Adaptive Critics for Dynamic Particle Swarm Optimization

Ganesh K. Venayagamoorthy
Missouri University of Science and Technology

Proceedings of the 2004 IEEE
International Symposium on Intelligent Control
Taipei, Taiwan, September 2-4, 2004

Adaptive Critics for Dynamic Particle Swarm Optimization

Ganesh K Venayagamoorthy, *Senior Member, IEEE*

Department of Electrical and Computer Engineering
University of Missouri-Rolla
132 Emerson Electric Co. Hall, Rolla, MO 65409, USA
Email: gkumar@ieee.org

Abstract—**This paper introduces a novel technique for Dynamic Particle Swarm Optimization (DPSO) using adaptive critic designs. The adaptation between global and local search in an optimization algorithm is critical for optimization problems especially in a dynamically changing environment or process over time. The inertia weight in particle swarm optimization (PSO) is dynamically adjusted in this paper in order to provide a nonlinear search capability for the PSO algorithm. Results on benchmark functions in the literature are provided.**

I. Introduction

Modern heuristic algorithms are increasingly seen as potential tools for nonlinear optimization in the last decade and have attracted a lot of attention for researchers to mathematically model the search process. The advantage of the heuristic algorithms is that they do not require the objective function to be differentiable or be continuous. The particle swarm optimization (PSO) algorithm is an evolutionary computation technique developed by Kennedy and Eberhart [1, 2].

The PSO method is one of the most powerful methods for solving unconstrained and constrained global optimization problems. Little is, however known about how the PSO method works or finds a globally optimal solution of an optimization problem. The performance of PSO is known to do well in the early iterations of the search process, but has problems in reaching a near optimal solution in several benchmark functions compared to other techniques.

To overcome this problem, many researchers have employed methods to adapt PSO parameters [3-8]. Some these approaches are based on deterministic rules, like decreasing or increasing the inertia weight according to the number of iterations. Some have used fuzzy rules to provide some feedback information for weight change. The fuzzy rules are developed based on experience.

The search process of PSO is non-linear and very complicated; it is hard, if not impossible to mathematically model the search process to carry out dynamic adjustment of its parameters such as the inertia weight and the acceleration constants. It is critical for the search process to be nonlinear i.e. move from a global to a local search and vice-versa, in other words move from explorations to exploitations and vice-versa respectively, especially if the environment is dynamic. This requires nonlinear dynamic adaptations of PSO parameters to be possible from iteration to iteration.

This paper proposes the use of combined concepts of approximate dynamic programming and reinforcement learning called adaptive critic designs (ACDs) for the nonlinear dynamic adaptation of the inertia weight in the particle swarm optimization algorithm. Section II of this paper describes the PSO algorithm, Section III gives a brief background on adaptive critics and the type of ACD used in this paper, Section IV describes the proposed dynamic PSO and Section V presents some initial results obtained with the dynamic PSO algorithm on Rosenbrock, Rastrigrin and Griewank benchmark functions.

II. Particle Swarm Optimization

Particle swarm optimization is an evolutionary computation technique (a search method based on a natural system) developed by Kennedy and Eberhart [1, 2]. PSO, like a generic algorithm (GA), is a population based optimization tool. However, unlike GA, PSO has no evolution operators such as crossover and mutation, and moreover, PSO has lesser parameters. PSO is an evolutionary algorithm that does not implement survival of the fittest, and unlike other evolutionary algorithms where an evolutionary operator is manipulated, the velocity is dynamically adjusted.

The system initially has a population of random solutions. Each potential solution, called a *particle*, is given a random velocity and is flown through the problem space. The particles have memory and each particle keeps track of its previous best position (called the *pbest*) and its corresponding fitness. There exist a number of *pbest* for the respective particles in the swarm and the particle with greatest fitness is called the *global best* (*gbest*) of the swarm. The basic concept of the PSO technique lies in accelerating each particle towards its *pbest* and the *gbest* location, with some random weighting at each time step and this is illustrated in Fig. 1, where *P(k)* is the current position of a particle, *P(k+1)* is its modified position, *V(k)* is its initial velocity, *V(k+1))* is its modified velocity, V_{pbest} is the velocity considering its *pbest* location and V_{gbest} is the velocity considering its *gbest* location.

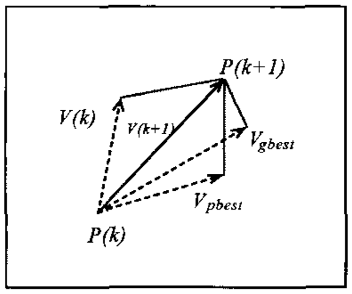

Fig. 1 Movement of a PSO particle

The main steps in the particle swarm optimization process are described as follows:

(i). Initialize a population of particles with random positions and velocities in *d* dimensions of the problem space and fly them.

(ii). Evaluate the fitness of each particle in the swarm.

(iii). For every iteration compare each particle's fitness with its previous best fitness (*pbest*) obtained. If the current value is better than *pbest*, then set *pbest* equal to the current value and the *pbest* location equal to the current location in the *d*-dimensional space.

(iv). Compare *pbest* of particles with each other and update the swarm global best location with the greatest fitness (*gbest*).

(v). Change the velocity and position of the particle according to (1) and (2) respectively. *V(k+1)* and *P(k+1)* represent the velocity and position of the i^{th} particle with *d* dimensions, respectively, *rand1* and *rand2* are two uniform random functions, and w_i is the inertia weight, which is chosen beforehand.

$$V(k+1) = w_i * V(k) + c_1 * rand1() * (P_{best}(k) - P(k)) + c_2 * rand2() * (G_{best}(k) - P(k)) \quad (1)$$

$$P(k+1) = P(k) + V(k) \quad (2)$$

(vi). Repeat steps (ii) to (v) until convergence is reached based on some desired single or multiple criteria.

The parameters used in PSO are described as follows: w_i called the inertia weight controls the exploration and exploitation of the search space because it dynamically adjusts velocity. Local minima are avoided by small local neighborhoods, but faster convergence is obtained by a larger global neighborhood, and in general a global neighborhood is preferred. Synchronous updates are more costly than the asynchronous updates. *Vmax* is the maximum allowable velocity for the particles (i.e. in the case where the velocity of the particle exceeds *Vmax*, then it is limited to *Vmax*). Thus, resolution and fitness of search depends on *Vmax*. If *Vmax* is too high, then particles will move beyond a good solution, and if *Vmax* is too low, particles will be trapped in local minima. The constants c_1 and c_2 in (1) and (2), termed as cognition and social components, respectively, are the acceleration constants which changes the velocity of a particle towards *pbest* and *gbest* (generally, somewhere between *pbest* and *gbest*). The velocities of the particles determine the tension in the swarm. A swarm of particles can be used locally or globally in a search space. In the local version of the PSO, *gbest* is replaced with *lbest* and the entire process is the same.

III. Adaptive Critic Designs

A. Background

Adaptive critic designs (ACDs) are neural network designs capable of optimization over time under conditions of noise and uncertainty. A family of ACDs was proposed by Werbos [9] as new optimization techniques combining concepts of reinforcement learning and approximate dynamic programming.

The adaptive critic method determines optimal control laws for a system by successively adapting two neural networks, namely, an action neural network (which dispenses the control signals) and a critic network (which learns the desired performance index for some function associated with the performance index). These two neural networks approximate the Hamilton-Jacobi-Bellman equation associated with optimal control theory. The adaptation process starts with a non-optimal, arbitrarily chosen control by the action network; the critic network then guides the action network toward the optimal solution at each successive adaptation. During the adaptations, neither of the networks needs any "information" of an optimal trajectory, only the desired cost needs to be known. Furthermore, this method determines optimal control policy for the entire range of initial conditions and needs no external training.

The design ladder of ACDs includes three basic implementations: Heuristic Dynamic Programming (HDP), Dual Heuristic Programming (DHP) and Globalized Dual Heuristic Programming (GDHP), in the order of increasing power and complexity. The interrelationships between members of the ACD family have been generalized and explained in [10-12]. The HDP method is applied in this paper without the use of a process model and therefore referred to the action-dependent HDP (ADHDP) [12]. In [11], a neural network model of the power system controlled was obtained and details are found in [13].

B. ADHDP Critic Network

Action Dependent Heuristic Dynamic Programming has a Critic neural network that estimates the function *J* (cost-to-go) in the Bellman equation of dynamic programming, expressed as follows:

$$J(x(k)) = \sum_{k=0}^{\infty} \gamma^k U(x(k)) \tag{3}$$

where γ is a discount factor for finite horizon problems ($0 < \gamma < 1$), $U(.)$ is the utility function or the local cost and $x(k)$ is an input vector to the Critic.

Fig. 2 shows the ADHDP Critic adaptation/training. The inputs to the Critic are outputs from the Action neural network. Two Critic neural networks are shown in Fig. 2 having the same inputs and outputs but at different time instants. Their outputs are $J(k+1)$ and $J(k)$.

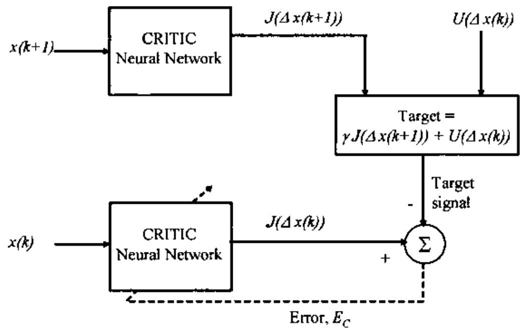

Fig. 2 HDP Critic neural network adaptation/training

The Critic network tries to minimize the following error measure over time

$$\|E_1\| = \frac{1}{2}\sum_t E_C^2(k) \tag{4}$$

where

$$E_C(k) = J(\Delta x(k)) - \gamma J(\Delta x(k+1)) - U(\Delta x(k)) \tag{5}$$

where $\Delta x(k)$ is the changes in $x(k)$, a vector of observables of the plant (or the states, if available). The utility function U is dependent on the system controlled or the problem in hand. The necessary condition for (4) to be minimal is given in (6).

$$\frac{1}{2}\frac{\partial}{\partial W_C}\left\langle E_C^2(k)\right\rangle = \left\langle E_{C1}\frac{\partial E_C(k)}{\partial W_C}\right\rangle = 0 \tag{6}$$

The weights' update for the Critic network using the backpropagation algorithm is given as follows:

$$\Delta W_C = -\eta \quad E_C(k)\frac{\partial E_C(k)}{\partial W_C} \tag{7}$$

$$\Delta W_C = -\eta\{J(\Delta x(k)) - \gamma J(\Delta x(k+1)) - U(\Delta x(k))\}\times \frac{\partial\{J(\Delta x(k)) - \gamma J(\Delta x(k+1)) - U(\Delta x(k))\}}{\partial W_C} \tag{8}$$

where η is a positive learning rate and W_C are the weights of the Critic neural network. The Critic network's output $J[\Delta x(k+1)]$is necessary in order to provide the training signal $\gamma J[\Delta x(k+1)]+ U(\Delta x(k))$, which is the desired/target value for $J[\Delta x(k)]$.

C. ADHDP Action Network

The objective of the Action neural network in Fig. 3, is to minimize $J(\Delta x(k))$ in the immediate future, thereby optimizing the overall cost expressed as a sum of all $U(\Delta x(k))$ over the horizon of the problem. This is achieved by training the Action neural network with an error signal $\partial J/\partial A$. The gradient of the cost function J, with respect to the outputs A, of the Action neural network, is obtained by backpropagating $\partial J/\partial J$ (i.e. the constant 1) through the Critic neural network to the Action neural network. This gives $\partial J/\partial A$ and $\partial J/\partial W_A$ for all the outputs of the Action neural network, and all the Action neural network's weights W_A, respectively. The weights' update in the Action neural network using backpropagation algorithm is given as follows:

$$\|E_2\| = \frac{1}{2}\sum_t E_A^2(k) \tag{9}$$

where
$$E_A = \frac{\partial J(k)}{\partial A(k)} \tag{10}$$

Weight change in the Action network ΔW_{A1} can be written as:

$$\Delta W_A \propto \frac{\partial E_A}{\partial W_A} \tag{11}$$

Equation (11) can be further written as:

$$\Delta W_A = -\alpha E_A(k)\frac{\partial E_A(k)}{\partial W_A} \tag{12}$$

$$\Delta W_A = -\alpha\frac{\partial J(k)}{\partial A(k)}\frac{\partial}{\partial W_A}\left(\frac{\partial J(k)}{\partial A(k)}\right) \tag{13}$$

where α is a positive learning rate.

With (8) and (13), the training of the Critic and the Action networks can be carried out. The training procedure with more details for the Critic and the Action networks are given in [10, 11].

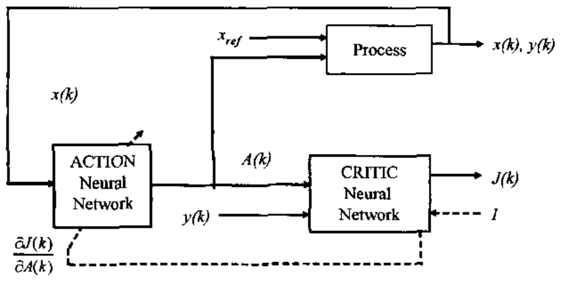

Fig. 3 ADHDP Action Critic neural network adaptation/training

IV. DYNAMIC PSO

The dynamic change in PSO algorithm is caused in the inertia weight w_i with the goal to minimize or maximize the objective function. To carry out the dynamic change in the inertia weight, the inputs to the Critic and Action neural network in Figs. 2 and 3 are the inertia weight at the time step k and the gbest fitness at the time step k of the PSO algorithm. The outputs of the Critic network is the J function of Bellman's equation in dynamic programming. The output of the Action network is the inertia weight for time step $(k+1)$. The Critic network is trained with constant and varying inertia weights. The objective of the Action neural network is to minimize the output of the Critic network by varying the inertia weight to improve the gbest fitness. The Critic and Action network training are interleaved in order to obtain convergence on both networks. The utility function plays a major role in variation of the inertia weight to achieve the best fitness desired. The proposed velocity equation is given in (14) where $w_i(k)$ is the output of the Action network.

$$V(k+1) = w_i(k) * V(k) + c_1 * rand1() * (P_{best}(k) - P(k)) + c_2 * rand2() * (G_{best}(k) - P(k)) \quad (14)$$

V. RESULTS

The proposed approach is experimented on the standard benchmark functions studied in [3, 6, 7] which are all minimization problems. These functions and their parameters are given below. Table I lists the initialization ranges of the three functions. The first benchmark function is the Rosenbrock function given by (15).

$$f_1(x) = \sum_{i=1}^{n} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \quad (15)$$

The second function is the generalized Rastrigrin function described by (16).

$$f_2(x) = \sum_{i=1}^{n} (x_i^2 - 100\cos(2\pi x_i) + 10) \quad (16)$$

The last function is the generalized Griewank function described by (17).

$$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1 \quad (17)$$

TABLE I: ASYMMETRIC INITIALIZATION RANGES

Function	Asymmetric initialization range
f_1	$(15, 30)^n$
f_2	$(2.56, 5.12)^n$
f_3	$(300, 600)^n$

For the minimization problems above, the desired *gbest* fitness is zero, therefore the following utility function *U(k)* is chosen.

$$U(k) = 2 \times gbest\_fitness \quad (18)$$

Table II below gives preliminary results obtained for the PSO algorithm for the above benchmark problems on a 10-dimensional problem (n). The number particles in the PSO algorithm are 20, the acceleration constants c_1 and c_2 in the PSO equations are 2 and the maximum of generations allowed are 1000. The results shown are over 10 trials. The preliminary results have shown improvements from those reported in literature.

TABLE II: BEST VALUES OBTAINED BY THE PSO

Function	Best Value (PSO gbest)
f_1	41.62
f_2	0.00
f_3	0.09

VI. CONCLUSIONS

This paper has proposed a dynamic particle swarm optimization algorithm based on adaptive critic designs. Preliminary results are promising that a nonlinear dynamic search process is achievable for improving PSO's performance using adaptive critics. The proposed method remains to be tried out on higher dimensions of above problems and statistical analysis on convergence needs to be carried out over a large of trials.

REFERENCES

[1] J. Kennedy, R. C. Eberhart, "Particle Swarm Optimization", *Proceedings IEEE International Conference on Neural Networks*, Volume: 4, 27 November - 1 December 1995, pp. 1942 -1998.

[2] J. Kennedy, R. C. Eberhart, *Swarm Intelligence*, Morgan Kauffman publishers, San Francisco, CA. ISBN 1-55860-595-9.

[3] S. Yuhui, R. C. Eberhart, "Fuzzy adaptive particle swarm optimization", *Proceedings of the 2001 Congress on Evolutionary Computation*, vol. 1, 27-30 May 2001, pp. 101 – 106.

[4] X. Hu, R C Eberhart, "Adaptive particle swarm optimization: detection and response to dynamic systems", *Proceedings of the Congress on Evolutionary Computation*, 12-17 May 2002, vol. 2, pp. 1666 - 1670.

[5] K. Yasuda, A. Ide, N. Iwasaki, "Adaptive particle swarm optimization", *IEEE International Conference on Systems, Man and Cybernetics*, Oct. 5-8, 2003 , vol. 2, pp. 1554 – 1559.

[6] X. Xie, W. Zhang, Z Yang, "Adaptive particle optimization on individual level", *6th IEEE International Conference on Signal Processing*, 26-30 Aug. 2002, vol. 2, pp. 1215 – 1218.

[7] Y. L. Zheng, L. Ma, L. Zhang, J. Qian, "On the convergence analysis and parameters selection in particle swarm optimization", *Proceedings of the Second IEEE International Conference on Machine Learning and Cybernetics*, November 2-5, 2003, pp. 1802-1807.

[8] M. Clerc, "The swarm and the queen: towards a deterministic and adaptive swarm optimization", *Proceedings of the Congress on Evolutionary Computation*, July 6 – 9, 1999, vol. 3, pp. 1951 - 1957.

[9] P. J. Werbos, "Approximate dynamic programming for real-time control and neural modeling", in Handbook of intelligent control, White and Sofge, Eds., Van Nostrand Reinhold, ISBN 0-442-30857-4, pp 493-525.

[10] J. W. Park, R. G. Harley, G. K. Venayagamoorthy, "Adaptive Critic Based Optimal Neurocontrol for Synchronous Generator in Power System Using MLP/RBF Neural Networks", *IEEE Transactions on Industry Applications*, vol. 39, no. 5, September/October 2003, pp. 1529-1540.

[11] G. K. Venayagamoorthy, R. G. Harley, D. C. Wunsch, "Comparison of Heuristic Dynamic Programming and Dual Heuristic Programming Adaptive Critics for Neurocontrol of a Turbogenerator", *IEEE Transactions on Neural Networks*, Volume: 13 Issue: 3 , May 2002, pp. 764 -773.

[12] D. Prokhorov, D.C. Wunsch, "Adaptive Critic Designs", *IEEE Transaction. on Neural Networks*, vol. 8, no. 6, pp. 997-1007, 1997.

[13] G. K. Venayagamoorthy, R. G. Harley, D. C. Wunsch, "Experimental Studies with Continually Online Trained Artificial Neural Network Identifiers for Multiple Turbogenerators on the Electric Power Grid", *IEEE-INNS International Joint Conference on Neural Networks*, Washington DC, USA, 15-19 July, 2001, pp. 1267-1272.