AWRCP-- 99002639

INTEGRATED EVOLUTIONARY COMPUTATION NEURAL NETWORK QUALITY CONTROLLER FOR AUTOMATED SYSTEMS

Sanjukta Patro and William J. Kolarik
Department of Industrial Engineering
Texas Tech University
Lubbock, TX 79409-3061
Phone: (806) 742-3543
E-mail: aiwsp@ttacs.ttu.edu

ABSTRACT

With increasing competition in the global market, more and more stringent quality standards and specifications are being demanded at lower costs. Manufacturing applications of computing power are becoming more common. The application of neural networks to identification and control of dynamic processes has been discussed. The limitations of using neural networks for control purposes has been pointed out and a different technique, evolutionary computation, has been discussed. The results of identifying and controlling an unstable, dynamic process using evolutionary computation methods has been presented. A framework for an integrated system, using both neural networks and evolutionary computation, has been proposed to identify the process and then control the product quality, in a dynamic, multivariable system, in real-time.

Key words: Evolutionary computation, Neural networks, Automated systems, Quality controller

INTRODUCTION

Current times are experiencing a revolution in the national and international manufacturing quality area. Quality has become a competitive weapon among companies. Consumer protection legislation and product liability legislation are demanding increasingly stringent standards and specifications at low costs. Contractors are demanding the same stringent standards from suppliers. Quality Control in general and automatic quality control in particular are assuming major importance in modern society as technological systems are becoming increasingly complex and highly interconnected. Manufacturing processes are increasingly being automated. This has to be matched by the quality and speed of source inspection. Thus, there is a growing demand for integrated dimensional control in real-time. Rapid changes are taking place in technologies relating to sensors, control vision, and intelligent quality control. Newly developed distributed processing systems make it feasible to handle data in real-time and relay immediate corrective feedback to the manufacturing process. A key function of such a system is process monitoring. The system must continuously monitor the manufacturing process by recording and analyzing, in real-time, the inspection data in order to allow intelligent feedback decisions.

Classical control methods work well for controlling linear, single input, single output systems; however many real-life control problems are complex nonlinear, multiple input, multiple output systems. Classical methods are unsuitable for the control of such systems. Modern manufacturing processes involve complex machinery with multiple control settings, highly nonlinear dynamics, increased flexibility due to parts with tailored geometries and various material types. The cost-effective control of such processes requires adaptability. Learning control techniques provide a flexible capability for designing and building adaptable, intelligent process controllers capable of improving control of complex manufacturing processes.

NEURAL NETWORKS

Most process control models are developed by fitting a model to a set of input/output data. These models are usually linear time invariant models. Neural networks

DISCLAIMER

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

are known to be effective for the identification of nonlinear dynamic systems. Universal function approximation, resistance to noisy or missing data, and accommodation of multiple nonlinear variables, are some of the many attractive properties of neural networks. Neural networks have been successfully tested for process model identification (Haesloop and Holt, 1990). Although many attempts have also been made to use neural networks for control (Chinnam and Kolarik, 1994; Narendra and Mukhopadhyay, 1994), several shortcomings limit their applications. Backpropagation, a commonly used learning algorithm, requires gradient information about the search space. For some problem domains, such as reinforcement learning, gradient information is unavailable or costly to obtain (Whitley, et al., 1993). Most networks in supervised learning use feed-forward networks with sigmoidal transfer functions or radial basis functions. These choices make gradient information relatively easy to obtain, but if more complex transfer neurodes, such as product neurodes, are used, or if fully recurrent networks are trained, then computing gradient information becomes far more costly (Schaffer, et al., 1992). The backpropagation learning algorithm cannot be used for direct control of a nonlinear process with an unknown model since the process dynamics are unknown and cannot be used to generate the desired partial derivative (Narendra and Parthasarathy, 1990). Even if the process model has been identified, a neural network controller will risk convergence to local optimal solutions, because, during the weight optimization, the search begins with a single point in the search space and therefore, has a high risk of getting stuck at a local minima or maxima. This "suboptimization" will not be a drawback if the solution space is unimodal but advanced manufacturing systems are typically complex, resulting in the solution space being multimodal. Hence, using a neural network to control such processes will not always result in efficient control. Another approach to implementing neural networks for process control resides in the category of inverse control. The inverse method uses the system state as the input vector to the network and the control signal of the plant as the function being learned. The network trains on the system input-output pairs until it learns to map the inverse plant dynamics. Once the network has been successfully trained to mimic the plant inverse, it can be used to predict the appropriate control signal from the measured plant states and the desired plant state. A major limitation of plant inverse identification occurs when the plant inverse is not uniquely defined. This occurs for a plant when more than one input vector u exists that corresponds to one output vector y. In such cases, the neural network modeling the plant inverse attempts to map a single input y to one of the many target responses $u_1, u_2, \ldots$ It may be that the eventual mapping learned would somewhat tend to average the more than one desired u values (Zurada, 1992).

EVOLUTIONARY COMPUTATION

It has been recognized that intelligence displayed by living creatures is a result of evolution and therefore, it would be prudent to model the evolutionary process in order to create entities capable of intelligent behavior. The term used for such simulation of evolution on a computer is Evolutionary Computation. Evolutionary computation methods are implemented as a population based search over a fitness response surface. Three decades of research in this area have shown that modeling the search process of natural evolution can yield very robust, direct computer algorithms. Evolutionary algorithms are based on the collective learning processes within a population of individuals, each of which represents a point in the space of potential solutions to a given problem. Emulating the optimization process inherent in natural selection results in iteratively improving solution quality. The rate of improvement depends on the shape of the response surface. Many studies have shown that the procedures generally converge to near optimal solutions despite difficult-to-optimize response surfaces. These stochastic optimization techniques often out-perform classical methods of optimization when applied to real-world problems (Fogel, 1995). Conventional optimization techniques process a single point of the search space and therefore, have a high risk of getting stuck at a local minima or maxima. In comparison, evolutionary techniques maintain a population of solutions, allowing a wider coverage of the solution space. They therefore, have a greater chance of finding the global optimum.

The implementation of simulated evolutionary algorithms begins with an arbitrarily chosen initial population of solutions which is allowed to evolve toward better and better regions of the search space through the randomized processes of selection, mutation, and recombination. The fitness of each trial solution is assessed by an objective measure of performance, and the selection process favors those individuals of higher fitness to reproduce more often than those of lower fitness. Parental information is passed on to the descendants through recombination, mutation introduces innovation into the population. Evolutionary computation encompasses three broadly

similar avenues of investigation in simulated evolution: Genetic Algorithms, Strategic Programming and Evolutionary Programming. The differences between the procedures are characterized by the methods employed for selecting new parents, and by the types of alterations that are imposed on solutions to create offspring.

Evolutionary computation methods require little *a priori* information and they are robust (Fogel, 1995). They belong to the class of probabilistic algorithms, yet they are different from random algorithms as they combine elements of both directed and stochastic searches. Many advances have been made in the design of adaptive controllers for linear systems but such controllers cannot be used for the global control of nonlinear systems. When applied to optimal control problems, evolutionary methods require no simplifying assumptions of linearity, continuity, etc. and they have therefore, great potential for addressing engineering problems, especially those that have resisted solution by classical techniques. Past applications of evolutionary techniques for control involved defining the controller structure and/or tuning controller parameters. For example Filipic and Juricic (1993) used genetic algorithms to perform optimization of a controller with a predefined structure. In this paper, we propose utilizing the optimization ability of evolutionary computation to generate the optimal control inputs by using the neural network process model to provide estimates of objective values for the individual control actions.

EXAMPLE APPLICATION

The evolutionary programming technique has been applied to control the cart-pole system, a dynamic, unstable system, in real-time[1]. Basically, the cart and pole can be considered as a trolley that moves in the plane of a track and upon which is mounted a hinged inverted pendulum. The configuration is shown in Figure 1. The control goal is to keep the pole as near the vertical as possible, and the cart within the bounds of the track. A bang-bang control of either a right or left force of 1N was applied.

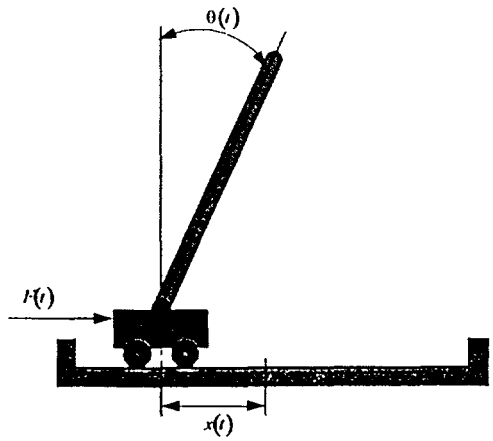

Figure 1. Configuration of the cart-pole system

The dynamics of the cart-pole system are:

$$\theta''(t) = \frac{(m_p + m_c)g\sin(\theta(t)) - \cos(\theta(t))[F(t) + m_p l(\theta'(t))^2 \sin(\theta(t))]}{(4/3)(m_p + m_c)l - m_p l\cos^2(\theta(t))}$$

$$x''(t) = \frac{F(t) + m_p l[(\theta'(t))^2 \sin(\theta(t)) - \theta''(t)\cos(\theta(t))]}{(m_p + m_c)}$$

where, $\theta(t)$ = pole angle with vertical, at time t (radians)
$\theta'(t)$ = angular velocity (radians /sec)
$x(t)$ = cart position at time t (meters)
$x'(t)$ = cart velocity (meters/sec)
$F(t)$ = force applied to cart at time t (Newtons)
m_c = mass of the cart (1.0 Kg)
m_p = mass of the pole (0.1 Kg)
l = length of the pole (0.5 meters)
g = gravitational constant (9.8 meters/sec^2)

For the evolutionary control, the cart and pole system model was represented as:

$$A_1(q)y_1(t) = B_1(q)u(t) + C_1(q)e_1(t) + D_1(q)y_2(t)$$

$$A_2(q)y_2(t) = B_2(q)u(t) + C_2(q)e_2(t) + D_2(q)y_1(t)$$

where

$$A_1(q), A_2(q), B_1(q), B_2(q), C_1(q), C_2(q), D_1(q),$$

and $D_2(q)$ are polynomials in a shift operator q^{-1}, $y_1(t)$ and $y_2(t)$ represent the pole and cart positions respectively, $u(t)$ represents the single-force input to the system, and $e_1(t)$ and $e_2(t)$ represent discrepancies between the true system and the hypothesized models.

[1] Control of the cart-pole system using evolutionary techniques has been attempted before (see Fogel, 1995) using different cart-pole dynamics. Some notations have been borrowed from Fogel, 1995.

A population of 50 models was randomly constructed. Each individual model was evaluated based on how well it fit past data. A failure of the system was said to have occurred if either the pole angle exceeded ± 12 degrees or the cart hit one of the ends of the track (± 2.4 meters). The fittest individual models were allowed to reproduce. The best individual in each generation was used to generate the control action for the next time step. The simulation was allowed to continue for 2000 time steps (0.02 sec. each). The evolutionary technique was able to keep the cart and pole system from going out of control for the entire simulation. During the simulation, the cart position did not deviate more than 0.003 meters from the center and the pole did not deviate more than 0.004 radians. The results for a typical set of 100 time steps is presented in Figure 2.

These results, although promising, showed that evolutionary methods are very expensive in terms of computation time, especially if used for both identification and control of a system. As noted earlier, using neural networks for such purposes has its own limitations. Instead of using either a neural network or evolutionary technique, to both identify the process and control the quality, it seems reasonable to combine the two. A framework for identification and control of a dynamic, multivariable process has been proposed, that will retain the neural network identification, but will have an evolutionary quality controller. The main idea in integrating neural networks with evolutionary programming is to use their strengths collectively and at the same time overcome any limitations.

PROTOTYPE DESCRIPTION

The prototype addresses zero defects product quality by monitoring and modifying, in real-time, the production process itself. It consists of a neural network identifier to extract the relationships between the production variables and the quality measures and an evolutionary algorithm to generate the control action (in order to achieve the target quality requirements). Figure 3 shows the schematic system overview and Figure 4 shows the evolutionary quality controller in more detail.

The target quality parameters are obtained by converting the customers' demands into technical specifications and requirements within the existing equipment capability. The starting weights for the neural network identifier are obtained by performing an evolutionary search. From then on, the network works alone to identify the process. All known and measurable inputs (controllable and uncontrollable variables) to the production process are also provided to the neural network identifier.

From the given inputs, the network generates a set of outputs. The network weights are adjusted, in real-time, to minimize the error between the network's predicted output and the output from the actual production process. The identified process model (in the form of the neural network) is then sent to the evolutionary controller which performs an evolutionary search to generate the best control action. The control design problem consists of finding the best set of control parameters that will result in the production process

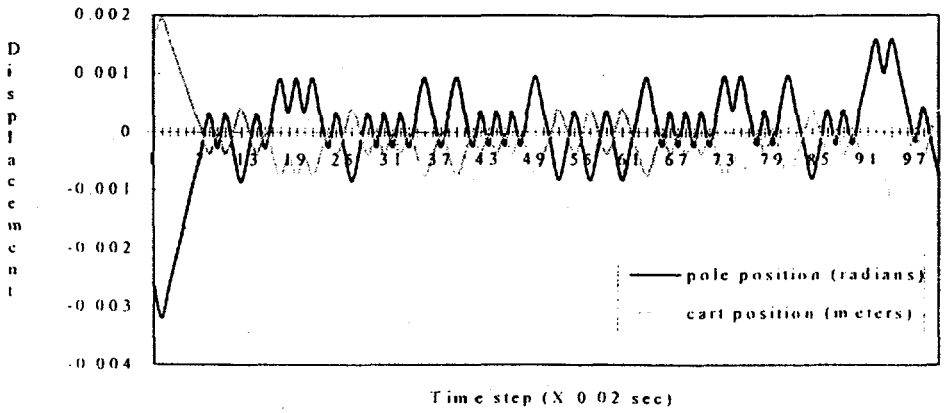

Figure 2. Control of the Cart-Pole System for a typical 100 steps

generating the required closed-loop quality control performance. The control goal is to minimize the error between the output predicted by the neural network process model (using the best control action) and the target product quality parameters.

Specifically, the technology involves three broad stages: (A) Evolutionary weight search stage, (B) Process identification stage, and (C) Quality control stage. Stage A, evolutionary weight search, is only performed once, initially. After finding the initial weights for the identifier, stages B and C are completed in real-time. Before beginning stage A, sets of input and output data are obtained from the process to form the training and test data set for the process identifier.

Stage A

1. Create an initial population of network weight matrices (coded as the genotypes).
2. Convert each genotype matrix to the phenotype (network architecture) and train it.
3. Evaluate each individual on the test set data and assign fitness.
4. Select parents for the next generation, based on the fitnesses.
5. Perform crossover and mutation to generate offspring.
6. Repeat steps 2-5 until the stop (tolerance) criterion is reached.

Stage B

1. Connect the identifier network to the actual production process to begin on-line process identification.
2. Input the identifier network with the required past data and the measurable inputs.
3. Compare the predicted outputs from the identifier network with the actual outputs from the production process and calculate the error.
4. Backpropagate the error and update the network weights.

Stage C

1. Copy the process identification model from the previous stage into the evolutionary controller.
2. Generate an initial population of parent control action vectors.
3. Generate offspring control vectors by performing recombination and mutation on the parents.
4. Use each individual control vector, along with the past data, as input to the process model and perform feedforward of the error. The identification model network weights remain unchanged.
5. Compare each individual's predicted output with the target quality requirements and assign fitness.
6. Use random competition (as suggested by Fogel, 1995) to rank the controller organisms.
7. Select parents and repeat steps 3-5 until stop criterion is reached.
8. Apply the control action vector with the highest fitness generated so far as the new control parameters on the production process and go to the beginning of stage B.

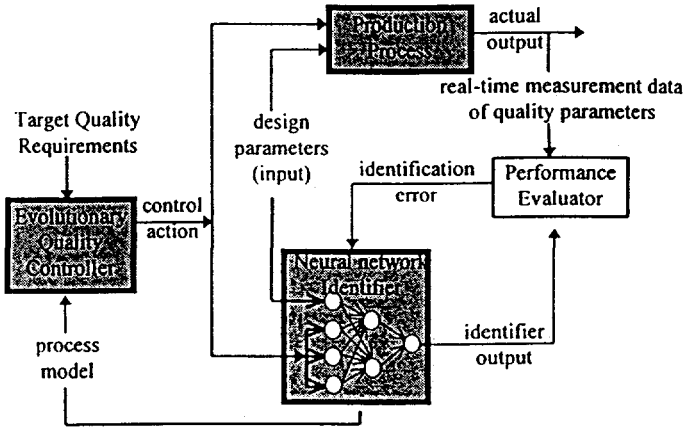

Figure 3. A schematic overview of the control

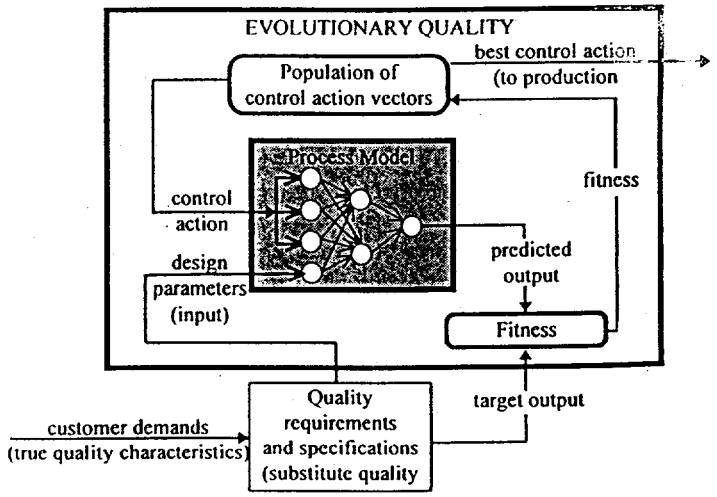

Figure 4. Evolutionary generation of the control action

As mentioned earlier, evolutionary techniques can, depending on the problem, become computationally expensive. To overcome this limitation, the proposed evolutionary optimizer will utilize heuristics like dynamic parameter encoding and controlled offspring generation in order to increase its efficiency. Dynamic parameter encoding involves adaptively controlling the

mapping from fixed-length binary codes to real values. By doing this, individuals will cover more of the search space in the beginning and will gradually concentrate more on promising regions than on less promising regions. Controlled offspring generation involves allowing certain individuals in the population to have more offspring than others. For example, more fit parents being allowed to produce more offspring will allow better solutions to have a higher survival probability.

CONCLUSIONS

The complexity of a dynamical system model and demanding closed loop system performance requirements, necessitate the use of sophisticated controllers. Based on our limited research to date, it appears that the integration of neural networks with evolutionary computation methods, for dynamic, multivariable process quality control, is feasible and enables each to overcome most of the other's limitation, thus resulting in an efficient and more robust quality control of a product or process. This paper proposed a framework of one such integrated process.

ACKNOWLEDGMENTS

The research work described in this paper is funded, in part, through the Texas Advanced Technology Program, Quality Controllers for Automated Systems Project (#003644-082).

REFERENCES

Chinnam, R. B. and Kolarik, W. J. (1994), "Quality controller for automated systems," *Proceedings of the 48th Annual Quality Congress*, ASQC, pp. 430 - 438.

Filipic, B. and Juricic, D. (1993), "An interactive genetic algorithm for controller parameter optimization," *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms*, pp. 458-462.

Fogel, D. B. (1995), *Evolutionary Computation--Toward a New Philosophy of Machine Intelligence*, IEEE Press.

Hacsloop, D. and Holt, B. R. (1990), "Neural networks for process identification," *International Joint Conference on Neural Networks*, vol. 3, pp. III-429 - III-434.

Narendra, K. S. and Mukhopadhyay, S. (1994), "Adaptive control of nonlinear multivariable systems using neural networks," *Neural Networks*, vol. 7, no. 5, pp. 737 - 752.

Narendra, K. S. and Parthasarathy, K. (1990), "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4-27.

Schaffer, J. D., Whitley, D., and Eshelman, L. J. (1992), "Combinations of genetic algorithms and neural networks: a survey of the state of the art," *Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks*.

Whitley, D., Dominic, S., and Das, R. (1993), "Genetic reinforcement learning for neurocontrol problems," *Machine Learning*, vol. 13, pp. 259 - 284.

Zurada, J. M. (1992), *Introduction to Artificial Neural Systems*, West Publishing Company.

BIOGRAPHICAL SKETCHES

SANJUKTA PATRO is a graduate student in the Industrial Engineering Department of Texas Tech University. She holds a BS degree in Mechanical Engineering from Andhra University, India, and an MS degree in Industrial Engineering from Texas Tech University. She has two years of experience in production planning and control. Her special interests are in intelligent control systems for process and quality control. Ms. Patro is an active member of IIE, ASQC, and SME.

WILLIAM KOLARIK is a professor in the Industrial Engineering Department at Texas Tech University. He currently serves as the director for the Center for Applied Research in Industrial Automation and Robotics. He has work experience in the manufacturing, service, and agricultural sectors. Dr. Kolarik has over 10 years experience in academic research and instruction, primarily in the areas of quality, reliability, and experimental design. He has published over 50 technical papers in the quality and reliability area. Based on his experiences in academic research and instruction, as well as professional practice. Dr. Kolarik is the author of Creating Quality, McGraw Hill, 1995.