

A Comparison between GAs and PSO in Training ANN to Model the TE Chemical Process Reactor

Malik Braik and Alaa Sheta and Amani Arieqat¹

Abstract. In this paper, the adaptation of network weights using Particle Swarm Optimization (PSO) was proposed as a mechanism to improve the performance of Artificial Neural Network (ANN) in modeling a chemical process. This is particularly useful for cases involving changing operating conditions as well as highly nonlinear processes. As a case study, a Tennessee Eastman (TE) chemical process reactor was considered. Four subsystems of the reactor were considered. They are the reactor level, the reactor pressure, the reactor cooling water temperature, and the reactor temperature. PSO is proposed to allow automatic update of network weights to increase the adaptability to dynamic environment. Comparisons were also made to training the ANN using Genetic Algorithms (GAs). The results obtained in this paper confirmed the potential of PSO-based ANN model to successfully model the TE process. The results are explored with a discussion using the Mean Square Error (MSE) and Variance-Account-For (VAF) to illustrate the usability of the proposed approach. Finally, conclusions and future works are derived.

1 Introduction

Over the years, the application of Artificial Neural Network (ANN) in process industries has been growing in acceptance. Today, there is a growing interest of using ANNs to assist building a reasonable model structure for physical nonlinear systems [25]. ANNs have a special capacity to approximate the dynamics of nonlinear systems in many applications in a black box manner. An example of dynamical nonlinear systems is a Tennessee Eastman (TE) chemical reactor process [6]. Given sufficient input-output data, ANN is able to approximate any continuous function to arbitrary accuracy. In general, the development of a good ANN model depends on several operators. The first operator is related to the input-output data driven, where model qualities are mostly influenced by the quality of data being used. The second operator is concerning with the network architecture. Different network architectures result in a different estimation performance. The third operator is the model size and its complexity. Where a small network may be not able to represent the real situation of the model estimation due to its limited capability, while a large network may has noise in the training data and hence fail to provide good generalization ability, and the last operator is related to the quality of the process model, and is strongly dependent on the network training. May be the last issue, is the most important among all, since it is essentially an identification of model parameters that fit with the given data. The work in this paper will focus on the last issue.

Until today, many researchers prefer of using gradient descent algorithms such as Back-Propagation (BP) method. Some of the advantages of this gradient-based technique include its efficient implementations, good at fine-tuning, and faster convergence when compared with other methods. However, these methods are subject to problems involving local minima-since they are local search methods and when applied to complex nonlinear optimization problems, may be sometimes result in unexpected performances. These gradient methods assess the error in the network's decision as compared to a supervisor, and propagate the error to the weights throughout the network, so that, one of the main obstacles due to the fact that searching of optimal weights is strongly dependent on initial weights, and if they are located near local minima, the algorithm would be stuck at a sub-optimal solution. So that, the conventional gradient search method is susceptible to be converged at local optima. This is however not the case for Evolutionary Algorithms (EAs) since the genetic search methods offer better chances to get to the global optima. Furthermore, since EAs does not use gradient information, it is advantageous for problems where such information is unavailable or very costly to obtain. These advantages make them more robust and appealing than many other search algorithms [7].

Several different attempts have been proposed by various researchers to propitiate this training problem. These include imposing constraints on the search space, restarting training at many random points, adjusting training parameters, and restructuring the ANN structure [25]. One of the most promising techniques is by introducing adaptation of network training using (EAs) such as Particle Swarm Optimization (PSO), and Genetic Algorithms (GAs). Unlike BP, PSO is a global search algorithm based on the principle "survival of fittest". PSO is quite suitable for problems with many local minima or problems where gradient information isn't readily available, PSO avoid trapping in a local minima, because it is not based on gradient information [1, 15].

PSO has been used to train neural networks, finds neural network architectures, tunes network learning parameters, and optimizes network weights. The global best or local best solution in PSO is only reported to the other particles in a swarm. Therefore, evolution only looks for the best solution and the swarm tends to converge to the best solution quickly and efficiently, thus increasing the probability of global convergence. In this way, the merging of EAs and ANN will gain adaptability to dynamic environment and lead to significantly better intelligent systems than relying on ANN, PSO, or GA alone. In [20], Montana and Davis reported the successful application of a GA to a relatively large ANN problem. They proved that GA produce results superior than BP method. In [32], Yao and Liu presented a new evolutionary algorithm, to evolve ANN architecture and connection weights simultaneously. When tested on a number of

¹ Information Technology Department, Prince Abdullah Bin Ghazi Faculty of Science and Information Technology, Al-Balqa Applied University, Jordan, email: {m.fjo,asheta2,amani_arieqat}@yahoo.com

benchmark problems such as medical diagnosis problems, and credit card assessment problems. In [34], Zuo and Wu used GA to determine the optimal feeding rate for a hybrid ANN model of a fermentation system.

In other words, the developed models using ANN-EAs should be more robust to dynamic nonlinear process system [33]. In this paper, GAs-based ANN and PSO-based ANN are introduced for connection weights in ANN modeling.

Following this introductory section, the rest of the paper is organized as follows: In Section II, a background of the case study is presented. In section III: the preparatory works for ANN model development such as sensitivity analysis, and model input selection are reported. Subsequently, data collection and scaling procedure for model development are described. Also, procedures of ANN model development have been discussed. This is followed in section IV: by discussion of PSO and GAs and the motivation to include evolution in ANN modeling using EAs, also, the research methodologies involved are described. In section V, The reported results are simulated and discussed, and finally, the paper closes with some concluding remarks and future research directions in Section VI.

2 Tennessee Eastman Chemical Process

Over the years, the processes involved in manufacture and production of purified chemicals have become increasingly more complex. Systems that were once controlled by operators based on the system performance have been converted to automatic control, where a mechanical or electrical controller adjusts valves and pumps [4, 17, 29]. Engineers have learned that, to develop the control algorithms for large complex systems, it is important to first create and test a model of the system offline. There are two reasons to do this in chemical plants, they are:

- Protecting humans and the environment from unsafe conditions.
- Shorten the design and analysis of the system [31].

2.1 Features of the TE Process

TE process, as presented by Downs and Vogel in [6], is based on an actual system, with slight changes made to protect the identity of the reactants and products. The system is a wellposed problem for analysis and control design of a nonlinear, open-loop unstable chemical process. The plant consists of five major operations: a two phase reactor, a product condenser, a vapor/liquid separator, a recycle compressor, and a product stripper. The nonlinear dynamics of the plant are mainly due to the chemical reactions within the reactor. A Nonlinear Predictive Control shown in Figure 1 was simulated [19]. The TE reactor process shown in this figure is borrowed from [3].

Downs and Vogel [6] proposed the following control objectives for this system:

1. Maintain process variables at desired values.
2. Keep process operating conditions within equipment constraints.
3. Minimize variability of product rate and product quality during disturbances.
4. Minimize movement of the gas valves which affect other processes.
5. Recover quickly and smoothly from disturbances, production rate changes, or product mix changes.

A suitable controller must be able to achieve at least these control objectives.

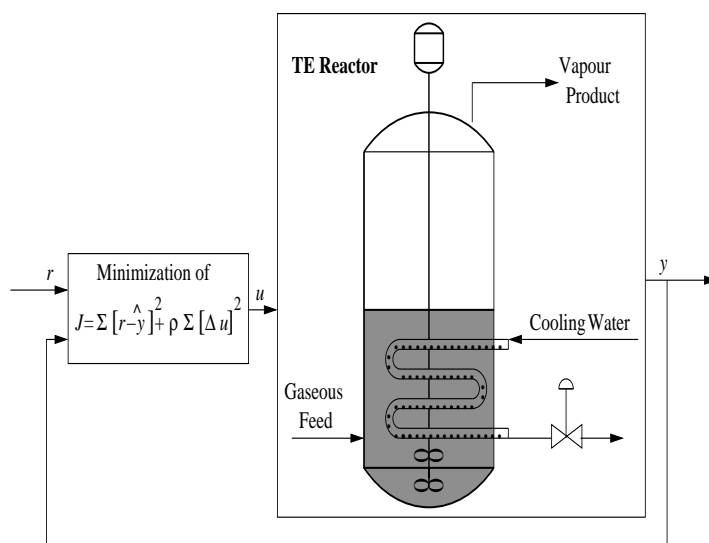


Figure 1. The Simulated Predictive Control Principle

2.2 Inputs and Outputs of the TE System

Typically, any chemical system can be represented by a set of mathematical equations that map the inputs and outputs of the system with a set of state variables. The main set of equations for normal operation of the chemical processes is the dynamic of equations associated with the reactions, unit operations, and flows of the system. The plant model has 12 manipulated variables, 50 states, and 41 measurements from different locations in the plant [12]. The 20 disturbance scenarios can be turned on and off to test the robustness of the system. Of the measured values, there are 22 values that monitored continuously. The other 19 measured values are retrieved from gas chromatographs (analyzers) that give information only at certain intervals. The three analyzers give information about chemical components in the reactor feed, the purge gas, and the product with sampling delays of 0.1 hour, 0.1 hour and 0.25 hour, respectively. The 50 states of the model closely correspond to the actual plant operation that represents the significant system dynamics. Fast dynamics, such as transmitter lags, are modeled as straight gains because the effects of these transients are minimal. The 12 manipulated variables in the TE process allow the designer 12 degrees of freedom with 9 flow valves, 2 temperature control valves, and control of the reactor agitation rate [12, 11].

3 Preparation for Model Development

Before the ANN model can be selected, some preparation stages must be completed. These tasks included: data collection, model structure selection. These preparations are necessary to provide model development.

3.1 Data Collection and Description

Dataset for training and evaluating is collected from various operating conditions to measure different outputs of the TE reactor. A total of 300 measurements were downloaded from [23]. The measurements describe the behavior of the TE reactor and show how it responds to various inputs. The dataset is split into two parts; the training dataset which is used to train the NN model, whilst a testing dataset is used to verify the accuracy of the trained NN model. The

training dataset consists of 50% of the total dataset with the testing dataset consisting of the remaining 50%.

Since the model was built using time-series data, measurements at past sampling time ($t - 1$) for input variables were also included as model input. This was due to the fact that chemical process variables were always auto-correlated. Perhaps, by including the delay measurement as a model input for current estimation, the generalization ability of a feedforward network would be improved, and hence, the performance of the network in estimating the TE process would be enhanced.

The TE reactor modeling problem is divided into four sub-problems. They are: The reactor level, the reactor pressure, the reactor cooling water temperature, and the reactor temperature. Each of the four sub-problems has four input variables [2, 29], they are:

- $u_1(k)$ stand for the flow to the reactor.
- $u_2(k)$ stand for the coolant valve position,
- $u_3(k)$ stand for the feed mole fraction, and
- the fourth input $y(k - 1)$ represents the past output variable value.

The output for each sub-problem is represented by $y(k)$.

3.2 Data Scaling

Dataset is not normally used directly in process modeling of ANN. This is due to the difference in magnitude of the process variables. The data was scaled to a fixed range to prevent unnecessary domination of certain variables, and to prevent data with larger magnitude from overriding the smaller and impede the premature learning process. The choice of range depends on transfer function of the output nodes in ANN. Typically, $[0, 1]$ for sigmoid function and $[-1, 1]$ for hyperbolic tangent function. However, due to nonlinear transfer function has asymptotic limits; the range of dataset is always set slightly less than the lower and upper limits. In this work, since the sigmoid function is adopted, the data is normalized in the range of $[0.1 - 0.9]$ using the relation in equation 1:

$$v_i^{norm} = \frac{0.8(v_i - v_i^{min})}{(v_i^{max} - v_i^{min})} \quad (1)$$

where v_i is the input variables i or the output variable to be scaled. v_i^{min} and v_i^{max} are the smallest and the largest value of the data. The choice of $[0.1 - 0.9]$ was recommended based on the facts that it eliminates some of the saturation region at both end of the sigmoid function [9].

3.3 Model Structure Selection

Selecting a model structure from a set of candidate models is a difficult problem since the TE is a nonlinear process [22, 24]. Architecture of ANN model includes type of network, number of input and output neurons, transfer function, number of hidden layers as well as number of hidden neurons. Normally, the input neurons and output neurons are problem specific. In this work, Multi-Input Single-Output (MISO) structure had been utilized to estimate the TE process; hence, there will be only one output neuron. Since the model was built using time-series data, measurements at past sampling time ($t - 1$) for input variables were also included as model input. This was due to the fact that chemical process variables were always auto-correlated. In this way, the performance of the network in estimating the TE process would be enhanced. Thus, together with current sampling time for 3 input variables, there were 8 input neurons in the input layer. In this work, the TE reactor has only one hidden layer

and 7 neurons in the hidden layer. It was utilized that an ANN with one hidden layer was sufficient in performing process mapping. Only one hidden layer with 7 neurons in the hidden layer was utilized in this work as it had proved that an ANN with one hidden layer was sufficient in performing process mapping arbitrarily [10]. From the optimum network topology, it was shown that the ANN model does not need a large number of hidden neurons. This revealed that the estimation task was not too complicated.

Also, it was important that the transfer function possessed the properties of continuity and differentiability. Commonly, log sigmoid function was utilized in the hidden layer and the output generated had a value between 0 and 1. However, the linear transfer function was more suitable in output layer. The equations for the log and linear transfer functions used in this work are shown in equations 2 and 3:

$$f(x) = \frac{1}{1 + \exp(-x)} \quad (2)$$

$$f(x) = x \quad (3)$$

4 Genetic Algorithms (GAs)

GAs belongs to a class of population-based stochastic search algorithm that is inspired from principles of natural evolution known as Evolutionary Algorithms [5]. GA is based on the principle of "survival of fittest", as in the natural phenomena of genetic inheritance and Darwinian strife for survival. GA operates on a population of individuals which represent potential solutions to a given problem. Imitating the biological principles in nature, a single individual of a population usually is affected by other individuals in its environment. Normally, the better an individual performs under these competitive conditions, has a greater chance to survive and reproduce. This in turn inherits the good parental genetic information. Hence, after several generations, the bad individual will be eliminated and better individuals will be produced. In general, GA is applicable to a wide range of optimization problems. Primarily, GA was designed to optimally solve sequential decision processes more than to perform function optimization but over the years, it has been used widely in both learning and optimization problems [30, 28]. There are two important issues in searching strategies for optimization problems: exploiting the best solution and exploring the search space [8, 27]. GA makes a balance between the exploitation and exploration of the search space. It allows the exploration of all solution space, which may reduce the converging to a local minimum. The exploitation in the neighborhood of possible solutions will perform as the high fittest solutions to be developed. Basically, the performance of weights evolution using GA depended on the number of populations and generations. If these parameters were set too low, the evolution may converge to immature solution. However, the larger number of populations and generations would require longer computation time for convergence [26].

In general, GAs used four steps to obtain the optimum connection weights of ANN:

- step 1** Generate an initial population of random weights and the corresponding ANN was constructed with those weights.
- step 2** ANN was evaluated using the population weights. This was done by computing its training error and assigning it to a fitness value according to how good the solutions are.
- step 3** Parents for genetic manipulation were selected and a new population of weights were created.

step i The best existing weights (reproduction) were copied.

step ii New weights were created by crossover and mutation operators.

step 4 The best population of weights that appeared in any generation was designated as the result of the performed GA, for the weights evolved using GA, the number of generation was used to stop the iteration.

In this work, 50 populations of weights were evolved for 50 generations. The performances in the validation sets were considered in the acceptable level; this proved that this scheme was adequate with a sufficient accuracy.

5 Particle Swarm Optimization (PSO)

PSO is a global optimization technique that has been developed by Eberhart and Kennedy in 1995 [13, 14], the underlying motivation of PSO algorithm was the social behavior observable in nature, such as flocks of birds and schools of fish in order to guide swarms of particles towards the most promising regions of the search space. PSO exhibits a good performance in finding solutions to static optimization problems. It exploits a population of individuals to synchronously probe promising regions of the search space. In this context, the population is called a swarm and the individuals (i.e. the search points) are referred to as particles. Each particle in the swarm represents a candidate solution to the optimization problem. In a PSO system, each particle moves with an adaptable velocity through the search space, adjusting its position in the search space according to own experience and that of neighboring particles, then it retains a memory of the best position it ever encountered, a particle therefore makes use of the best position encountered by itself and the best position of neighbors to position itself towards the global minimum. The effect is that particles "fly" towards the global minimum, while still searching a wide area around the best solution [15, 21, 16]. The performance of each particle (i.e. the "closeness" of a particle to the global minimum) is measured according to a predefined fitness function which is related to the problem being solved. For the purposes of this research, a particle represents the weight vector of NNs, including biases. The dimension of the search space is therefore the total number of weights and biases.

The iterative approach of PSO can be described by the following steps:

step 1 Initialize a population size, positions and velocities of agents, and the number of weights and biases.

step 2 The current best fitness achieved by particle p is set as $pbest$. The $pbest$ with best value is set as $gbest$ and this value is stored.

step 3 Evaluate the desired optimization fitness function fp for each particle as the Mean Square Error (MSE) over a given data set.

step 4 Compare the evaluated fitness value fp of each particle with its $pbest$ value. If $fp < pbest$ then $pbest = fp$ and $best_{xp} = xp$, xp is the current coordinates of particle p , and $best_{xp}$ is the coordinates corresponding to particle p 's best fitness so far.

step 5 The objective function value is calculated for new positions of each particle. If a better position is achieved by an agent, $pbest$ value is replaced by the current value. As in Step 1, $gbest$ value is selected among $pbest$ values. If the new $gbest$ value is better than previous $gbest$ value, the $gbest$ value is replaced by the current $gbest$ value and this value is stored. if $fp < gbest$ then $gbest = p$, where $gbest$ is the particle having the overall best fitness over all particles in the swarm.

step 6 Change the velocity and location of the particle according to Equations 4 and 5, respectively [13, 18].

step 7 Fly each particle p according to Equation 5.

step 8 If the maximum number of a predetermined iterations (epochs) is exceeded, then stop; otherwise Loop to step 3 until convergence. In this work, 25 populations of weights were evolved for 200 generations.

$$V_i = wV_{i-1} + acc * rand() * (best_{xp} - xp) + acc * rand() * (best_{xgbest} - xp) \quad (4)$$

Where acc is the acceleration constant that controls how far particles fly from one another, and $rand$ returns a uniform random number between 0 and 1.

$$xp = xpp + V_i \quad (5)$$

V_i is the current velocity, V_{i-1} is the previous velocity, xp is the present location of the particle, xpp is the previous location of the particle, and i is the particle index. In step 5 the coordinates $best_{xp}$ and $best_{xgbest}$ are used to pull the particles towards the global minimum.

5.1 Tuning Parameters for GAs and PSO

To develop an accurate process model using ANN, the training, and validation processes are among the important steps. In the training process, a set of input-output patterns is repeated to the ANN. From that, weights of all the interconnections between neurons are adjusted until the specified input yields the desired output. Through these activities, the ANN learns the correct input-output response behavior. The model training stage includes choosing a criterion of fit (MSE) and an iterative search algorithm to find the network parameters that minimizes the criterion. PSO as well as GAs are used in an effort to formalize a systematic approach to training ANN, and to insure creation of a valid model. They are used to perform global search algorithms to update the weights and biases of neural network. The combinations of control parameters used for running PSO and GAs are shown in Table 1 and Table 2 respectively:

Table 1. The Control Parameters used for Running PSO

Parameter	Value
Number of population	50
Number of generations	2000
Learning factors	1.3
Inertia weights	0.6 and 0.9
Fitness	MSE

Table 2. The Control Parameters used for Running GAs

Parameter	Value
Number of population	50
Number of generations	2000
Crossover probability	0.9
Mutation probability	0.03
Selection function	Ranking
Fitness	MSE

For the validation process, the expected model is compared graphically with the behavior of the target model.

6 Training and Validation Results

The performance of the proposed models in tracking the actual process data during the training and validation testing stages of the reactor level is illustrated in Figures 2,3, respectively. The modeling process of the reactor Level is shown as a sample, in overall, all estimator models display good performance in the training and validation sets indicating that the models developed are able to represent the behavior of the TE process.

The final convergence value of the proposed models reached 0. It can be seen from Figure 4 that the convergence process results in smooth curves, with a rapid increase at the start that gradually slows down. The experiments were implemented five times to ensure that MSE converges to a minimum value. In the reactor level problem, the MSE error converges to a value of 0.00034721 using PSO, and to 0.00038193 using GAs, while in the reactor pressure problem, the MSE converges to a value of 0.0001550 using PSO, and to 0.0001705 using GAs. In the other models, the MSE convergence value is nearly 0.

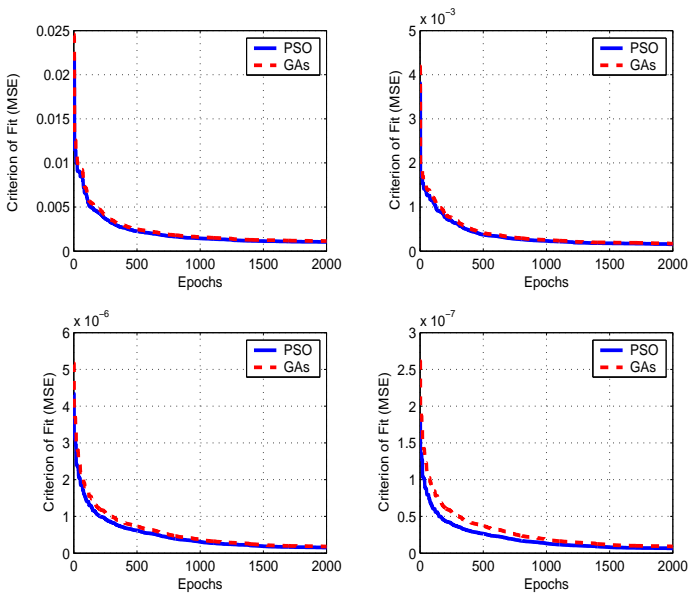


Figure 4. Convergence Curves of TE sub-problems; upper left-Reactor Level; upper right-Reactor Pressure; lower left-Reactor Cooling Temperature; lower right-Reactor Temperature

6.1 Evaluation Criteria

The performance of the proposed approach is evaluated by measuring the estimation accuracy. The estimation accuracy can be defined as the difference between the actual and estimated values. The first typical fitting criterion (MSE) is defined as in Equation 6:

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (6)$$

where N is the total number of data, y is actual target value, and \hat{y} its estimated target value.

Since the calculation started with random initial weights, each run produced different results even though the network architecture was

maintained. Thus, in order to obtain an optimal solution, tedious approaches such as repeating runs for the same topology cannot be avoided. The experiments are implemented many times to ensure that MSE converges to a minimum value. It was found that the TE reactor composition could be best estimated using a network with 7 hidden neurons. Using this network topology, the training and validation errors of the models developed based MSE are tabulated in Table 3.

Table 3. Training and Validation Results of The Estimator Models

problem	Train-PSO	Valid-PSO	Train-GAs	Valid-GAs
1	1.9321e-1	3.7736e-2	2.0820e-1	0.8226e-1
2	4.2778e-3	4.0076e-3	6.5994e-3	5.1018e-3
3	3.3147e-6	7.6189e-7	1.0179e-4	2.2881e-5
4	5.1391e-6	6.0090e-6	4.7434e-5	5.3340e-5

The second criterion fit, Variance-Account-For (VAF) is used to test the ability of the proposed approach to model the TE reactor in recall or testing phase, VAF is given in Equation 7.

$$VAF = 1 - \frac{var(y - \hat{y})}{var(y)} \times 100\% \quad (7)$$

where, $y(t)$ is the actual system output, and $\hat{y}(t)$ is the predicted ANN output.

The results of modeling the TE reactor using ANN-PSO and ANN-GAs based on the VAF criterion are computed and reported in Table 4.

Table 4. VAF Values of the Proposed Models

Sub-problem	ANNs-PSO	ANNs-GAs
Reactor Level	90.9445	87.3228
Reactor Pressure	91.9994	89.8116
Reactor Cooling Temperature	99.7988	98.3863
Reactor Temperature	99.9944	99.7300

From the results shown in Tables 3, 4, when the comparison of training and validation performances was made between ANN-PSO and ANN-GAs model, ANN-PSO can perform better, meaning that PSO is very effective in training the NNs, learned to successfully navigate the course on the majority of test runs, and often reached an optimal MSE. ANN-GAs is not effective in modeling TE process; this is due to that population of weights for ANN-GA may not be able to reach a global optimum when the evolution was simulated over the same number of generations as with ANN-PSO model. The higher number of generations may be used but this is not recommended due to longer convergence time.

7 Conclusions and Future Work

This work has proposed an enhancement to neural network model. Aiming to improve the model robustness, evolution in network connection weights using EAs was explored. Based on the results obtained in this study, the main conclusions of this paper are: ANN is an efficient and effective empirical modeling tool for estimating the chemical process variable by using other easily available process measurements and the use of multilayer feedforward network with delay values in model input variables are sufficient to give estimation to any arbitrary accuracy. Also, this paper has taken advantage

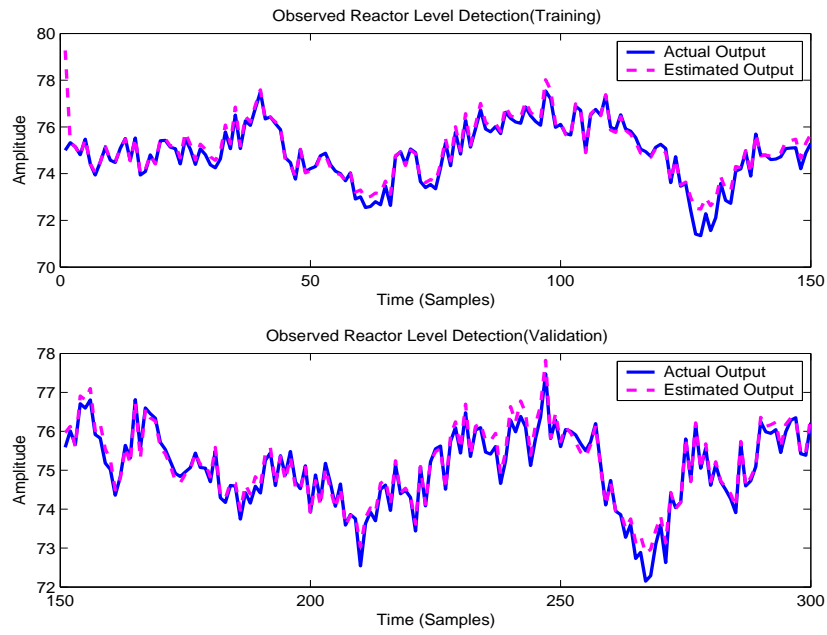


Figure 2. Observed Reactor Level: Training and Validation Performance of ANN-PSO Model

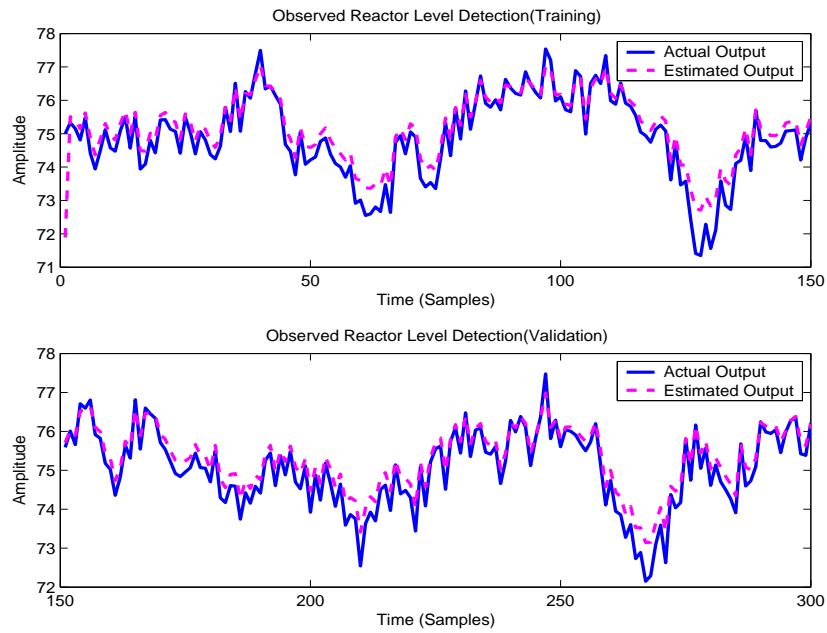


Figure 3. Observed Reactor Level: Training and Validation Performance of ANN-GA Model

of flexibility EAs techniques by applying it to the problem of training neural networks. In particular the global optimization method PSO is employed to provide a sense of the directivities of optimization the weights and biases of neural networks, and seek a good starting weight vector for subsequent neural networks learning algorithm. The preliminary results give a positive indication of the potential offered by EAs; this ensures the ability to effectively train the neural networks using the optimization techniques. In addition, PSO offered an increased level of adaptability of neural networks and is more preferable as the optimal solution searching to model a TE reactor problem than GAs. Despite the encouraging finding was obtained, there are still several further works to be considered. These include: The inclusion of adaptive feature using EAs to improve model robustness can be extended to evolution of a network architecture, which is typically number of hidden nodes as well as number of hidden layers; also the evolution of network architecture requires new set of connection weights.

Acknowledgements

Authors would like to acknowledge the financial support of Al-Balqa Applied University, Al-Salt, Jordan.

REFERENCES

- [1] Hussein A. Abbass, Ruhul Sarker, and Charles Newton, 'PDE: A Pareto-frontier Differential Evolution Approach for Multi-objective Optimization Problems', in *Proceedings of the Congress on Evolutionary Computation 2001 (CEC'2001)*, volume 2, pp. 971–978, Piscataway, New Jersey, (May 2001). IEEE Service Center.
- [2] Heba Al-Hiary and Alaa Sheta, 'A new neural networks model for industrial processes', in *4th International Multiconference on Computer Science and Information Technology CSIT*, volume 2, pp. 189–198, (2006).
- [3] I.Paul Barton and S.Wade Martinsen, 'Equation-oriented simulator training', in *Proceedings of the American Control Conference, Albuquerque, New Mexico*, pp. 2960–2965, (1997).
- [4] N. Bhat and T. J. McAvoy, 'Use of neural nets for dynamic modelling and control of chemical process systems', *Computers & Chemical Engineering*, **14**(4), 573–583, (1990).
- [5] M. Choenaer and Z. Michalewicz, 'Evolutionary computation control and cybernetics', *Proceedings of the IEEE*, **26**(3), 307–338, (1997).
- [6] J. J. Downs and E. F. Vogel, 'A plant-wide industrial process control problem', *Computers Chemical Engineering*, **17**, 245–255, (1993).
- [7] D.B. Fogel, 'An introduction to simulated evolutionary optimization', *IEEE Transactions on Neural Networks*, **5**(1), 3–14, (1994).
- [8] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design*, United States of America: John Wiley & Son, Inc., 1997.
- [9] M.H. Hassoun, *Fundamentals of Artificial Neural Networks*, MIT Press, Cambridge, MA., 1995.
- [10] K. J. Hornik, D. Stinchcombe, and H. White, 'Multilayer feedforward networks are universal approximators', *Neural Networks*, **2**(5), 359–366, (1989).
- [11] J.C. Hoskins and D. M. Himmelblau, 'Artificial neural network models of knowledge representation in chemical engineering', *Computers & Chemical Engineering*, **12**(9), 881–890, (1988).
- [12] T. Jockenhovel, L. T. Biegler, and A. Wachter, 'Dynamic optimization of the tennessee eastman process using the opt-control centre', in *Proceedings of the IEEE*, (2003).
- [13] J. Kennedy and R. C. Eberhart, 'Particle swarm optimization', *Proceedings of IEEE International Conference on Neural Networks (Perth, Australia)*, *IEEE Service Center, Piscataway, NJ*, **5**(3), 1942–1948, (1995).
- [14] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*, Morgan Kaufmann Publishers, San Francisco, 2001.
- [15] R. Kiran, S. R. Jetti, and G. K. Venayagamoorthy, 'Online training of generalized neuron with particle swarm optimization', in *International Joint Conference on Neural Networks, IJCNN 06, Vancouver, BC, Canada*, pp. 5088–5095. Institute of Electrical and Electronics Engineers, (2006).
- [16] N. Kwok, D. Liu, and K. Tan, 'An empirical study on the setting of control coefficient in particle swarm optimization', in *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2006), Vancouver, BC, Canada*, pp. 3165–3172, Vancouver, BC, Canada, (16-21 July 2006). IEEE Press.
- [17] Ungar L.H., E. J. Hartman, J. D. Keeler, and G. M. Martin, 'Process modeling and control using neural networks', *AIChE Symposium Series*, **92**, 57–67, (1990).
- [18] Jiann-Horng Lin and Ting-Yu Cheng, 'Dynamic clustering using support vector learning with particle swarm optimization', in *Proceedings of the 18th International Conference on Systems Engineering*, pp. 218–223, (2005).
- [19] M.Norgaard, O.Ravn, Poulsen, and L.K.Hansen, *Neural Networks for Modelling and Control of Dynamic Systems*, Springer, London, 2000.
- [20] D. Montana and L. Davis, 'Training feedforward neural networks using genetic algorithms', in *Proceedings of Eleventh International Joint Conference on Artificial Intelligence*, pp. 762–767, (1989).
- [21] T.J. Richer and T.M. Blackwell, 'When is a swarm necessary?', in *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, eds., Gary G. Yen, Simon M. Lucas, Gary Fogel, Graham Kendall, Ralf Salomon, Byoung-Tak Zhang, Carlos A. Coello Coello, and Thomas Philip Runarsson, pp. 1469–1476, Vancouver, BC, Canada, (16-21 July 2006). IEEE Press.
- [22] N. L. Ricker, 'Nonlinear model predictive control of the tennessee eastman challenge process', *Computers and Chemical Engineering*, **19**(9), 961–981, (1995).
- [23] N. L. Ricker, 'Nonlinear modeling and state estimation of the tennessee eastman challenge process', *Computers and Chemical Engineering*, **19**(9), 983–1005, (1995).
- [24] N. L. Ricker, 'Optimal steady state operation of the tennessee eastman challenge process', *Computers and Chemical Engineering*, **19**(9), 949–959, (1995).
- [25] R.S. Sexton, R.E. Dorsey, and N.A. Sikander, 'Simultaneous optimization of neural network function and architecture algorithm', in *Decision Support Systems*, pp. 1034–1047. IEEE, (2002).
- [26] A. Sheta and K. Eghneem, 'Training artificial neural networks using genetic algorithms to predict the price of the general index for amman stock exchange', in *Midwest Artificial Intelligence and Cognitive Science Conference, DePaul University, Chicago, IL, USA*, volume 92, pp. 7–13, (2007).
- [27] A. Sheta, H. Turabieh, and P.Vasant, 'Hybrid optimization genetic algorithms (HOGA) with interactive evolution to solve constraint optimization problems', *International Journal of Computational Science*, **1**(4), 395–406, (2007).
- [28] Alaa Sheta and H. Turabieh, 'A comparison between genetic algorithms and sequential quadratic programming in solving constrained optimization problems', *ICGST International Journal on Artificial Intelligence and Machine Learning (AIML)*, **6**(1), 67–74, (2006).
- [29] Alaa F. Sheta, *Modeling the Tennessee Eastman Chemical Reactor Using Fuzzy Logic*, volume 181, ISE Book Series on Fuzzy System Engineering-Theory and Practice, published by Nova Science, 2005.
- [30] Chen Wah Sit, *Application of Artificial Neural Network-Genetic Algorithm In Inferential Estimation And Control of A Distillation Column*, Ph.D. dissertation, Universiti Teknologi Malaysia, 2005.
- [31] John C. Sozio, *Intelligent Parameter Adaptation for Chemical Processes*, Ph.D. dissertation, Electrical Engineering Department, Virginia Polytechnic Institute, 1999.
- [32] X. Yao, 'Global optimization by evolutionary algorithms', in *Proceedings of the 2nd AIZU International Symposium on Parallel Algorithms / Architecture Synthesis*, pp. 282–291, (1997).
- [33] X. Yao, *Evolving artificial neural networks*, 1999.
- [34] K. Zuo and W.T. Wu, 'Semi-realtime optimization and control of a fed-batch fermentation system', in *Computers & Chemical Engineering*, volume 24, pp. 1105–1109, (2000).