A Particle Swarm Optimization Approach for Estimating Parameter Confidence Regions

Praveen Koduru
Electrical & Computer Engineering
Kansas State University
Manhattan, KS, USA
praveen@ksu.edu

Stephen M. Welch
Department of Agronomy
Kansas State University
Manhattan, KS, USA
welchsm@ksu.edu

Sanjoy Das
Electrical & Computer Engineering
Kansas State University
Manhattan, KS, USA
sdas@ksu.edu

ABSTRACT

Point estimates of the parameters in real world models convey valuable information about the actual system. However, parameter comparisons and/or statistical inference requires determination of parameter space confidence regions in addition to point estimates. In most practical applications, the relation of the parameters to model fitness is highly nonlinear and noisy data leads to further deviations. Thus the confidence regions obtained by using locally linearized models are often misleading. Uniform covering by probabilistic rejection (UCPR) is a robust technique that has been developed to solve this problem, and has been proven to be more efficient than other approximate random search techniques. In this paper, we propose a contour particle swarm optimization (C-PSO) technique and compare its performance against UCPR in predicting the confidence regions. Results indicate that for problems with low number of parameters, both the algorithms are quite comparable. However, real world models such as genetic networks have a large number of parameters and the UCPR fails in finding good convergence due to its limited search capabilities. In such problems, the C-PSO technique was able to find the confidence regions with better resolution and efficiency.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search – *graph and tree search strategies, heuristic methods.*

General Terms

Algorithms, Theory.

Keywords

Evolutionary algorithms, genomics, confidence regions, particle swarm optimization.

1. INTRODUCTION

Particle Swarm Optimization (PSO) is one of the most recent biologically inspired algorithms for optimization [1-4]. In PSO a population of candidate solutions, called particles are maintained. The search space is explored by these particles that are allowed to move about inside it in a manner similar to the movement of birds or fishes in swarms. These particles have their own positions (locations within the search space, i.e. candidate solutions) as well as their velocities. The particle' trajectories are guided through iterative velocity updates, by their individual memories, i.e. stored previous best positions, as well as by their interaction with other particles. Eventually, the particles converge to suitable optima [3].

In this research, a PSO algorithm is proposed to address a specific need in modeling gene networks as differential equations, which is discussed below.

1.1 Problem Definition

For over 40 years plant physiologists and systems modelers have used simulation models to predict plant growth and development varietal characteristics and environmental inputs. Recent advancements in genomic science has opened the possibility of doing the same thing using dynamic models of gene expression [7-9]. Gene networks are apparently modular at the small scale [10]. Simple single-gene models, when interconnected into one to four-gene networks, demonstrate rich signal processing capabilities including Boolean logic gates, linear arithmetic units, coincidence detectors, delays, differentiators, integrators, oscillators, and bi-stable devices [10]. The latter are particularly important in linking events at the genome level to whole-plant, phenotypic responses because many developmental processes are initiated by state changes in a biological switch [*e.g.*, 11].

Unfortunately, estimating differential equation model parameters to high accuracy is complicated by (*i*) model imperfections, i.e. discrepancies between the model and the real system, (*ii*) experimental errors in collected data, and (*iii*) inconsistencies present when compiling data from multiple sources. Under these circumstances, it is desired to determine for each model parameter, a range of values, between which the actual parameter are located with a specified probability.

For all practical purposes, it can be assumed that the real world model is nonlinear and with the presence of noise in the prediction data the deviations are more pronounced. This leads to a highly nonlinear relationship between the fitness of the model and the estimated parameters. UCPR [13] is an approach that aids in visualizing confidence regions by plotting a cloud of points. The interior of such a cloud is assumed to approximate the confidence region. In comparison, with other standard benchmark methods this approach has been proven to be efficient [13].

However, the search technique used in this approach is trivial and becomes highly inefficient in finding the optima for problems that have noisy landscapes. PSO have been applied to such difficult problems and have been proven to be able to converge to the optima [3]. Another added advantage of the PSO is that at any given iteration in the algorithm the population is a cloud of points that can potentially be utilized to plot the confidence region.

In this paper, we present a comparison of the performance of UCPR and C-PSO in robust and efficient prediction of a confidence region around the best-fit parameters. The formal problem formulation is given in the next section.

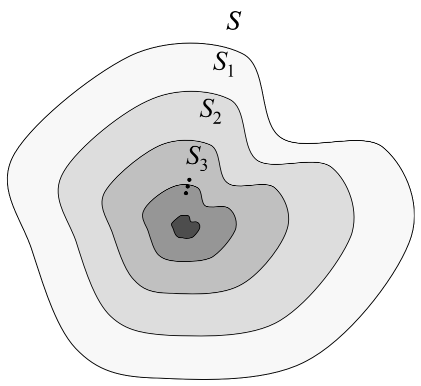

Figure 1. A schematic of the search space of a function illustrating how contours are defined by the sets S_k.

1.2 Problem Formulation

Given any function to be minimized, $f : S \subset \Re^n \to \Re$, where S is the search space, and an ascending sequence of level sets, l_k, k = 1, 2… L, such that $\inf(f(\mathbf{x})) \leq l_1 < l_2 <\ldots< l_L \leq \sup(f(\mathbf{x}))$, where L is the number of levels, the objective is to find the subsets S_k, k = 1, 2…L of S, where,

$$S_k = \{\mathbf{x} \mid \mathbf{x} \in S, f(\mathbf{x}) \leq L_k\}. \tag{1}$$

These sets defined above in equation (1) implicitly describe the contours of the function $f(\cdot)$. This is because a boundary separating the region S_k from another region $S_{k-1} - S_k$ (where $S \equiv S_0$) is the contour defined by $f(\mathbf{x}) = l_k$ (see also, figure 1 above). Appropriate levels l_k can be easily defined *a priori* depending on where the contours are desired.

From each region S_k, one can easily determine the range of any given parameter x_i, the ith component of **x** as,

$$R_i^k = \left[\min_i(S^k), \max_i(S^k)\right] \tag{2}$$

The $\min_i(\cdot)$ and $\max_i(\cdot)$ operators are simply the minimum and maximum of the ith component of their argument. Figure 2 shows this clearly. A special case of this problem is when the levels are defined from probabilities, thus delimiting confidence regions for point parameter estimates. This case is the principle motivation for the work reported here.

Since PSO is only capable of sampling a finite subset of S, the algorithm described in this paper can only produce a finite, reasonably small number of samples of each S_k. Because of this restriction, it is judicious to seek solutions that cover as extensive a region of each S_k as possible and are also regularly spaced, a desired feature that is termed as *diversity* in evolutionary computation parlance. Diversity is an important consideration in multi-objective optimization where the absence of a single

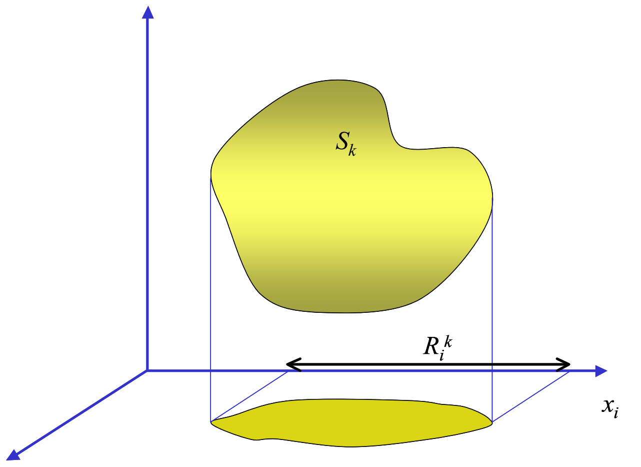

Figure 2. Schematic showing the relationship between ranges of acceptable values of a parameter x_i and S_k.

optimal solution makes it necessary for any algorithm to be able to produce an entire set of near-optimal sample solutions [5]. Multi-objective optimization algorithms therefore must maintain diversity during their search process [5]. Multi-objective versions of PSO routinely incorporate features specifically to address diversity too (*cf.* [6]). As shown later in this paper, diversity maintenance being a necessary requirement here, the algorithm suggested in this paper also implements features borrowed from other evolutionary approaches. We will henceforth refer to our algorithm as C-PSO (Contour Particle Swarm Optimization). To illustrate the effectiveness of C-PSO, we have chosen the following minimization problems.

1.3 Test Functions

The first three problems were originally used in [13] to evaluate the UCPR algorithm. The fourth problem is a simple gene network system of higher dimensionality.

1.3.1 Problem f_1

This is a model of a simple biological organism, that responds to two different external inputs (x_1^i and x_2^i) yielding a trait y_i via a multiplicative response [13]. The objective is to find the optimal pair (p_1, p_2) that minimizes the least squares function f_1 as defined in equation (3) .

$$f_1 = \sum_{i=1}^{4}\left(\frac{1}{\left((1+(x_1^i - p_1)) * (1+(x_2^i - p_2))\right)} - y_i\right)^2 \tag{3}$$

where, $x_1^{i=1,2,3,4} = \{15, 15, 20, 20\}$, $x_2^{i=1,2,3,4} = \{15, 20, 15, 20\}$, $y_1^{i=1,2,3,4} = \{0.6, 0.7, 0.7, 0.8\}$ and parameters p_1 and p_2 are in the range of $[12, 22]^2$. With the given data this function has three local optima and one global minimum.

1.3.2 Problem f_2

This model that represents a common case with poorly spaced experimental data (x^i) [13]. The objective is to minimize the value of f_2 as defined in equation (4), by finding the optimal pair (p_1, p_2).

$$f_2 = \sum_{i=1}^{4} \left(\frac{\exp(p_1) * x_i}{(\exp(p_2) + x_i)} - y_i \right)^2 \tag{4}$$

where, $x_1^{i=1,2,3,4} = \{0.9, 0.9, 1, 1\}$, $y_1^{i=1,2,3,4} = \{0.4, 0.55, 0.4, 0.6\}$ and p_1 and p_2 are the parameters to be predicted in the range of [-1,3] and [-2,3] respectively.

1.3.3 Problem f_3

This is an analytical test function that represents a disconnected confidence region [13]. The objective is to minimize the value of f_3 as defined in equation (5), by finding the optimal pair (p_1, p_2) in the range of $[0, 4.34]^2$.

$$f_3 = 3 - \sin(p_1^2) - \sin(p_2^2) \tag{5}$$

1.3.4 Parameter Estimation in a Gene Network

The effectiveness of the proposed C-PSO algorithm is also tested with a real world problem involving a single-gene model. The levels of messenger RNA was measured every 3 hours under short-days (9 hours of light, 15 hours of darkness each day) and long-days (15 hours of light, 9 hours of darkness per day) for the gene *HEADING DATE-1* (*Hd1*), an important flowering time control gene in rice (*Oryza sativa*). The experimental data provided [16] had two time series: $Hd1_{LD}^{(\exp)}(t)$ and $Hd1_{SD}^{(\exp)}(t)$ for $t = 0$ to 54 hours.
In [10] this data has been modeled with the equations,

$$\frac{d}{dt}(Hd1) = R_D g_{NN}(C(t)) - (Hd1)\lambda_D \tag{6}$$

under conditions of darkness, and as,

$$\frac{d}{dt}(Hd1) = R_L g_{NN}(C(t)) - (Hd1)\lambda_L \tag{7}$$

during light periods. Here, R's and λ's are constants associated with the gene and the subscripts L and D refer to light and dark periods. An input from the plant's diurnal clock is $C(t)=A*\sin(2\pi/p + \theta) + \mu$, where A is amplitude, p is period, θ is a phase angle, μ is a phase factor and $g_{NN} = \frac{1}{1+\exp(-c)}$ [7]. The state variable, $Hd1$, is dimensionless as expression levels are routinely normalized against laboratory standards. When simulating equation (7), two initial conditions for $Hd1$, are required, coresponding to light and dark periods, which are H_L and H_D respectively. Each simulation of $Hd1$'s RNA level defined in equation 4 should be carried out through alternating periods of darkness and light for each 24 hour cycle.

The ten parameters associated with the simulation can be regarded as a solution vector **x** in S as,

$$\mathbf{x} = [\lambda_L\ \lambda_D\ R_L\ R_D\ A\ p\ \theta\ \mu\ H_L\ H_D]. \tag{8}$$

When the simulation is carried out separately for long day and short day conditions using any parameter vector **x**, two time series can be obtained, $Hd1_{LD}^{(sim)}(t)$ and $Hd1_{SD}^{(sim)}(t)$. The fitness function E, is defined as the sum of the root mean square errors between the simulated and a given k number of experimental data points,

$$E = \sqrt{\frac{\sum\left(Hd1_{iSD}^{(sim)}(t) - Hd1_{iSD}^{(\exp)}(t)\right)^2}{k}} + \sqrt{\frac{\sum\left(Hd1_{iLD}^{(sim)}(t) - Hd1_{iLD}^{(\exp)}(t)\right)^2}{k}} \tag{9}$$

The objective is to obtain reliable estimates for the range of values of each parameter in equations (6) and (7) that minimize E in equation (9).

2. OUTLINE OF THE ALGORITHMS

In this section we explain in detail both the algorithms UCPR and C-PSO.

2.1 UCPR

The main idea of the algorithm is to start with a set of randomly generated N number of points in the search region. In each iteration the worst point (G_w) is selected and replaced with a better solution. This is achieved by randomly generation solutions in the search space within a specified boundary region (D). This process is iterated until the worst fitness is lies within a boundary of an α confidence region of the target boundary value fitness (G_c), i.e. until $G_w < G_c$,. The value of G_c is evaluated using equation (10).

$$Gc = \left[1 + \frac{nF(n, K-n, \alpha)}{K-n}\right] G_{\min} = WG_{\min} \tag{10}$$

where, n is the number of parameters, K is the sample size, $F()$ is the Fisher F-value [14] and G_{min} represents the best minimum fitness value found. The overall algorithm as detailed in [13], can be summarized briefly as below:

1. Initialize sample size N, parameter domain range D, safety factor c and an initial value for G_c,.
2. Randomly initialize N points uniformly in the search domain D, which forms the set X_0.
3. Set $j = 0$
4. Calculate the fitness $(G(P))$ for each point in X_j.
5. Termination criteria: If $max(G(P)) < G_c$, then exit and output the current population. Otherwise go to step 6.
6. Generate a new individual P_{try} uniformly distributed over D.
7. If the minimum Euclidian distance to any point in X_j is greater than the safety factor c, then go to step 6. Otherwise go to step 8.
8. Evaluate the fitness of P_{try}, $:G(P_{try})$. If $G(P_{try}) > max(G(P))$, then go to step 6. Otherwise, replace current worst point in X_j. If $G(P_{try}) < G_{min}$, then replace G_{min} with the new value and revaluate the value of G_c is updated. Increment j by 1, and return to step 5.

2.2 C-PSO

2.2.1 Particle Swarm Optimization

This section describes the variant of the standard PSO algorithm that has been used for the rest of the paper. The algorithm maintains a population of M particles whose positions,

$X(i)$, $i = 1, 2, \dots M$, are initialized to random values at the start. These positions are updated in each iteration of the algorithm by adding the particles instantaneous velocity $V_t(i)$ during iteration t to it, as follows,

$$X_{t+1}(i) = X_t(i) + V_t(i) \tag{11}$$

The velocity is updated in each iteration, so that the particle can eventually move towards a better location. The velocity update takes place using each particle's recorded previous best position, and the current location of the other particles. It is given by,

$$V_{t+1}(i) = \chi(V_t(i) + C_1 \times U[0,1] \times (X_{ib}(i) - X_t(i)) + C_2 \times U[0,1] \times (X_{gb,t} - X_t(i))) \tag{12}$$

In the above equation, C_1 and C_2 are two constants, called the *cognitive* and the *social* constants, and χ is called the constriction coefficient, that helps in maintaining stability [3]. The quantity $U[0,1]$ is a uniformly distributed random number in [0, 1]. The quantity X_{ib} is the *individual best* recorded position of the i^{th} particle so far, $X_{ib}(i) = X_{t'}(i)$, such that $\forall s \in \{0, 1, \dots t\}$, $e(X_{t'}(i)) \leq e(X_s(i))$, where $e(\cdot)$ is the objective function to be minimized. The other quantity, $X_{gb,t}$ is called the *global best*, and is the position of the best particle in the current iteration t. In other words, $X_{gb,t} = X_t(j)$, for some j, such that $\forall k \in \{0, 1, \dots N\}$, $e(X_t(j)) \leq e(X_t(k))$.

2.2.2 Hybridized PSO Algorithm

Hybridizing the Nelder-Mead simplex approach with evolutionary algorithms has been a very popular approach to speed up convergence. Koduru *et al*. have extended this work to hybridizing PSO [1, 2] using Nelder-Mead simplex [15]. In this approach, the *k*-means algorithm is used within each iteration to divide the entire population into separate clusters, each containing points in close proximity. Each cluster is then improved separately through the Nelder Mead search. The results presented in [1,2] suggest that using the clustering approach has provided better convergence results with the simplex hybrid PSO. For this reason we have opted to use this approach of hybrid simplex PSO for the current work.

2.2.3 Archiving

The hybridized PSO algorithm has a good convergence rate but is unsuitable, as given, for problem at hand. This section describes the modifications we have made in response. In general, the objective of a PSO algorithm is to direct the swarm of particles to the global minima. However, in the current work, our goal is form a cloud of points that shows the confidence region. Hence, it is necessary to have an archive that stores all that solutions that have a fitness value less than G_c in equation (11). Such an archive when plotted gives a confidence region with the target boundary fitness of G_c. Hence we modify the basic algorithm to have an external archive in addition to the global best solution, which stores all the individuals that have a better fitness than G_c in every generation.

2.2.4 Crossover and Mutation

In general, when any PSO algorithm is allowed to run for a sufficient number of iterations, the swarm converges to single optimum. However, in order to form the confidence regions, we need the swarm to form a cloud of points rather than a dense region. This effect is achieved by using the crossover and mutation operators in the hybrid PSO algorithm.

The mutation operator is applied as turbulence to the velocity update of particle i as:

$$V_{t+1}(i)_{wt} = V_{t+1}(i) + \delta \times U[-1,1] \times V_{t+1}(i) \tag{13}$$

Where δ is a turbulence factor, and the quantity U[-1,1] is a uniformly distributed random number in [-1, 1]. This updated, turbulent velocity ($V_{t+1}(i)_{wt}$) is then used to update the particles position [17].

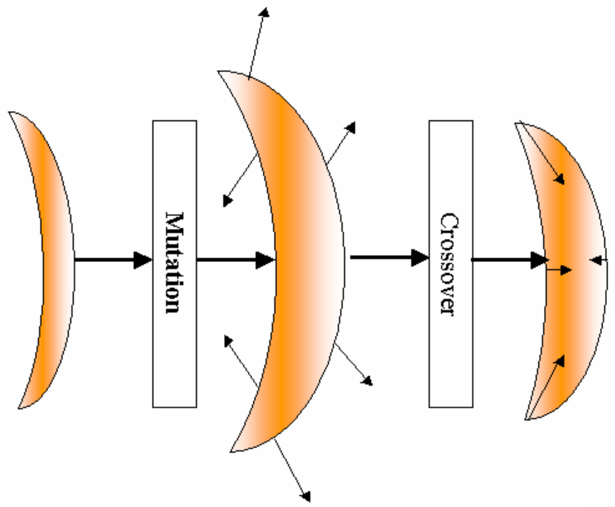

Figure 3. Schematic showing the effect of mutation and crossover on a moving cloud

The mutation operator creates a spreading cloud of particles, some of which are inevitably far outliers in fitness. Hence a countervailing correction mechanism, crossover, is incorporated into the algorithm. This can be visualized as a solar flare in space that expands as it progresses. The flare continues in directions where the conditions are favorable and expands in that direction. However, in other directions the flame loses intensity and eventually disappears. The mutation drives the expansion, while the crossover limits the effect of turbulence by maintaining the flow in favorable directions (see Figure 3).

Uniform crossover is implemented by replacing the worst fitness particle (outlier, X_t^w) with a new particle (X_t'), which is generated using equation (14), using a good fitness particle (core, X_t^b).

$$X_t' = \eta X_t^w + (1-\eta) X_t^b \tag{14}$$

where η is a uniformly distributed random number in [0.5, 1]. In the present work, we subjected any particles outside the 95% confidence region to crossover with a randomly selected particle within the 10% confidence region.

2.2.5 C-PSO Algorithm

The complete algorithm that is implemented is as follows:

1. $t = 0$.
2. Randomly initialize the particle positions $X_0(i)$.
3. Initialize all velocities, $V_0(i)$, to zeroes.
4. Check for termination criteria. If condition met then

terminate, otherwise continue to step 5.

5. Assign each particle i to clusters using k-means.
6. Evaluate the objective function at each $X_t(i)$.
7. Update each particle i's individual best.
8. Update global best and G_{min} and reevaluate G_c.
9. Remove any solutions in the archive have fitness greater than or equal to G_c, as defined by Equation (10). Add any solutions in the new population that do satisfy the condition.
10. Apply the simplex approach to each cluster.
11. Update positions $X_t(i)$ according to (11).
12. Apply uniform crossover on outlier particles.
13. Update velocity according to (12) and (13).
14. $T = t + 1$. If $t > t_{max}$, stop, else go to 4.

3. EXPERIMENTAL SETUP

The performance of both the algorithms was compared on the different test functions that have been described in Section 1.3. To permit comparisons, the termination criterion for both algorithms was the same as in [13]. Specifically, iteration stops when the search domain D contains a set of 200 points that all satisfy the condition $G(P) < G_c$. The value of G_c is evaluated using equation (10), where the value of W is fixed at 1.2 for all the test problems except for test function f_1, for which W is set to be 1.4 [13]. Both the algorithms have many settings to be specified *a priori*. These quantities are discussed next.

In UCPR, an initial population of 200 points is uniformly generated in the specified search domain range (D) for each problem. However, such large size populations are not commonly used in PSO algorithm [2, 3 and 6]. In order to maintain uniformity over all the problems, we have fixed the population size such that a total of 10 k-means clusters can be formed with each cluster each having $(n+1)$ points, where n represents the number of parameters to be estimated. This number is determined by the prerequisites necessary to apply the Nelder-Mead simplex within each cluster. The total number of k-means steps to divide the population into different clusters is restricted to a maximum of 10 iterations. With each cluster, the Nelder-Mead simplex is used to improve the worst individuals by doing reflection and contraction operations as necessitated. The parameters used with the simplex operation are the reflection and contraction coefficients, which are fixed at 1.5 and 0.5 respectively. In the preliminary runs, it was observed that use of more simplex operations would cause the swarm to rapidly converge to local optima. Hence, to avoid the loss of diversity in the swarm, one simplex flip per instance is used. This allowed for a faster convergence rates while maintaining the diversity of the swarm. For the mutation operation, a turbulence factor (δ) of 0.1 is used. Each of the algorithms is run five times for each test problem and the average of the results obtained are reported.

4. RESULTS

Table I lists the average number function evaluations utilized by each algorithm in order find the 200 points in the parameter space that satisfy the condition $G(P) < G_c$.

It can be clearly seen in all the problems that C-PSO was able to satisfy the termination criteria with least number of function evaluations. In the case of the gene network problem, the UCPR method failed to satisfy the termination condition in any of the 5 different runs even after 30,000 function evaluations. This suggests the failure of UCPR method in converging to the optima in a high dimensional search space. Table II and III provide the average of the best and worst fitness obtained in the final population. Again it is clearly evident that C-PSO was able to converge to better optima than UCPR. Hence, C-PSO is not only efficient in convergence, but also robust in finding the optima in comparison with UCPR.

Figures 4 and 5 show comparison of the confidence regions obtained from all the different runs by UCPR and C-PSO on the test functions $f1$ and f_3, respectively. Comparing the contour formed by joining the exterior of the point clouds with the actual nonlinear shaped contour in the background, it can be clearly observed that C-PSO has been able to get a much better approximation of the confidence regions. The algorithms were also capable of finding the confidence regions in disconnected regions. Figure 6 shows the scatter plot of the particle cloud form one of the run of UCPR on gene network problem for the two parameters R_D and R_L. The RMS prediction error is the vertical axis, and the points have a spectrum that is proportional to the goodness of fit of the solution. The algorithm was not able to converge to any good solution in all the runs and the near even distribution of the points search space, strongly suggests the failure of the algorithm in forming the confidence region.

Table I. Comparison of Number of Function Evaluations necessary for convergence for both algorithms

PROBLEM	UCPR	C-PSO
f_1	893	820
f_2	1304	1220
f_3	1274	860
Gene Network	30000*	20240

Table II. Comparison of Best solution obtained

PROBLEM	UCPR	C-PSO
f_1	1.231474	1.2302278
f_2	0.03130862	0.0312501
f_3	1.00875927	1.00000
Gene Network	3.6190199*	2.4919667

Table III. Comparison of Worst solution obtained

PROBLEM	UCPR	C-PSO
f_1	1.4159113	1.43431408
f_2	0.04692051	0.03585649
f_3	1.38366209	1.19575342
Gene Network	5.099656*	2.570873

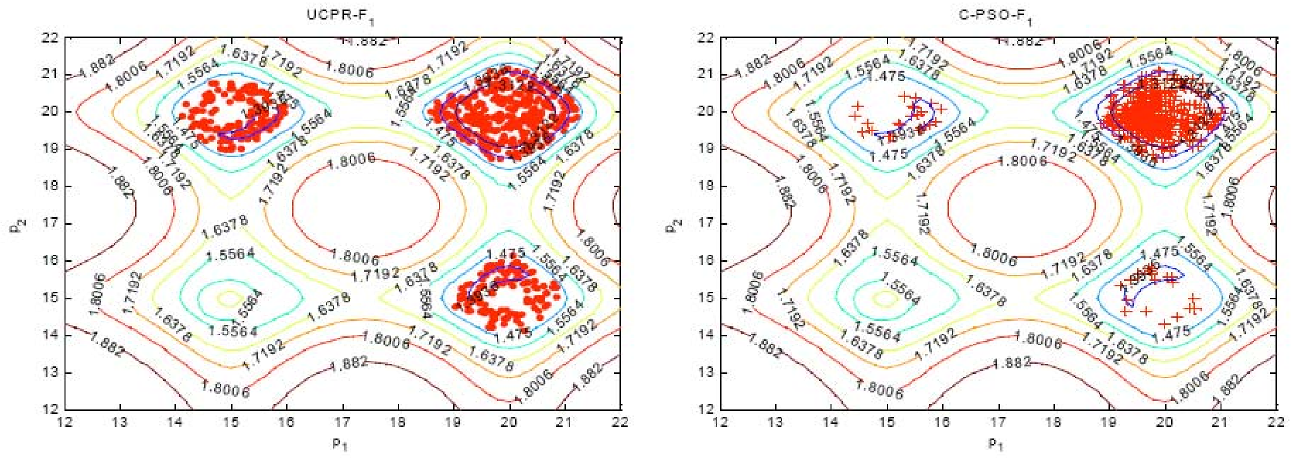

Figure 4. Comparison of the confidence regions obtained by UCPR and C-PSO on test function f_1

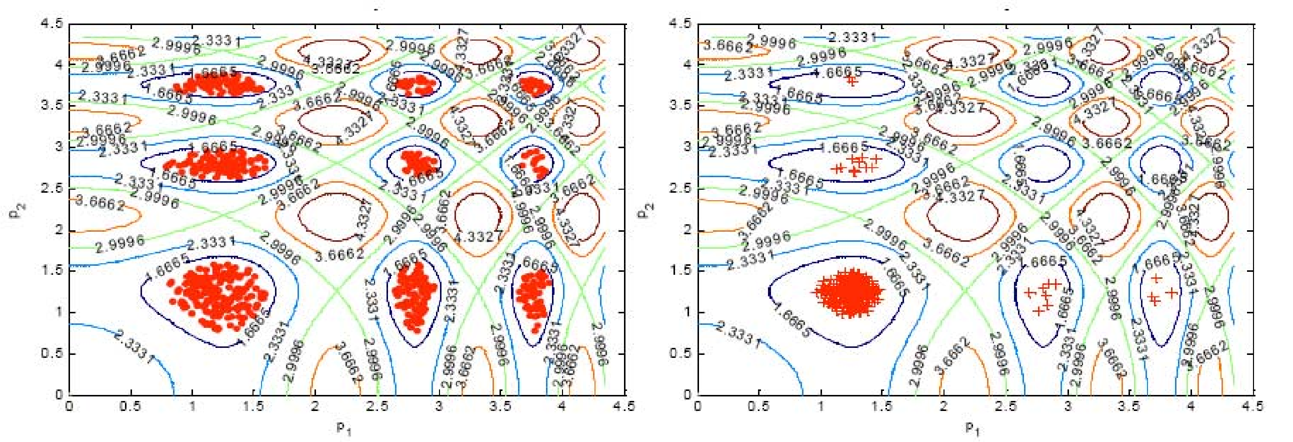

Figure 5. Comparison of the confidence regions obtained by UCPR and C-PSO on test function f_3

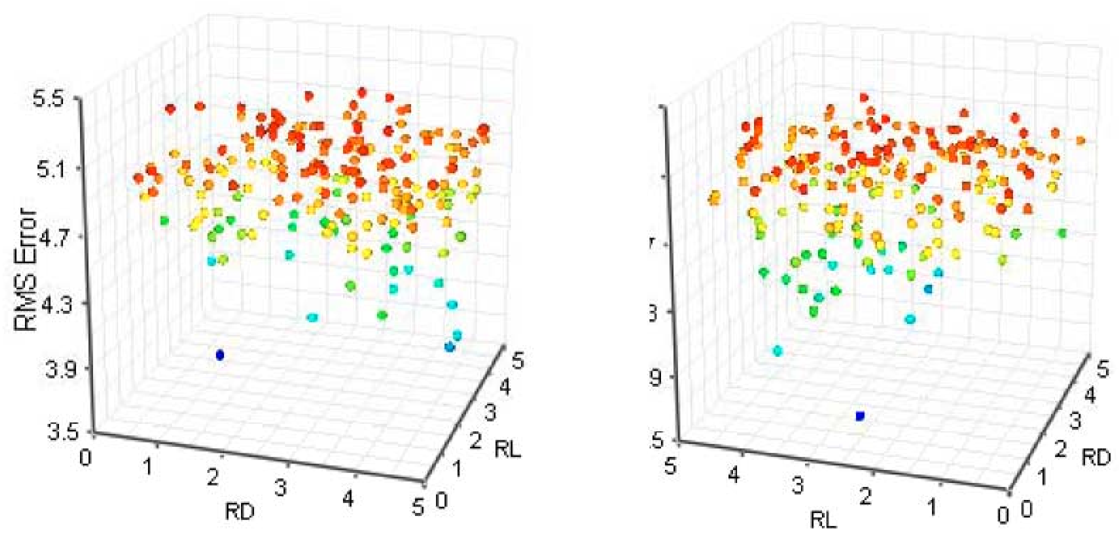

Figure 6. Scatter plots showing the distribution of the parameters R_D and R_L vs RMS prediction error for a single run of UCPR

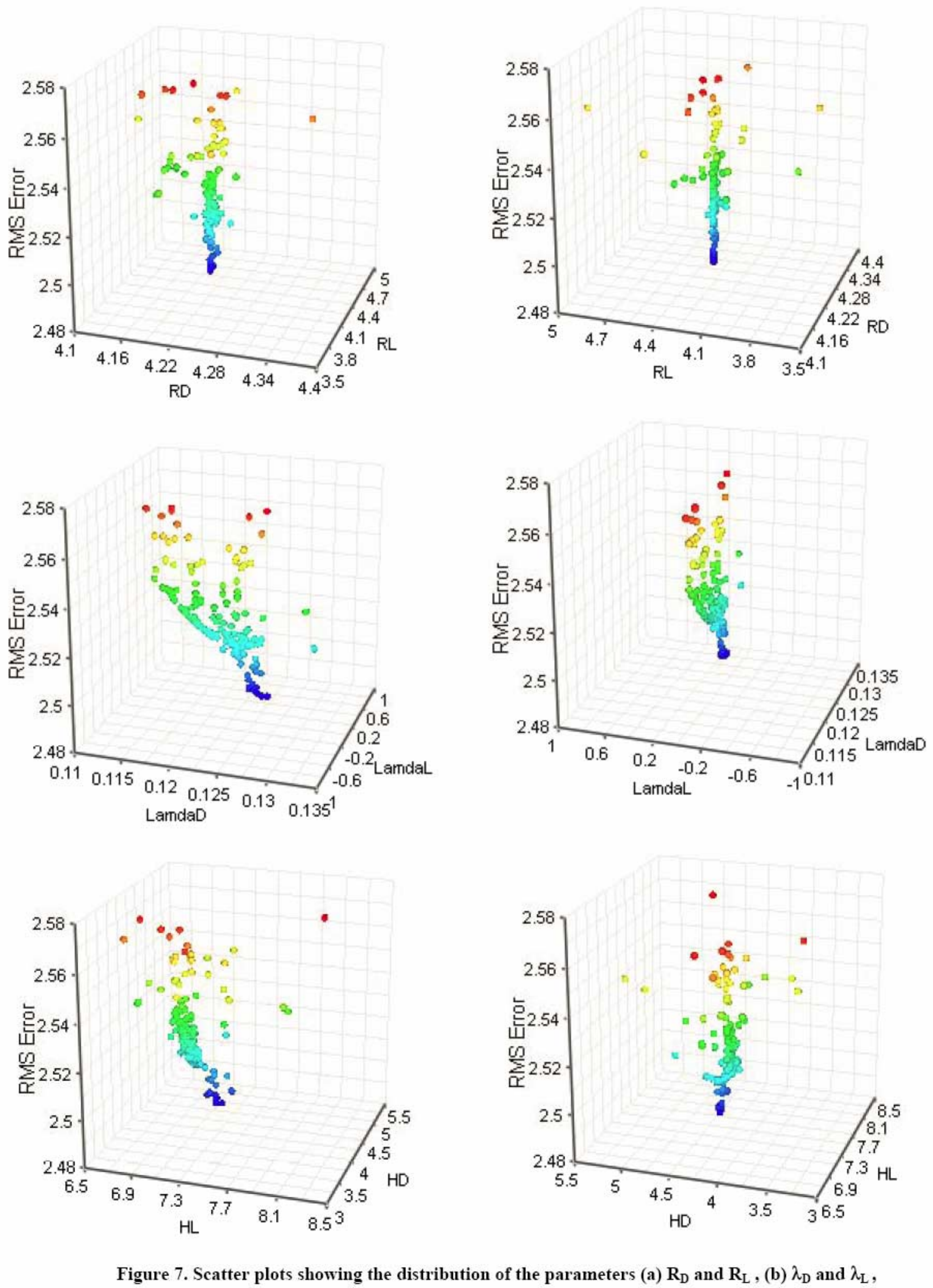

Figure 7. Scatter plots showing the distribution of the parameters (a) R_D and R_L , (b) λ_D and λ_L , (c) H_L and H_D vs RMS prediction error for a single run of C-PSO

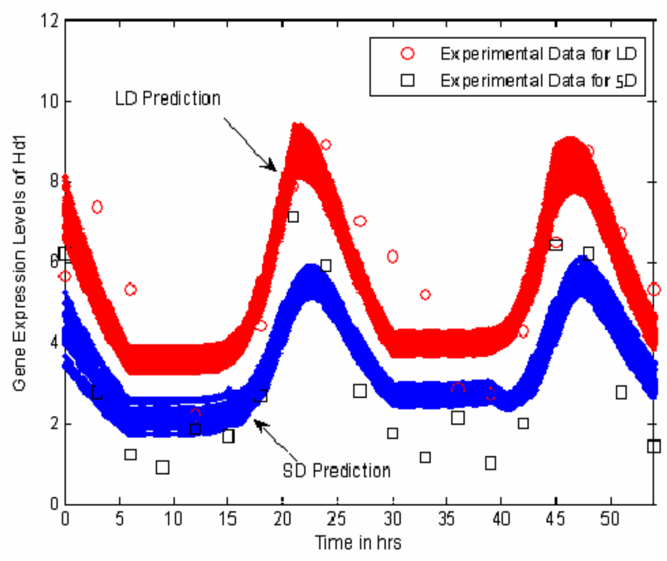

Figure 8. Comparison of Predicted and Actual values of Hd1 expression levels for LD and SD for all the 200 solutions obtained in a single run of C-PSO

In comparison, Figure 7 shows the particle distribution in the parameter space for three different sets of parameters. From these plots it can be noted that this real world problem has narrow valley optima surfaces, which renders common purpose optimization algorithms incapable of finding good fit solutions. To illustrate the effect of the varying parameters a plot of the simulated output and the actual experimental data is provided in Figure 8. The wide band of solid lines depicts the regions of prediction for the various parameters values that are present in the cloud of points shown in Figure 7.

5. CONCLUSIONS

Statistical inferences for complex models must be based on accurate confidence regions. Such regions may have convoluted shapes not readily revealed by linearized analysis. We have compared UCPR with a novel algorithm (C-PSO) based on particle swarm optimization.. For lower dimensional problems, C-PSO was able to find somewhat better minima in somewhat less time. But, to the eye at least, UCPR had better uniformity and tended to find more portions of disconnected confidence intervals. More importantly, however, is the fact that C-PSO was able to find confidence regions for higher dimensioned problems where UCPR failed. This result provides justification for additional work that should be able to enhance C-PSO performance across the board. Such work is underway.

6. REFERENCES

[1] S. Das, P. Koduru, S. M. Welch, M. Gui, M. Cochran, A. Wareing, B. Babin, Adding Local Search to Particle Swarm Optimization, *Proceedings*, *World Congress on Computational Intelligence*, Vancouver, Canada, 2006.

[2] P. Koduru, S. Das, S. M. Welch, A Particle Swarm Optimization-Nelder Mead Hybrid Algorithm for Balanced Exploration and Exploitation in Multidimensional Search Space, *Proceedings*, *International Conference on Artificial Intelligence*, Las Vegas, Nevada, 457–464, 2006.

[3] M. Clerc and J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, 58-73, 2002.

[4] Mendes, R.; Kennedy, J.; Neves, J., The fully informed particle swarm: simpler, maybe better, IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, 204 – 210, June 2004.

[5] Kalyanmoy Deb, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley, London, 2001.

[6] Coello, C.A.C., Pulido, G.T., Lechuga, M.S., Handling multiple objectives with particle swarm optimization, IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, 256 – 279, June 2004.

[7] S. M. Welch, J. L. Roe and Z. Dong, Z., A genetic neural network model of flowering time control in Arabidopsis thaliana, Agron. J. Vol. 95, 71-81, 2003.

[8] S. M. Welch, Z. Dong and J. L. Roe, Modelling gene networks controlling transition to flowering in Arabidopsis, Proceedings of the 4th International Crop Science Congress, Brisbane, Australia, Sep 26 – Oct 1, 2004.

[9] Z. Dong, Incorporation of genomic information into the simulation of flowering time in Arabidopsis thaliana, Ph.D. dissertation, Kansas State University, 2003.

[10] S. M. Welch, J. L. Roe, S. Das, Z. Dong, R. He, and M. B. Kirkham, Merging genomic control networks with soil-plant-atmosphere-continuum (SPAC) models, Agricultural Systems, 2004.

[11] A. V. Hill, The possible effect of aggregation of molecules of haemoglobin on its dissociation curves, J. Physiol. 40 (1910), iv-viii.

[12] S. Das, P. Koduru, S. M. Welch, M. Gui, M. Cochran, A. Wareing, B. Babin, Adding Local Search to Particle Swarm Optimization, *Proc. of* the *World Congress on Computational Intelligence*, Vancouver, BC, Canada, 2006.

[13] O. Klepper, E.M.T. Hendrix, A comparison of algorithms for global characterization of confidence region for nonlinear models, Environmental Toxicology and Chemistry, vol. 13, 1887-1899, 1994.

[14] N.R. Draper, H. Smith, Applied Regression Analysis, John Wiley, New York, 1966.

[15] J. A. Nelder and R. A. Mead, A simplex method for function minimization, *Computer Journal*, Vol. 7 no. 4, 308-313, 1965.

[16] S. Kojima, Y. Takahashi, Y. Kobayashi, L. Monna, T. Saski, T. Araaki and M. Yano, Hd3a, a rice ortholog of the Arabidopsis FT gene, promotes transistion to flowering downstream of Hd1 under short-day conditions, *Plant Cell Physiology*, 43, 1096-1105, 2002.

[17] J.E. Fieldsend and S. Singh, A multi-objective algorithm based upon particle swarm optimization, an efficient data structure and turbulence, In Proceedings of the 2002 U.K. Workshop on Computational Intelligence, Birmingham, UK, 37-44, September, 2002.