Multi-Objective Hybrid PSO Using ε-Fuzzy Dominance

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

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

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

ABSTRACT
This paper describes a PSO-Nelder Mead Simplex hybrid multi-objective optimization algorithm based on a numerical metric called ε -fuzzy dominance. Within each iteration of this approach, in addition to the position and velocity update of each particle using PSO, the *k*-means algorithm is applied to divide the population into smaller sized clusters. The Nelder-Mead simplex algorithm is used separately within each cluster for added local search. The proposed algorithm is shown to perform better than MOPSO on several test problems as well as for the optimization of a genetic model for flowering time control in *Arabidopsis*. Adding the local search achieves faster convergence, an important feature in computationally intensive optimization of gene networks.

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
Multi-objective optimization, evolutionary algorithms, fuzzy dominance, particle swarm optimization.

1. INTRODUCTION
Biologically inspired metaphors such as those based on Darwinian evolution or swarm intelligence are increasingly being applied to hard optimization problems. The great success of these approaches is because biologically inpired optimization algorithms: (*i*) are derivative-free techniques, (*ii*) do not easily 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.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

A very recent algorithm belonging to the class of biologically inspired approaches is Particle Swarm Optimization (PSO) [8]. PSO is a population-based approach that maintains a set of candidate solutions, called particles, which move within the search space. The trajectory followed by each particle is guided by its own memory, as well as by its interaction with other particles. The specific method of adjusting the particles trajectory is motivated by the interaction of birds, fishes, or other organisms that move in swarms. Eventually, the particles converge to suitable optima. We will use the terms particle and solution interchangeably henceforth.

Multi-objective optimization has been the focus of much recent research. Unlike in single-objective optimization where it is easy to compare one solution to another, in multi-objective problems, a solution that is inferior to another one in one objective, may in fact be better in another. Under these circumstances, the concept of Pareto optimality is used. Given a population S of solutions, a solution *u* is considered to *dominate* another solution *v* iff it is at least as good as *v* along all objectives, and furthermore, better in at least one. The *non-dominated set* is defined as the subset of S that contains all the non-dominated solutions. The non-dominated set of the entire solution space is called the *Pareto set*. Its corresponding image in the space of all objective functions is known as the *Pareto front*. Since all the solutions in the Pareto set are non-dominated, they must be treated as all equally good. Therefore, the goal of an effective multi-objective optimization algorithm is to find candidate solutions whose images in the objective function space are (*i*) are as close to the true Pareto front as possible, and (*ii*) are also as spread out and evenly spaced as possible, thereby sampling an extensive region of the Pareto front. These two conditions will be referred to as *convergence* and *diversity* respectively for the remainder of this paper.

Multi-objective versions of PSO have been recently proposed [9, 10, 11]. In [11], an *archive* of all non-dominated solutions found is maintained. The velocities of the particles in the population are redirected towards archive solution. In order to preserve diversity, this process is carried out in a probabilistic manner, with archive solutions in sparser regions influencing more particles in the population. In [9], archive solutions that dominate the least number of population solutions can be used as a reorient the velocities of the latter as a method to enforce diversity. The concept of ε-dominance, which is an extension of the dominance, has been explored in [10].

While biologically motivated algorithms such as PSO are very effective in providing optimal solutions, they can be further

improved by maintaining the correct balance between exploration and exploitation. Adding an exploitative component allows the algorithm to make use of local information to guide the search towards better regions in the search space. This property lets the algorithm convergence towards the Pareto front using fewer function evaluations – a much-desired characteristic in applications such as gene regulatory network modeling, where a substantial amount of computation is involved in evaluating each objective function. PSO hybrid algorithms for single objective optimization was proposed in [6, 7]. Although in [6], gradient term has been used indirectly through an extra term in the particles' velocity updates, the algorithm suggested in [7] explicitly implements local search through a separate Nelder-Mead operator [4], which iteratively replaces the worst particles with more promising ones.

This paper introduces a novel multi-objective PSO algorithm as well as hybrid approach that combines PSO with Nelder-Mead based local search as in [7]. It also introduces a new metric, *ε-fuzzy dominance*., for measuring the relative fitness of solutions in a multi-objective setting. This metric, ε-fuzzy dominance is a variation of the recently proposed concept of fuzzy dominance [1, 5]. Fuzzy dominance has proved to be highly effective in multi-objective genetic algorithms [1, 5], producing significantly faster convergence for the most difficult multi-objective test functions as well as when applied to gene network parameter estimation in comparison to NSGA-II [2]. In this paper, ε-fuzzy dominance has been used in two ways. In the strict PSO algorithm, it is applied to the population individuals in order to discriminate between ones that are closer to the non-dominated front less dominated from those further behind. Additionally, when PSO is hybridized, ε-fuzzy dominance provides an effective metric for the Nelder-Mead algorithm to pick out the worst solutions for replacement.

The hybrid algorithm proposed here combines the Nelder-Mead search algorithm with PSO in a manner similar to [7]. In each iteration, the particles are divided into more localized clusters by means of the well-known *k*-means algorithm. The Nelder-Mead operation is carried out separately for each cluster. A single step of PSO is then carried out to compute the new positions and velocities of the particles.

The hybrid algorithm was specifically applied to the problem of estimating the parameters of a differential equation model of gene networks. Unlike standard test functions that are developed specifically to have rugged fitness landscapes, in applications such as this, the landscape is relatively smooth. This makes it more suitable for local search operations. In an earlier work, it has been observed that the Nelder Mead process has a greater role in speeding up the hybrid algorithm's convergence towards the Pareto front, in comparison to benchmarks such as the ZDT test suite [2].

2. APPROACH

2.1 Fuzzy ε-Dominance

Without a loss of generality, let us assume that the multi-objective problem entails the minimization of each of M objective functions $e_i(.)$, $i = 1 \ldots M$. The solution space is denoted as $\Psi \subset \Re^n$. Given a monotonically non-decreasing function $\mu_i^{dom}(\cdot)$, whose range is in [0, 1], $i \in \{1,2,\ldots,n\}$, a solution $u \in \Psi$ is said to *i*-dominate solution $v \in \Psi$, if and only if $e_i(u) < e_i(v)$. This relationship can be denoted as $u \succ_i^F v$. If $u \succ_i^F v$, the degree of fuzzy *i*-dominance is equal to $\mu_i^{dom}(e_i(v) - e_i(u)) \equiv \mu_i^{dom}(u \succ_i^F v)$. Fuzzy dominance can be regarded as a fuzzy relationship $u \succ_i^F v$ between u and v. Solution $u \in \Psi$ is said to fuzzy dominate $v \in \Psi$ if and only if $\forall i \in \{1,2,\ldots,M\}$, $u \succ_i^F v$. This relationship can be denoted as $u \succ^F v$. The degree of fuzzy dominance can be defined by invoking the concept of fuzzy intersection and using a *t*-norm,

$$\mu^{dom}(u \succ^F v) = \bigcap_{i=1}^{M} \mu_i^{dom}(u \succ_i^F v) \quad (1)$$

In the previous implementation of fuzzy dominance [1, 5], the membership functions $\mu_i^{dom}(\cdot)$ used to compute the fuzzy *i*-dominances were defined to be zero for negative arguments. Therefore, whenever $e_i(u) > e_i(v)$, the degree of fuzzy dominance $u \succ_i^F v$ was necessarily zero. In this paper, we allow non-zero values. The membership functions used are trapezoidal, yielding nonzero values whenever their arguments are to the right of a threshold ε, as shown in figure 1 below.

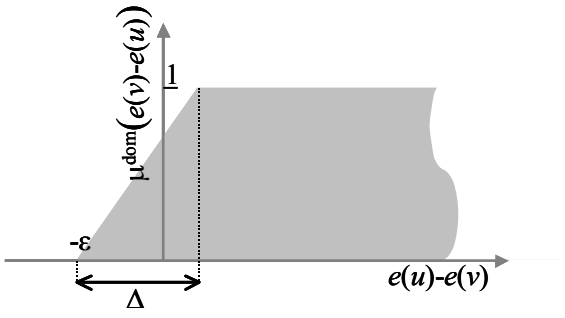

Figure 1. Fuzzy membership functions used here to compute ε-dominances.

Mathematically, the memberships $\mu_i^{dom}(u \succ_i^F v)$ are defined as,

$$\mu_i^{dom}(\Delta e_i) = \begin{cases} 0, & \Delta e_i \le -\varepsilon \\ (\Delta e_i)/\Delta_i & -\varepsilon < \Delta e_i < \Delta_i - \varepsilon \\ 1, & \Delta e_i \ge \Delta_i - \varepsilon \end{cases} \quad (2)$$

where, $\Delta e_i = e_i(v) - e_i(u)$. Given a population of solutions $S \subset \Psi$, a solution $v \in S$ is said to be fuzzy dominated in S iff it is fuzzy dominated by any other solution $u \in S$. In this case, the degree of fuzzy dominance can be computed by performing a union operation over every possible $\mu^{dom}(u \succ^F v)$, carried out using *t*-co norms as,

$$\mu^{dom}(S \succ^F v) = \bigcup_{u \in S} \mu^{dom}(u \succ^F v) \quad (3)$$

In this manner, each solution can be assigned a single measure to reflect the amount it dominates others in a population. Better solutions within a set will be assigned lower fuzzy dominances, although, unlike in [1, 5] non-dominated solution may not necessarily be assigned zero values. The union and intersection operators follow the standard min and max definitions [3]. Further details about fuzzy dominance can be obtained from [1]. For optimization problems that involve constraints in either parameter or objective domain, we use a penalty term, which is equal to the total number of constraints violated by a solution and add the value to its fuzzy dominance. Since the value of fuzzy dominances obtained from equation (3) can never exceed unity, invalid solutions are always assigned higher fuzzy dominances than valid ones, and hence are least favored.

In order to compare multiple solutions having similar fuzzy dominance values we use a diversity fitness value which is equal to the perimeter of the largest *M*-dimensional hypercube in the objective space which encloses the same solution but no others [2]. The value of the perimeter $I(v)$ of the enclosing cuboid for any solution v is given as,

$$I(v) = \sum_{i=1}^{M} \left(e_i(u) - e_i(w)\right) / \left(\max(e_i) - \min(e_i)\right) \quad (4)$$

where u, and w are solutions adjacent to v when the merged population is sorted in ascending order according to the i^{th} objective, e_i. Boundary solutions are assigned ∞ values. Larger values of $I(v)$ imply sparseness in the region of the solution v. In comparing multiple solutions with similar fuzzy dominance values, the preference is given to solutions with higher values of $I(v)$.

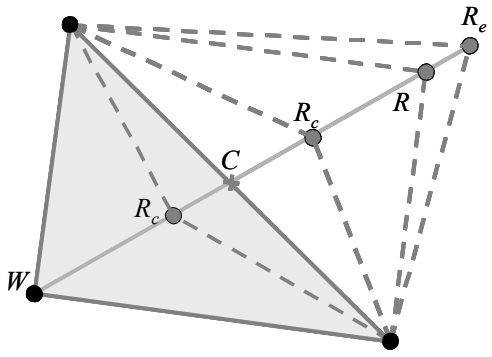

Figure 2. The simplex used by Nelder-Mead search in 3 dimensions. The worst point *W* is replaced with either *R*, R_e or R_c.

2.2 Clustering and Local Search

The Nelder-Mead approach employs a *simplex*, which, in *n*-dimensions, consists of $n+1$ solutions, $X(k)$, $k = \{1, 2, \ldots n+1\}$ [4] (*i.e.*, a triangle in a two dimensional plane – see figure 2). The solutions are evaluated in each step and the worst solution W is identified. The centroid of the simplex is computed as,

$$C = (\Sigma X(k))/n \quad (5)$$

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

$$R = C + (C - W) \quad (6)$$

Usually, the worst point W is just replaced by the reflected point R. However, if the reflected point is better than any other solution in the simplex, the simplex is further expanded as,

$$R_e = C + \eta(C - W) \quad (7)$$

where η is an expansion coefficient. But, if R is worse than W, the simplex is contracted and the reflected solution is placed on the same side of the centroid. When R is better than W and 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,

$$R_c = C \pm \kappa(C - W) \quad (8)$$

where κ is a 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*.

The *k*-means algorithm is useful to divide a population of solutions into separate clusters, where each cluster consists of proximally located solutions 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 for a predefined number of steps:

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

In this application, clusters must be disjoint, with only $n+1$ points each, so the algorithm was modified accordingly to assign only the only nearest $n+1$ points to a cluster. The unassigned points are then assigned to the next closest cluster with less than $n+1$ points.

2.3 Standard PSO

We first describe the variant of the standard PSO algorithm that has been used in the rest of the paper. This algorithm maintains a population of N particles whose positions, $X(i)$, $i = 1, 2, \ldots N$, are initialized to random values. These positions are incremented in each iteration t of the algorithm according to the instantaneous velocity $V_t(i)$, as follows,

$$X_{t+1}(i) = X_t(i) + V_t(i) \quad (9)$$

The velocity is also updated in each iteration, using the particle's own recorded previous best position, as well as the current location of the other particles. The update is

$$V_{t+1}(i) = \chi(V_t(i) + C_1 \times U[0,1] \times (X_{ib}(i) - X_t(i)) + C_2 \times U[0,1] \times (X_{gb,t} - X_t(i))) \quad (10)$$

In the above equation, C_1 and C_2 are two constants, called the *cognitive* and the *social* constants, and χ is a constriction coefficient, that helps in maintaining stability [1]. $U[0,1]$ 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 $\forall s \in \{0, 1, \ldots t\}$, $e(X_{t'}(i)) \le e(X_s(i))$, where $e(\cdot)$ is the objective function to be minimized. The other quantity, $X_{gb,t}$ is the *global best* position of any particle in the current iteration t.

In other words, $X_{gb,t} = X_t(j)$, for some j, such that $\forall k \in \{0, 1, \ldots N\}$, $e(X_t(j)) \leq e(X_t(k))$. For all the test problems, we have used: $\chi = 0.4$, $C_1 = 2.1$ and $C_2 = 2.1$.

2.4 Multi-objective PSO

In the case of multi-objective PSO, the choice of global best and individual best solutions is not straightforward [11]. There could be multiple non-dominated solutions in the current population that are potential candidates for being a global solution. At the same time, comparison of the so far best recorded position of a particle with its current position may not always be possible, as they could be incomparable in multiple objectives. Due to these reasons the standard PSO is generally modified to be applicable for multi-objective optimization [11, 13, 14]. In addition to that main objective of a multi-objective optimization algorithm is to find multiple uniformly distributed solutions on the Pareto front. Hence it is quite common in multi-objective algorithms to use an external archive that stores the non-dominating solutions obtained at end of every run [2, 11].

In the current work, we have also modified the standard PSO as detailed in Section 2.3 to be applicable for multi-objective optimization to have an archive. The archive stores the best N non-dominated solutions found so far by the PSO. This is obtained by using top best N solutions from the union of the solutions in current iteration and archive from previous generation, which is then sorting based on fuzzy dominance values.

As explained in Section 2.1, when multiple solutions have similar fuzzy dominance values, then solutions with the highest value of $I(v)$ are preferred. In equation (10), the global best solution ($X_{gb,t}$), is selected by doing a binary tournament selection from the individuals in the current archive. In deciding the personal best ($X_{pb,t}$), for a solution, we compare the particles current position and its position from previous iteration. The non-dominating solution is assigned as the new personal best. However, if the solutions are incomparable i.e. mutually non-dominant, then the current position of the particle is used as the updated personal best.

To avoid getting stuck in a local minimum a mutation is used in multi-objective PSO algorithms [11]. In the current work, we have implemented a turbulence factor into the velocity update equation (10), which is similar to a mutation operator in evolutionary algorithms. The modified update equation is given as:

$$\begin{aligned} V_{t+1}(i) = \chi(V_t(i) &+ C_1 \times U[0,1] \times (X_{ib}(i) - X_t(i)) \\ &+ C_2 \times U[0,1] \times (X_{gb,t} - X_t(i))) \\ &+ \exp(-\delta \times t) \times U[-1,1] \end{aligned} \quad (11)$$

where δ is the turbulence coefficient. The negative exponential term assures that the turbulence in the velocities is higher at the initial generations that aids in exploration, and later in the algorithm would help it in exploitation.

2.5 Multi-objective Hybrid PSO

The hybrid algorithm is detailed below:

1. $t = 0$.
2. Randomly initialize the particle positions $X_0(i)$, and set the initial archive A_0 to X_0.
3. Initialize all velocities, $V_0(i)$, to zeroes.
4. Evaluate the objective function for each $X_t(i)$.
5. Evaluate the fuzzy dominance in the population.
6. Update the archive A_i
7. Update each particle i's individual best and global best.
8. Randomly initialize k cluster centers.
9. Assign each particle i to a cluster using k-means.
10. For each cluster apply Nelder-Mead simplex.
11. Update velocity according to Equation (11).
12. Update positions $X_t(i)$ according to Equation (9).
13. $t = t + 1$.
14. If $t > t_{max}$, stop, else go to Step 4.

3. TEST PROBLEMS

In order to assess the performance of the proposed algorithm, we compare its performance with MOPSO [11], on the following standard test problems (KUR, ZDT1, ZDT3, ZDT4 and ZDT6) and a three gene network-modeling problem.

3.1 Standard Test Problems

The following standard test problems are commonly used to give a fair comparison of the multi-objective optimization algorithms in finding the Pareto front in different scenarios [2]. For all the standard test problems, the maximum number of allowed function evaluations was set at 12,000, except for ZDT4, which was given 20,000. The details of the different test functions are as given below.

$$\begin{aligned} KUR: \quad f_1(\mathbf{x}) &= \sum_{i=1}^{n-1} -10\exp(-0.2\sqrt{x_i^2 + x_{i+1}^2}) \\ f_2(\mathbf{x}) &= \sum_{i=1}^{n} (|x_i|^{0.8} + 5\sin(x_i)^3 \end{aligned} \quad (12)$$

where $n = 3$, $x_i \in [-5, 5]$, and has a discontinued Pareto front.

$$\begin{aligned} ZDT1: \quad f_1(\mathbf{x}) &= x_1 \\ g(\mathbf{x}) &= 1 + \frac{9\left(\sum_{i=2}^{n} x_i\right)}{(n-1)} \\ f_2(f_1(\mathbf{x}), g(\mathbf{x})) &= g(x)\left(1 - \sqrt{f_1(\mathbf{x})/g(\mathbf{x})}\right) \end{aligned} \quad (13)$$

where $n = 10$, $x_i \in [0, 1]$, and at the convex Pareto front, the function $g(\mathbf{x}) = 1$.

$$\begin{aligned} ZDT3: \quad f_1(\mathbf{x}) &= x_1 \\ g(\mathbf{x}) &= 1 + \frac{9\left(\sum_{i=2}^{n} x_i\right)}{(n-1)} \\ f_2(f_1(\mathbf{x}), g(\mathbf{x})) &= g(x) \times \\ &\left(1 - \sqrt{f_1(\mathbf{x})/g(\mathbf{x})} - (f_1(\mathbf{x})/g(\mathbf{x}))\sin(10\pi f_1(\mathbf{x}))\right) \end{aligned} \quad (14)$$

where $n = 10$, $x_i \in [0, 1]$, and a discontinuous Pareto front is formed with $g(\mathbf{x}) = 1$.

$$ZDT4: \; f_1(\mathbf{x}) = x_1$$

$$g(\mathbf{x}) = 1 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 - 10\cos(4\pi x_i)) \qquad (15)$$

$$f_2(f_1(\mathbf{x}), g(\mathbf{x})) = g(x)\left(1 - \sqrt{f_1(\mathbf{x})/g(\mathbf{x})}\right)$$

where n = 10, $x_1 \in [0, 1]$ and $x_2, x_3, \ldots x_n \in [-5, 5]$. The Pareto front is defined as the locus of all points such that $g(\mathbf{x}) = 1$. The ZDT4 problem and tests an algorithm's robustness against multi-modality.

$$ZDT6: \; f_1(\mathbf{x}) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$$

$$g(\mathbf{x}) = 1 + 9\left(\sum_{i=2}^{n} x_i \Big/ (n-1)\right)^{0.25} \qquad (16)$$

$$f_2(f_1(\mathbf{x}), g(\mathbf{x})) = g(x)\left(1 - \left(f_1(\mathbf{x})/g(\mathbf{x})\right)^2\right)$$

where n = 10, $x_i \in [0, 1]$ and the non-convex Pareto front is formed with $g(\mathbf{x}) = 1$.

3.2 Modeling Flowering Time Control in *Arabidopsis*

In the molecular genetic model plant, *Arabidopsis thaliana*, three genes *TERMINAL FLOWERING 1* (*TFL1*), *APETALA 1* (*AP1*), and *LEAFY* (*LFY*) play a special role in flowering [6]. OFF to ON state changes in two of them (*AP1* and *LFY*) signal plant commitment to flowering. All three interact so closely in regulating each other that is not possible to completely disentangle their linkages based on extant experimental data. One possible model, however, is that of a three-element positive feedback loop comprising a bistable switch. A model for this switch is the coupled set of differential equations,

$$\frac{d}{dt}LFY = R_L g(SOC1, TFL1) - \lambda_L LFY$$

$$\frac{d}{dt}AP1 = R_H h_{up}(LFY) - \lambda_H AP1 \qquad (17)$$

$$\frac{d}{dt}TFL1 = R_T h_{dwn}(AP1) - \lambda_T TFL1$$

where h_{up} and h_{dwn} are, respectively, promotive (n=3) and repressive (n = -3) Hill functions [15] defined as,

$$h_i(x) = \frac{a_i^n}{a_i^n + K_x^n} \qquad (18)$$

where n is a *cooperativity coefficient*. The function g is a repressive Hill function whose input is (*TFL1*-*SOC1*). The difference input to g is restricted to positive values; negative biochemical concentrations are impossible.

The *TFL1* gene becomes progressively more active with time in the apex of the growing shoot [16]. It influences the activity of *LFY* and *AP1* in leaf primordia that sequentially emerge from the apex. It is not known precisely how this molecular influence is exerted, but the result is to slow the plant's development toward flowering. Increasing levels of *LFY* and *AP1* within each primordium offset this effect. Ultimately, the switch changes state, causing the primordium within which this happens to initiate inflorescence development. Equation (17) models all these effects as if they are direct, but they may not be so.

External switch input is provided by the expression level of the *SUPRESSOR 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. An equation for *SOC1* expression levels is,

$$SOC1(t_m) = b_s t_m + \frac{(a_s - b_s)}{2} t_m \sin\left(\frac{2\pi t_m}{p_s}\right) \qquad (19)$$

A synthetic data set of 244 data points was generated covering a period of nine days and the emergence of four leaf primordia. Each primordia was simulated until it could either be assumed to be committed to leaf development (at 30 hrs) or until its floral switch copy changed state, which ever came first.

The objective is to find the set of nine parameters in equation (17) :[R_L, R_H, R_T, λ_L, λ_H, λ_T, K_{LFY}, K_{AP1}, and K_{TFL1}], such that it simultaneously minimizes the mean squared error between the actual and simulated data for *AP*1 and *TFL*1.

4. PERFORMANCE COMPARISON

4.1 Experimental Setup

We compared the performance of the proposed algorithm with MOPSO [11], on standard test problems and a gene network problem, which are detailed in the previous section. In order to do a fair comparison, for each problem 10 independent runs are done using both the algorithms. The results presented are the average of all the runs. For all the runs, both the algorithms use a population size of 100 and an archive (or repository) size of 100. MOPSO was implemented using 30 divisions of adaptive grid and with a mutation rate of 0.5. For the hybrid PSO, as we have a population size of 100 and all the problems that have nearly 10 parameters we have used 9 k-means clusters. In case of test problem KUR that has only 3 parameters, a population size of 40 was used. The Nelder-Mead simplex algorithm was implemented with η=1.5 and κ=0.5. Two simplex flips were done per iteration within each cluster. When a point in the cluster is flipped using the simplex, the position of the particle changes. Hence the old velocity would have no relevance with the particle's new position. To over come this problem we have opted two different methods to update the velocity of the new particle.

1. Random: Reinitialize the velocity of the particle randomly.
2. Flip Update: The velocity of the particle undergoes the exact same operations as the position in Nelder-Mead simplex.

4.2 Performance Metrics

The following metrics are used to provide a qualitative assessment of the performance of both the algorithms on the standard test problems.

4.2.1 Generational Distance

In case of the benchmark test problems, a target set can be created to compute the convergence by distributing points on the Pareto front, which is known a priori. The minimum distance from the non-dominated solutions found by each algorithm to the uniformly distributed solutions on the known front is a good convergence measure [17]. The procedure for calculating this metric is:

1. Create a target set Γ^*, of non-dominated solutions from the known Pareto front
2. For every generation of the evolutionary algorithm, identify the set of non-dominated solutions $P^*(t)$, in the

current population $P(t)$

3. For each solution i, in $P^*(t)$ compute the smallest Euclidian distance d_i to all points in Γ^* as:

$$d_i = \min_{j=1}^{|\Gamma^*|} \sqrt{\sum_{k=1}^{M} \left(f_k(i) - f_k(j)\right)^2} \tag{20}$$

where M is the number of objectives and $f_k(.)$ is the fitness of an individual for objective k, and $|.|$ denotes the cardinality of the set.

4. The generational distance, GD, is the average of d_i, for all non-dominated solutions in the current population:

$$GD = \frac{\sum_{i=1}^{|P^*(t)|} d_i}{|P^*(t)|} \tag{21}$$

4.2.2 Spacing Metric

In addition to comparing the convergence to the actual Pareto front, we also need to get a qualitative measure of the spread of the solutions in the final solution. The spacing metric (S) measures the spread of the solutions in the non-dominated front by evaluating the variance of the nearest distance between neighboring solutions [18]. The spacing metric can be evaluated as:

$$S = \sqrt{\frac{1}{(n-1)} \sum_{i=1}^{n} (\bar{d} - d_i)^2} \tag{22}$$

where $\bar{d}$ and d_i are defined in equation (23) and n is the number of solutions in the non-dominated front.

$$d_i = \min_{i,i \neq j} \left[\sum_{k=1}^{M} \left| f_k^i - f_k^j \right| \right]$$
$$\bar{d} = \frac{1}{n} \sum_{i=1}^{n} d_i \tag{23}$$

Lower values of S indicate a more even spread of solutions in the front, and a value of zero suggests that all the solutions are evenly spread out.

It should be noted that a comparison of the spacing metric is only meaningful when the different algorithms have similar generational distance values. In other words, a comparison of the diversity is only applicable when both algorithms have a similar convergence to the actual Pareto front. In addition to this, in the case of gene network problem, the actual Pareto front is unknown, hence the metrics are not evaluated for that problem. However, the comparison of the final non-dominated solutions obtained by both the algorithms would provide a good performance

5. RESULTS

Figures 3-8 show the plot of the final non-dominated solutions obtained by both the algorithms in one of the runs. Tables I and II show the performance metrics that have been averaged over all runs. Comparing these results, it is evident that the proposed multi-objective hybrid PSO algorithm has outperformed MOPSO on all the problems except in KUR, where both algorithms had similar performance. However, the number of parameters to be found in KUR is only 3. When dealing with problem with higher dimensions there is a significant difference between MOPSO and the proposed method. Comparing the different approaches to update the velocities in the simplex algorithm, the flip update scheme shows better results than random approach.

As we have no information on the detail of the actual Pareto front for the gene network problem, to compare we select an individual from the non-dominated solutions obtained and plot its response to the actual data in Figure 9. It can be noted that the prediction closely follows the actual data and hence a good fit of parameters have been obtained for the problem.

6. CONCLUSION

To achieve a better exploration and exploitation, global search algorithm are generally optimized with local search methods. Nelder-mead simplex algorithm has been a popular choice for hybridization due to its gradient free, fast local search capabilities. In this paper, the first multi-objective PSO hybrid algorithm has been presented. The hybrid algorithm was also shown to outperform MOPSO. Future work would be directed in using the hybrid algorithm for parameter estimation for more complex gene network models.

Table I. Results of Generations Distance for the two different approaches of hybrid PSO and MOPSO

PROBLEM	UPDATED HYBRID	RANDOM HYBRID	MOPSO
KUR	0.044856	0.063236	0.053350
ZDT1	0.004143	0.020045	0.585582
ZDT3	0.001961	0.065657	0.686700
ZDT4	0.004248	0.014319	0.280784
ZDT6	0.002295	1.552199	1.359427

Table II. Results of Spacing metric for the two different approaches of hybrid PSO and MOPSO

PROBLEM	UPDATED HYBRID	RANDOM HYBRID	MOPSO
KUR	0.074627	0.609093	0.190415
ZDT1	0.008679	0.068562	0.025590
ZDT3	0.012893	0.054150	0.009207
ZDT4	0.008132	0.127192	0.009876
ZDT6	0.007747	0.196983	0.076379

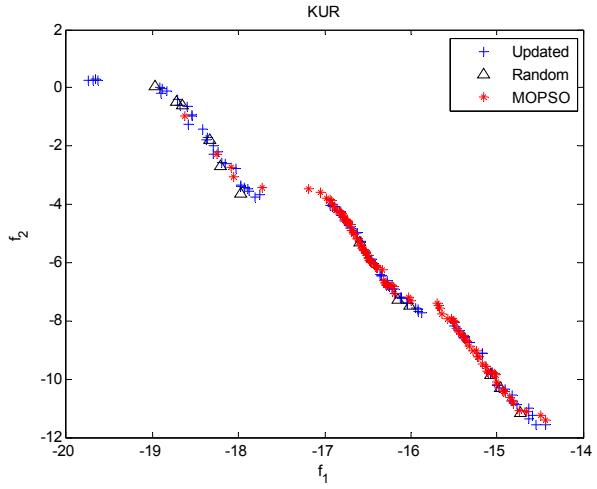

Figure 3. Sample Pareto fronts obtained by two different approaches of hybrid PSO and MOPSO for KUR

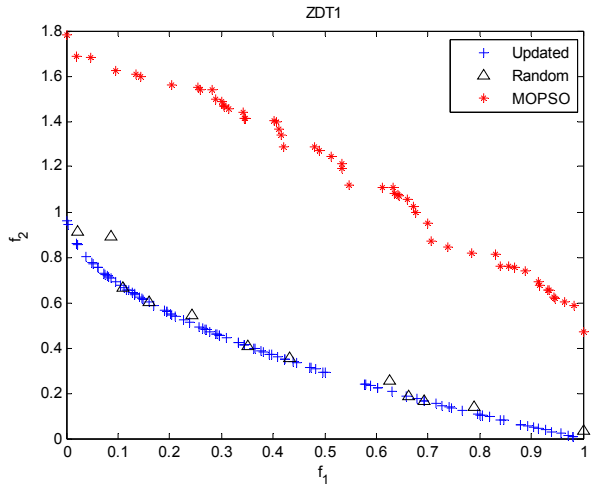

Figure 4. Sample Pareto fronts obtained by two different approaches of hybrid PSO and MOPSO for ZDT1

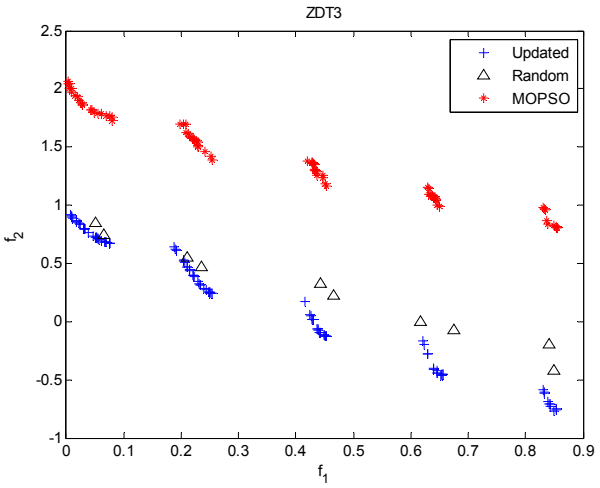

Figure 5. Sample Pareto fronts obtained by two different approaches of hybrid PSO and MOPSO for ZDT3

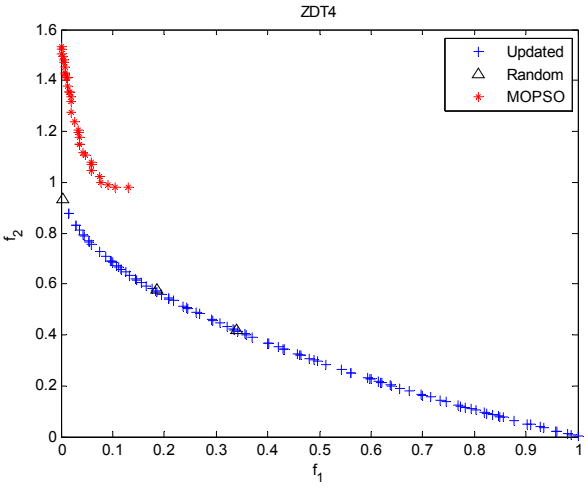

Figure 6. Sample Pareto fronts obtained by two different approaches of hybrid PSO and MOPSO for ZDT4

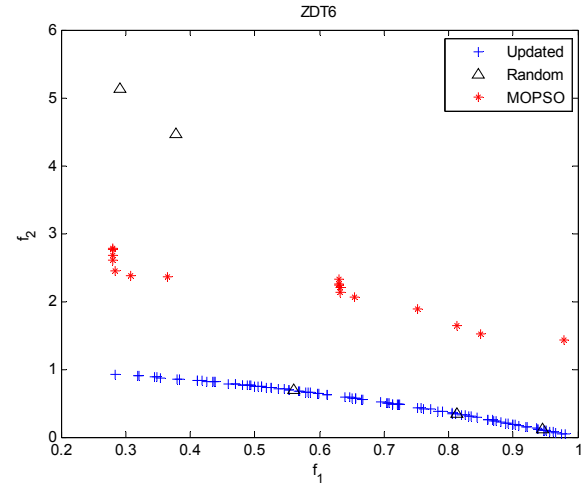

Figure 7. Sample Pareto fronts obtained by two different approaches of hybrid PSO and MOPSO for ZDT6

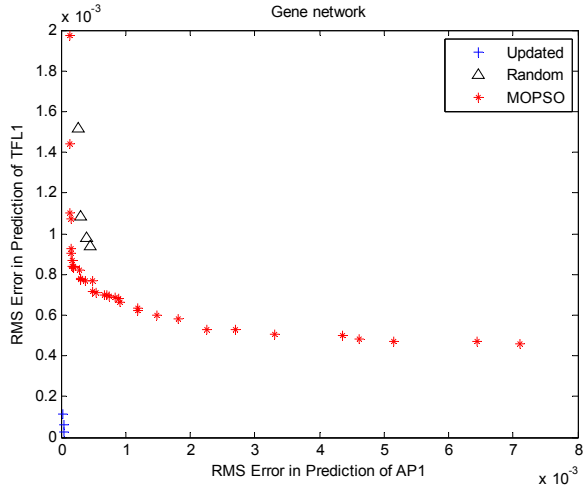

Figure 8. Sample Pareto fronts obtained by two different approaches of hybrid PSO and MOPSO for Gene Network

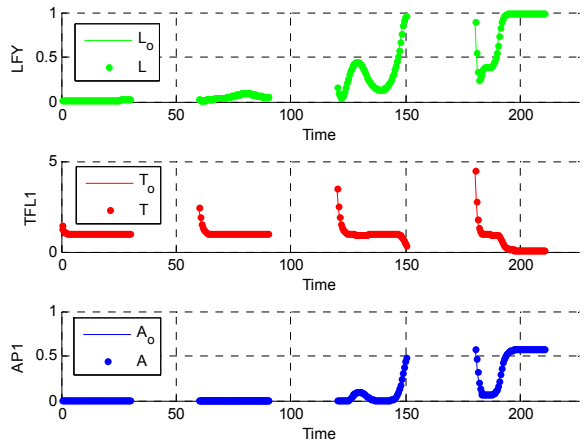

Figure 9. Simulation of one of the solution obtained by updated hybrid algorithm. (L_o, T_o, and A_o, represent the predicted output for LFY, TFL and AP1. L, T, and A, represent actual values for LFY, TFL and AP1)

7. REFERENCES

[1] 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, 356–367, 2004.

[2] Deb, K., Agrawal,S., Pratap, A. and Meyarivan, T., A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2), 182-197, 2002.

[3] Mendel, J.M.: Fuzzy logic systems for engineering: A tutorial, Proceedings of the IEEE, 83(3), 345-377, March 1995.

[4] J. A. Nelder and R. A. Mead, A simplex method for function minimization, *Computer Journal*, 7(4), 308-313, 1965.

[5] 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., 393–399, 2005.

[6] 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, BC, Canada, 2006.

[7] 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.

[8] M. Clerc and J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, *IEEE Transactions on Evolutionary Computation*, 6(1), 58-73, 2002.

[9] Julio E. Alvarez-Benitez, Richard M. Everson and Jonathan E. Fieldsend, A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts, in Carlos A. Coello Coello and Arturo Hernández Aguirre and Eckart Zitzler (editors), *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, 459-473, Springer. Lecture Notes in Computer Science Vol. 3410, Guanajuato, México, March 2005.

[10] Margarita Reyes Sierra and Carlos A. Coello Coello, Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and e-Dominance, in Carlos A. Coello Coello and Arturo Hernández Aguirre and Eckart Zitzler (editors), *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, 505-519, Springer. Lecture Notes in Computer Science Vol. 3410, Guanajuato, México, March 2005.

[11] Coello, C.A.C.; Pulido, G.T.; Lechuga, M.S., Handling multiple objectives with particle swarm optimization, IEEE Transactions on Evolutionary Computation, 8(3), 256-279, June 2004.

[12] Mendes, R.; Kennedy, J.; Neves, J., The fully informed particle swarm: simpler, maybe better, IEEE Transactions on Evolutionary Computation, 8(3), 204-210, June 2004.

[13] J. Moore and R. Chapman, Application of particle swarm to multiobjective optimization, Dept of Computer Science Software Engg., Auburn University, 1999.

[14] K.E. Parsopoulos and M.N. Vrahatis, Particle swarm optimization methods in multiobjective problems, in Proc. 2002 ACM Symp. Applied Computing, SAC, Spain, 603-607, 2002.

[15] A.V. Hill, The possible effect of aggregation of molecules of haemoglobin on its dissociation curves, *Journal of Physiology*, 40:iv-viii, 1910.

[16] O.J. Ratcliffe, D.J. Bradley and E.S. Coen, Separation of shoot and floral identity in Arabidopsis, *Development*, 126, 1109-1120, 1999.

[17] D. A. Van Veldhuizen and G.B. Lamont, Multiobjective evolutionary algorithms research: A history and analysis, Dept. Elec. Comput. Eng., Graduate School of Eng., Air Force Inst. Technol., Wright-Patterson AFB, OH, Tech. Rep. TR-98-03, 1998.

[18] Y. Collette and P. Siarry, Multiobjective Optimization, Springer, Berlin, 2003.