

A Particle Swarm Optimization-Nelder Mead Hybrid Algorithm for Balanced Exploration and Exploitation in Multidimensional Search Space

Praveen Koduru*, Sanjoy Das*, Stephen M. Welch[†]

Abstract—Maintaining an appropriate balance between exploration and exploitation is the key to faster convergence. This paper proposes a method to add an exploitative component to particle swarm optimization, a recently proposed biologically inspired metaphor. This is accomplished by applying the well-known Nelder Mead simplex algorithm to the population of solutions at the end of each iteration. It has been shown that this proposed hybridization method speeds up the significantly the rate of convergence for several well known benchmark problems as well as for the problem of fitting a gene model with observable data.

Keywords: *Swarm Intelligence, Evolutionary Algorithms, Bio-inspired computing, Optimization and Genomics*

I. Introduction

BIOLOGICALLY inspired algorithms are stochastic optimization algorithms that are based on biological phenomena. Perhaps the most commonly used biologically inspired algorithms are genetic algorithms that are modeled after Darwinian evolution. More recently, swarm intelligence based algorithms have been proposed for discrete and continuous optimization. Algorithms based on biological phenomena are immensely popular as they (i) are derivative-free techniques, (ii) do not get trapped in local minima, (iii) sample a wide region of the search space, (iv) can be tailored specifically to suit the problem, and (v) can be hybridized with other algorithms for improved performance.

Particle Swarm Optimization (PSO) is a biological approach based on swarm intelligence that is receiving a lot of research attention lately [1-4]. PSO is a population-based approach that holds a set of candidate solutions, called particles, which move along the search space. The trajectory followed by each particle is guided by the particle's own memory,

as well as its interaction with other particles. The specific method of adjusting the particles trajectory is modeled after the way a flock of birds or a school of fish would interact with each other. This interaction helps in guiding the particles towards good minima in the problem's search space. At the end of the algorithm's execution, a few of the particles would have converged to optimal solutions.

While biologically motivated algorithms such as PSO are very effective in computing optima, they can be further improved by maintaining the correct balance between exploration and exploitation. An exploitative component to the search allows the algorithm to make use of local information to better guide the search towards improving regions of the search space. This feature enables the algorithm to provide faster convergence, using less number of function evaluations.

This research explores the possibility of improving the particle swarm optimization by hybridizing it with the Nelder Mead simplex algorithm, an algorithm that exploits local information and converges to the nearest optimal point. The well-known k -means clustering algorithm is used to divide the particles into clusters immediately before applying the Nelder Mead algorithm.

The improved algorithms' effectiveness was tested for the optimization of gene differential equation models. The model has to be run as many times as the number of function evaluations required by the optimization algorithm. Because differential equation models are simulated using the Runge-Kutta or other similar numerical techniques, evaluating them requires substantial computer time. Therefore, fast convergence is of critical importance in this application.

Although intended for fitting gene differential equation models with observable data, the proposed algorithms can be applied in many other practical situations where objective function evaluation is too expensive, requiring at least several minutes of simulation time per evaluation. It is a common situation in many applications.

*Department of Electrical and Computer Engineering, Kansas State University, Manhattan, KS

[†]Department of Agronomy, Kansas State University, Manhattan, KS

This work was supported in part by the U.S. Department of Agriculture under Grant No. USDA 2003-35304-13217 and the National Science Foundation under Grant No. NSF FIBR 0425759.

II. Background

A. Particle Swarm Optimization

We first describe the variant of the standard PSO algorithm that has been used for the rest of the paper. This algorithm maintains a population of N particles whose positions, $X(i)$, $i = 1, 2, \dots, N$, 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,

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

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

$$V_{t+1}(i) = \chi(V_t(i) + C_1 \times U[0,1] \times (X_{ib}(i) - X_t(i)) + C_2 \times U[0,1] \times (X_{gb,t} - X_t(i))) \quad (2)$$

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 [1]. The quantity $U[0,1]$ is a uniformly distributed random number in $[0, 1]$. The quantity X_{ib} is called the *individual best*, and is the best recorded position of the i^{th} particle so far, $X_{ib}(i) = X_r(i)$, such that $\forall s \in \{0, 1, \dots, t\}$, $e(X_r(i)) \leq e(X_s(i))$, 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 $\forall k \in \{0, 1, \dots, N\}$, $e(X_t(j)) \leq e(X_t(k))$.

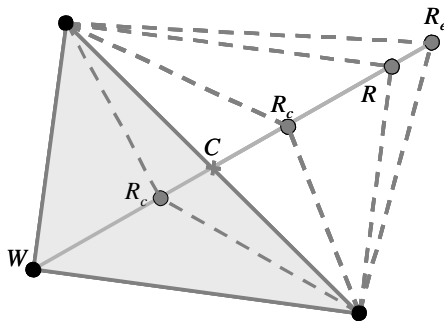


Fig. 1. Illustration of the Nelder-Mead simplex algorithm. Note that the worst point W is replaced with R , R_c or R_e .

B. Nelder-Mead Simplex Based Local Search

The Nelder-Mead approach makes use of a construct called a *simplex*. A simplex in n -dimensions consists of $n+1$ solutions, $X(k)$, $k = \{1, 2, \dots, n+1\}$ [5]. In a

two dimensional plane, this corresponds to a triangle. The solutions are evaluated in each step and the worst solution W is identified. The centroid of the simplex is computed as,

$$C = (\sum X(k))/n \quad (3)$$

where the worst point, W , is excluded from the summation. Then the worst point is reflected along the centroid. The reflected solution is given by,

$$R = C + (C - W) \quad (4)$$

Usually, the worst point w is replaced with the reflected point r in the simplex, but if the reflected point is better than any solution in the simplex, the simplex is further expanded as,

$$R_e = C + \eta(C - W) \quad (5)$$

where η is called the expansion coefficient. However, if the reflected solution R is worse than W , the simplex is contracted and the reflected solution is placed on the same side of the centroid. When solution R is not worse than W , but worse than any other solution in the simplex, the simplex is still contracted, but the reflection is allowed to remain on the other side of the centroid. Reflection is carried out as follows,

$$R_c = C \pm \kappa(C - W) \quad (6)$$

In the above equation, κ is called the contraction coefficient. Solution W is replaced with the new one, R , R_e , or R_c from the next step onwards. The simplex algorithm is allowed to run for multiple steps before being terminated. We will refer to each step of the simplex as a *flip*.

Figure 1. shows a schematic of the Nelder-Mead algorithm in two dimensions.

C. K-means Clustering

The k -means approach is useful to divide a population of solutions into separate clusters, where each cluster consists of proximally located individuals only. Here k refers to the total number of clusters. It is a simple approach that begins by randomly placing k cluster centers in the solution space. In each iteration, it computes the distance of each $X(i)$ to the clusters, and applies two steps:

1. For each $X(i)$, determine its closest cluster center $C(X(i))$,
2. Replace each center with $(\sum X(k))/n$, the summation being carried out over all particles belonging to that cluster center, and n being the total number of such particles.

III. The Hybrid Algorithm

Hybridizing the Nelder-Mead approach with evolutionary algorithms has been a very popular approach to speed up convergence. In recent years, several papers have reported this approach [6-11].

Koduru *et al.* have extended this method for multi-objective optimization problems as well [9 - 11]. Several schemes have been proposed, that fall into one of the two broad categories (i) tandem, and (ii) cascade. In a tandem hybridization strategy, within each iteration of the approach, the entire population is divided into two subsets. While one set of individuals are subject to the main stochastic optimization algorithm, one or more flips of the Nelder-Mead algorithm is applied to the other one. At the end, both subsets are merged together. In cascade hybridization, the approach is to apply the stochastic optimization process to all the individuals of the population, and further improve the solutions obtained using Nelder-Mead simplex search. The two schemes are illustrated in figure 2.

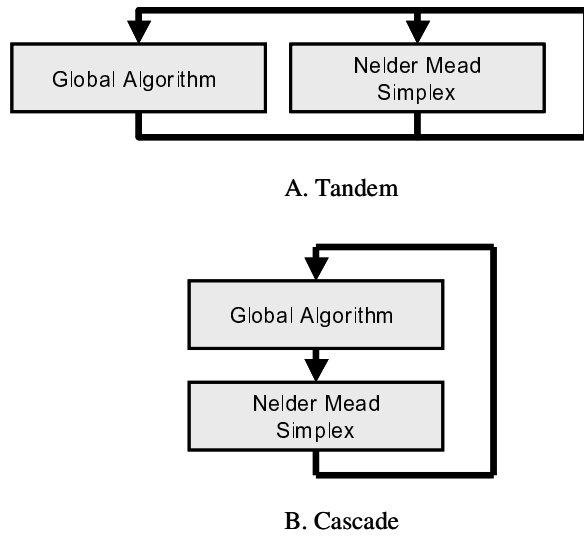


Fig. 2. Two schemes for hybridization of the Nelder-Mead simplex approach with any stochastic global optimization algorithm as in PSO.

The available literature on evolutionary algorithms appears to be fairly evenly divided on the scheme used. However, our initial experiments (not reported here) have suggested that executing the PSO and Nelder Mead routines in tandem does not yield results as good as those from a cascade approach. Hence the latter scheme has been adopted.

Since the Nelder Mead algorithm is for local optimization, it works best when the solutions provided to it are closely spaced in the search space. In order to do so, the k -means algorithm has been adopted to divide the particles into separate clusters, each of which is then improved separately through the Nelder Mead search. The overall algorithm for this approach is outlined below:

1. $t = 0$.
2. Randomly initialize the particle positions $X_0(i)$.
3. Initialize all velocities, $V_0(i)$, to zeroes.
4. Randomly initialize k cluster centers.
5. Update positions $X_t(i)$ according to (1).
6. Evaluate the objective function at each $X_t(i)$.
7. Update each particle i 's individual best.
8. Update global best.
9. Update velocity according to (2).
10. Assign each particle i to clusters using k -means.
11. Apply the simplex approach to each cluster.
12. $t = t + 1$.
13. If $t > t_{\max}$, stop, else go to 5.

IV. Problems

A. Benchmark Problems

The following benchmarks have been used to evaluate the performance of the proposed algorithm:

Sphere function:

$$f_1(x_i) = \sum_{i=1}^n x_i^2 \quad (7)$$

Rosenbrock function:

$$f_2(x_i) = \sum_{i=1}^{(n-1)} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \quad (8)$$

Rastrigin function:

$$f_3(x_i) = \sum_{i=1}^{n-1} [x_i^2 - 10\cos(2\pi x_i) + 10] \quad (9)$$

Griewank function:

$$f_4(x_i) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (10)$$

Schwefel function:

$$f_5(x_i) = 418.9829n + \sum_{i=1}^n x_i \sin\left(\sqrt{|x_i|}\right) \quad (11)$$

For all the problems we have considered three different cases with the search space dimensionality of $n = 10, 20$, and 30 . The range of the search space for each problem is provided in the following table.

Table 1

	Function name	Range
1.	Sphere	$(-100, 100)^n$
2.	Rosenbrock	$(-100, 100)^n$
3.	Rastrigin	$(-10, 10)^n$
4.	Griewank	$(-600, 600)^n$
5.	Schwefel	$(-500, 500)^n$

B. Gene Modeling

For over 40 years plant physiologists and systems modelers have used simulation models to predict plant growth and development varietal characteristics and environmental inputs [1]. Recently, the explosion of genomic science has opened the possibility of doing the same thing using dynamic models of gene expression [13-15]. Gene networks are apparently modular at the small scale [16]. 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 [16]. 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., 17].

In the molecular genetic model plant, *Arabidopsis thaliana*, the three genes *TERMINAL FLOWERING 1 (TFL1)*, *APETALA 1 (API)*, and *LEAFY (LFY)* play a special role in flowering. Connected together into a positive (actually, “double network”) feedback loop, these genes comprise a developmental switch, which, when flipped, signals plant commitment to flower. We have modeled this switch using a coupled set of differential equations,

$$\begin{aligned} \frac{d}{dt} LFY &= R_L g(SOC1, TFL1) - \lambda_L LFY \\ \frac{d}{dt} API &= R_H h_{up}(LFY) - \lambda_H API \\ \frac{d}{dt} TFL1 &= R_T h_{down}(API) - \lambda_T TFL1 \end{aligned} \quad (12)$$

where h_{up} and h_{down} are, respectively, promotive ($n = 3$) and repressive ($n = -3$) Hill [17] functions

$$h_i(x) = \frac{a_i^n}{a_i^n + K_i^n}, i = LFY, API, TFL1 \quad (13)$$

The external input to the switch is the expression level of the *SUPPRESSOR OF OVEREXPRESSION OF CO (SOC1)* gene, which is a linear, ramped sinusoid. The steepness of the ramp and the amplitude of its

oscillations relate to the rate of progress toward flowering. The *SOC1* equation is,

$$SOC1(t_m) = b_s t_m + \frac{(a_s - b_s)}{2} t_m \sin\left(\frac{2\pi t_m}{p_s}\right)$$

(14)

The $g(\cdot)$ function in equation (12) folds the *SOC1* input into the system dynamics. This function uses a repressive Hill function whose input is (*TFL1-SOC1*), limited to prevent biochemically impossible negative values. The complete list of parameters to be estimated is : $R_L, R_H, R_T, \lambda_L, \lambda_H, \lambda_T, K_{LFY}, K_{API},$ and K_{TFL1} , a total of 9 parameters. The parameters of *SOC1* in equation (14) are known *a priori*.

The network is simulated and the predicted *LFY* and *API* values are compared with corresponding generated data. The mean squared error (MSE) in each of the *LFY* and *API* predictions with the actual data is computed. The goal is to find a set of parameters that minimize the sum of the MSEs for both the genes.

V. Results

A. Benchmark Problems

The proposed algorithm was applied to the five benchmarks discussed earlier. Simulations were carried out to study the outcome of each of the following cases on the performance of the algorithm,

- (i) effect of increasing the number of dimensions,
- (ii) effectiveness of k -means clustering, and,
- (iii) effect of the number of simplex flips.

Therefore the algorithm was applied to 10, 20 and 30 dimensional versions separately. In each case, experiments were carried out with the total number of flips of the Nelder Mead algorithm being kept at 0, 5, 10, 15, and 20 flips per iteration of the PSO. Two separate sets of data were collected. In the first set, the Nelder Mead algorithm was applied to randomly formed clusters of $(n+1)$ points each, while in the other, the clusters were determined by applying 10 steps of the k -means per PSO iteration.

The results reported here are averaged over thirty runs in each case. In all runs, the constants, C_1 and C_2 were maintained at equal values of 2.05 each, so that they add up to 4.1 [1].

The results are shown in the form of tables 3. through 6. Each table shows the average value obtained by the algorithm for $n = 10, 20$ and 30 dimensions at the end of a fixed number of iterations. The best average solution is indicated in bold italics. In each case, it is seen that with $n = 10$ (low dimensionality) the algorithm performed best with an intermediate number of flips. However, when n is increased to 20, the best performing algorithm was the

one 20 flips, except Rosenbrock function, which was best at 15 flips and for Schwefel's function where zero flips was optimal. When the dimensions was further increased to $n = 30$, with the exception of Schwefel's function, in all cases, a total of 20 flips per iteration was best. The results show that in most cases, when the dimensionality of the problem increases, applying the Nelder Mead algorithm becomes more effective.

The tables also report simulations with both random and k -means based clustering. In only 3 of the 15 cases did random clustering was able to provide good performance. In two cases (Schwefel's with $n = 20, 30$), did both random and k -means produce equally good results. However, in the majority of the cases (10 out of 15), it was found that applying k -means clearly improves the performance of the algorithm.

Table 2.

Sphere – 10 Dim – After 2750 Fn Eval		
Number of Flips	With K-Means	Without K-Means
0	0.00040485	0.00040485
5	0.00042472	0.00066955
10	0.00014924	1.2174e-005
15	4.9375e-005	5.5108e-005
20	0.00015451	0.00019932
Sphere – 20 Dim – After 10500 Fn Eval		
0	0.0035386	0.0035386
5	0.0024636	0.014566
10	2.7063e-005	0.00049023
15	1.4602e-005	5.5457e-006
20	3.8031e-006	1.2349e-005
Sphere – 30 Dim – After 15500 Fn Eval		
0	4.2537	4.2537
5	0.73664	1.7161
10	0.021013	0.40639
15	0.0014054	0.43383
20	0.0010956	0.13405

Table 3.

Rosenbrock – 10 Dim – After 2750 Fn Eval		
Number of Flips	With K-Means	Without K-Means
0	32.9772	32.9772
5	38.134	0.43291
10	3.2352	2.572
15	0.32936	0.45436
20	0.31413	0.43519
Rosenbrock – 20 Dim – After 10500 Fn Eval		
0	11540.5982	11540.5982
5	18.3852	4155.4821
10	6.0476	9.2883
15	0.65431	3.5174
20	0.74177	25.78

Rosenbrock – 30 Dim – After 15500 Fn Eval		
0	6.2794	6.2794
5	6.7476	7.8239
10	5.6826	4.0287
15	4.8945	9.0754
20	3.7156	4.1428

Table 4.

Rastrigin– 10 Dim – After 2750 Fn Eval		
Number of Flips	With K-Means	Without K-Means
0	0.93684	0.93684
5	1.5267	1.1091
10	1.1943	1.2097
15	0.76329	0.4098
20	0.39816	0.33204
Rastrigin – 20 Dim – After 7875 Fn Eval		
0	2.6908	2.6908
5	2.3281	2.7591
10	2.5585	3.2984
15	3.0857	3.2855
20	3.2512	1.9938
Rastrigin – 30 Dim – After 11625 Fn Eval		
0	6.5327	6.5327
5	6.8979	8.538
10	6.0263	6.034
15	4.9698	9.3011
20	3.7702	5.6372

Table 5.

Griewank – 10 Dim – After 1650 Fn Eval		
Number of Flips	With K-Means	Without K-Means
0	4.6045e-005	4.6045e-005
5	3.9455e-005	0.0001805
10	0.00021789	0.00050014
15	7.7409e-005	0.00012245
20	7.06e-005	0.00019776
Griewank – 20 Dim – After 5250 Fn Eval		
0	0.034408	0.034408
5	0.0068742	0.0031565
10	0.00079559	0.00088602
15	0.0014959	0.0021034
20	0.00017033	0.00054215
Griewank – 30 Dim – After 7750 Fn Eval		
0	0.11102	0.11102
5	0.022642	0.084634
10	0.013758	0.029152
15	0.0052929	0.090387
20	0.0039921	0.069751

Lastly, we see that only in the case of Schwefel's function with $n = 20$ and 30 , that performance without the use of simplex is better than a hybrid approach. In 13 of the 15 cases, the hybrid algorithm was more effective.

Sample plots from our experiments are shown later on for additional clarity (Figs. 5 – 8).

Table 6.

Schwefel – 10 Dim – After 5500 Fn Eval		
Number of Flips	With K-Means	Without K-Means
0	31.6847	31.6847
5	46.8197	27.7368
10	31.0268	39.5817
15	21.9379	35.634
20	42.8208	47.376
Schwefel – 20 Dim – After 15750 Fn Eval		
0	37.6273	37.6273
5	92.1217	73.0983
10	77.6462	111.2048
15	131.607	110.5525
20	125.0749	134.8983
Schwefel – 30 Dim – After 31000 Fn Eval		
0	111.9715	111.9715
5	120.3109	115.5724
10	123.2762	114.7503
15	171.7393	157.3345
20	132.9723	155.3036

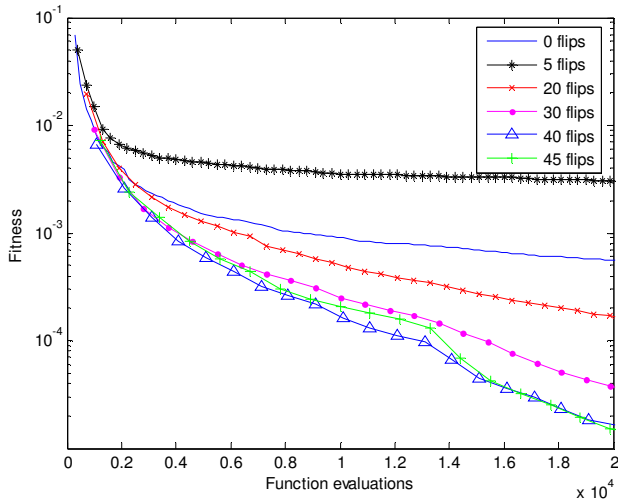


Fig. 3. Simulations to compare different number of simplex flips in the gene modeling problem

B. Benchmark Problems

The proposed algorithm was applied to the genomics problem discussed earlier. The algorithm was run with 0, 5, 20, 30, 40, and 45 flips of the Nelder Mead algorithm per iteration of PSO. The results are shown in Fig. 3 and Fig. 4. In Fig 3., it is clear that when increasing the number of flips to 40 the algorithm converged the fastest, deteriorating only when it reached 45 flips.

Fig 4. shows the data and the output gene expressions produced by the model using the parameters obtained by the best sample run.

VI. Conclusions

A hybrid algorithm that performs significantly better than basic particle swarm optimization have been proposed in this research. When tested with several standard benchmarks the results indicate that the hybridization significantly increases the performance over basic PSO. The results with parameter estimation of gene network modeling has shown that the proposed modifications to PSO reduce the function evaluations required by the algorithm is indeed effective. The approach is currently being extended to multi-objective PSO, where fuzzy dominance is used as a metric by the simplex [9 – 11].

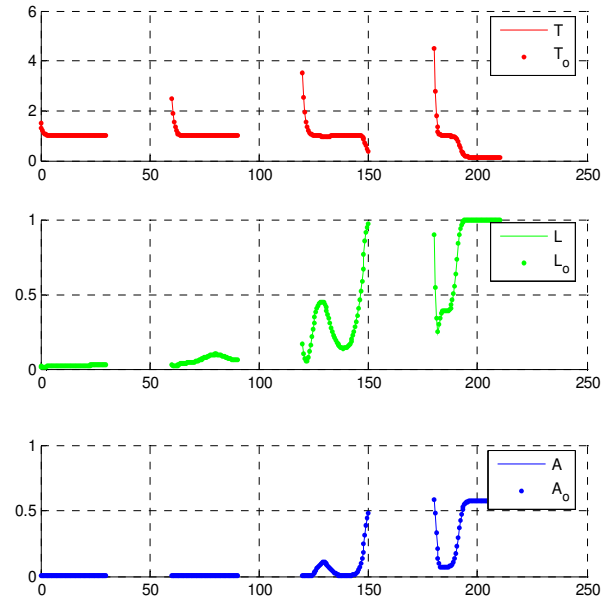


Fig. 4. Simulations of the gene network model. Along the x-axis is shown time in hrs, and the y-axis pertains to the gene expression level. The discrete points T_o , L_o , and A_o are collected data points while the solid lines T, L, and A are the results of matching the differential equations to the data using PSO.

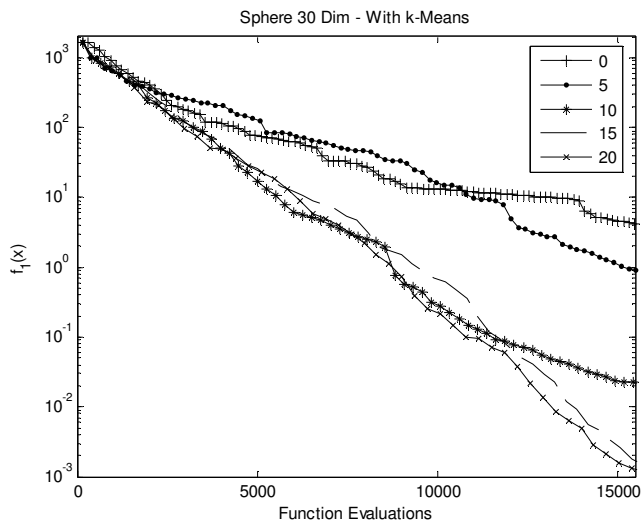


Fig 5. Convergence of the mean of the best solution found for 30 different runs for Sphere function vs. Function evaluations, with 30 parameters (using k -means)

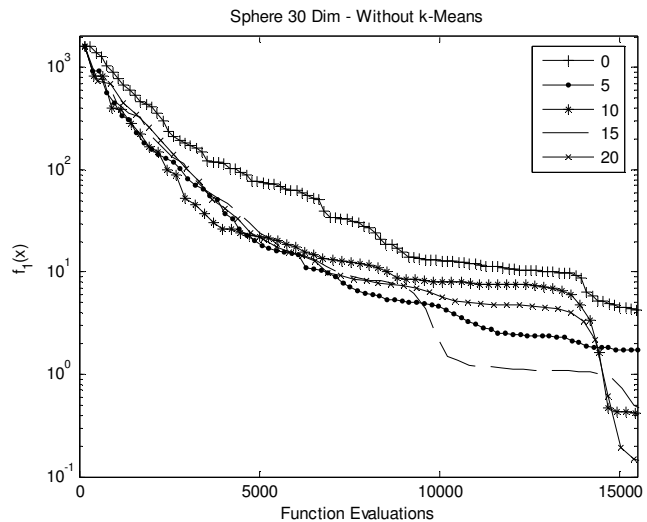


Fig 6. Convergence of the mean of the best solution found for 30 different runs for Sphere function vs. Function evaluations, with 30 parameters (without using k -means)

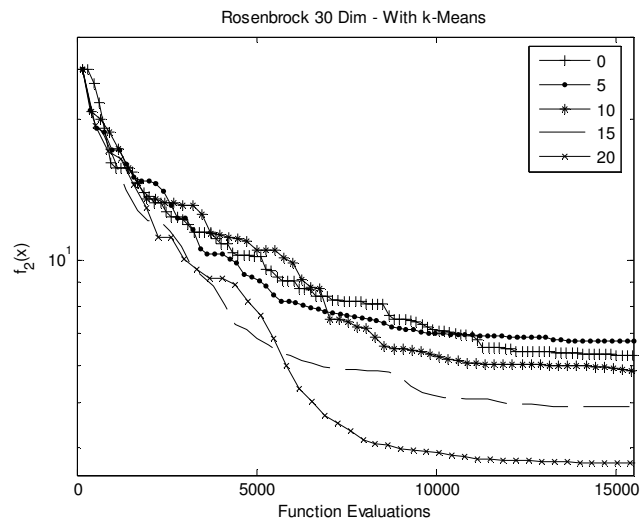


Fig 7. Convergence of the mean of the best solution found for 30 different runs for Rosenbrock function vs. Function evaluations, with 30 parameters (using k -means)

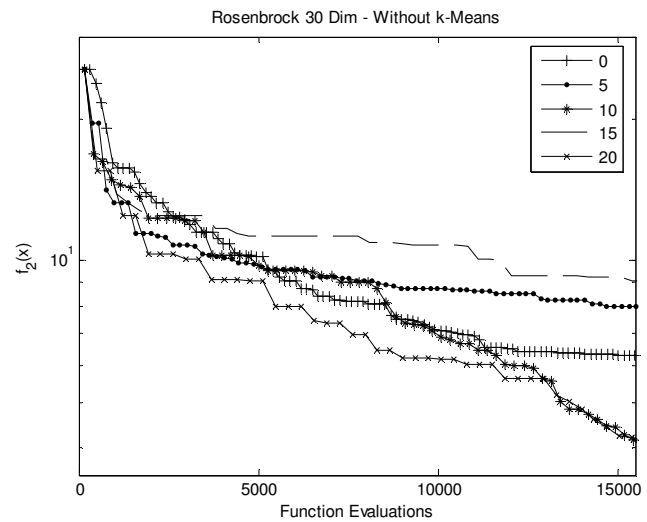


Fig 8. Convergence of the mean of the best solution found for 30 different runs for Rosenbrock function vs. Function evaluations, with 30 parameters (without using k -means)

REFERENCES

- [1] 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, pp. 58-73, 2002.
- [2] K. E. Parsopoulos and M. N. Vrahatis, M. N., "Recent approaches to global optimization problems through particle swarm optimization," *Natural Computing*, vol. 1, no. 2-3, pp. 235-306, 2002.
- [3] A. Carlisle and G. Dozier, "An off-the-shelf PSO", *Proceedings of the Workshop on Particle Swarm Optimization*, 2001, Indianapolis, IN. 2001.
- [4] B. Birge, "PSOT: a particle swarm optimization toolbox for use with MATLAB". Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003), Indianapolis, Indiana, USA. pp. 182-186, 2003.
- [5] J. A. Nelder and R. A. Mead, "A simplex method for function minimization", *Computer Journal*, Vol 7 no. 4, pp. 308-313, 1965.
- [6] Renders, J.M., and Flasse, S.P., Hybrid methods using genetic algorithms for global optimization, *IEEE Transactions on Systems, Man and Cybernetics Part-B*, Vol. 28 no. 2 , pp. 73-91, Apr. 1998.
- [7] Mohamed B. Trabia , "A Hybrid Fuzzy Simplex Genetic Algorithm", *Journal of Mechanical Design*, Volume 126, No. 6, pp. 969-97, November 2004.
- [8] Yen, J., Liao, J.C., Lee, B., and Randolph, D., A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method, *IEEE Transactions on Systems, Man and Cybernetics Part-B*, Vol. 28 no. 2, pp. 173-191, April. 1998.
- [9] P. Koduru, S. Das, S. M. Welch, J. L. Roe, "Fuzzy Dominance Based Multi-objective GA-Simplex Hybrid Algorithms Applied to Gene Network Models", *Lecture Notes in Computer Science: Proceedings of the Genetic and Evolutionary Computing Conference*, Seattle, Washington, (Eds. Kalyanmoy Deb *et al.*), Springer-Verlag, Vol 3102, pp. 356-367, 2004.
- [10] P. Koduru, S. Das, S. M. Welch, J. L. Roe, "A Multi-objective GA-Simplex Hybrid Approach for Gene Regulatory Network Models" *IEEE International Congress on Evolutionary Computation*, Portland, Oregon, pp. 2084-2090, June 2004.
- [11] P. Koduru, S. Das, S. M. Welch, J. Roe, Z. P. Lopez-Dee, "A Co-evolutionary Hybrid Algorithm for Multi-objective Optimization of Gene Regulatory Network Models", *Proceedings of the Genetic and Evolutionary Computing Conference*, Washington D. C., 2005.
- [12] T. R. Sinclair and N. G. Seligman, N.G., "Crop modelling, From infancy to maturity", *Agron. J.*, Vol. 88, pp. 698-704, 1966.
- [13] 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, pp. 71-81, 2003.
- [14] 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, Au., Sep 26 – Oct 1, 2004.
- [15] Z. Dong, "Incorporation of genomic information into the simulation of flowering time in *Arabidopsis thaliana*", Ph.D. dissertation, Kansas State University, 2003.
- [16] 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.
- [17] A. V. Hill, "The possible effect of aggregation of molecules of haemoglobin on its dissociation curves", *J. Physiol.* 40 (1910), iv-viii.