01 Jan 2006

Recurrent Neural Network Based Predictions of Elephant Migration in a South African Game Reserve

Parviz Palangpour

Ganesh K. Venayagamoorthy
Missouri University of Science and Technology

Kevin Duffy

Recommended Citation

P. Palangpour et al., "Recurrent Neural Network Based Predictions of Elephant Migration in a South African Game Reserve," *Proceedings of the IEEE International Joint Conference on Neural Network, 2006*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2006.
The definitive version is available at https://doi.org/10.1109/IJCNN.2006.246953

2006 International Joint Conference on Neural Networks
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Recurrent Neural Network Based Predictions of Elephant Migration in a South African Game Reserve

Parviz Palangpour, *Student Member, IEEE*, Ganesh K. Venayagamoorthy, *Senior Member, IEEE*, and Kevin Duffy

Abstract— **A large portion of South Africa's elephant population can be found on small wildlife reserves. When confined to enclosed reserves the elephant densities are much higher than observed in the wild. The large nutritional demands and destructive foraging behavior of elephants threaten rare species of vegetation. If conservation management is to protect threatened species of vegetation, knowing how long elephants will stay in one area of the reserve as well as which area they will move to next is essential. The goal of this study is to train a recurrent neural network (RNN) to continuously predict an elephant herd's next position in the Pongola Game Reserve. Accurate predictions would provide a useful tool in assessing future impact of elephant populations on different areas of the reserve. The particle swarm optimization (PSO) algorithm is used to adapt the weights of the neural network. Results are presented to show the effectiveness of RNN-PSO for elephant migration prediction.**

I. Introduction

Converting farmland into small wildlife reserves is becoming common throughout South Africa. These reserves introduce the major species that attract tourists: lion, leopard, elephant, rhino, giraffe and hippopotamus. The last four are often classified as *mega* herbivores due to the large size and great nutritional demand of the species. It is known that several mega herbivores, such as the elephant, are destructive foragers and their overpopulation can adversely impact the ecology of their range. Elephants are also known to exhibit preferences for certain species of vegetation [1]. As a result, elephants confined to small areas can have a heavy impact on the diversity of vegetation throughout their habitat.

Knowing which areas of vegetation are being threatened by elephant overpopulation is essential for effective conservation management. A short term prediction system that could model an elephant herds migration would provide a powerful tool to conservation management. As the elephants migrate, the herd is continually reducing the amount of vegetation available in different areas of the reserve. This in return changes the migration behavior of the herd since the availability of vegetation is a large factor of habitat selection. Because of the labor required to continually survey the health of the vegetation in an entire reserve, this information is rarely known. Using the previous positions of an elephant herd without knowing the state of vegetation throughout the reserve makes elephant migration prediction very difficult.

The neural network (NN) has been applied successfully to many problems involving time series prediction and modeling of non-linear systems [2]. These NN's typically use a fixed number of previous states to predict the next state of the system. Recurrent Neural Networks (RNN), utilize feedback within the network which allows them to memorize previously presented patterns. This capability makes RNN's superior to feedforward NN's when modeling dynamic temporal systems because the networks output is a function of both the current inputs as well as all of the previous inputs. In this paper, RNN's are used to model the short term dynamics of elephant herd migration in a small reserve. The particle swarm optimization (PSO) is applied for training two RNN's, one network predicts the x coordinate and the other predicts the y coordinate of the herds position.

This paper is organized as follows. Section II describes the game reserve from which data was collected. Section III explains the RNN architecture. Section IV presents the techniques used to train the RNN's to predict elephant migration. Finally, Section V analyzes the results obtained from the study and offers some future research directions.

II. Case Study

Located on the southeastern border of Swaziland in South Africa is the Pongola Game Reserve. In June 1997, a family group of 17 elephants from another park were introduced to the reserve. While the reserve is 73.6 km^2, the same family group had grown to 37 individuals by January 2004. Three bulls which move independently of each other also live on the reserve. In addition to the family group and bulls, a group of five young elephants also shared the habitat during this period. Each of these three groups migrate to different areas of the reserve as separate herds. A study has indicated at least one rare tree, the *Sclerocarya birrea* is being removed at a rate higher than annual regeneration [1]. The diversity of vegetation in the reserve can be seen in Fig. 1.

A. Data Collection

In February 2000, a cow from the family group was fitted with a GPS satellite collar to monitor their movement. The GPS positions of the collar were recorded on semi-regular intervals until March 2002. Because elephants in a family herd move as a group, the movement of a single elephant represents the general movement of the herd.

This work was supported in part by the National Science Foundation CAREER grant ECS #0348221.

Parviz Palangpour and Ganesh K. Venayagamoorthy are with the Real-Time Power and Intelligent Systems (RTPIS) Laboratory, Department of Electrical and Computer Engineering, University of Missouri - Rolla, 1870 Miner Circle, Rolla, MO 65409, USA (email: pmpv3b@umr.edu, gkumar@ieee.org)

Kevin Duffy is with the Centre for Systems Research, Durban Institute of Technology, Durban, South Africa.

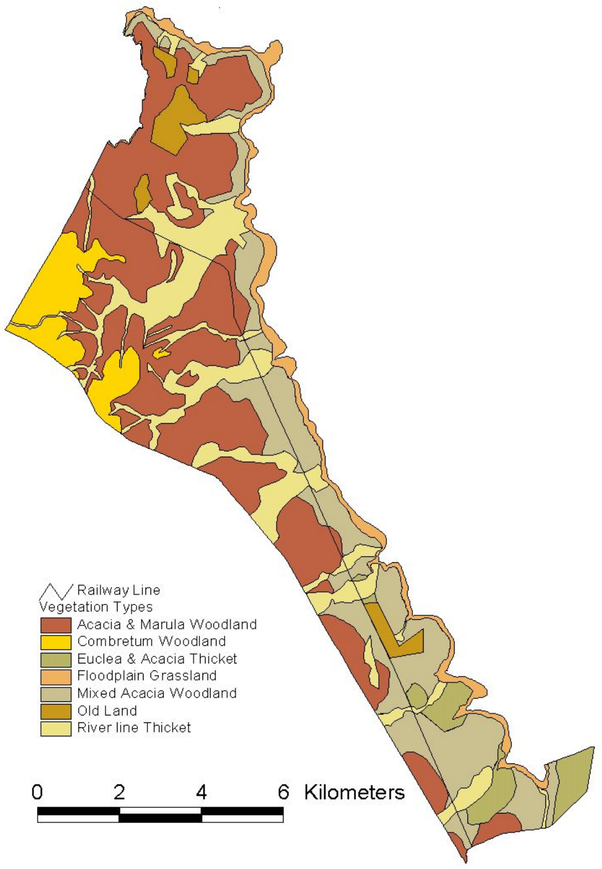

Fig. 1. Pongola game reserve

B. Preprocessing of GPS Data

The longitude and latitude coordinates from each set of recorded positions are first transformed to a projected cartesian coordinate system. The data from each group is then interpolated to produce a dataset consisting of one position for every twenty-four hour interval. All positions are normalized to be in [-1,1].

III. RNN for Elephant Migration Prediction

The Elman RNN model adopted here stores the previous state of the hidden layer into what is known as the context layer. When a pattern is presented to the network, the values in the context layer serve as additional inputs to the hidden layer. This feedback in the RNN topology allows RNN's to incorporate all of the previous input patterns. The weights of the connections from the hidden layer to the context layer are fixed and equal to 1. The layers which are fully connected are denoted by bold arrows in Fig. 2. The RNNs used are of size 4×16×1. To predict the elephant herd position at the time $t+1$, the network is presented the herds' current position $Z(t)$, and a number of time-delayed positions, $Z(t-1)$, $Z(t-2)$, $Z(t-3)$. The neurons in the hidden layer use the hyperbolic tangent activation function. As the input values are normalized to be in [-1,1], the output layer denormalizes the output to scale the predicted position in units of kilometers.

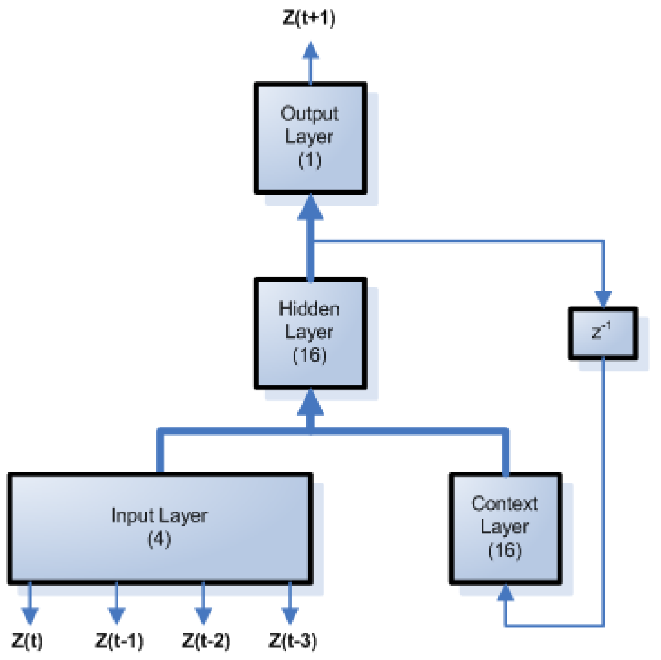

Fig. 2. RNN (size-4×16×1) architecture for elephant migration prediction

IV. Training The RNN

A batch learning technique is used to update the weights of the RNN's after each presentation of the entire training dataset. The training dataset consists of 180 days of recorded positions. After each iteration, the weights are updated based on the cumulative error of the networks predictions over the training dataset.

There are a number of different training algorithms used for this purpose, though backpropagation and other forms of gradient descent have been used the most extensively. In addition, evolutionary algorithms such as PSO and genetic algorithms have also been used to optimize the weights of NN's [3]-[4]. When used to train the same NN, comparisons of backpropagation, genetic algorithms and PSO have shown that PSO requires the fewest number of training iterations to achieve the same error [5]-[6]. PSO is selected because it has been shown to be efficient for training NN's and performs well on a large variety of problems [8].

Two separate RNNs are needed to predict the position (x,y), the network that predicts the x position (NN_x) and the network that predicts the y position (NN_y). Attempts are made to use both the x and y coordinates to train each network, but this did not prove feasible.

A. Particle Swarm Optimization

Particle swarm optimization is used to find the optimal weights of the RNNs. The PSO algorithm was developed by Kennedy and Eberhart and is based on the evolution of a population of particles [3]. Each particle in the population has a position vector which represents a potential solution to the problem. In this problem, each value in the position vector of a particle corresponds to a weight in a RNN, thus each particle represents a complete RNN. The particles are initialized to random positions throughout the search space and for each iteration of the algorithm a velocity vector is computed and used to update each particles position. Each particles velocity is influenced by the particles own experience as well as the experience of its neighbors.

In this study, the *global* version of the PSO algorithm is applied. For each iteration, a cost function f is used to measure the fitness of each particle i in the population. The position of each particle i is then updated, which is influenced by three terms, the particles velocity from the last iteration, the difference between the particles known best position and the particles current position, and the difference between the swarms best known position and the particles current position. The latter two terms are each multiplied by a random number in [0,1] to randomly vary the influence of each term, as well as an acceleration coefficient to scale and balance the influence of each term. The best position each particle attained is stored in the vector p_i, while the best position attained by any particle in the population is stored in the vector p_g. The velocity vector $v_i(t)$ for each particle is then updated.

$$\begin{aligned} v_i(t+1) = v_i(t) + c_1\rho_1(p_i - x_i(t)) \\ +c_2\rho_2(p_g - x_i(t)) \end{aligned} \tag{1}$$

where c_1 and c_2 are positive and ρ_1 and ρ_2 are uniformly distributed random numbers in [0,1]. The term c_1 is called the cognitive acceleration term and c_2 is called the social acceleration term. These two values balance the influence between the particles own best performance and that of the population. The velocity is constrained between the parameters V_{min} and V_{max} to limit the maximum change in position.

$$v_i(t+1) = \begin{cases} v_{Max} & \text{if } v_i(\text{t + 1}) > V_{max} \\ v_{Min} & \text{if } v_i(\text{t + 1}) < V_{min} \\ v_i(t+1) & \text{else} \end{cases} \tag{2}$$

The position of each particle is then updated using the new velocities.

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{3}$$

The position in each dimension is limited between the parameters X_{min} and X_{max}.

$$x_i(t+1) = \begin{cases} x_{Max} & \text{if } x_i(\text{t + 1}) > X_{max} \\ x_{Min} & \text{if } x_i(\text{t + 1}) < X_{min} \\ x_i(t+1) & \text{else} \end{cases} \tag{4}$$

B. PSO Parameter Selection

The original PSO algorithm uses static parameters, though several studies have shown that using dynamic parameters for PSO can greatly improve the convergence speed and reduce the probability of converging on a local minima [4]-[7]. Many approaches use parameters that are time-varying and typically favor global exploration at the beginning of the search and local search toward the end. Shi and Eberhart suggested using an inertia term, w, in the velocity update equation to control the amount of momentum added [12].

$$\begin{aligned} v_i(t+1) = w * v_i(t) + c_1\rho_1(p_i - x_i(t)) \\ +c_2\rho_2(p_g - x_i(t)) \end{aligned} \tag{5}$$

By starting the algorithm with a large inertia term and gradually decreasing it toward the end of the search, PSO performs global exploration in the early stages while transitioning towards finer improvements with iterations. Another strategy, is to vary the acceleration coefficients so either the cognitive component or the social component has a larger influence on the particles at different stages of the search [13].

The approach used in this study uses the velocity update equation (5) where c_1 is a fixed value, c_2 is an increasing function of iterations, and the inertia term w is a decreasing function of iterations. The value for c_1 is 1.5 and c_2 is increased linearly with iterations from 0.1 to 1.5. During the early stages of training, c_2 is very small and thus the social term has little impact on the particles velocity, this prevents premature convergence on the global best. The inertia term is held fixed for the first 75% of the allowed number of PSO iterations and then decreased linearly with iterations. The inertia term is

$$w = \begin{cases} 0.8 & \text{if } m < (0.75)e \\ 0.8 - \frac{0.4[m-(0.75)e]}{(0.25)e} & \text{else} \end{cases} \tag{6}$$

where m is the current iteration and e is the total number of allowed iterations. The parameters X_{max} and X_{min} are set to 1 and -1 respectively, to constrain the weights and biases of each RNN to be in [-1,1]. The V_{max} and V_{min} parameters are used to constrain the velocity for any dimension to be in [-1,1], or half of the dynamic range of the search space.

C. Fitness Function

Each distinct area of the Pongola Game Reserve has been classified by the dominate vegetation, as seen in Fig. 1. For the purpose of assessing the elephant population's impact on distinct areas of vegetation, even a very small error in the predicted position can result in the wrong vegetation being classified; therefore, the network with the least mean square error (MSE) is the most ideal. The fitness of particle p_i is

$$f(p_i) = \frac{1}{|P|}\sum_{t=1}^{|P|}(Z(t) - P(t))^2 \tag{7}$$

where P is the dataset of training patterns, $Z(t)$ is the output of the NN and $P(t)$ is the target output. Using the MSE as the fitness function will put a non-linear emphasis on removing errors in the predictions that are greater than the mean error.

V. Results

The optimal number of hidden neurons and time-delayed inputs for both networks is found empiricly to be 16 and 4, respectively. Each network is trained for 350 iterations. The population size is equal to 40 particles, where each particle in the population represents a RNN for each iteration. The networks are trained using 180 days of recorded positions.

The trained networks are then used to predict positions one day in advance, for the following 180 days. Table I shows our training and testing results for networks NN_x and NN_y. It can be observed from Table I that both networks achieved a lower MSE on the testing dataset than the training dataset. This is due to much larger variations in movement during the first six months when compared with the second six months, which can be seen in Figs. 3-4 and 5-6, respectively.

TABLE I
COMPARISON OF THE TRAINING AND TESTING ERRORS

NN	Training MSE (km^2)	Testing MSE (km^2)
NN_x	0.8641	0.6740
NN_y	5.1063	3.4510

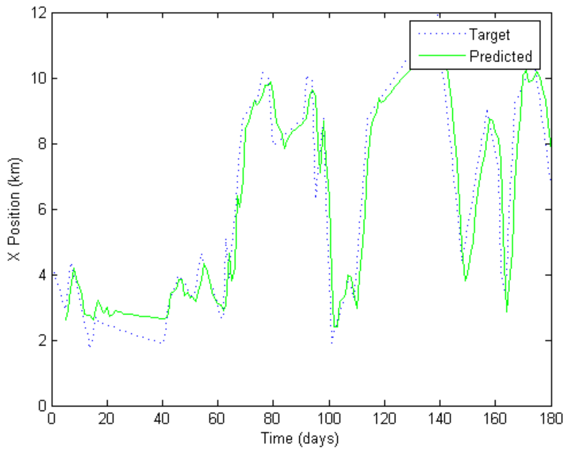

Fig. 3. Comparison of NN_x training results against the recorded data

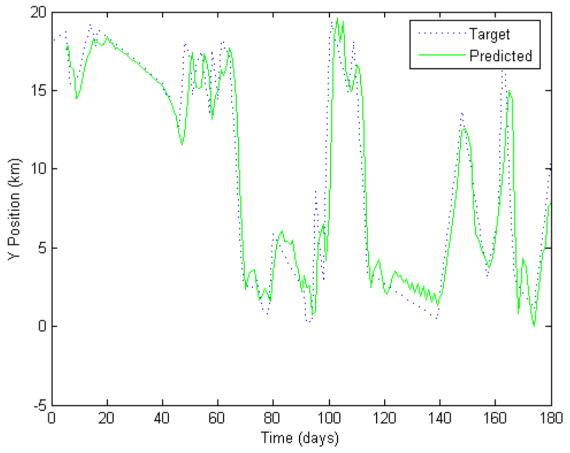

Fig. 4. Comparison of NN_y training results against the recorded data

A. Predict-Correct Technique

Because the output of a RNN is a function all the previous input patterns presented, any error in the output of the hidden layer will influence future predictions. A predict-correct technique is used to compensate for the networks current output error as a function of the networks previous output error during testing. Using this technique, the corrected output of the network is

$$Z^*(t) = Z(t) + 0.5 * (P(t-1) - Z(t-1)) \qquad (8)$$

where $Z(t)$ is the output of the network at time t and $P(t)$ is the recorded position at time t. Using this predict-correct technique, we were able to improve the accuracy of both networks NN_x and NN_y; our results are compared in Table II. Figs. 7 and 8 show the networks predictions using predict-correct to compensate for previous errors.

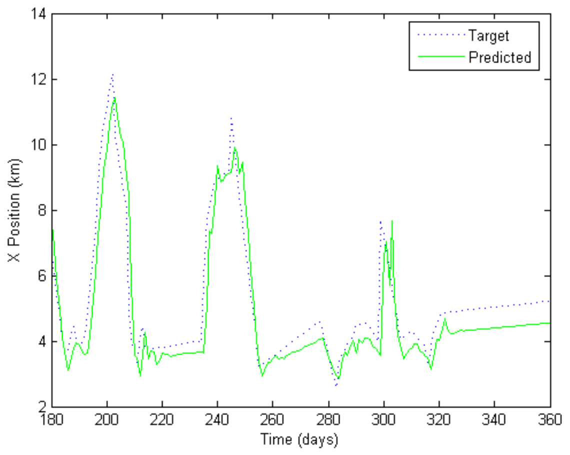

Fig. 5. Comparison of NN_x testing results against the recorded data

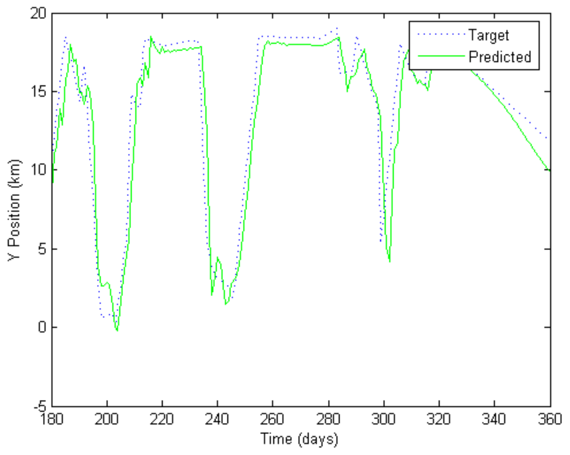

Fig. 6. Comparison of NN_y testing results against the recorded data

VI. Conclusions

A short term prediction system for elephant movement in a South African game reserve has been presented. RNN trained with PSO have been shown to provide some degree of success, though research in neural networks and optimization algorithms are constantly progressing. Better results

TABLE II

TESTING ERROR IMPROVEMENT FOR PREDICT-CORRECT

NN	Testing MSE (km^2)	Predict-Correct Testing MSE (km^2)	Predict-Correct Improvement (%)
NN_x	0.6740	0.3308	50.91
NN_y	3.4510	0.8301	75.94

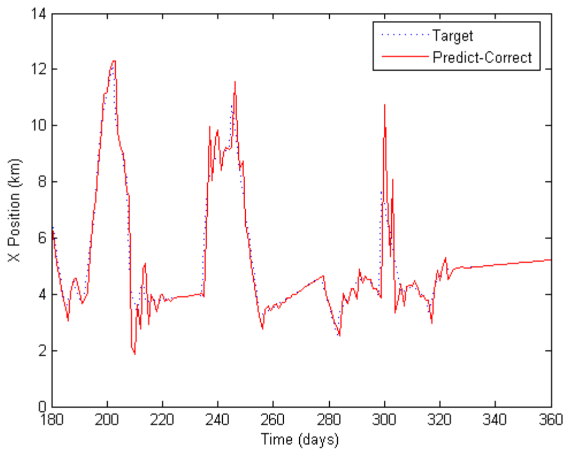

Fig. 7. Comparison of NN_x predict-correct testing results against the recorded data

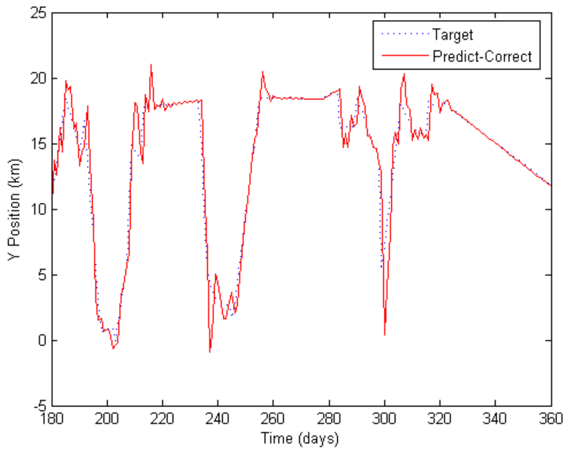

Fig. 8. Comparison of NN_y predict-correct testing results against the recorded data

are surely attainable given the inherent flexibility of neural networks.

The results indicate that short term prediction is feasible. While the networks are not accurate enough to be used for vegetation classification, much can be done in terms of network architecture, training parameters and the amount of data used for training. In this experiment, the network topologies are fixed, no neurons are added or removed during the training process. Using PSO to optimize the topology of the network in addition to the weights is being investigated.

REFERENCES

[1] G. Shannon, B. Page, K. Duffy, and R. Slotow, "African elephant home range and habitat selection in Pongola Game Reserve, South Africa," Submitted to African Journal of Ecology.

[2] E. W. Saad, D. V. Prokhorov, and D. C. Wunsch, "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks," *IEEE Transactions on Neural Networks*, vol.9, pp. 1456-1470, Nov. 1998.

[3] D. Srinivasan, W. H. Loo, and R. L. Cheu, "Traffic Incident Detection Using Particle Swarm Optimization," *Proc. IEEE Swarm Intelligence Symposium*, April 2003, pp. 144-151.

[4] S. H. Ling, F. H. F. Lenung, H. K. Lam, and P. K. S. Tam, "Short-Term Daily Forecasting in an Intelligent Home with GA-Based Neural Network," *Proc. IEEE International Joint Conference on Neural Networks*, May 2002, pp. 997-1001.

[5] E. A. Grimaldi, F. Grimaccia, M. Mussetta, and R. E. Zich, "PSO as an effective learning algorithm for neural networks applications," *Proc. IEEE International Conference on Computational Electromagnetics and Its Applications*, Nov. 2004, pp. 557-560.

[6] V. Gudise, and G. Venayagamoorthy, "Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithms for Neural Networks," *Proc. IEEE Swarm Intelligence Symposium*, April 2003, pp. 110-117.

[7] J. Kennedy, and R. Eberhart, "Particle swarm optimization," *Proc. IEEE International Conference on Neural Networks*, Nov/Dec 1995, pp. 1942-1948.

[8] R. Eberhart, and Y. Shi, "Particle swarm optimization: developments, applications and resources," *Proc. IEEE Congress on Evolutionary Computation*, 2001, pp. 81-86.

[9] G. Ueno, K. Yasuda, and N. Iwasaki, "Robust Adaptive Particle Swarm Optimization," *Proc. IEEE International Conference on Systems, Man and Cybernetics*, Oct. 2005, pp. 3915-3920.

[10] R. A. Krohling, "Gaussian swarm: a novel particle swarm optimization algorithm," *Proc. IEEE Conference on Cybernetics and Intelligent Systems*, Dec. 2004, pp. 372-376.

[11] Y. Shi, and R. C. Eberhart, "Empirical study of particle swarm optimization," *Proc. IEEE Congress on Evolutionary Computation*, 1999, pp. 1945-1950.

[12] Y. Shi, and R. Eberhart, "A modified particle swarm optimizer," *Proc. IEEE Congress on Evolutionary Computation*, May 1998, pp. 69-73.

[13] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol.8, pp. 240-255, June 2004.