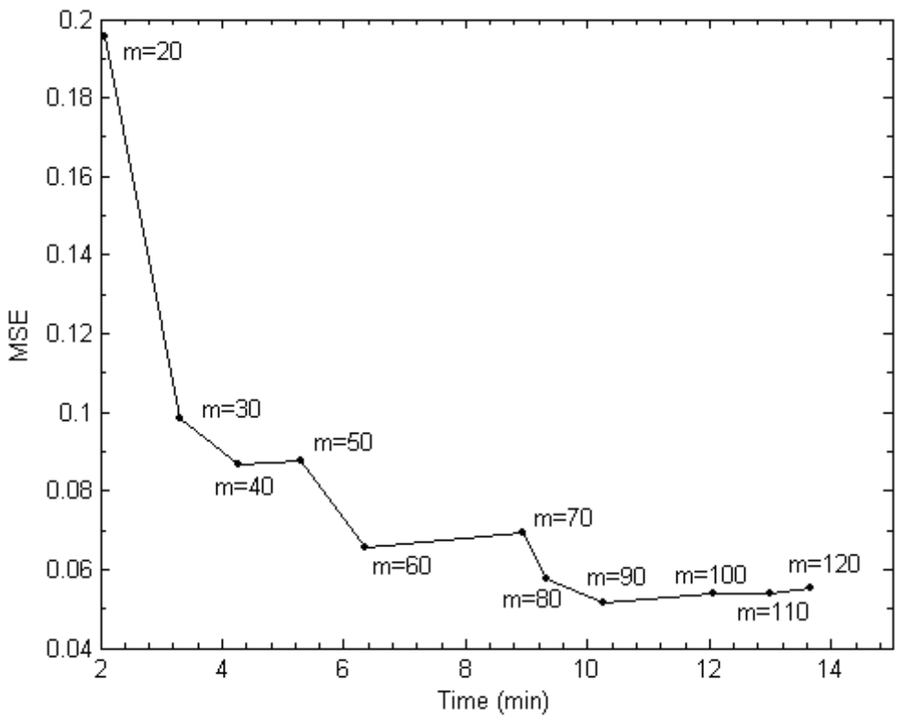


Fig. 5. ACO_R is executed for different numbers of ants. The other parameters are left same. (k=15, nos=8, ρ =0.2, γ =1.1, q=0.21)

7.3 Standard deviation

In this section, two methods of standard deviation assignment to PDFs are evaluated (see equations 25 and 26). The ACO_R algorithm was executed ten times for each method. The results illustrated in figure 6, indicate that the proposed equation (26) can decrease the found minimum for 29% but the convergence happens later and the computation time increases. The parameter u is set to 1 and 2 and the experiments suggest ACO_R is converged earlier if u is set to a large value and this results in lower quality solutions. Therefore, the parameter u can be used to compromise computation time with solution quality.

7.4 Exploration and exploitation

The experimental results summarized in previous sections, indicate ACO_R usually suffers from lack of simultaneous exploitation and exploration strategies. For instance, when the number of ants m is set to high values in comparison with k, exploitation is accomplished since the solution archive is fed with high quality solutions but the lack of enough exploration in this strategy results in fast convergence and suboptimal solutions. On the other hand, when k, ρ and γ are set to high values, exploration is achieved but the

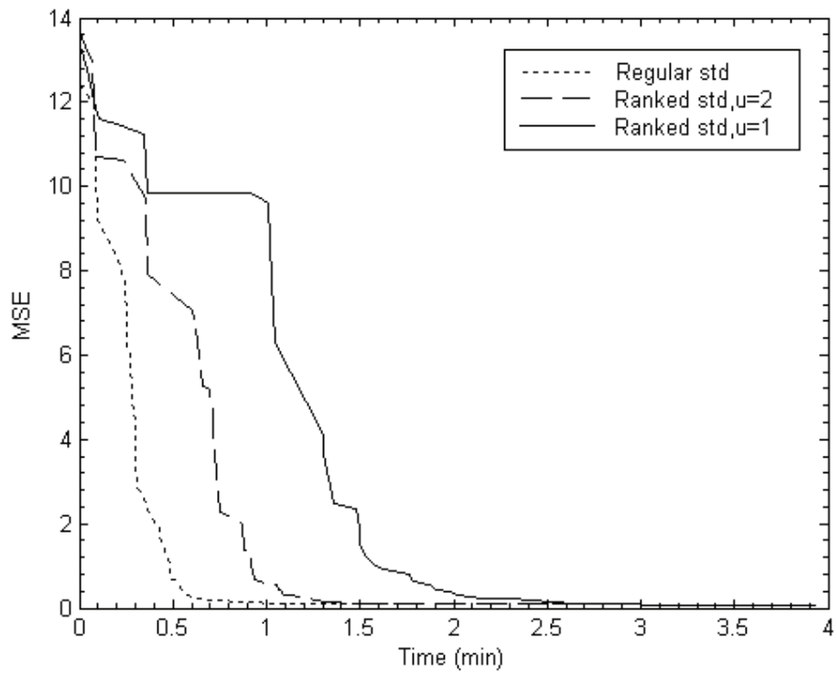


Fig. 6a. Different methods for standard deviation calculation are investigated. "Regular std" denotes equation (25) and "Ranked std" refers to equation (26). The other parameters are left same. (k=15, nos=8, ρ =0.2, γ =1.1, q=0.21)

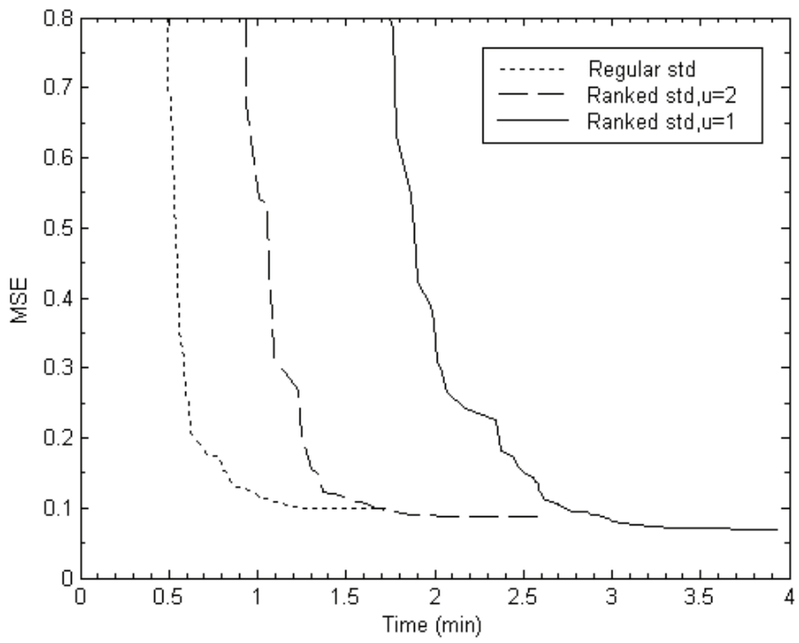


Fig. 6b. The convergence behaviour of algorithm in figure 6a is magnified.

algorithm suffers from lack of exploitation to take advantage of explored areas and construct high quality solutions. Starting the algorithm with dominant exploration strategy, provides a substantial information about the search space and then if it enters exploitation phase, this information will guide the ants to promising areas of search space and better solutions. Hence, it is proposed to implement two phases of exploration and exploitation in ACO_R and apply two different sets of parameter settings to implement each phase. This approach was experimented and best solutions were achieved among all different parameter settings with slight increased computation time (see figures 7a and 7b).
Different parameter sets used in this experiment are summarized in Table 3.

	k	nos	m	ρ	γ	q
Exploration set	30	10	35	0.25	1.15	0.2
Exploitation set	15	8	85	0.2	1.1	0.2
Exploration-Exploitation (Phase I)	30	10	35	0.25	1.15	0.2
Exploration-Exploitation (Phase II)	15	8	85	0.2	1.1	0.2

Table 3. The "Exploration", "Exploitation" and "Exploration-Exploitation" configuration.

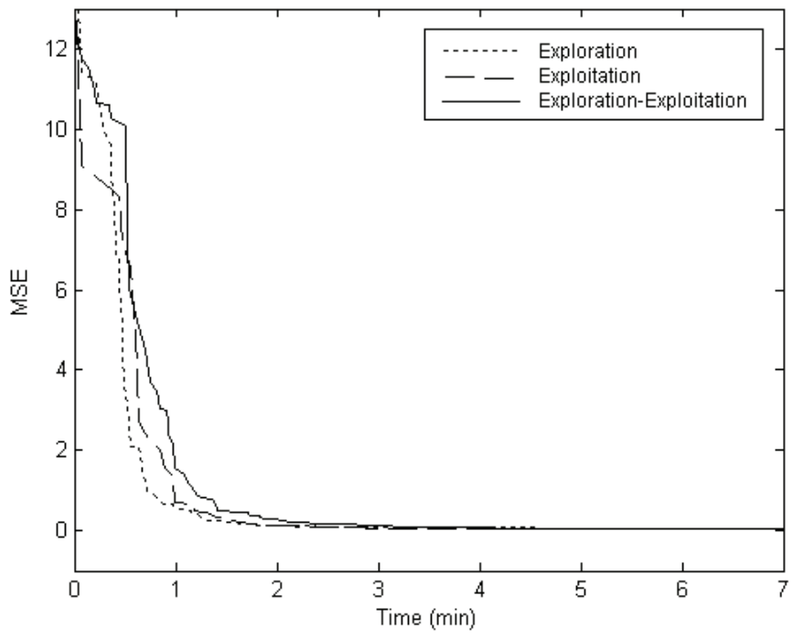


Fig. 7a. Two different parameter sets are defined for exploration and exploitation phases and three configurations of these parameter sets are tested. ACO_R with "Exploration", Exploitation" and "Exploration-Exploitation" parameters are executed for 10 times and the best run of each configuration is presented for comparison. In "Exploration-Exploitation", first ACO_R is executed with "Exploration" parameter set and then after 400 iterations, algorithm enters a new phase and uses "Exploitation" parameter set.

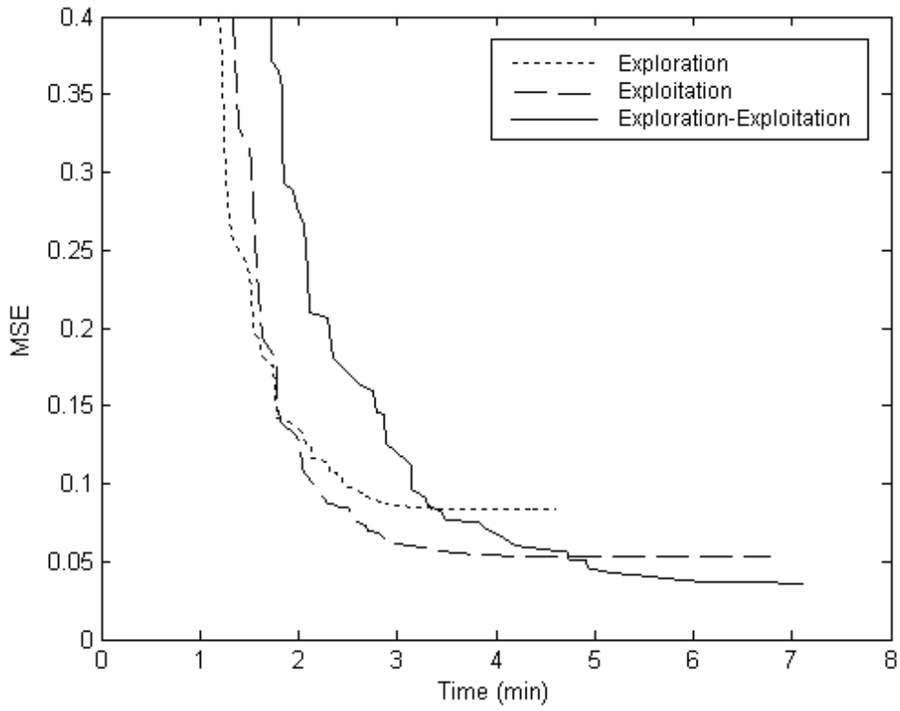


Fig. 7b. The convergence behaviour of algorithm in figure 7a is magnified.

In the exploration set, *k* is set to a higher value with comparison to exploitation set, so that the algorithm keeps a larger amount of information about the history of search. However, *m* is set to smaller value and *nos* is almost one third of *m*, implying that the most solutions found by ants are stored in the solution archive. In the exploitation phase, *nos*=0.09*m* and that points out just a small number of very high performing solutions (8 out of 85) are stored in the solution archive. In this way, high-quality solution archive guides the ants to explore more deeply, the areas that were found to be promising in the exploration phase.
We ran ACO_R algorithm with "Exploration", Exploitation" and "Exploration-Exploitation" parameters for 10 times and the result of each execution is summarized in Table 4.

	MSE of the converged solution in each execution									
Exploration	0.1085	0.0953	0.1088	0.1084	0.1308	**0.0836**	0.1062	0.1107	0.4797	0.0960
Exploitation	0.0636	0.0691	0.0578	0.0628	0.0573	0.0665	0.0777	0.0599	**0.0534**	0.0603
Exploration-Exploitation	**0.0362**	0.0629	0.0516	0.0524	0.0515	0.0549	0.0511	0.0526	0.0657	0.0641

Table 4. MSE per execution for each configuration

The bold numbers in Table 4 are related to the executions with lowest MSE and their computation time are summarized in Table 5. The complete iterations of these executions are illustrated in figure 7a and 7b. This approach resulted in the lowest MSE of 0.0362 that was the best-found solution in all of our experiments with various parameter settings.

	Min	*Time*
Exploration	0.0836	3.76
Exploitation	0.0534	4.52
Exploration-Exploitation	0.0362	6.88

Table 5. Minimum MSE and the computation time for best executions of each configuration

8. Conclusion

The application of ACO_R algorithm to train an interpolator feed-forward neural network and perform DoA estimation have been studied in this chapter. The ACO_R based neural network has been used to model DoA estimation and it demonstrated superior performance in comparison with RBFNN as the known approach for DoA problem.

Various parameter settings in ACO_R were investigated and the behaviour of algorithm was discussed accordingly. The possibilities to enhance ACO_R behaviour were examined. Standard deviation calculation in positive update phase is proposed to be inversely proportional to the rank function. In addition, it was investigated how the algorithm performance is improved by guiding the algorithm through exploration and exploitation phases by different parameter sets during execution.

9. References

Christodoulou, C. (2001). Applications of Neural Network in Electromagnetics, Artech House.

Danneville, E; Brault, J & Laurin, J. (2005). Implementation of an MLP-based DOA System Using a Reduced Number of MM-wave Antenna Elements, Proceedings of International Joint Conference on Neural Networks, Montreal, Canada, July 31-August 4, 2005.

Deneubourg, J.-L; Aron, S.; Goss, S. & Pasteels, J.-M. (1990). The self-organizing exploratory pattern of the Argentine ant. Journal of insect behavior, 3, 159-168.

Dorigo, M.; Maniezzo, V. & Colorni, A. (1991). Positive feedback as a search strategy, Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991.

Dorigo, M.; Maniezzo, V. & Colorni, A. (1996). The Ant System: Optimization by a colony of cooperating agents, IEEE Trans SMC Part B, 1996.

Dorigo, M. & Stutzle, T. (2004). Ant Colony Optimization, MIT Press, Cambridge, MA.

Godara, L. (2002). Handbook of antennas in wireless communications, CRC Press.

Kuwahara; Matsumoto (2005). Experiments of direction finder by RBF neural network with post processing, IEEE Electronic Letters, Volume 41, Issue 10, May 2005.

Lee, J. (2004). A first course in combinatorial optimization, Cambridge texts in mathematics.

Movahedipour, H.; Atlasbaf, Z. & Hakkak, M. (2006). Performance of Neural Network Trained with Genetic Algorithm for Direction of Arrival Estimation, IEEE MCWC Conference, 2006.

Movahedipour, H; Atlasbaf, Z; Mirzaee, A & Hakkak, M. (2008). A Hybrid Approach Involving Artificial Neural Network and Ant Colony Optimization for Direction of Arrival Estimation, IEEE CCECE Conference, Canada, 2008.

Papadimitriou, CH & Steiglitz, K. (1982) Combinatorial optimization—Algorithms and complexity, Dover.

Schoonderwoerd, R.; Holland, O. & Bruten, J. (1997). Ant-based Load Balancing in Telecommunications Networks", Adaptive behaviors, 1997.

Socha, K. (2004). ACO for continuous and mixed-variable optimization, in M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada and T. Stutzle (eds), Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004, Vol. 3172 of LNCS, Springer-Verlag, Berlin, Germany, pp. 25–36.

Socha, K. & Blum, C. (2007). Hybrid ant algorithms applied to feed-forward neural network training: An application to medical pattern classification, Neural Computing and Applications, 16(3): 235–248, 2007.

Socha, K. & Dorigo, M. (2008). Ant colony optimization for continuous domains, European Journal of Operations Research 185(3): 1155–1173.

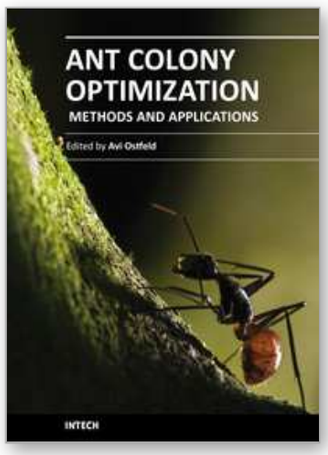
Titus, K ; Leung, H. & Litva, J. (1994). Radial Basis Function Neural Network for Direction-of-Arrival Estimation, IEEE Signal Processing Letters, vol. 1, No. 2, February 1994.

Titus, K ; Leung, H. & Litva, J. (1994). Artificial neural network for AOA estimation in a multipath environment over the sea, IEEE Journal of Oceanic Engineering, vol. 19, (Oct. 1994), pp 555-562.

Zooghby, A.; Christodoulou, C. & Georgiopoulos, M. (1997). Performance of Radial-Basis Function Networks for Direction of Arrival Estimation with Antenna Arrays, IEEE Transactions on Antenna and Propagation, Vol. 45, No. 11, November 1997.

Zooghby, A.; Christodoulou, C. & Georgiopoulos, M. (1997i). Antenna array signal processing with neural networks for direction of arrival estimation", IEEE Antennas and Propagation Society International Symposium, Volume 4, Issue 13, July 1997.

Zooghby, A.; Christodoulou, C. & Georgiopoulos, M. (2000). A neural network-based smart antenna for multiple source tracking, IEEE Trans. Antennas Propagation, pp. 768-776, May 2000.



Ant Colony Optimization - Methods and Applications

Edited by Avi Ostfeld

ISBN 978-953-307-157-2
Hard cover, 342 pages
Publisher InTech
Published online 04, February, 2011
Published in print edition February, 2011

Ants communicate information by leaving pheromone tracks. A moving ant leaves, in varying quantities, some pheromone on the ground to mark its way. While an isolated ant moves essentially at random, an ant encountering a previously laid trail is able to detect it and decide with high probability to follow it, thus reinforcing the track with its own pheromone. The collective behavior that emerges is thus a positive feedback: where the more the ants following a track, the more attractive that track becomes for being followed; thus the probability with which an ant chooses a path increases with the number of ants that previously chose the same path. This elementary ant's behavior inspired the development of ant colony optimization by Marco Dorigo in 1992, constructing a meta-heuristic stochastic combinatorial computational methodology belonging to a family of related meta-heuristic methods such as simulated annealing, Tabu search and genetic algorithms. This book covers in twenty chapters state of the art methods and applications of utilizing ant colony optimization algorithms. New methods and theory such as multi colony ant algorithm based upon a new pheromone arithmetic crossover and a repulsive operator, new findings on ant colony convergence, and a diversity of engineering and science applications from transportation, water resources, electrical and computer science disciplines are presented.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen