Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2008, Article ID 126970, 9 pages
doi:10.1155/2008/126970

Research Article

A Strongly Interacting Dynamic Particle Swarm Optimization Method

S. Kok and J. A. Snyman

Department of Mechanical and Aeronautical Engineering, University of Pretoria, Pretoria 0002, South Africa

Correspondence should be addressed to S. Kok, schalk.kok@up.ac.za

Received 24 July 2007; Accepted 18 December 2007

Recommended by Riccardo Poli

A novel dynamic interacting particle swarm optimization algorithm (DYN-PSO) is proposed. The algorithm can be considered to be the synthesis of two established trajectory methods for unconstrained minimization. In the new method, the minimization of a function is achieved through the dynamic motion of a strongly interacting particle swarm, where each particle in the swarm is simultaneously attracted by all other particles located at positions of lower function value. The force of attraction experienced by a particle at higher function value due to a particle at a lower function value is equal to the difference between the respective function-values divided by their stochastically perturbed position difference. The resultant motion of the particles under the influence of the attracting forces is computed by solving the associated equations of motion numerically. An energy dissipation strategy is applied to each particle. The specific chosen force law and the dissipation strategy result in the rapid collapse (convergence) of the swarm to a stationary point. Numerical results show that, in comparison to the standard particle swarm algorithm, the proposed DYN-PSO algorithm is promising.

1. INTRODUCTION

A new direct search method using only function values is proposed for finding a local minimizer $\mathbf{x}^*$ with associated function value f^* of a real valued function $f(\mathbf{x})$, $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \in R^n$. The method proposed may be considered as the synthesis of two unconventional trajectory methods for the unconstrained minimization of a multivariable function. The first is the *dynamic* method of Snyman [1, 2], and the second method is the particle swarm optimization (PSO) method of Eberhardt and Kennedy [3]. In the dynamic method, also known as the leap-frog algorithm, the minimum of the function is sought by considering the dynamic motion of a single particle of unit mass in an n-dimensional force field, where the potential energy of the particle is represented by the function to be minimized. In the computation of the numerical trajectory (by means of the leap-frog integration scheme of Greenspan [4]), an interfering strategy is applied to the motion of the particle by extracting kinetic energy whenever it moves "uphill" along its trajectory. In this way, the particle is forced to converge to a local minimum. This method requires the availability of the gradient vector of the function, denoted ∇f, the negative of which represents the force acting on the particle. The particle's acceleration is therefore proportional to $-\nabla f$. This is in contrast to classical gradient-based optimization, where position updates (or parts thereof) are proportional to $-\nabla f$. In the PSO method, the motion of a swarm of loosely interacting particles is considered. In this method, each particle is attracted to the best location (lowest function value position) along its path, as well as to the globally overall best position over all the particle trajectories to date. This method requires no gradient information and may therefore be considered a direct search method.

In the new method proposed here, the minimization of a function is achieved through the dynamic motion of a *strongly interacting* particle swarm, where *each particle* in the swarm is *simultaneously* attracted by all other particles located at positions of lower function value. The specific force law for the interaction between the individual particles within the swarm dictates that the force of attraction experienced by a particle at higher function value position

due to a particle at a lower function value position be equal to the difference between the respective function values divided by their distance of separation. The resultant motion of the particles under the influence of the attracting forces is computed by solving the associated equations of motion numerically by using again the leap-frog numerical integration scheme. An energy dissipation strategy, similar to that used in the original dynamic method, is applied to each particle by extracting kinetic energy from a particle whenever it moves "uphill". The specific chosen force law and the dissipation strategy result in the rapid collapse (convergence) of the swarm to a stationary point. To prevent the collapse of the swarm to a minimum in a subspace of R^n, the computed components of the respective attracting forces are individually stochastically perturbed in computing the respective particle trajectories. Because of the strong interaction, the resulting algorithm converges rapidly to a local minimum, as does the original dynamic algorithm, but now without the need of explicit gradient information. Note that the method proposed here is not similar to the method proposed by Engelbrecht [5], where the position of *only* the global best particle is adjusted using the dynamic method, with the remainder of the swarm using a standard PSO algorithm.

The proposed dynamic particle swarm optimization (DYN-PSO) algorithm promises to be an extremely reliable, robust, and easy-to-use method. As a preliminary test, it was decided to evaluate its performance against that of both trajectory methods it was constructed from, namely, a standard PSO algorithm [6] and the dynamic method of Snyman. The choice of a standard PSO algorithm is a deliberate one. The intention with this paper is to determine whether or not DYN-PSO deserves further investigation, not to suggest that this is a superior algorithm. A comparison to a standard PSO rather than a highly refined PSO variant serves this purpose better.

Before proceeding, the constrained molecular dynamics PSO by Poli and Stephens [7] deserves to be mentioned. In essence, their method is similar is spirit, but the details differ. They embed the cost function (the height above the search space) as an artificial coordinate, and then constrain the particles "to be on the fitness landscape" via an equality constraint. Their general formulation allows for different types of forces, one of which is a gravity-like force. In our implementation, this gravity-like force is the only external force. Poli and Stephens also consider particle-interaction forces. In particular, they discuss the case of particles connected by springs and energy dissipation that occur via friction/viscous damping. They then also numerically integrate the resulting differential equation, but elect to use the forward Euler method. Their method still requires cost function gradients, as well as second-order derivatives, which they approximate numerically. Limited numerical testing was performed.

2. DESCRIPTION OF THE DYNAMIC-PSO METHOD

2.1. Computation of particle trajectories

The DYN-PSO method is started by generating at time $t = 0$, a swarm of np particles, each of unit mass, with random initial positions denoted by $\mathbf{x}\{i\}(0) = \mathbf{x}^0\{i\}$, $i = 1, 2, \ldots, \text{np}$, within the region ("box") of interest in R^n. Initially, at $t = 0$, these particles all have zero velocities, that is, $\mathbf{v}\{i\}(0) = \mathbf{v}^0\{i\} = \mathbf{0}$, $i = 1, 2, \ldots, \text{np}$. We now postulate that at time t each particle i experiences a force $\mathbf{a}\{i\}(t)$, which is to be a function of the positions $\mathbf{x}\{j\}(t)$ and corresponding function values $f(\mathbf{x}\{j\}(t))$, $j = 1, 2, \ldots, \text{np}$ of all the particles at time t. The explicit analytical form of the force law giving $\mathbf{a}\{i\}(t)$ will be discussed in the next subsection. Thus, the trajectories $\mathbf{x}\{i\}(t)$ of the particles are given by the solution to the system of *initial value problems*

$$\ddot{\mathbf{x}}\{i\}(t) = \mathbf{a}\{i\}(t) \tag{1}$$

with initial conditions

$$\begin{aligned} \mathbf{x}\{i\}(0) &= \mathbf{x}^0\{i\}, \\ \dot{\mathbf{x}}\{i\}(0) &= \mathbf{v}\{i\}(0) = \mathbf{v}^0\{i\} = \mathbf{0}; \end{aligned} \tag{2}$$

for $i = 1, 2, \ldots, \text{np}$.

In practice, these equations are solved numerically by discretizing the time interval into time steps δ, and computing for $i = 1, 2, \ldots, \text{np}$ approximations $\mathbf{x}^k\{i\}$ to $\mathbf{x}\{i\}(t_k)$ and $\mathbf{v}^k\{i\}$ to $\mathbf{v}\{i\}(t_k)$ at discrete time mesh points $t_k = k\delta$, $k = 0, 1, 2, \ldots$, by some suitable numerical integration scheme. Here, we use the simple leap-frog numerical integration scheme of Greenspan [4].

Given $\mathbf{x}^0\{i\}$ and $\mathbf{v}^0\{i\}$ for $i = 1, 2, \ldots, \text{np}$, then for iterations $k = 0, 1, 2, \ldots$, compute for $i = 1, 2, \ldots, \text{np}$:

$$\begin{aligned} \mathbf{v}^{k+1}\{i\} &= \mathbf{v}^k\{i\} + \mathbf{a}^k\{i\}\delta, \\ \mathbf{x}^{k+1}\{i\} &= \mathbf{x}^k\{i\} + \mathbf{v}^{k+1}\{i\}\delta, \end{aligned} \tag{3}$$

where $\mathbf{a}^k\{i\}$ denotes the resultant force on particle i due to the individual forces of all the other particles at respective positions $\mathbf{x}^k\{j\}$, $j = 1, 2, \ldots, \text{np}$, $j \neq i$ (see next subsection).

This scheme has been found to be stable for sufficiently small time steps δ. It is also approximately energy conserving in the absence of energy dissipating forces, and was successfully used in the original single-particle dynamic method of Snyman [1].

2.2. The interacting force law

In the original dynamic method of Snyman [1, 2], a single particle is considered. The force $\mathbf{a}$ acting on this particle is equal to the negative of the function gradient. Since we propose a direct search method, we no longer have gradient information available. Rather, we will use information available in the swarm to generate the particle forces.

We now postulate that at iteration k each particle i in the swarm is simultaneously attracted by all other particles located at *current* computed positions of lower function value. No force is exerted on particle i by particles at higher function value positions. The explicit force law that we assume here dictates that the force of attraction experienced by a particle at higher function value due to a particle at a lower function value position is equal to the difference between the respective function values, divided by their distance of separation.

If the particle positions are randomly distributed, this force law can be viewed as a coarse finite difference computation, with the finite difference step equal to the distance of separation. As the particle positions become biased, the analogy between this proposed force law and a finite difference computation breaks down. This breakdown occurs since multiple forces in the same direction add up, opposed to standard finite difference perturbations which form an orthogonal set. Even considering generalized finite differences [8], the gradient components due to multiple perturbations in the same direction are averaged, rather than added together as in our proposed force law. A stochastic element is introduced into the proposed force law by randomly perturbing the direction of action of the attracting force. More explicitly and precisely, at each iteration k the resultant force exerted on particle i at *current* position $\mathbf{x}^k\{i\}$ with function value $f(\mathbf{x}^k\{i\})$ by the other np $-$ 1 particles at current positions $\mathbf{x}^k\{j\}$, $j = 1, 2, \ldots, \text{np}$, $j \neq i$ with corresponding function values $f(\mathbf{x}^k\{j\})$ is given by the force $\mathbf{a}^k\{i\}$ with components

$$a_m^k\{i\} = \frac{\sum_{j=1,\, j\neq i}^{\text{np}} [x_m^k\{j\} - x_m^k\{i\}]c_j 2(\text{rd})}{||\mathbf{x}^k\{j\} - \mathbf{x}^k\{i\}||} \tag{4}$$

for $m = 1, 2, \ldots, n$. Here $c_j = \max[0, f(\mathbf{x}^k\{i\}) - f(\mathbf{x}^k\{j\})]$ and rd is, for each j, an independent random number in [0, 1]. Note that the multiplication of a uniform random number in [0,1] by 2 results in a scheme that on average assigns unit weight to each of the computed components. If this stochastic element is omitted, the proposed algorithm suffers from premature convergence due to collapse of the swarm to a subspace in R^n. Similar observations have been made for the linear PSO [9].

This proposed force law is in contrast to the standard PSO, where particles are attracted to historic positions (personal and global bests) rather than the current position of other particles. In addition, this force law automatically results in a dynamic neighborhood, where it is possible that completely different particles exert forces on particle i in iteration $k + 1$, as compared to iteration k. In a given iteration k, the complete range of interaction is also covered. One extreme is the particle with the current worst function value, which experiences forces from all other particles. The other extreme is the particle with the current best function value, which experiences no force and travels at constant velocity.

2.3. Energy dissipation strategy

In computing the trajectories of the particles $\mathbf{x}^k\{i\}$, $i = 1, 2, \ldots, \text{np}$, for $k = 0, 1, 2, \ldots$, the function values $f_k(i) = f(\mathbf{x}^k\{i\})$ at $\mathbf{x}^k\{i\}$ are monitored at each iteration k so that the best (lowest) function value $f_b\{i\}$ and the corresponding best position $\mathbf{x}^b\{i\}$ along each trajectory i are recorded. The current overall globally best function value $f_g = \min_i(f_b\{i\})$ and the corresponding position $\mathbf{x}^g$ are also recorded.

The following energy dissipation strategy is now applied to ensure local descent of a particle and the overall collapse of the swarm to a local minimum. Whenever a particle i moves "uphill" at iteration k, that is, when $f_{k+1}\{i\} > f_k\{i\}$, then set $\mathbf{x}^{k+1}\{i\} := [2\mathbf{x}^k\{i\} + \mathbf{x}^b\{i\} + \mathbf{x}^{k+1}\{i\}]/4$ and set $\mathbf{v}^{k+1}\{i\} := [\mathbf{v}^{k+1}\{i\} + \mathbf{v}^k\{i\}]/4$. Notice that the current velocity $\mathbf{v}^{k+1}\{i\}$ is recomputed to be half the average velocity over the past two time steps. This interference will normally result in a decrease in kinetic energy and together with the "backward" adjustment of the position $\mathbf{x}^{k+1}\{i\}$ will initiate controlled motion of the particle towards a position of lower function value. The trajectory of particle i is assumed to have converged if the relative function value difference from iteration k to $k + 1$, that is, $|f_{k+1}\{i\} - f_k\{i\}|/(1 + |f_{k+1}\{i\}|)$, is less than some prescribed tolerance ε. The computation of the trajectories are continued until a sufficient number (npc) of the particles have converged to a stationary point. In practice, we choose npc = min$[n, 0.1\text{np}]$.

A formal presentation of the basic DYN-PSO algorithm is presented in Algorithm 1. For the sake of clarity and simplicity of presentation of the algorithm, the function value computation and monitoring procedure, and the recording of best local $f_b\{i\}$ and global f_g function values and corresponding best local and global positions, $\mathbf{x}^b\{i\}$ and $\mathbf{x}^g$ are not explicitly listed, but are implicitly assumed to be done in the execution of Algorithm 1. Also the computation of the forces $\mathbf{a}^k\{i\}$, $i = 1, 2, \ldots, \text{np}$, according to (4), is assumed to have been done as the need for the $\mathbf{a}^k\{i\}$ arises in Algorithm 1.

2.4. Selection of suitable integration time step

An outstanding matter is the selection of an appropriate time step δ to be used in the leap-frog integration scheme. This value can be chosen arbitrarily, but if chosen too large may result in individually erratic and unstable trajectories that fail to converge because of very large zigzagging steps being taken in space. On the other hand, if δ is too small, the steps in space will be correspondingly small and the collapse of the swarm may be very slow, requiring an excessively large number of iterations for convergence.

Many different schemes may be proposed to automatically select and control the time step so that acceptable convergence rates are obtained. Here, we firstly select an initial time step which guarantees sufficiently large trajectory steps in space for all the particles. Secondly, after the trajectories are initiated, for each iteration k and for each particle i, the magnitude of the actual step taken in space is monitored; and if larger than some specified step limit, the time step associated with the particular trajectory is reduced, that is, we now allow for different time steps $\delta\{i\}$, $i = 1, 2, \ldots, \text{np}$, for the different particles. The details of these additional automatic time step control procedures follow below.

In Step 1 of Algorithm 1, after the generation of the np random particles, compute the average magnitude $\bar{a}$ of the forces acting on the particles $\bar{a} = \sum_i ||\mathbf{a}\{i\}||/\text{np}$. An associated average computed initial step size follows from the leap-frog scheme as $\Delta x = \bar{a}\delta^2$. Requiring initially, on average, a step size of $\Delta x = D$, where D is the diameter of the initial variable "box", we initially select as sufficiently large initial time steps $\delta\{i\} = \delta = \sqrt{D/\bar{a}}$, $i = 1, 2, \ldots, \text{np}$, which is now inserted in Step 1 just after the generation of the initial random particle positions. However, for a particular particle, this initial choice for the time step may still be too large and result

Given the following:

(i) function $f(\mathbf{x})$, $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \in R^n$;
(ii) np = number of particles;
(iii) $\mathbf{y}$ = central point of region ("box") of interest;
(iv) range(i) = range of ith dimension of variable "box";
(v) δ = integration time step;
(vi) ε = tolerance for convergence of a trajectory on function value;
(vii) npc = number of particle trajectories required to converge before termination (default: npc = min[n, np]);
(viii) max = maximum number of iterations allowed,

then perform the following steps.

Step 1.

(i) For $i = 1, 2, \ldots,$ np, generate random particle starting positions $\mathbf{x}^0\{i\}$ within the specified box with components: $x_j^0\{i\} = y_j + \text{range}(j)(\text{rd} - 1/2)$; $j = 1, 2, \ldots, n$, where rd is an independent random number in the interval $[0, 1]$;
(ii) set initial velocities equal to zero: $\mathbf{v}^0\{i\} = \mathbf{0}$ and set iteration number $k := 0$; and convergence counter ic := 0.

Step 2.

(i) Compute trajectory step k to $k+1$ for each particle $i = 1, 2, \ldots,$ np, using the leap-frog integration scheme:
for $i = 1, 2, \ldots,$ np,
$\mathbf{v}^{k+1}\{i\} := \mathbf{v}^k\{i\} + \mathbf{a}^k\{i\}\delta$,
$\mathbf{x}^{k+1}\{i\} := \mathbf{x}^k\{i\} + \mathbf{v}^{k+1}\{i\}\delta$,
end for

Step 3.

(i) for $i = 1, 2, \ldots,$ np:
if $f_{k+1}\{i\} > f_k\{i\}$, then
$\mathbf{x}^{k+1}\{i\} := [2\mathbf{x}^k\{i\} + \mathbf{x}^b\{i\} + \mathbf{x}^{k+1}\{i\}]/4$,
$\mathbf{v}^{k+1}\{i\} := [\mathbf{v}^{k+1}\{i\} + \mathbf{v}^k\{i\}]/4$
end if
if $|f_{k+1}\{i\} - f_k\{i\}|/(1 + |f_{k+1}\{i\}|) < \varepsilon$ and $k > 0$, then set ic = ic + 1;
if ic = npc, then set $\mathbf{x}^* := \mathbf{x}^g$ and $f^* := f_g$ and *stop*.
if k = max, then *stop*.
end for
(ii) set $k := k + 1$; ic := 0 and go to Step 2.

Algorithm 1: Basic DYN-PSO algorithm.

in erratic behavior along the computed trajectory. Thus, as a further control measure, the magnitude of the step taken by each particle is monitored at each iteration, and if it exceeds a specified limit x_{lim}, the current velocity is scaled down by a factor $x_{\text{lim}}/\|\mathbf{x}^{k+1}\{i\} - \mathbf{x}^k\{i\}\|$, the particle's time step for subsequent steps is reduced by a factor α, and the step to $\mathbf{x}^{k+1}\{i\}$ is recomputed using the rescaled velocity and new time step. More specifically, we introduce at the end of Step 2 of Algorithm 1, for each iteration k and for each particle i, the additional procedure given in Algorithm 2.

In practice, good choices for the additional parameters introduced here are $x_{\text{lim}} = D/2$ and $\alpha = 0.5$. These values are used in the numerical tests that are reported here.

if $\|\mathbf{x}^{k+1}\{i\} - \mathbf{x}^k\{i\}\| > x_{\text{lim}}$ then
$\delta\{i\} := \alpha\delta\{i\}$
$\mathbf{v}^{k+1}\{i\} := x_{\text{lim}}\mathbf{v}^{k+1}\{i\}/\|\mathbf{x}^{k+1}\{i\} - \mathbf{x}^k\{i\}\|$
$\mathbf{x}^{k+1}\{i\} := \mathbf{x}^k\{i\} + \mathbf{v}^{k+1}\delta\{i\}$
end if

Algorithm 2: Adjustment of time step δ.

One complication of the above strategy arises if the nature of the cost function changes dramatically as the search proceeds. The initial time steps are appropriate during the initial stages of the search. However, the average force acting on the particles often decreases over time. Hence, the average particle step size also decreases over time. This seems appropriate for swarm convergence, but the step size decrease may be excessive if the average magnitudes of the forces computed by (4) decrease dramatically. This typically happens in the neighborhood of a local minimum of highly nonlinear functions. To overcome this drawback, we simply recompute the time steps $\delta\{i\} = \delta = \sqrt{D/\overline{a}}$, $i = 1, 2, \ldots,$ np, whenever appropriate, using the recomputed average force $\overline{a}$. For all the results presented in this paper, the time steps are recomputed every 100 iterations.

3. NUMERICAL PERFORMANCE OF THE DYN-PSO METHOD

3.1. Illustrative two-dimensional trajectories

To illustrate the mechanics of the DYN-PSO algorithm, we compute the trajectories for $f(\mathbf{x}) = x_1^2 + 2x_2^2$ with 3 particles, that is, the case $n = 2$ with np = 3. As starting points, we select the vertices $(40, 40)$; $(-40, 0)$; $(40, -40)$. Figure 1 depicts the three computed trajectories (using diameter $D = 100$) up to the 30th iteration at which point $f_g = 9.48 \times 10^{-4}$, with $x_1^g = -0.0201$; $x_2^g = 0.0165$.

3.2. Choice of number of particles

Throughout the numerical experiments done here, the number of particle trajectories required to converge before termination of the algorithm is taken as npc = min[n, np]. It is now required to obtain an indication of what is a good or an optimum choice for the number of particles np to be used for "normal" problems where one expects a single unique global minimum in the region of interest. To get an indication of what it should be, some experiments were performed (with function value tolerance $\varepsilon = 10^{-8}$) on the extended homogeneous quadratic test function (see list of test problems) for $n = 10$ and $n = 20$, using different values for np and determining the average number of functions evaluations (over 100 independent runs) required for successful convergence in each case. The variation of the number of function evaluations against np is shown in Figure 2. Note that for $n = 20$, the number of function evaluations appears to be almost insensitive to np over the range np = 15 to np = 25, with a

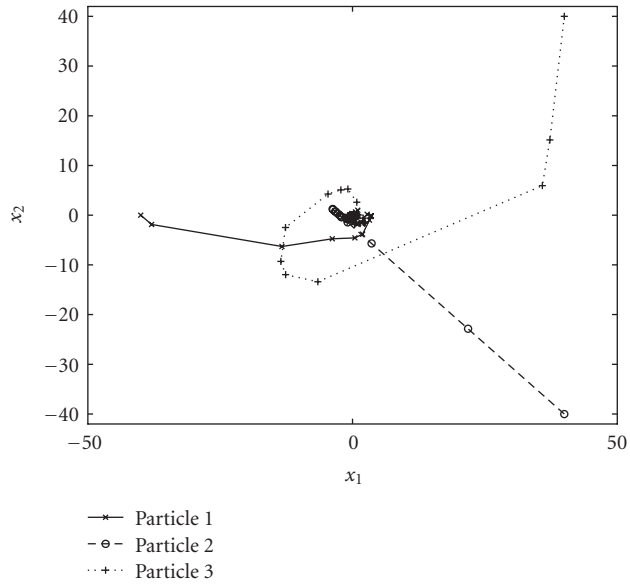

Figure 1: Example of dynamic PSO particle trajectories for the 2D function $f(x_1, x_2) = x_1^2 + 2x_2^2$.

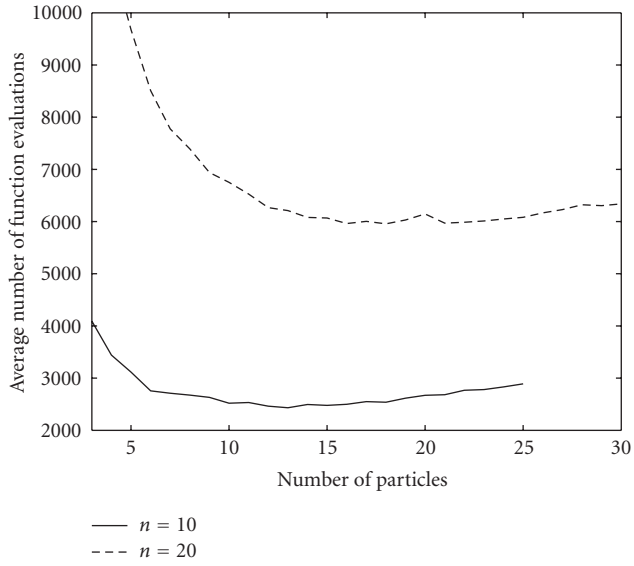

Figure 2: Average number of function evaluations at convergence versus number of particles.

variation of less than 4 percent being recorded. The corresponding variation of average function value at convergence with np is depicted in Figure 3. The results appear to indicate that in general a choice of np $= n + 1$ (i.e., with the positions of the particles defining the vertices of an n-dimensional simplex in R^n) is probably a good one. Consequently, the choice np $= n + 1$ is used throughout the experiments performed in the next subsection. Since this guideline is based on a very narrow test, some future effort should be directed towards improved guidelines to determine the number of particles. It is anticipated that highly multimodal problems might benefit from an increased number of particles.

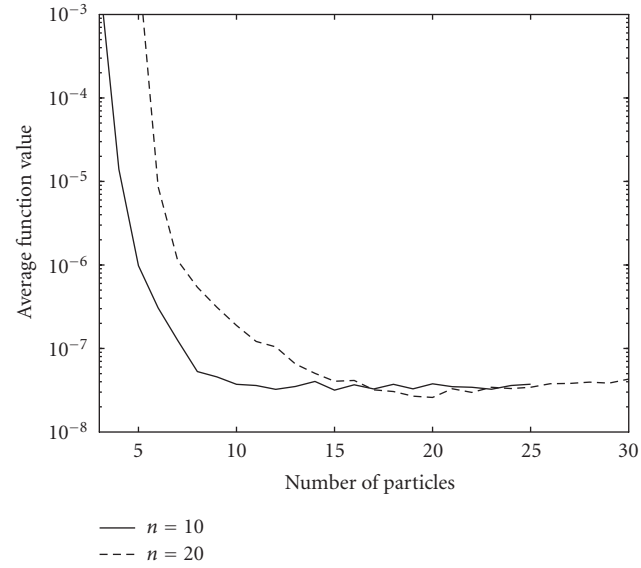

Figure 3: Average function value at convergence versus number of particles.

3.3. Performance on a set of test functions

The newly proposed DYN-PSO algorithm is tested on the following eight test problems.

(i) Homogeneous quadratic (unimodal)

$$f(\mathbf{x}) = \sum_{i=1}^{n} i x_i^2 \tag{5}$$

subject to $-5 < x_i < 5$ for $i = 1, 2, \ldots, n$; $f^* = 0$, at $\mathbf{x}^* = [0, 0, \ldots, 0]^T$.

(ii) Oren's power function (unimodal) [1]

$$f(\mathbf{x}) = \left(\sum_{i=1}^{n} i x_i^2 \right)^2 \tag{6}$$

subject to $-10 < x_i < 10$ for $i = 1, 2, \ldots, n$; $f^* = 0$, at $\mathbf{x}^* = [0, 0, \ldots, 0]^T$.

(iii) Extended Rosenbrock (multimodal) [10]

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right] \tag{7}$$

subject to $-2.048 < x_i < 2.048$ for $i = 1, 2, \ldots, n$; $f^* = 0$, at $\mathbf{x}^* = [1, 1, \ldots, 1]^T$.

(iv) Neumaier 3 (multimodal) [10]

$$f(\mathbf{x}) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1} \tag{8}$$

subject to $-n^2 < x_i < n^2$ for $i = 1, 2, \ldots, n$; $f^* = -n(n + 4)(n - 1)/6$, at $x_i^* = i(n + 1 - i)$.

(v) Extended Manevich (unimodal) [11]

$$f(\mathbf{x}) = \sum_{i=1}^{n} \frac{(1 - x_i)^2}{2^{i-1}} \tag{9}$$

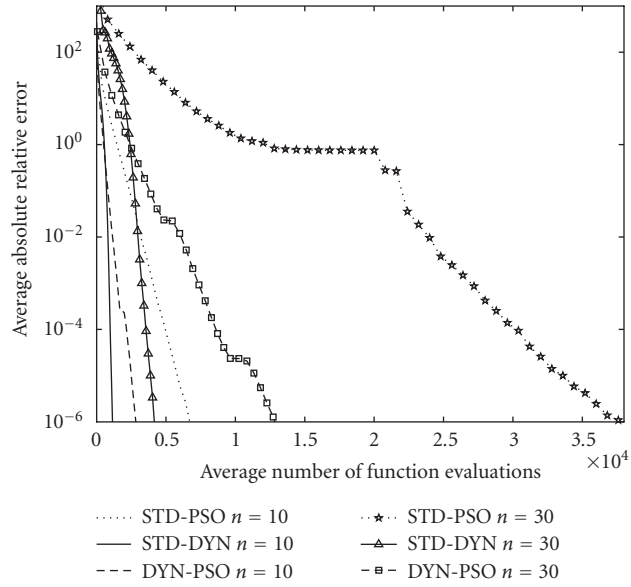

Figure 4: Convergence history on the homogeneous quadratic test function.

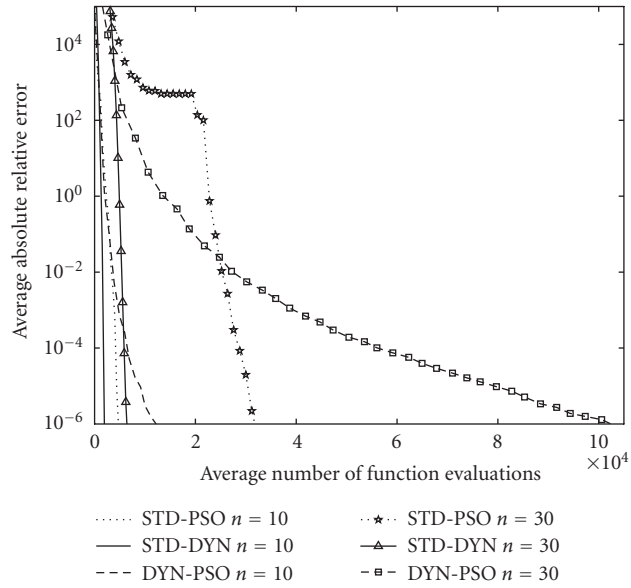

Figure 5: Convergence history on Oren's power test function.

subject to $-10 < x_i < 10$ for $i = 1, 2, \ldots, n$; $f^* = 0$, at $\mathbf{x}^* = [1, 1, \ldots, 1]^T$.

(vi) Zakharov (unimodal) [12]

$$f(\mathbf{x}) = \left(\sum_{i=1}^{n} x_i^2\right) + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^4 \tag{10}$$

subject to $-5 < x_i < 10$ for $i = 1, 2, \ldots, n$; $f^* = 0$, at $\mathbf{x}^* = [0, 0, \ldots, 0]^T$.

(vii) Griewank (multimodal) [10]

$$f(\mathbf{x}) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) \tag{11}$$

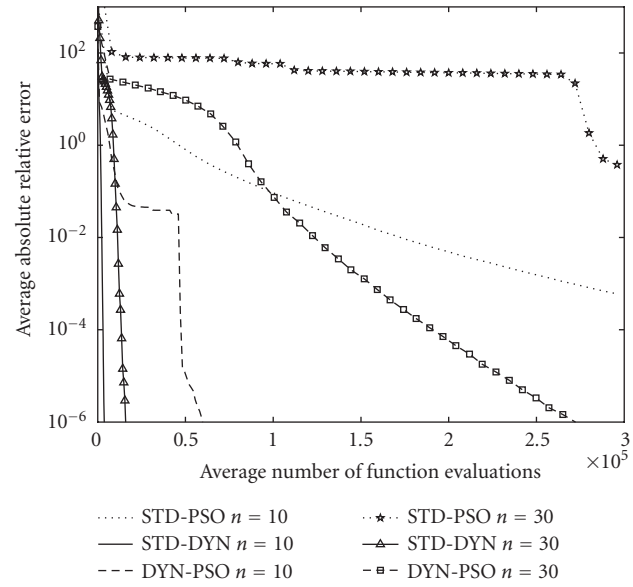

Figure 6: Convergence history on the Rosenbrock test function.

subject to $-600 < x_i < 600$ for $i = 1, 2, \ldots, n$; $f^* = 0$, at $\mathbf{x}^* = [0, 0, \ldots, 0]^T$.

(viii) Rastrigin (multimodal) [10]

$$f(\mathbf{x}) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)] \tag{12}$$

subject to $-5.12 < x_i < 5.12$ for $i = 1, 2, \ldots, n$; $f^* = 0$, at $\mathbf{x}^* = [0, 0, \ldots, 0]^T$.

Each test problem is solved 100 times for problem dimension $n = 10$ and 30. The DYN-PSO algorithm is compared to the dynamic method of Snyman [1, 2] (STD-DYN) and a standard PSO algorithm (STD-PSO) (http://www.particleswarm.info/Standard_PSO_2006.c), making use of settings proposed by Eberhart and Shi [6] ($c_1 = c_2 = 1.49445$, constant inertia weight of 0.729 and swarm size of 20). The gradients required by the dynamic method are computed using a forward difference method, using a perturbation size of 10^{-6}. These additional function evaluations are taken into consideration in the results of STD-DYN presented below. The convergence histories of the STD-DYN, DYN-PSO, and STD-PSO algorithms are depicted in Figures 4 to 11, for the eight test functions. In these graphs, at each iteration $k = 0, 1, 2, \ldots$, the absolute relative error

$$e_r = \frac{|f_g - f^*|}{1 + |f^*|}, \tag{13}$$

averaged over the 100 runs, is plotted against the corresponding average number of function evaluations.

Two common scenarios exist to terminate optimization algorithms. First, some convergence criteria are satisfied, which indicates that no substantial improvement is likely and search can terminate. Alternatively, a maximum number of function evaluations may be specified. For the test problems considered here, both these scenarios occur. The many local

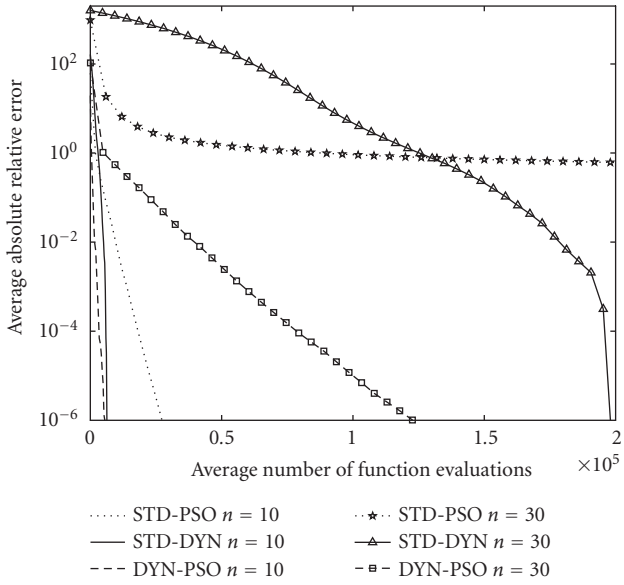

Figure 7: Convergence history on the Neumaier 3 test function.

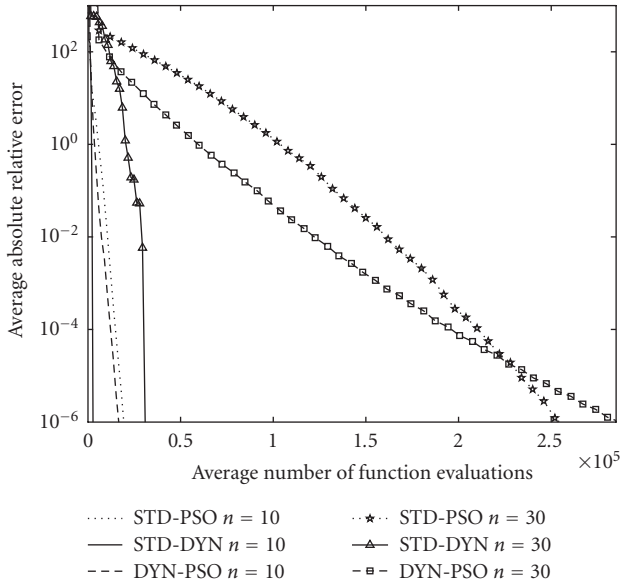

Figure 9: Convergence history on the Zakharov test function.

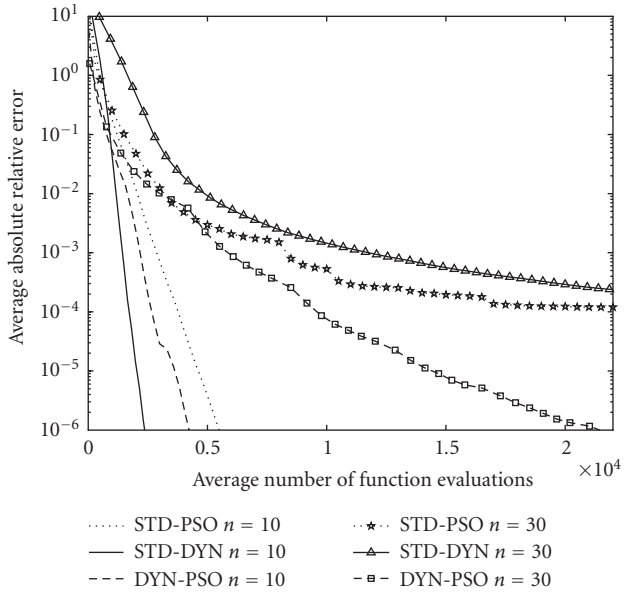

Figure 8: Convergence history on the Manevich test function.

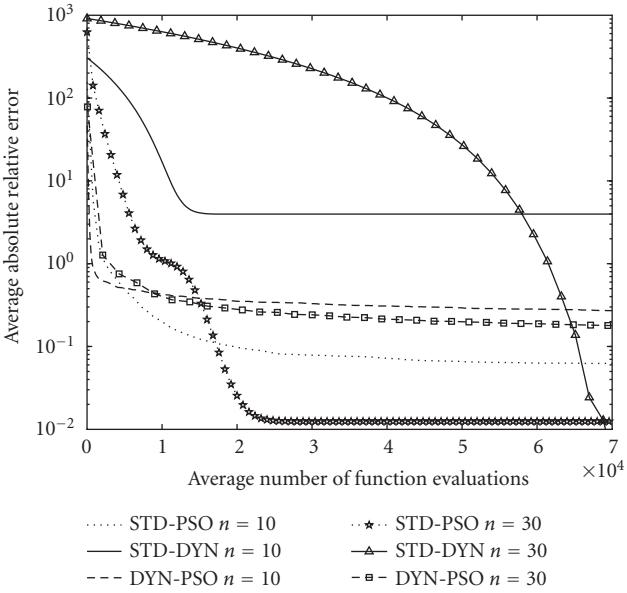

Figure 10: Convergence history on the Griewank test function.

minima present in the multimodal Griewank and Rastrigin test functions prevent the majority of particle trajectories to converge, hence these problems are terminated based on a maximum allowable number of function evaluations. The proposed convergence criteria work well for all the remaining test functions, where we used a convergence tolerance of $\varepsilon = 10^{-8}$.

The STD-DYN and DYN-PSO algorithms always locate the global minimum for the homogeneous quadratic, Oren's power function, Neumaier 3, Manevich and Zakharov problems. The STD PSO fails to do so for the 30D Neumaier problem. Also note that out of these 5 test problems, the STD-DYN algorithm is most efficient on the homogeneous quadratic, Oren's power function, and the Zakharov function, all of which are unimodal. The DYN-PSO algorithm performs best on the Neumaier 3 problem and the 30D Manevich function. STD PSO is only more efficient than the DYN-PSO algorithm for Oren's power function.

In the case of the Rosenbrock test function, the DYN-PSO algorithm does not always locate the global minimum. In those cases in which the global minimum is not found, the algorithm converges to the only other local minimum, as reported by Shang and Qiu [13]. In the Rosenbrock experiments, 89 and 96 of the 100 runs converged to the global minimum, for $n = 10$ and 30, respectively. The STD-DYN method locates the global minimum 83 and 88 times, for

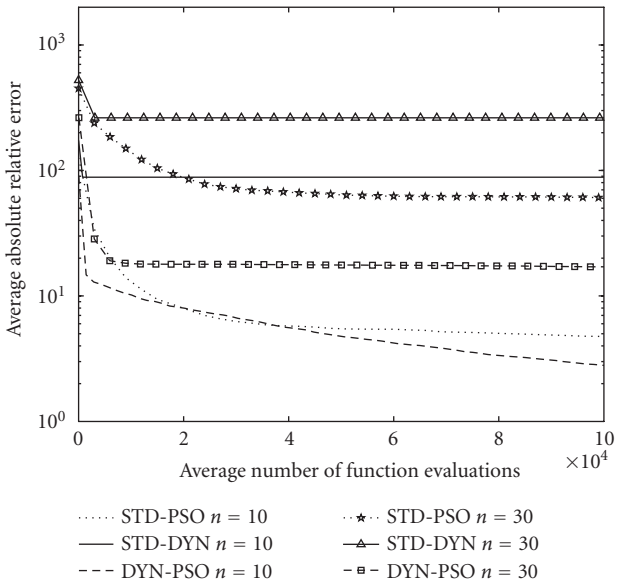

Figure 11: Convergence history on the Rastrigin test function.

$n = 10$ and 30, respectively. Note however that the number of function evaluations is more than a factor 10 less compared to the DYN-PSO algorithm. In the case of the STD-PSO algorithm, the majority of runs do not converge to either the global or local minimum. The number of runs that has a global best less than 1 after 3×10^5 function evaluations is 98 and 4, respectively, for n=10 and 30. The absolute relative error in Figure 6 is computed by only averaging over those runs that converge to the global minimum (DYN PSO and STD DYN) or those that have a global best less than unity after 3×10^5 function evaluations (STD PSO).

Finally, the DYN-PSO algorithm never locates the global minimum for the multimodal Griewank and Rastrigin test functions. The dynamic method, which is the backbone of the DYN-PSO algorithm, was developed as a local minimizer. However, the energy dissipation strategy we opted in the DYN-PSO algorithm is unchanged from that of the original dynamic method. This strategy is quite severe on "uphill" moves and hence energy is lost quickly if many local minimizers are present, such as in the Griewank and Rastrigin test functions. Nevertheless, compared to the performance of the STD-DYN method, the DYN PSO is vastly superior on the Rastrigin test function. The STD-DYN method simply locates the first strong local minimum and cannot escape it. On the Griewank function, DYN PSO is superior to STD DYN for 10D, and vice versa for 30D. The STD PSO however far outperforms both STD DYN and DYN PSO for the Griewank function. However, in our experience, the performance of the DYN-PSO algorithm is still comparable to that of more traditional global optimization algorithms. The current algorithm does demonstrate the ability to escape some local minima, similar to the standard PSO algorithm. The performance on the 30D Rastrigin function is especially noteworthy, with a mean function value less than 20 after only 5 000 function evaluations. This is obtained with the standard settings proposed here, with no tuning of parameters to suit the problem.

In summary, the DYN-PSO algorithm seems to inherit the desired properties from both its ancestors. It can efficiently solve unimodal functions, sometimes even more efficiently than the gradient-based local minimizer it is based on, for example, the 30D Manevich problem. This is achieved without making use of gradient information. Therefore, the DYN-PSO method might even work for discontinuous problems. This efficient local character is blended with nonlocal behavior, where the swarm provides sufficient information to solve multimodal problems, illustrated best on the Rastrigin function.

4. CONCLUSIONS

We have proposed a novel dynamic interacting particle swarm optimization algorithm. The algorithm compares well to a standard PSO implementation, especially in terms of efficient solution of high-dimensional problems containing few local minimizers. Based on these promising results, the DYN-PSO algorithm deserves further development.

A number of outstanding issues remain. The importance of recomputing appropriate time-step sizes is already recognized, but more refined criteria should be developed. A guideline for the number of particles for efficient search is already proposed, but since it is based on a very narrow test, additional experiments are required. Also, the convergence criterion can be refined in order to also work for functions containing a very large number of local minimizers. The mechanism that probably requires the most attention is the energy dissipation scheme, which can be modified to increase the probability of convergence to the global minimum in the case of multiple local minima. This could be achieved by less aggressive energy dissipation during "uphill" moves, but will necessarily retard convergence.

The initial results indicate that the proposed DYN-PSO algorithm shows much promise as an alternative direct search method for solving large scale unconstrained optimization problems. The algorithm seems capable of solving unimodal problems economically, and it also has competitive performance on functions containing many local minima.

REFERENCES

[1] K. W. Kolodziej and J. Hjelm, *Local Positioning Systems: LBS Applications and Services*, CRC Press, Boca Raton, Fla, USA, 2006.

[2] A. LaMarca, Y. Chawathe, S. Consolvo, et al., "Place lab: device positioning using radio beacons in the wild," in *Proceedings of the 3rd International Conference on Pervasive Computing (PERVASIVE '05)*, vol. 3468, pp. 116–133, Munich, Germany, May 2005.

[3] J. Letchner, D. Fox, and A. LaMarca, "Large-scale localization from wireless signal strength," in *Proceedings of the 20th National Conference on Artificial Intelligence and the 17th Innovative Applications of Artificial Intelligence Conference (AAAI '05)*, pp. 15–20, Pittsburgh, Pa, USA, July 2005.

[4] H. Lim, L.-C. Kung, J. C. Hou, and H. Luo, "Zero-configuration, robust indoor localization: theory and experimentation," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM '06)*, pp. 1–12, Barcelona, Spain, April 2006.

[5] P. Bahl and V. N. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '00)*, vol. 2, pp. 775–784, Tel Aviv, Israel, March 2000.

[6] K. Kaemarungsi and P. Krishnamurthy, "Properties of indoor received signal strength for WLAN location fingerprinting," in *Proceedings of the 1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS '04)*, pp. 14–23, Boston, Mass, USA, August 2004.

[7] D. Niculescu and B. Nath, "Ad hoc positioning system (APS)," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '01)*, vol. 5, pp. 2926–2931, San Antonio, Tex, USA, November 2001.

[8] D. Niculescu and B. Nath, "Ad hoc positioning system (APS) using AOA," in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '03)*, vol. 3, pp. 1734–1743, San Francisco, Calif, USA, March-April 2003.

[9] R. Peng and M. L. Sichitiu, "Angle of arrival localization for wireless sensor networks," in *Proceedings of the 3rd Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '06)*, vol. 1, pp. 374–382, Reston, Va, USA, September 2006.

[10] R. L. Moses, D. Krishnamurthy, and R. M. Patterson, "A self-localization method for wireless sensor networks," *EURASIP Journal on Applied Signal Processing*, vol. 2003, no. 4, pp. 348–358, 2003.

[11] D. Niculescu and B. Nath, "Position and orientation in ad hoc networks," *Ad Hoc Networks*, vol. 2, no. 2, pp. 133–151, 2004.

[12] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low-cost outdoor localization for very small devices," *IEEE Personal Communications*, vol. 7, no. 5, pp. 28–34, 2000.

[13] C. Savarese, J. M. Rabaey, and K. Langendoen, "Robust positioning algorithms for distributed Ad-Hoc wireless sensor networks," in *Proceedings of USENIX Annual Technical Conference*, pp. 317–327, Monterey, Calif, USA, June 2002.