Fundamenta Informaticae ?? 1–47

IOS Press

A Convergence Proof for the Particle Swarm Optimiser

F van den Bergh

CSIR, Meiring Naude Road, Pretoria, South Africa

fvdbergh@csir.co.za

AP Engelbrecht

Department of Computer Science, University of Pretoria, Pretoria, South Africa

engel@cs.up.ac.za

Abstract. The Particle Swarm Optimiser (PSO) is a population based stochastic optimisation algorithm, empirically shown to be efficient and robust. This paper provides a proof to show that the original PSO does not have guaranteed convergence to a local optimum. A flaw in the original PSO is identified which causes stagnation of the swarm. Correction of this flaw results in a PSO algorithm with guaranteed convergence to a local minimum. Further extensions with provable global convergence are also described. Experimental results are provided to elucidate the behavior of the modified PSO as well as PSO variations with global convergence.

Keywords: Particle swarm optimisation, convergence proofs, stochastic optimisation, local convergence, global convergence

Address for correspondence: University of Pretoria, Pretoria, South Africa, engel@cs.up.ac.za

1. Introduction

The particle swarm optimiser (PSO), introduced by Kennedy and Eberhart [13, 8], is a population based optimisation algorithm, originally derived from a simplified sociological model. The members of the population, called *particles*, represent potential solutions to the optimisation task at hand. Each particle has a current *position*, a *personal best position*, and a *velocity* associated with it. The particles then move through the search space under the influence of the attractive forces of previously discovered promising solutions, as well as the influence of their own inertia.

The velocity update of each particle depends on: 1) the distance that the particle is from the best solution found by the swarm, called the *global best position*, and 2) the distance that the particle is from its *personal best position*, defined as the best solution that the particle has personally come across during its lifetime. Particles start from random positions in the search space, so that initially some particles will have large velocities, allowing them to explore a large region of the search space. Over time, particles move closer together, eventually converging on a single position.

The ability of the PSO to find the global optima of functions has mainly been studied experimentally (refer to the list of papers at http://www.swarmintelligence.org/). Claims like "The particle swarm paradigm found the global optimum each run,..." have been made regarding Schaffer's f_6 function [13]. These claims may give the impression that the PSO is a global optimisation method, with only empirical evidence to back up such claims. Theoretical analyses have been done to show that, under certain conditions, particles converge to a stable point [7, 1, 26, 29, 4, 19]. While existing research proved that particles converge to a stable point, nothing has been said about the optimality of this point. A formal analysis is presented in this paper to prove that this stable point is not necessarily a local optimum. It is also shown that a flaw in the original PSO causes the particles to stagnate. A variation of the PSO is described to address this flaw, and it is proven that the adapted PSO has guaranteed convergence to a local optimum, but not a global optimum. The PSO is then further adapted to provide algorithms with global convergence.

Section 2 provides an overview of the original PSO. The main findings of published theoretical studies are summarised in Section 3. Criteria for convergence, as applicable to global and local stochastic

search algorithms are discussed in Section 4. Section 5 provides a novel proof that the PSO is not a local or global minimiser. The flaw in the original PSO algorithm, described analytically here for the first time, is discussed in Section 6, and a PSO variation is presented with guaranteed convergence to a local optimum. Section 7 discusses the global convergence properties of the PSO, followed by the description of globally convergent PSO variants in Section 8. Experimental results are summarised in Section 9.

2. Particle Swarm Optimisation

Particle swarm optimisation (PSO) is a stochastic optimisation approach which maintains a swarm of candidate solutions, referred to as *particles* [13, 8]. Particles are "flown" through an n-dimensional search space, with each particle being attracted towards the best solution found by the swarm and the best solution found by the particle. The position, $\mathbf{x}_i$, of the i-th particle is adjusted by a stochastic velocity $\mathbf{v}_i$ which depends on the distance that the particle is from its own best solution and that of the swarm. For the original PSO [13, 8],

$$v_{ij,t+1} = v_{ij,t} + \phi_{1j,t}(y_{ij,t} - x_{ij,t}) + \phi_{2j,t}(\hat{y}_{j,t} - x_{ij,t}) \quad (1)$$

$$x_{ij,t+1} = x_{ij,t} + v_{ij,t+1} \quad (2)$$

for $i \in \mathbb{Z}, 1 \leq i \leq h$ and $j \in \mathbb{Z}, 1 \leq j \leq n$, where:

$\phi_{1j,t} = c_1 r_{1j,t}$ and $\phi_{2j,t} = c_2 r_{2j,t}$

h is the total number of particles in the swarm

n is the dimension of the problem, i.e., the number of parameters of the function being optimised

c_1 and c_2 are acceleration coefficients

$r_{1j,t}, r_{2j,t} \sim U(0,1)$

$\mathbf{x}_{i,t}$ is the position of particle i at time step t

$\mathbf{v}_{i,t}$ is the velocity of particle i at time step t

$\mathbf{y}_{i,t}$ is the personal best solution of particle i, up to time step t

$\hat{\mathbf{y}}_t$ is the best position found by the swarm, up to time step t.

In Equation (1) the social component, $\hat{\mathbf{y}}_t$, represents the best solution found by the swarm. There are variations of the original PSO in which the global best particle position, $\hat{\mathbf{y}}_t$, is replaced by a *neighbourhood best* position which is dependent on the neighbourhood to which the particle belongs, and is thus denoted $\hat{\mathbf{y}}_{i,t}$. This neighbourhood is usually defined in terms of the particle's index number i, rather than a Euclidean-distance based neighbourhood.

Different neighborhood topologies can be used to constrict the information flow between particles. A number of neighbourhood topologies have been investigated [12, 15, 14, 11], of which the star, ring, and Von Neumann topologies have shown to be the most popular. This paper concentrates on the star topology where the neighbourhood of each particle is the entire swarm, with the resulting PSO algorithm referred to as the *gbest* PSO.

The *gbest* PSO algorithm is summarised in Figure 1.

1. Create and initialise a n-dimensional swarm, P
2. While stopping condition is not true
   (a) For each particle $i \in \mathbb{Z}, 1 \leq i \leq h$
      i. If ($f(P.\mathbf{x}_i) < f(P.\mathbf{y}_i)$) then $P.\mathbf{y}_i = P.\mathbf{x}_i$
      ii. If ($f(P.\mathbf{y}_i) < f(P.\hat{\mathbf{y}})$) then $P.\hat{\mathbf{y}} = P.\mathbf{y}_i$
   (b) For each particle $i \in \mathbb{Z}, 1 \leq i \leq h$
      i. update the velocity using equation (1)
      ii. update the position using equation (2)

Figure 1. The Original *gbest* PSO.

The remainder of this section summarises problems experienced with the original version of the PSO, and some early variations to address these problems.

2.1. Velocity Clamping

Initial PSO studies used $c_1 = c_2 = 2.0$. Although good results have been obtained, it was observed that velocities quickly exploded to large values. To combat this effect, it was proposed that each component of the velocity should be clamped [11]. Later studies found that the velocity clamping can be avoided by using a constriction coefficient [7] .

2.2. Inertia Weight

The inertia weight was introduced by Shi and Eberhart to restrict divergent behaviour [21, 10]. The inertia weight, w, controls the momentum of the particle by weighting the contribution of the previous velocity, controlling how much the previous flight direction will influence the new velocity. The velocity equation becomes

$$v_{ij,t+1} = wv_{ij,t} + \phi_{1j,t}(y_{ij,t} - x_{ij,t}) + \phi_{2j,t}(\hat{y}_{j,t} - x_{ij,t}) \tag{3}$$

Initial empirical studies of PSO with inertia have shown that the value of w is critical in ensuring convergent behaviour [22, 9]. For $w \geq 1$, velocities increase over time causing divergent behaviour, so that particles fail to change direction in order to move back towards promising areas. For $w < 1$, particles decelerate until their velocities reach zero, provided that $2w > (c_1 + c_2) - 2$, as shown in [4, 1].

Empirical results have shown that a constant inertia of $w = 0.7298$ and acceleration coefficients with $c_1 = c_2 = 1.49618$ provide good convergent behaviour [9]. While static inertia values have been used successfully, adaptive inertia values have also shown to lead to convergent behaviour [25, 30, 6, 23, 27].

3. Particle Trajectories

Early published theoretical analyses of the PSO have concentrated on analyzing the behavior of particles by studying particle trajectories. Ozcan and Mohan concluded from their studies that particle trajectories follow periodic sinusoidal waves [16, 17].

Clerc and Kennedy provided a theoretical analysis of particle trajectories to ensure convergence to a

stable point [7],

$$\mathbf{p} = \frac{\phi_1 \mathbf{y} + \phi_2 \hat{\mathbf{y}}}{\phi_1 + \phi_2} \tag{4}$$

The main result of Clerc's work is the introduction of the constriction coefficient and different classes of constriction models. The objective of this theoretically derived constriction coefficient is to prevent the velocity from growing out of bounds, with the advantage that, theoretically, velocity clamping is no longer required. As a result of this study, the velocity equation changes to [5, 7]

$$v_{ij,t+1} = \chi[v_{ij,t} + \phi_{1j,t}(y_{ij,t} - x_{ij,t}) + \phi_{2j,t}(\hat{y}_{ij,t} - x_{ij,t})] \tag{5}$$

where χ is the constriction coefficient calculated as

$$\chi = \frac{2\kappa}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|} \tag{6}$$

with $\phi = \phi_1 + \phi_2 \geq 4$ and $\kappa \in [0, 1]$. The constant κ controls the rate of convergence. For $\kappa \approx 0$, rapid convergence to a stable point is obtained, while a $\kappa \approx 1$ results in slow convergence. The reader is referred to [7] for a more detailed derivation of the constriction coefficient and models.

Van den Bergh and Engelbrecht extended the analysis of a simple deterministic PSO system to also include the inertia term [1, 4]. This was also independently done by Trelea [26]. The analysis independently arrived at the conclusion that particles converge to the point

$$(1 - a)\mathbf{y} + a\hat{\mathbf{y}} \tag{7}$$

where $a = \frac{c_2}{c_1 + c_2} \in [0, 1]$. The same result has been derived in [26, 29].

If $\{\mathbf{x}_t\}_{t=0}^{+\infty}$ denotes the sequence of positions for a specific particle, then all that the current theoretical analyses have shown is that

$$\lim_{t \to +\infty} \mathbf{x}_t = (1 - a)\mathbf{y} + a\hat{\mathbf{y}} \tag{8}$$

From this, nothing can be inferred about the optimality of the point $(1 - a)\mathbf{y} + a\hat{\mathbf{y}}$.

The rest of this paper is devoted to investigating whether the original PSO converges to a local minimum. That is, to show whether the sequence $\{\mathbf{x}_t\}_{t=0}^{+\infty}$ converges to $\mathbf{x}_B^*$ in region $B \subset \mathcal{S}$, where $\mathcal{S}$ is the feasible space, i.e., if

$$\lim_{t \to +\infty} \mathbf{x}_t = \mathbf{x}_B^* \tag{9}$$

where $f(\mathbf{x}_B^*) \leq f(\mathbf{x})$, for all $\mathbf{x} \in B$. It is also shown that, for the original PSO,

$$P\left(\lim_{t \to +\infty} \mathbf{x}_t \neq \mathbf{x}^*\right) > 0 \tag{10}$$

where $f(\mathbf{x}^*) < f(\mathbf{x})$, for all $\mathbf{x} \neq \mathbf{x}^*$.

4. General Conditions for Convergence

The stochastic nature of the particle swarm optimiser makes it more difficult to prove (or disprove) properties like global convergence. Solis and Wets have studied the convergence of stochastic search algorithms, most notably that of pure random search algorithms, providing criteria under which algorithms can be considered to be global search algorithms, or merely local search algorithms [24]. This paper uses the definitions from [24] to study the convergence characteristics of the PSO. For convenience, some of these definitions are reproduced below.

4.1. Simple Random Search

The convergence conditions are based on the following problem and the conceptual algorithm described in Section 4.2:

Problem 1. *Given a measurable function $f : \mathbb{R}^n \to \mathbb{R}$ and $\mathcal{S} \subseteq \mathbb{R}^n$. The objective is to find a point $\mathbf{x} \in \mathcal{S}$ which minimises f on $\mathcal{S}$.*

This provides a definition of what a global optimiser must produce as output, given the function f and the search space $\mathcal{S}$.

4.2. Conditions for Local Convergence

It is sufficient to show that a random search algorithm will at least converge to a local minimum if the algorithm satisfies the *algorithm condition* and the *convergence condition*. The algorithm in Figure 2 serves as a conceptual simple random search algorithm.

1. Start with an initial solution, $\mathbf{z}_0 \in \mathcal{S}$, and set $t = 0$.
2. Generate a vector ξ_t from $\mathbb{R}^n$, using μ_t as probability measure.
3. Set $\mathbf{z}_{t+1} = D(\mathbf{z}_t, \xi_t)$, choose μ_{t+1}, set $t := t + 1$, return to step 2.

Figure 2. Simple Random Search Algorithm.

Referring to Figure 2, D is a mapping that combines the new sample, ξ_t, with the current solution, $\mathbf{z}_t$. The algorithm condition, presented next, stipulates that the new solution suggested by D is to be no worse than the current solution.

Algorithm condition: *The mapping $D : \mathcal{S} \times \mathbb{R}^n \rightarrow \mathcal{S}$ should satisfy $f(D(\mathbf{x}, \xi)) \leq f(\mathbf{x})$ and if $\xi \in \mathcal{S}$, then $f(D(\mathbf{x}, \xi)) \leq f(\xi)$.*

Let M_t be the support of the probability measure μ_t. That is, M_t is the smallest closed subset of $\mathbb{R}^n$ with measure 1 under μ_t. Almost all random search algorithms are adaptive, with μ_t depending on the previous solutions, $\mathbf{x}_0, \ldots, \mathbf{x}_{t-1}$ generated by the algorithm. The μ_t are thus viewed as conditioned probability measures. Let m be the Lebesgue measure of a set. For a local search method, the μ_t with bounded support M_t have, for all except a possibly finite t, $m(\mathcal{S} \cap M_t) < m(\mathcal{S})$.

To avoid searching for an element in a set of null measure, the search will be for the essential infimum of f. The algorithm will thus be said to have found a solution if it is able to generate a point in the *optimality region*, R_ϵ, defined as

$$R_\epsilon = \{\mathbf{z} \in \mathcal{S} \mid f(\mathbf{z}) < \psi + \epsilon\} \tag{11}$$

where ψ denotes the essential infimum of f on $\mathcal{S}$, and $\epsilon > 0$.

Convergence condition: *Sufficient condition for convergence to at least a local minimum: For any* $\mathbf{x}_t \in \mathcal{S}$ *there exists a* $\gamma > 0$ *and an* $0 < \eta \leq 1$ *such that*

$$\mu_t(\text{dist}(\mathbf{x}_{t+1}, R_\epsilon) \leq \text{dist}(\mathbf{x}_t, R_\epsilon) - \gamma \text{ or } \mathbf{x}_t \in R_\epsilon) \geq \eta \tag{12}$$

Therefore, an algorithm is a local optimisation algorithm if a non-zero η can be found such that at every step the algorithm can move $\mathbf{x}$ closer to the optimality region by at least distance γ, or $\mathbf{x}$ is already in the optimality region with a probability greater than or equal to η.

Theorem 4.1. Local Search: *Assume that* f *is a measurable function,* $\mathcal{S}$ *is a measurable subset of* $\mathbb{R}^n$ *and both the algorithm and the convergence condition for local search are satisfied. Let* $\{\mathbf{x}_t\}_{t=0}^{\infty}$ *be a sequence generated by the algorithm,* D*. Then,*

$$\lim_{t\to\infty} P(\mathbf{x}_t \in R_\epsilon) = 1 \tag{13}$$

The proof of this theorem can be found in [24].

4.3. Conditions for Global Convergence

In the case of a global search algorithm, the following condition is sufficient to prove convergence to a global minimum:

Convergence condition: *Sufficient condition for convergence to a global minimum: For any (Borel) subset* A *of* $\mathcal{S}$ *with* $m(A) > 0$*,*

$$\prod_{t=0}^{\infty}(1 - \mu_t(A)) = 0 \tag{14}$$

where $\mu_t(A)$ *is the probability of* A *being generated by* μ_t*.*

This convergence condition means that for any subset A of $\mathcal{S}$ with positive measure m, the probability

of repeatedly missing the set A using random samples (e.g., the ξ_t), must be zero.

Theorem 4.2. Global Search: *Assume that f is a measurable function, $\mathcal{S}$ is a measurable subset of $\mathbb{R}^n$, and the algorithm condition and convergence condition for global search are satisfied. If $\{\mathbf{x}_t\}_{t=0}^{+\infty}$ is a sequence generated by the algorithm, D, then*

$$\lim_{t \to +\infty} P(\mathbf{x}_t \in R_\epsilon) = 1 \tag{15}$$

The proof of this theorem can be found in [24].

5. Local Convergence of PSO

This section tests the hypothesis that the basic PSO is a local optimiser. The proof first considers unimodal optimisation problems. The objective of the proof is to show whether the basic PSO satisfies both the algorithm and convergence conditions as given in Section 4.2.

5.1. Unimodal Optimisation Problems

Let $\mathbf{x}_0$, defined as

$$\mathbf{x}_0 = \underset{\mathbf{x}_i}{\operatorname{argmax}}\{f(\mathbf{x}_i)\},\ i \in \mathbb{Z}, 1 \leq i \leq h, \tag{16}$$

be an initial position which represents the worst particle in the swarm. Note that minimisation problems are considered for which the worst particle yields the largest f value of all particles in the swarm. Define the compact set,

$$L_0 = \{\mathbf{x} \in \mathcal{S} : f(\mathbf{x}) \leq f(\mathbf{x}_0)\} \tag{17}$$

to be the set of all particles with f values smaller than or equal to that of the worst particle position, $\mathbf{x}_0$. From the velocity and position equations (1) and (2), function D (as used in the algorithm condition) is

defined for the PSO as

$$D(\hat{\mathbf{y}}_t, \mathbf{x}_{i,t}) = \begin{cases} \hat{\mathbf{y}}_t & \text{if } f(\mathbf{g}(\mathbf{x}_{i,t})) \geq f(\hat{\mathbf{y}}_t) \\ \mathbf{g}(\mathbf{x}_{i,t}) & \text{if } f(\mathbf{g}(\mathbf{x}_{i,t})) < f(\hat{\mathbf{y}}_t) \end{cases} \tag{18}$$

where $\mathbf{g}(\mathbf{x}_{i,t})$ denotes the application of $\mathbf{g}$, which is the function performing the PSO updates, defined in Equation (19).

The definition of D in equation (18) satisfies the algorithm condition, since the sequence $\{f(\hat{\mathbf{y}}_t)\}_{t=0}^{T}$ is monotonic by definition. This sequence represents the best positions amongst all the positions that all the particles visited up to time step T.

This proof views the computation of the value of $\mathbf{x}_{i,t+1}$ as the successive application of three functions $\mathbf{g}_1, \mathbf{g}_2$ and $\mathbf{g}_3$. Each function adds a term to the previous result. That is,

$$\mathbf{x}_{i,t+1} = \mathbf{g}(x_{i,t}) = \mathbf{g}_1(\mathbf{x}_{i,t}) + \mathbf{g}_2(\mathbf{x}_{i,t}) + \mathbf{g}_3(\mathbf{x}_{i,t}) \tag{19}$$

where

$$\mathbf{g}_1(\mathbf{x}_{i,t}) = \mathbf{x}_{i,t} + w\mathbf{v}_{i,t} \tag{20}$$

and

$$g_2(\mathbf{x}_{i,t})_j = c_1 r_{1j,t}(y_{ij,t} - x_{ij,t}), \;\; j \in \mathbb{Z}, 1 \leq j \leq n \tag{21}$$

$$g_3(\mathbf{x}_{i,t})_j = c_2 r_{2j,t}(\hat{y}_{j,t} - x_{ij,t}), \;\; j \in \mathbb{Z}, 1 \leq j \leq n \tag{22}$$

where $g_k(x_{i,t})_j$ denotes the j-th dimension of the vector function $\mathbf{g}_k$.

Now, from the definition of the compact set L_0 (refer to equation (17)), and the assumption that all the particles are initially in L_0,

$$\mathbf{y}_{i,0}, \mathbf{x}_{i,0} \in L_0 \tag{23}$$

and therefore also, $\hat{\mathbf{y}}_0 \in L_0$. Let $\mathbf{g}^N(\mathbf{x}_{i,t})$ denote N successive applications of $\mathbf{g}$ on $\mathbf{x}_{i,t}$. The particle

will follow a convergent trajectory under the following conditions:

Convergent parameter region: *Let* $\phi_1 = c_1 r_1$ *and* $\phi_2 = c_2 r_2$*, and let*

$$\lambda_1 = \frac{1 + w - \phi_1 - \phi_2 + \gamma}{2} \tag{24}$$

$$\lambda_2 = \frac{1 + w - \phi_1 - \phi_2 - \gamma}{2} \tag{25}$$

with

$$\gamma = \sqrt{(1 + w - \phi_1 - \phi_2)^2 - 4w} \tag{26}$$

For all $0 < w < 1$ *and* $0 < \phi_1, \phi_2 < 2$*, if*

$$w > \frac{(\phi_1 + \phi_2)}{2} - 1,$$

then

$$\max\{\|\lambda_1\|, \|\lambda_2\|\} < 1.$$

For a detailed discussion of this convergent parameter region, see [4, 1]. Recently, Poli pointed out that this region only guarantees *order-1* convergence, that is, convergence of the mean $E[\mathbf{x}_t]$ [19]. Poli has shown that the region that offers *order-2* stability, that is, the region for which the standard deviation of $\mathbf{x}_t$ tends to zero, is a large subset of the convergent region specified above. In particular, the common choice of $c_1 = c_2 = 1.4961798$ and $w = 0.729844$ falls within the order-2 stability region. In addition to correct parameter choices, Poli has further shown that order-2 stability of the stochastic PSO requires that $\mathbf{y}_{i,t} = \hat{\mathbf{y}}_t$. This additional constraint will be revisited after Lemma 5.3 below.

Lemma 5.1. *For all* w*,* $\phi_1 = c_1 r_1$ *and* $\phi_2 = c_2 r_2$ *falling inside the convergent parameter region, there exists a finite* N *and a positive* ϵ *such that, for all* $k \geq N$*,*

$$||\mathbf{g}^k(\mathbf{x}_{i,t}) - \mathbf{g}^{k+1}(\mathbf{x}_{i,t})|| < \epsilon$$

Proof: The trajectories of the particles will now be considered for a single particle only, thus the particle subscript will be omitted. Furthermore, the individual dimensions are independent, thus a one-dimensional notation will be used here. In [4, 1] it was shown that the deterministic trajectory of a particle is described by

$$x_t = k_1 + k_2\lambda_1^t + k_3\lambda_2^t \tag{27}$$

where

$$\begin{aligned} k_1 &= \frac{\phi_1 y + \phi_2 \hat{y}}{\phi_1 + \phi_2} \\ k_2 &= \frac{\lambda_2(x_0 - x_1) - x_1 + x_2}{\gamma(\lambda_1 - 1)} \\ k_3 &= \frac{\lambda_1(x_1 - x_0) + x_1 - x_2}{\gamma(\lambda_2 - 1)} \end{aligned}$$

and $x_0 = x(0)$, $x_1 = x(1)$, and $x_2 = (1 + w - \phi_1 - \phi_2)x_1 - wx_0 + \phi_1 y + \phi_2 \hat{y}$.

From equation (2),

$$x_{t+1} - x_t = v_{t+1} \tag{28}$$

Therefore,

$$\begin{aligned} v_{N+1} &= (x_{N+1} - x_N) \\ &= k_1 + k_2\lambda_1^{N+1} + k_3\lambda_2^{N+1} - (k_1 + k_2\lambda_1^N + k_3\lambda_2^N) \\ &= (k_2\lambda_1^N(\lambda_1 - 1) + k_3\lambda_2^N(\lambda_2 - 1)) \end{aligned} \tag{29}$$

Note that $\max\{||\lambda_1||, ||\lambda_2||\} < 1$, so that $\lim_{N\to+\infty} \lambda_1^N = 0$ and $\lim_{N\to+\infty} \lambda_2^N = 0$, which implies that there exists a finite N such that

$$(k_2\lambda_1^N(\lambda_1 - 1) + k_3\lambda_2^N(\lambda_2 - 1)) < \epsilon' \tag{30}$$

for any $\epsilon' > 0$. Note that this holds for every dimension of the vector $\mathbf{v}$, thus

$$\|\mathbf{g}^k(\mathbf{x}_{i,t}) - \mathbf{g}^{k+1}(\mathbf{x}_{i,t})\| = \|\mathbf{v}_{t+1}\| < \epsilon, \quad \forall k \geq N$$

■

This is sufficient to show that the particles will stop moving in the deterministic version of the PSO. The next two lemmas indicate where the particles will come to rest.

Lemma 5.2. The global best particle will converge onto the position $\hat{\mathbf{y}}$.

Proof: Let τ represent the index of the global best particle. Then $\hat{\mathbf{y}} \triangleq \mathbf{y}_\tau$. From Equation (27) it is clear that

$$\lim_{t \to +\infty} x_t = k_1 = \frac{\phi_1 y + \phi_2 \hat{y}}{\phi_1 + \phi_2} \tag{31}$$

This holds for each dimension of $\mathbf{x}_t$, so that

$$\lim_{t \to +\infty} \mathbf{x}_{\tau,t} = \frac{\phi_1 \mathbf{y}_\tau + \phi_2 \hat{\mathbf{y}}}{\phi_1 + \phi_2} = \hat{\mathbf{y}} \tag{32}$$

■

This result applies directly to the deterministic version of the PSO. Using the recent discoveries of Poli [19], this result can directly be extended to the stochastic case, because $\mathbf{y}_\tau = \hat{\mathbf{y}}$. This result will now be extended to the other particles.

Lemma 5.3. All particles $1 \leq i \leq h$ will converge onto the position $\hat{\mathbf{y}}$, so that

$$\lim_{t \to +\infty} \mathbf{x}_{i,t} = \hat{\mathbf{y}}$$

Proof: Lemma A.1 in Appendix A shows that there is a non-zero probability that a particle i will move closer to $\hat{\mathbf{y}}$ by at least a distance $\rho > 0$. This satisfies the local *convergence condition* given in Section 4.2.

Step 2.a.i of the PSO algorithm (refer to Figure 1) ensures that $\mathbf{y}_i$ will be updated if the new position, $\mathbf{x}_{i,t+1}$, yields a smaller value in f. This implies that the *algorithm condition* of Section 4.2 holds, where $\mathbf{x}_{i,t+1}$ substitutes ξ and $\mathbf{x}$ is replaced with $\mathbf{y}_i$.

Application of Theorem 4.1 then implies that $\lim_{t\to+\infty} \mathbf{y}_{i,t} = \hat{\mathbf{y}}$. In the deterministic version of the PSO, direct application of Lemma 5.1 implies that $\lim_{t\to+\infty} \mathbf{x}_{i,t} = \mathbf{y}_{i,t} = \hat{\mathbf{y}}$. Because $\lim_{t\to+\infty} \mathbf{y}_{i,t} = \hat{\mathbf{y}}$, the approach of Poli [19] can be applied to extend this result to the stochastic case to infer that $\lim_{t\to+\infty} \mathbf{x}_{i,t} = \mathbf{y}_{i,t} = \hat{\mathbf{y}}$. ■

The lemmas above identify a defect of the basic PSO algorithm: There exists states in which the particles will stop moving, regardless of whether the global best particle is located at a minimiser. One trivial example is the state where all $\mathbf{x}_i = \mathbf{y}_i = \hat{\mathbf{y}}, \forall i \in \mathbb{Z}, 1 \leq i \leq h$. In this state, no further progress can be made and the search should be terminated. This problem, also empirically observed in [28], is explained in more detail next.

Consider the components of $\mathbf{g}$ separately. Application of $\mathbf{g}_2$ can be viewed as sampling a point from a distribution with hyper-rectangular support $M''_{i,t}$ as illustrated in Figure 3. The side lengths of $M''_{i,t}$ depend on the distance between $\mathbf{x}_{i,t}$ and $\mathbf{y}_{i,t}$. From Lemma 5.3, this distance tends to zero, since in the limit, $\mathbf{x}_{i,t} = \mathbf{y}_{i,t} = \hat{\mathbf{y}}_t$, which violates the convergence condition for local search, since the probability that a point is sampled closer to the optimality region R_ϵ becomes zero before $\hat{\mathbf{y}}$ necessarily reaches R_ϵ. The same argument applies to $\mathbf{g}_3$ and $\hat{\mathbf{y}}$. The basic PSO is therefore said to *converge prematurely*, or *stagnate*.

Examples of states that converge prematurely can easily be constructed. Consider a two-dimensional search space and a swarm with only two particles. One of the particles, say particle 1, will be the global best particle. Let the symbols a_1 through a_5 and p_1, p_2 denote arbitrary constants. Then the swarm will stagnate whenever it reaches the following state:

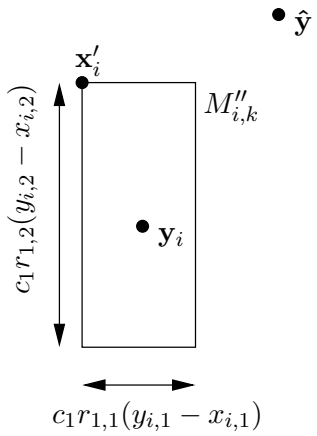


Figure 3. A depiction of the hyper-rectangular support of the space sampled by the function $\mathbf{g}_2$. Note that $\mathbf{y}_i$ need not be included in $M''_{i,k}$ — this depends on the value of c_1.

Particle 1	Particle 2
$\mathbf{v}_1 = a_1(0,1)$	$\mathbf{v}_2 = a_2(0,1)$
$\mathbf{x}_1 = (p_1, p_2)$	$\mathbf{x}_2 = (p_1, p_2) + a_3(0,1)$
$\mathbf{y}_1 = (p_1, p_2) + a_4(0,1)$	$\mathbf{y}_2 = (p_1, p_2) + a_5(0,1)$

This is the state where all the particles are constrained to move only along one of the axes of the search space. There is a non-zero probability that the swarm could reach this state, or it could even have been initialised to this state. If the minimiser of the function is not of the form (p_1, p_3), where p_3 is an arbitrary value, then the swarm will not be able to reach it. The fundamental problem here is that all movement in the swarm is relative to other particles in the swarm.

In summary, there exists initial states in which the original PSO algorithm can start that will lead to a stagnant state in a finite number of steps. By using a large number of particles, the probability of becoming trapped in such a stagnant state is reduced dramatically. It is shown experimentally in Section 9, however, that with only 2 particles the swarm can rapidly stagnate. This supports the theory that the basic PSO algorithm is not a local search algorithm, since it is not guaranteed to locate a minimiser from an arbitrary initial state.

6. Guaranteed Convergence PSO

A solution to the problem of stagnation is to change the velocity update equation for the global best particle to force a position change of the global best. One approach to achieve this was developed by Van den Bergh and Engelbrecht [3], referred to as the guaranteed convergence PSO (GCPSO). Let τ_t be the index of the global best particle at time t, so that

$$\tau_t \triangleq \underset{i}{\operatorname{argmin}} f(\mathbf{y}_{i,t}),\ i \in \mathbb{Z}, 1 \leq i \leq h \tag{33}$$

which, from the definition of $\hat{\mathbf{y}}_t$, implies that

$$\mathbf{y}_{\tau_t} \triangleq \hat{\mathbf{y}}_t. \tag{34}$$

In the absence of an explicit time index, it is assumed that $\hat{\mathbf{y}}$, and thus $\mathbf{y}_\tau$, refers to the most recent value of the global best position of the swarm. Recall that $f(\mathbf{y}_\tau)$ is monotonically non-increasing on a minimisation problem.

The original PSO algorithm applies equation (3) to all of the particles to compute the velocity update for each particle, which in turn is used to update the position of each particle using equation (2). In the GCPSO, the objective is to change the update equation for only the global best particle (particle τ) to

$$x_{\tau j,t+1} = \hat{y}_{j,t} + w v_{\tau j,t} + \rho_t(1 - 2r_t). \tag{35}$$

This new position of the global best particle thus has three terms: the first term which ensures the new position is calculated relative to the current global best position, the second term which carries over the momentum of the global best particle, and the third term which samples a point from a uniform distribution on a hypercube with side lengths 2ρ; the value of ρ is defined below. Provided that ρ is strictly greater than zero, this update equation ensures that the global best particle can never stop moving completely.

All that is required to implement the GCPSO is a new step in the algorithm that computes a different

velocity update for the global best particle. Thus, instead of using equation (3), the velocity update of the global best particle is computed using

$$v_{\tau j,t+1} = -x_{\tau j,t} + \hat{y}_{j,t} + wv_{\tau j,t} + \rho_t(1 - 2r_t). \tag{36}$$

Observe that substitution of equation (36) into equation (2) yields equation (35). The scaling factor ρ_t is defined as

$$\rho_{t+1} = \begin{cases} 2\rho_t & \text{if } \#successes > s_c \\ 0.5\rho_t & \text{if } \#failures > f_c \text{ and } \rho_t > \epsilon_m \\ \rho_t & \text{otherwise} \end{cases} \tag{37}$$

where ϵ_m represents the smallest allowable value of ρ — typically set to machine precision. Assuming minimisation, a 'failure' occurs when $f(\hat{\mathbf{y}}_t) \geq f(\hat{\mathbf{y}}_{t-1})$, and the counter variable $\#failures$ is subsequently incremented (i.e., no apparent progress has been made). A success then occurs when $f(\hat{\mathbf{y}}_t) < f(\hat{\mathbf{y}}_{t-1})$. Upon a success, the failure count is reset to zero; conversely, when the failure count is increased, the success count is reset to zero. The control threshold values were fixed at $f_c = 5$ and $s_c = 5$.

Equation (35) represents the action of sampling a point from a hypercube with side lengths 2ρ centered around $\hat{\mathbf{y}} + w\mathbf{v}$. Let M_k denote this hypercube, and let μ_k denote the uniform probability measure defined on the hypercube M_k. Figure 4 illustrates how a new sample $\mathbf{x}_{\tau,k+1}$ is generated. Stagnation is prevented by ensuring that $\rho > 0$ for all time steps. Before proving that the GCPSO is a local search algorithm, a few details regarding $\mathbf{x}_{\tau,k}$ must first be discussed.

Note that $\hat{\mathbf{y}}_k$ is always in L_0. It is possible, however, that $\mathbf{x}_{\tau,k} \notin L_0$, due to the cumulative effect of a growing $\mathbf{v}$ vector, so that $\hat{\mathbf{y}}_k + w\mathbf{v}_k \notin L_0$. One of two scenarios now unfolds: $\hat{\mathbf{y}}_k \in M_k$ or $\hat{\mathbf{y}}_k \notin M_k$. In the first case, this means that a point arbitrarily close to $\hat{\mathbf{y}}_k$ may be sampled, including $\hat{\mathbf{y}}_k$ itself. Since $\hat{\mathbf{y}}_k \in L_0$, this means that $m[M_k \cap L_0] > 0$. The second case implies that ρ is such that M_k does not include $\hat{\mathbf{y}}_k$. This happens when $\mathbf{v}_{\tau,t}$ points outwards from L_0. Since $\hat{\mathbf{y}}_k$ is only updated when a better solution is found, and from the definition of L_0, it is therefore clear that none of the points outside of

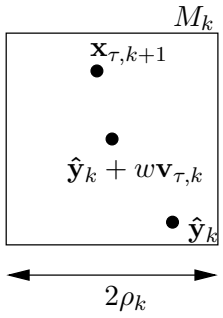


Figure 4. The hyper-cubic support M_k of the sample space μ_k, centered around the point $\hat{\mathbf{y}}_k + w\mathbf{v}_k$, as defined by the position update step of the GCPSO. The point $\mathbf{x}_{k+1}$ is an example of a point sampled by the new velocity update step.

L_0 will be selected to replace $\hat{\mathbf{y}}_k$. On the other hand, $\mathbf{x}_{\tau,t}$ is able to move outside of L_0 because of the residual velocity $\mathbf{v}_{\tau,t}$. Assume for the moment that ρ is insignificantly small. Equation (3) shows that, if $w < 1$, then clearly $||\mathbf{x}_{\tau,t+1} - \hat{\mathbf{y}}_k|| < ||\mathbf{x}_{\tau,t} - \hat{\mathbf{y}}_k||$, assuming that $\mathbf{x}_{\tau,t} \neq \hat{\mathbf{y}}_k$. After a finite number of time steps l, $\mathbf{x}_{\tau,t+l}$ will be in L_0 once more. This implies that $m[M_{k+l} \cap L_0] > 0$, so that a point arbitrarily close to $\hat{\mathbf{y}}_k$ can be sampled once more.

Both cases imply that a new sampled point arbitrarily close to $\hat{\mathbf{y}}_k$, and thus in L_0, can be generated. Note that the second case only comes into play when $\hat{\mathbf{y}}_k$ is close to the boundary of L_0. In low-dimensional search spaces, the first case, where $M_k \subset L_0$, can be considered the norm, however, the opposite may happen more frequently in high-dimensional search spaces. This would seem to indicate that the efficiency of the algorithm will deteriorate in high-dimensional search spaces.

The existence of a non-degenerate sampling volume μ_k with support M_k has thus been shown for the GCPSO algorithm. Using this fact, it is now possible to consider the local convergence property of the GCPSO algorithm.

If it is assumed that $\mathcal{S}$ is compact and has a non-empty interior, then L_0 will also be compact with a non-empty interior. Further, L_0 will include the essential infimum, contained in the optimality region R_ϵ, by definition. Now R_ϵ is compact with a non-empty interior, thus a ball B' centered at $\mathbf{c}'$ contained in R_ϵ can be defined as shown in Figure 5. Pick the point $\mathbf{x}' \in \text{argmax}_{\mathbf{x}}\{\text{dist}(\mathbf{c}', \mathbf{x}) | \mathbf{x} \in L_0\}$, as illustrated in Figure 5. Let B be the hypercube centered at $\mathbf{c}'$, with sides of length $2(\text{dist}(\mathbf{c}', \mathbf{x}') - 0.5\rho)$.

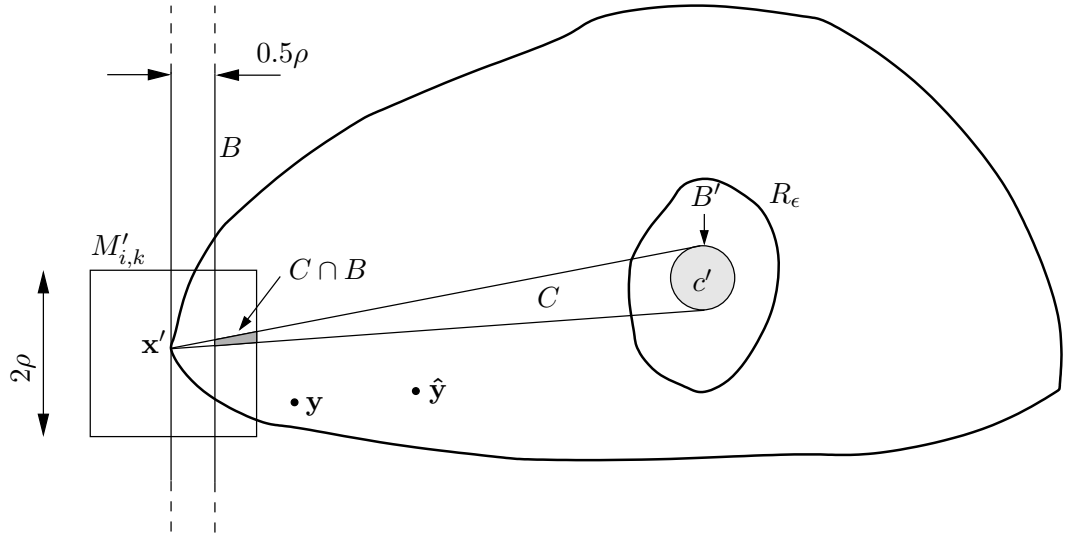


Figure 5. The intersection $C \cap B$, with B centered on $c' \in R_\epsilon$.

Let C be the convex hull of $\mathbf{x}'$ and B' (see Figure 5). Consider a line tangent to B', passing through $\mathbf{x}'$ (i.e., one of the edges of C). This line is the longest such line, for $\mathbf{x}'$ is the point furthest from B'. This implies that the angle subtended by $\mathbf{x}'$ is the smallest such angle of any point in L_0. In turn, this implies that the volume $C \cap B$ is smaller than that of $C' \cap B$ for any other convex hull C' defined by any arbitrary point $\mathbf{x} \in L_0$.

Then for all $\mathbf{x}$ in L_0

$$\mu_k[\text{dist}(D(\hat{\mathbf{y}}, \mathbf{x}_\tau), R_\epsilon) < \text{dist}(\mathbf{x}, R_\epsilon) - 0.5\rho] \geq \eta = \mu[C \cap B] > 0 \tag{38}$$

where μ_k is the uniform distribution on the hypercube centered at $\mathbf{x}$, with side length 2ρ. It was shown above that the modified PSO can provide such a hypercube.

Since $\mu[C \cap B] > 0$, the probability of selecting a new point $\mathbf{x}$ so that it is closer to the optimality region R_ϵ is always non-zero. This is sufficient to show that the Guaranteed Convergence Particle Swarm Optimiser (GCPSO) complies with the convergence condition for local search, because

1. The GCPSO can always generate a sample around a point in L_0, assuming $\hat{\mathbf{y}}, \mathbf{y}_i \in L_0$;

2. Given any starting point in L_0, the GCPSO algorithm guarantees a non-degenerate sampling volume with a non-zero probability of sampling a point closer to the optimality region, R_ϵ.

The GCPSO algorithm thus satisfies both the algorithm and convergence conditions for local search.

This completes the proof that the sequence of values $\{\hat{\mathbf{y}}_k\}_{k=0}^{\infty}$ generated by the GCPSO algorithm will converge to the optimality region, under the constraints of a local optimisation algorithm, regardless of the initial state of the swarm.

6.1. Functions with multiple minima

It was assumed above that L_0 was convex-compact. A non-unimodal function, with $\mathcal{S}$ including multiple minima, may result in a non-convex set L_0.

Of particular interest would be the class of functions that can be decomposed into a set of convex-compact subsets, each subset containing only one local minimum. By Lemma 5.3, it is clear that all the particles in the swarm will eventually converge onto the position of the global best particle. In the GCPSO, the global best particle, in turn, is guaranteed to converge on a local minimum — there is just no guarantee that this will be any specific minimum, for example, a global minimum.

The convergence proofs presented above do not provide any insight into the behaviour of the GCPSO on functions that can not be decomposed into such convex-compact subsets.

7. Global Convergence of PSO

In Section 5 it was shown that the original PSO is not guaranteed to converge on a local minimum. From this, the following trivial proof follows:

Lemma 7.1. Let $\mathbf{x}^*$ be the global minimiser of f on $\mathcal{S}$, and let $\mathbf{x}_t$ be the sequence of proposed solutions generated by the original PSO algorithm. Then $P\left(\lim_{t\to+\infty} \mathbf{x}_t \neq \mathbf{x}^*\right) > 0$.

Proof by counter-example: Assume a unimodal function f, with its global minimum located in the convex-compact set, L_0, as defined in Equation (17). As shown in Section 5, the original PSO may converge on a point $\mathbf{x} \neq \mathbf{x}^*$, and is thus not guaranteed to converge on the minimum located in L_0. ■

In contrast with the original PSO, it was shown in Section 6 that the GCPSO is guaranteed to converge on the local minimiser of f. It is stated without proof that the GCPSO is not able to guarantee convergence to a global minimiser, owing to its strong local convergence properties. Instead, several new algorithms are constructed in Section 8 in a way that guarantees convergence onto a global minimiser.

8. Stochastic Global PSOs

In Section 6 it was proved that the GCPSO algorithm converges on a local minimiser with probability 1, as the number of iterations approaches infinity. It is possible to extend the GCPSO to become a stochastic global search algorithm, so that it will locate the global minimiser of the objective function. Two algorithms with this property are introduced below.

8.1. Random Particle Approach

The simplest way to construct a PSO-based global search algorithm is to directly address the global search convergence condition. This can be achieved by adding *randomised particles* to the swarm. Particle i can be made a randomised particle by simply resetting its position to a random position in search space periodically.

Any number of particles in the swarm can be made random particles, but the optimal ratio of random versus normal particles depends on the swarm size. Let s_{rand} denote the number of random particles in the swarm. One possible implementation of the random particle approach is outlined in Figure 6. This implementation resets a specific particle's position only every s_{rand} iterations, allowing the particle to explore the region in which it was initialised before resetting it again. The resulting algorithm is called the Randomised Particle Swarm Optimiser, or RPSO.

It is trivial to show that the RPSO algorithm is a global search algorithm. The personal best position update equations are unaltered, thus the algorithm clearly satisfies the algorithm condition, as was shown for the original PSO in Section 5.1. During each iteration one particle assumes a random position in the search space. The sample space from which this sample is drawn has support $M_k = \mathcal{S}$, so that

Create and initialise an n-dimensional PSO : P
$s_{idx} \leftarrow 0$
repeat:
  if $s_{idx} \neq \tau$
    then $P.\mathbf{x}_{s_{idx}} = \text{random\_vector}()$
  $s_{idx} = (s_{idx} + 1) \text{ modulo } s_{rand}$
  for each particle $i \in [1..s]$ **:**
    if $f(P.\mathbf{x}_i) < f(P.\mathbf{y}_i)$
      then $P.\mathbf{y}_i = P.\mathbf{x}_i$
    if $f(P.\mathbf{y}_i) < f(P.\hat{\mathbf{y}})$
      then $P.\hat{\mathbf{y}} = P.\mathbf{y}_i$
  endfor
  Perform PSO updates on P using equations (1)–2)
until stopping condition is true

Figure 6. Pseudo code for the RPSO algorithm.

$m[M_k] = m[\mathcal{S}]$. This satisfies the global search convergence condition, so by Theorem 4.2 this is a global search algorithm.

8.2. Multi-start Approach

A different method of extending the GCPSO algorithm to become a global search algorithm can be constructed as follows:

1. Initialise all the particles to random positions in the search space.
2. Run the GCPSO algorithm until it converges on a local minimiser. Record the position of this local minimiser, and return to step 1.

The above algorithm is referred to as the Multi-start PSO (MPSO). There exist several criteria that can be used to determine whether the GCPSO algorithm has converged.

Maximum Swarm Radius: The maximum swarm radius can be computed directly using

$$r = \|\mathbf{x}_m - \mathbf{x}_\tau\|, \; m \in \mathbb{Z}, 1 \leq m \leq h$$

where

$$\|\mathbf{x}_m - \mathbf{x}_\tau\| \geq \|\mathbf{x}_i - \mathbf{x}_\tau\|, \; \forall i \in \mathbb{Z}, 1 \leq i \leq h$$

The normalised radius,

$$r_{\text{norm}} = \frac{r}{\text{diam}(\mathcal{S})}$$

can then be used to decide how close the particles are to the global best particle. The analysis in [1, 4] (also refer to Section 5.1) suggests that the swarm stagnates when all the particles coincide with the global best particle in search space. When r_{norm} is close to zero, the swarm has little potential for improvement, unless the global best particle is still moving. Alternatively, a single particle may still be wandering around while all the other particles have already coincided with the global best particle. Therefore this method is not robust, especially since it does not take the objective function values into account.

It was found empirically that re-starting the swarm when $r_{\text{norm}} < 10^{-6}$ produced acceptable results on a test set of benchmark functions. Clearly, larger thresholds will increase the sensitivity of the algorithm, but note that re-starting the swarm too frequently will prevent it from performing a fine-grained search.

Cluster Analysis: A more aggressive variant of the Maximum Swarm Radius method can be constructed by clustering the particles in search space. The clustering algorithm works as follows:

1. Initialise the cluster C to contain only the global best position
2. All particles such that $\text{dist}(\mathbf{x}_i, C) < r_{\text{thresh}}$ are added to the cluster
3. Repeat step 2 at least 5 times.

The ratio $|C|/h$ is then computed, where $|C|$ denotes the number of particles in the cluster — note that only a single cluster is grown. If the ratio is greater than some threshold, say 0.6, then the swarm is considered to have converged. Note that this method has the same flaws as the Maximum Swarm Radius technique, except that it will more readily decide that the swarm has converged.

Empirical results obtained on a small set of test functions indicated that a value of $r_{\text{thresh}} = 10^{-6}$ produced acceptable results; the swarm was declared to have converged when more than 60% of the particles were clustered around the global best particle. Using a ratio of more than 60% decreases the sensitivity of the algorithm, bringing with it the possibility of failing to detect stagnation early on. Larger r_{thresh} values will increase the sensitivity, similarly to the corresponding threshold value in the Maximum Swarm Radius technique.

Objective Function Slope: This approach does not take into account the relative positions of the particles in search space; instead it bases its decision solely on the rate of change in the objective function. A normalised objective function value is obtained by using

$$f_{ratio} = \frac{f(\hat{\mathbf{y}}_t) - f(\hat{\mathbf{y}}_{t-1})}{f(\hat{\mathbf{y}}_t)}$$

Note that f_{ratio} is undefined if $f(\hat{\mathbf{y}}) = 0$. However, if the global minimum value of f is in fact 0, then the algorithm has already found the global minimum, and there is no need to restart the swarm. Note that this normalisation may cause problems for functions that do not have a function value of 0 at their global minima. All of the test functions examined in Section 9 have this property, though. If this normalised value is smaller than some threshold, a counter is incremented. Once the counter reaches a certain threshold, the swarm is assumed to have converged. This approach is superior to the other two methods mentioned first in that it actually determines whether the swarm is still making progress, instead of trying to infer it from the positions of the particles. There is one remaining flaw with this approach, though. If half of the particles (including the global best particle) are trapped in the basin of some minimum, the other half may yet discover a better minimum in the next few iterations. This possibility can be countered by using one of the first two methods to check for this scenario.

The optimal threshold for the f_{ratio} value depends on the range of the objective function values, as well as the machine precision of the platform on which the algorithm is implemented. Empirical results indicated that a value of 10^{-10} works well. Smaller thresholds increase the sensitivity of

```
Create and initialise an n-dimensional PSO : P
repeat:
  if f(P.ŷ) < f(z)
    then z = P.ŷ
  if P has converged
    then re-initialise all particles
  for each particle i ∈ [1..s] :
    if f(P.x_i) < f(P.y_i)
      then P.y_i = P.x_i
    if f(P.y_i) < f(P.ŷ)
      then P.ŷ = P.y_i
  endfor
  Perform PSO updates on P using equations (1)–(2)
until stopping condition is true
```

Figure 7. Pseudo code for the MPSO algorithm.

the algorithm, possibly causing the algorithm to mistake a period of slow progress for stagnation.

Figure 7 is the outline of an algorithm making use of the multi-start (or restart) approach. Any of the convergence criteria mentioned above can be used. This type of algorithm is called the Multi-start Particle Swarm Optimiser (MPSO).

The MPSO algorithm is a global search algorithm, a property that will now be proved using Theorem 4.2. The MPSO satisfies the algorithm condition, similarly to the RPSO in the previous section. To satisfy the global search convergence condition, the MPSO must be able to restart an infinite number of times. This requires that the GCPSO algorithm converges onto a local minimum, which was proved in Section 6, and that the convergence-detection mechanism subsequently detects this. The Maximum Swarm Radius and Cluster Analysis methods indicate that the swarm has converged when the particles are arbitrarily close to the global best position, $\hat{\mathbf{y}}$. Lemma 5.3 has shown that the particles tend to the state $\mathbf{x}_i = \mathbf{y}_i = \hat{\mathbf{y}}$. Since the swarm is guaranteed to reach this state, these two convergence criteria will always detect convergence and trigger a restart. The Objective Function Slope criterion will detect convergence whenever the value of $f(\hat{\mathbf{y}})$ stops changing. This state is guaranteed, upon discovery of a local minimum, by the local convergence property of the GCPSO. This implies that, regardless of the

convergence-detection criterion used, the MPSO algorithm will be able to re-initialise the positions of all the particles an infinite number of times, if it is allowed to run for an infinite number of iterations.

The re-initialisation process assigns to each particle a position sampled from the whole search space $\mathcal{S}$. Since the support of this sample space, M_k, is equal to the search space, $m[M_k] = m[\mathcal{S}]$. This satisfies the global search convergence condition, which means that, by Theorem 4.2, the MPSO is a global search algorithm.

8.3. Rate of Convergence

The rate of convergence of a stochastic global method like the MPSO is directly dependent on the volume of the sample space, since the number of points in the sample space grows exponentially with the dimension of the search space. This implies that the MPSO algorithm will easily find the global minimiser of a 2-dimensional objective function in a relatively small number of iterations. If a function of comparable complexity in 20 dimensions is considered, the algorithm will take significantly longer to find the global minimiser. An indication of the severity of the problem can be obtained by considering the following simple example. Let d be the side length of a hypercube defining the optimality region R_ϵ. The volume of the optimality region is then d^n, where n is the number of dimensions. If the search space S is a hypercube with side lengths l, then its volume will be l^n. The probability of generating a sample in the optimality region in the first iteration of the algorithm, assuming a uniform distribution function on S, is then

$$\frac{d^n}{l^n} = \left(\frac{d}{l}\right)^n$$

Since the optimality region is certainly smaller than the search space itself, this implies that $d/l < 1$, so that

$$\lim_{n \to +\infty} \left(\frac{d}{l}\right)^n = 0$$

If a pseudo-random number algorithm is used to generate the samples used by the search algorithm, and the period of the generator is sufficiently large, then the sampling process will be equivalent to sampling without replacement. The probability of sampling from the optimality region will thus increase slightly

on successive samples, but this will not have a significant impact since the number of points in the search space still grows exponentially with the number of dimensions. This implies that the ability of the MPSO to find the global minimiser of a function in a finite number of iterations deteriorates very rapidly as the number of dimensions is increased.

The stretching technique proposed by Parsopoulos *et al.* [18] uses the equivalent of a MPSO using the original PSO instead of the GCPSO for the local search component of their algorithm. They do not provide any proofs of global convergence, but they present empirical results to support the claim that their method is global. Their algorithm modifies the objective function, and re-initialises the particles once a local minimiser is discovered. It is important to realise, though, that their algorithm's ability to locate the global minimiser comes from the periodic re-initialisation, and not from the transform that they apply to the objective function. This means that their algorithm is subject to the same limitation that the MPSO suffers: the curse of dimensionality. The examples they present are all restricted to two dimensions, which means that the re-initialisation method will have a good chance of finding the global minimiser within a small number of iterations. These results are misleading, since the algorithm will perform significantly worse on, say, 100-dimensional problems.

8.4. Stopping Criteria

While it is relatively simple to design a stopping criterion for a local search algorithm, it is rather more involved to do so for a global search algorithm, unless the value of the objective function in the global minimiser is known in advance. To illustrate: when the GCPSO algorithm fails to improve the value of $f(\hat{\mathbf{y}})$ over many consecutive iterations, it is relatively safe to assume that it has found a local minimum. When the MPSO exhibits the same behaviour, that is, it fails to improve on the best solution discovered so far after, say, 100 restarts, no such conclusion can be drawn. As illustrated above, the probability of sampling a point from the optimality region decreases exponentially as the number of dimensions increases. This means that the number of restarts required before giving up must grow accordingly, as will now be shown.

Solis and Wets provided some guidelines for choosing the correct number of iterations required for a

stochastic search algorithm to discover the global minimiser [24]. For example, they defined the number of iterations required to reach the optimality region, N_λ, with at least probability $1-\lambda$ as follows:

$$P[\hat{\mathbf{y}}_k \notin R_\epsilon] \leq \lambda, \quad \forall k > N_\lambda$$

Let k denote the number of restarts in the MPSO algorithm. Then, if a value m is known so that $0 < m \leq \mu_k[R_\epsilon]$ for all k, then

$$P[\hat{\mathbf{y}}_k \notin R_\epsilon] \leq (1-m)^k$$

Choosing an integer

$$N_\lambda \geq \left\lceil \frac{\ln \lambda}{\ln(1-m)} \right\rceil$$

yields the required property, since when $k \geq N_\lambda$, then $\ln\lambda / \ln(1-m) \leq k$, because $(1-m)^k \geq \lambda$. Note that this calculation requires that a lower bound m is known for $\mu_k[R_\epsilon]$, which implies that the size and shape of R_ϵ is known in advance. If the example from Section 8.3 is continued, we can use the value $m = (d/l)^n$ as such a lower bound. This implies that the number of iterations required to reach R_ϵ with probability $1-\lambda$ is

$$N_\lambda \geq \frac{\ln \lambda}{\ln\left(1-\left(\frac{d}{l}\right)^n\right)}$$

However, $\ln\left(1-\left(\frac{d}{l}\right)^n\right) \rightarrow 0$ as $n \rightarrow +\infty$, which means that $N_\lambda \rightarrow +\infty$, confirming our earlier suspicions.

9. Experimental Results

This section contains some brief results regarding the behavior of the original PSO and the GCPSO, as well as the stochastic global PSOs. There are two main themes that were investigated:

1. Does the guaranteed convergence of the GCPSO translate into improved performance on unimodal and/or multimodal functions, compared to the original PSO? How does the swarm size (h) influence the results?

2. Do the proposed global variants of the PSO (Section 8.2) improve on the performance of the GCPSO on multimodal functions?

To address these questions, the following functions were used for testing:

Sphere:

$$f_0(\mathbf{x}) = \sum_{j=1}^{n} x_j^2 \tag{39}$$

Rosenbrock (or banana-valley):

$$f_1(\mathbf{x}) = \sum_{j=1}^{n/2} [100(x_{2j} - x_{2j-1}^2)^2 + (1 - x_{2j-1})^2] \tag{40}$$

Quadric:

$$f_2(\mathbf{x}) = \sum_{j=1}^{n} \left(\sum_{k=1}^{j} x_j \right)^2 \tag{41}$$

Generalised **Rastrigin**:

$$f_3(\mathbf{x}) = \sum_{j=1}^{n} (x_j^2 - 10\cos(2\pi x_j) + 10) \tag{42}$$

Generalised **Griewank**:

$$f_4(\mathbf{x}) = 1 + \frac{1}{4000} \sum_{j=1}^{n} x_j^2 - \prod_{j=1}^{n} \cos(\frac{x_j}{\sqrt{j}}) \tag{43}$$

Schwefel:

$$f_5(\mathbf{x}) = \sum_{j=1}^{n} x_j \sin(\sqrt{|x_j|}) + 418.9829n \tag{44}$$

Ackley:

$$f_6(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{j=1}^{n} x_j^2}} - e^{\frac{1}{n}\sum_{j=1}^{n}\cos(2\pi x_j)} + 20 + e \tag{45}$$

This set of functions contains representatives of both the unimodal and multimodal objective function classes. Table 1 indicates the dimensions for which the functions were evaluated, as well as the size of the search domain.

Table 1. Parameters used for all experiments.

Function	**n**	**Domain**
f_0	30	[-100,100]
f_1	30	[-2.048,2.048]
f_3	30	[-100,100]
f_4	30	[-5.12,5.12]
f_5	30	[-600,600]
f_6	30	[-500,500]

The details of the PSO algorithms that featured in the experiments are provided in Sections 9.1 and 9.2. To provide an approximate idea of PSO performance relative to evolutionary algorithms, results obtained with a genetic algorithm were included in the global optimisation experiments of Section 9.2. The GA employed a binary coding scheme with 16 bits allocated to each variable. A two-point bitwise crossover operator was used alongside a single-bit mutation operator. The crossover rate was set to 0.6 and the mutation rate to 0.02. The population size was fixed at 100 members, with selection pressure exerted using a fitness-proportionate model. A one-element elitist strategy ensured that the algorithm retained the best solution over all generations. This specific configuration corresponds to the GA used by Potter in his CCGA experiments [20], and was included here to preserve continuity with a previous comparison between a genetic algorithm and a PSO [2].

9.1. PSO vs GCPSO

The two algorithms used for the first experiments are as follows:

PSO: The original PSO algorithm, using equations (3) and (2).

GCPSO: The modified PSO, using equations (36) and (35) to update the global best particle, while using equations (3) and (2) for all the other particles. The critical thresholds s_c and f_c, defined in equation (37), were both set to 5.

Both algorithms were configured to use parameter settings of $c_1 = c_2 = 1.4961798$ and $w = 0.729844$, since these values lead to rapid convergence [9, 4]. The swarm size, h, was varied from 2 to 20 in

Table 2. Mean function values of global best particle after 2×10^5 evaluations, over 500 simulations.

Fn	PSO (h=2)	GCPSO (h=2)
f_0	3.97e+04 ± 9.87e+03	4.59e−320 ± 0.00e+00
f_1	1.45e+03 ± 8.00e+02	3.58e−01 ± 3.93e−01
f_2	8.60e+04 ± 4.96e+04	6.64e−13 ± 1.64e−12
f_3	2.93e+02 ± 4.29e+01	1.72e+02 ± 4.11e+01
f_4	3.53e+02 ± 9.18e+01	1.17e−02 ± 1.41e−02
f_5	8.82e+03 ± 7.63e+02	7.31e+03 ± 1.41e+03
f_6	1.91e+01 ± 6.95e−01	1.83e+01 ± 6.40e−01

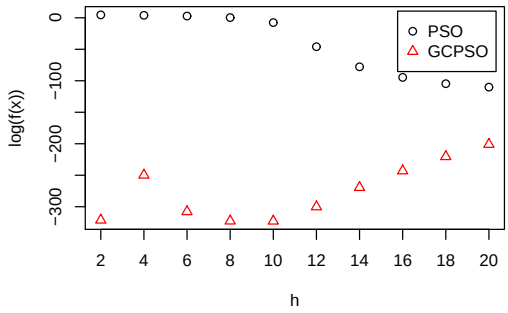
Fn	PSO (h=20)	GCPSO (h=20)
f_0	1.92e−93 ± 4.30e−92	1.28e−191 ± 0.00e+00
f_1	1.15e−02 ± 2.53e−02	3.86e−02 ± 4.45e−02
f_2	4.27e−15 ± 5.68e−14	1.12e−21 ± 1.12e−20
f_3	6.91e+01 ± 1.85e+01	7.42e+01 ± 1.93e+01
f_4	1.16e−01 ± 2.86e−01	2.86e−02 ± 8.53e−02
f_5	4.50e+03 ± 6.16e+02	4.85e+03 ± 6.68e+02
f_6	3.47e+00 ± 1.55e+00	3.14e+00 ± 1.85e+00

increments of 2 to investigate the effects of swarm size on stagnation. In order to keep the comparisons across swarm sizes fair, the total number of function evaluations was kept constant at 2×10^5 evaluations, which implied that the number of generations was reduced as the swarm size was increased. It is expected that the PSO will stagnate more easily with small swarm sizes, while the GCPSO is expected to be able to avoid stagnation even for swarm sizes of $h = 2$. Table 2 presents the mean function value over 500 runs obtained from the global best particle after 2×10^5 function evaluations for each of the four configurations indicated, where h denotes the swarm size. The median function values of the global best particle after 2×10^5 function evaluations over 500 runs are plotted against swarm sizes in the range 2 to 20 in Figure 8; the median was selected to produce cleaner plots through its greater robustness to outliers.

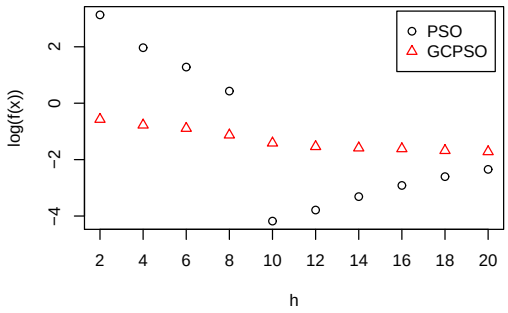
The functions belong to two different categories: functions f_0–f_2 are strictly unimodal, while functions f_3–f_6 contain many local minima. Since neither of the two algorithms tested here are designed to find the global minimum in the presence of many local minima, the results for functions f_3–f_6 should be seen only as an indication of their ability to converge in the presence of multiple minima.

The trend is clearly visible: when considering swarms containing only 2 particles, the GCPSO performs significantly better than the PSO, especially on functions f_0–f_2. On the other functions, the improvement is less dramatic, as both algorithms tend to become trapped in the same type of local minimum.

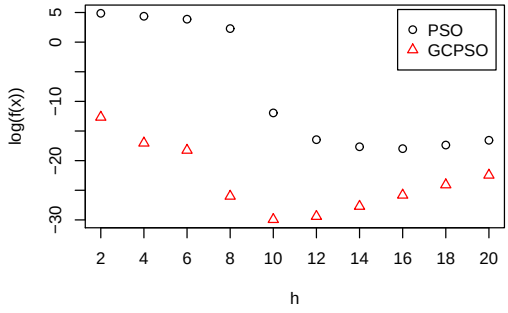
When considering the experiments with a more typical 20 particles per swarm, the GCPSO does not appear to have a definite advantage, with four of the test functions favouring the GCPSO, and three favouring the PSO. The graphs presented in Figure 8 show a clear pattern for functions f_0 and f_2: the performance of the GCPSO *deteriorates* when the swarm size increases beyond $h = 10$. Even Griewank's function (f_4) appears to exhibit decreasing performance with increasing GCPSO swarm sizes; this may be because the product term in equation (43) is dominated by the quadratic term for large n, so that the function behaves more like f_0. On multi-modal functions f_3 (not shown in Figure 8), f_5, and f_6 the performance of the GCPSO improves with increasing swarm size, just like the original PSO. On the unimodal Rosenbrock function (f_1) the original PSO attains peak performance at $h = 10$;



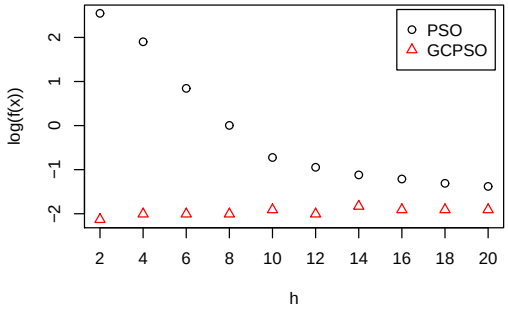
(a) Sphere (f_0)



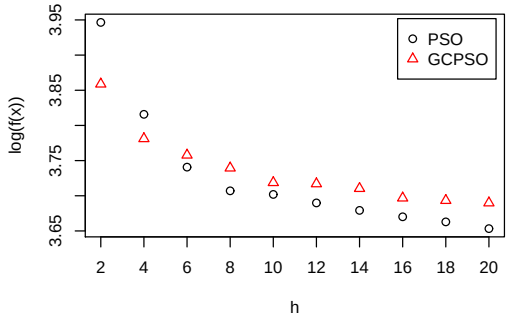
(b) Rosenbrock (f_1)



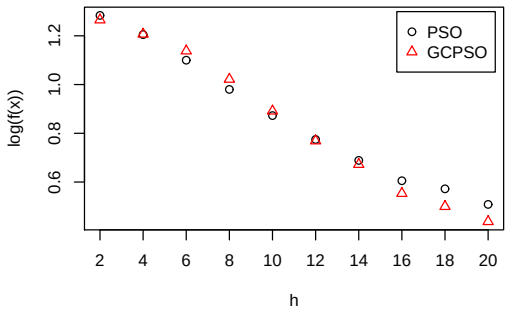
(c) Quadric (f_2)



(d) Griewank (f_4)



(e) Schwefel (f_5)



(f) Ackley (f_6)

Figure 8. Median function value of the global best particle after 2×10^5 function evaluations, computed over 500 simulations. The h-axis represents the swarm size. The profile for Rastrigin's function (f_3) is not shown, but has the same shape and relative ranking as that of f_5.

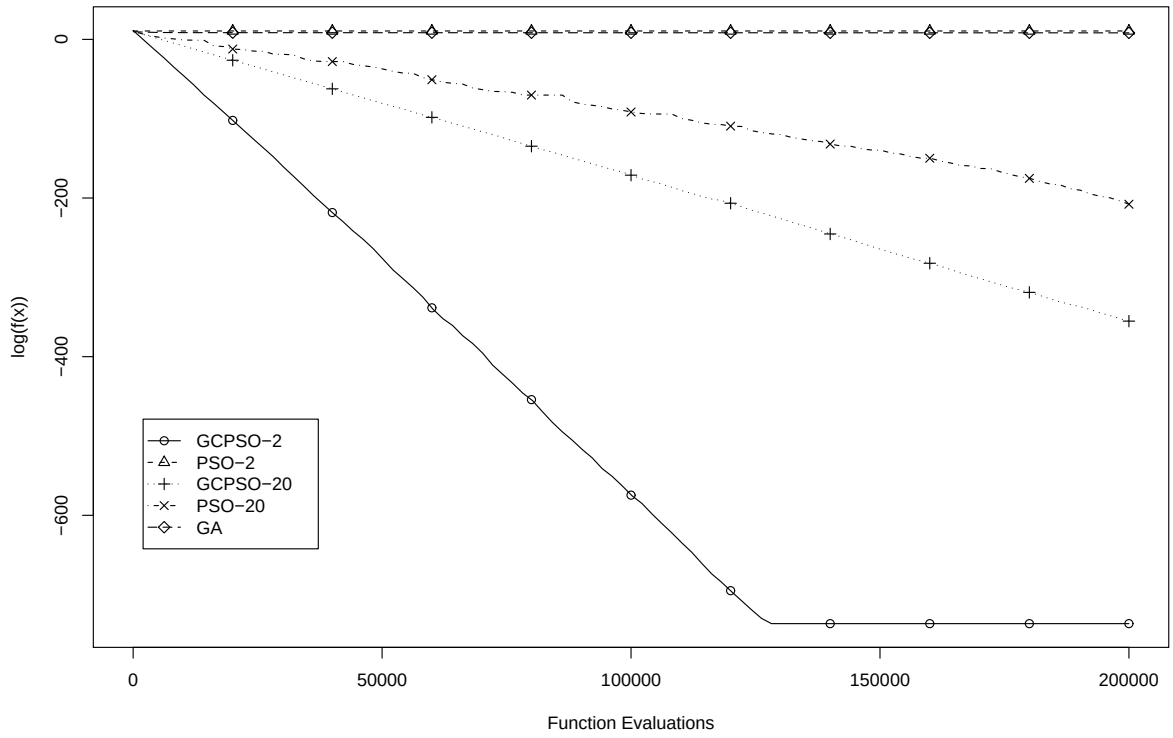


Figure 9. Error profiles for the Sphere function.

although not clearly visible in Figure 8(a) and (c), similar behaviour is observed on the other unimodal functions. For a fixed number of function evaluations, both the original PSO and the GCPSO have an optimal swarm size for strongly unimodal functions; once a sufficient level of diversity is guaranteed by the optimal number of particles, any additional particles will just consume additional function evaluations without contributing to performance. From the results in Table 2 and Figure 8, it would appear that the GCPSO has a smaller optimal swarm size compared to the original PSO. This is a desirable property for cooperative PSO algorithms such as the CPSO [2], where smaller swarm sizes help to decrease the overall computational complexity of the algorithm.

Figures 9 and 10 are profiles of the mean function value of the global best particle over 500 runs, plotted against the number of function evaluations. These values were sampled at every iteration of the respective algorithms, and then downsampled by a factor of 10 to aid visualisation. The PSO-based algo-

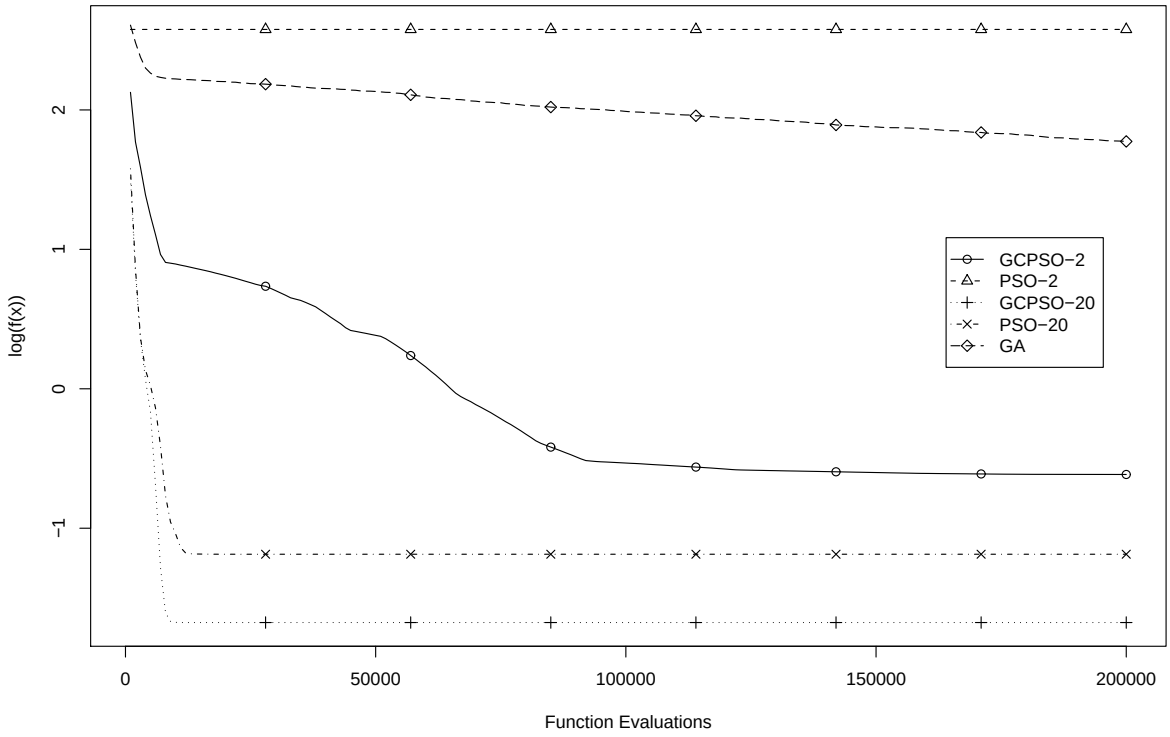


Figure 10. Error profiles for the Griewank function.

rithms have the number of particles used appended to their labels in the legend. Note that the GCPSO-2 curve in Figure 9 becomes flat at around 1.3×10^5 function evaluations – this is most likely caused by the limited machine precision, as the final value returned by the configuration is very close to the smallest number that can be represented on the test machine. Figure 10 shows that none of the PSO algorithms tested here could escape the local minima into which they settled, for their graphs are flat after the first 100000 function evaluations.

9.2. Global PSOs

Section 8 introduced two strategies for extending the PSO algorithm to ensure guaranteed convergence on a global minimiser. Both of these strategies only guarantee convergence on a global optimiser as the number of iterations approach infinity. It therefore remains to be seen if these strategies offer any practical advantages in a finite number of iterations. Experiments involving both the RPSO and the MPSO algorithm were implemented to investigate this matter.

The MPSO algorithm has three different mechanisms (see Section 8.2) that can be used to detect convergence, signaling that the algorithm must re-start. These three convergence-detection methods are included in the comparison, yielding the following algorithms:

MPSO$_{radius}$: The MPSO algorithm, using the *maximum swarm radius* convergence detection technique described in Section 8.2. The algorithm was configured so that the swarm was declared to have stagnated when $r_{\text{norm}} < 10^{-6}$. This value was found (informally) to produce good results on a small set of test functions.

MPSO$_{cluster}$: The MPSO algorithm, using the *cluster analysis* convergence detection technique described in Section 8.2. The value of r_{thresh} was set to 10^{-6}; the swarm was re-initialised whenever more than 60% of the particles were clustered around the global best particle. Again, these values were found empirically to result in acceptable performance on a small set of test functions.

MPSO$_{slope}$: The MPSO algorithm, using the *objective function slope* convergence detection technique described in Section 8.2. The algorithm re-initialised the swarm whenever $f_{ratio} < 10^{-10}$ for more

Table 3. Comparing various global algorithms on Ackley's function.

Algorithm	**Mean** $f(\mathbf{x})$	**Median** $f(\mathbf{x})$
GCPSO	2.96e+00 $\pm$ 6.36e$-$01	2.70e+00
MPSO$_{radius}$	7.51e$-$01 $\pm$ 1.35e$-$01	9.31e$-$01
MPSO$_{cluster}$	1.65e+00 $\pm$ 1.46e$-$01	9.31e$-$01
MPSO$_{slope}$	1.65e+00 $\pm$ 1.70e$-$01	1.34e+00
RPSO	2.96e+00 $\pm$ 3.53e$-$01	2.96e+00

than 500 consecutive iterations. These values were chosen based on previous experience with the algorithm, where it was found that they result in acceptable performance.

RPSO: The RPSO algorithm, described in Section 8.1. No extra particles were added to act as randomised particles, so that three of the 20 normal particles in the swarm were converted into randomised particles, leaving only 17 normal particles. This value was chosen to limit the possibly disruptive influence that the randomised particles could have on the overall behaviour of the swarm.

All experiments were performed using a swarm size of 20, with parameter settings of $c_1 = c_2 = 1.4961798$ and $w = 0.729844$. The GCPSO algorithm used in these experiments was configured identically to the one employed in Section 9.1.

These algorithms are all *stochastic* global optimisers, thus they are not guaranteed to find the global (or even a good local) minimum on every finite run. Since only one poor solution, say on the order of 10^{-3}, can skew the mean of a population otherwise consisting of values on the order of 10^{-19}, the median of each simulation run is also provided along with the mean.

Table 3 presents the results of applying the four global PSO-based algorithms to the task of minimising Ackley's function. The MPSO$_{radius}$ algorithm performed significantly better than any other algorithm. Note that all three of the MPSO algorithms performed significantly better than the GCPSO and RPSO algorithms. On this function it does not appear that the RPSO algorithm had any positive influence on the performance of the algorithm, since it had the same mean performance as the GCPSO algorithm.

Table 4. Comparing various global algorithms on Rastrigin's function.

Algorithm	**Mean** $f(\mathbf{x})$	**Median** $f(\mathbf{x})$
GCPSO	7.61e+01 $\pm$ 5.07e+00	7.61e+01
MPSO$_{radius}$	4.58e+01 $\pm$ 1.45e+00	4.58e+01
MPSO$_{cluster}$	4.48e+01 $\pm$ 1.41e+00	4.48e+01
MPSO$_{slope}$	4.97e+01 $\pm$ 1.59e+00	4.97e+01
RPSO	7.41e+01 $\pm$ 3.49e+00	7.41e+01

Table 5. Comparing various global algorithms on Griewank's function.

Algorithm	**Mean** $f(\mathbf{x})$	**Median** $f(\mathbf{x})$
GCPSO	2.21e$-$02 $\pm$ 4.84e$-$03	1.23e$-$02
MPSO$_{radius}$	1.99e$-$09 $\pm$ 1.87e$-$10	1.52e$-$09
MPSO$_{cluster}$	3.69e$-$02 $\pm$ 8.19e$-$03	1.48e$-$02
MPSO$_{slope}$	1.68e$-$03 $\pm$ 7.39e$-$04	2.17e$-$19
RPSO	4.53e$-$02 $\pm$ 7.72e$-$03	1.23e$-$02

The results in Table 4 were obtained by minimising Rastrigin's function using the various algorithms. Note that there was once again no significant difference between the performance of the RPSO algorithm and that of the GCPSO algorithm. All three the MPSO algorithms performed significantly better than the GCPSO. Amongst themselves the MPSO$_{radius}$ and MPSO$_{cluster}$ algorithms performed significantly better than the MPSO$_{slope}$ algorithm, although there was no significant difference between the performance of the MPSO$_{radius}$ and MPSO$_{cluster}$ algorithms.

Table 5 presents the results of minimising Griewank's function using the various algorithms. Both the MPSO$_{cluster}$ and RPSO algorithms performed significantly *worse* than the GCPSO. Clearly, the MPSO$_{cluster}$ algorithm had no beneficial effect on this function, most likely because it failed to trigger, i.e., it could not detect that the swarm has stagnated. This phenomenon can be attributed to the fact that the adjustable parameter (r_{thresh}) in the MPSO$_{cluster}$ algorithm was set to a value that was too small for Griewank's function. If the function contains many local minima very close to one another, the swarm could stagnate with a few particles scattered over several neighbouring minima. In this case, the cluster

Table 6. Comparing various global algorithms on Schwefel's function.

Algorithm	**Mean** $f(\mathbf{x})$	**Median** $f(\mathbf{x})$
GCPSO	4.87e+03 $\pm$ 1.31e+02	4.87e+03
MPSO$_{radius}$	3.71e+03 $\pm$ 8.07e+01	3.71e+03
MPSO$_{cluster}$	4.85e+03 $\pm$ 1.32e+02	4.85e+03
MPSO$_{slope}$	4.08e+03 $\pm$ 8.25e+01	4.08e+03
RPSO	4.73e+03 $\pm$ 1.38e+02	4.73e+03

detection algorithm could fail, since there are too few particles close to the global best particle.

Note the large difference between the mean and the median values of the MPSO$_{slope}$ algorithm results. Based on the median, it is clear that the MPSO$_{slope}$ algorithm performed better than the MPSO$_{radius}$ algorithm on most of the runs, although the mean does not reflect this. This is an indication that the MPSO$_{slope}$ algorithm did not trigger often enough to detect stagnation during all the simulation runs. This suggests that future research should investigate automated methods for adjusting the sensitivity of the MPSO$_{slope}$ algorithm.

Applying the various algorithms to the task of minimising Schwefel's function yielded the results presented in Table 6. The RPSO and MPSO$_{cluster}$ algorithms did not perform significantly better than the baseline GCPSO algorithm. The other two MPSO algorithms, MPSO$_{slope}$ and MPSO$_{radius}$, did show a significant improvement over the GCPSO algorithm. The MPSO$_{radius}$ algorithm further showed a significant improvement in performance over the MPSO$_{slope}$ algorithm, making it the overall best algorithm on this function.

Figure 11 further elucidates the characteristics of the various algorithms when applied to Griewank's function. The GCPSO, MPSO$_{cluster}$ and RPSO algorithms all behave similarly, clearly failing to make any further improvements after about 10000 function evaluations. This behaviour is normal for the GCPSO, since it is not explicitly designed to deal with multiple local minima. For the RPSO, this implies that the randomised particles have no effect. In fact, on none of the problems did the RPSO offer any improvement in performance over the GCPSO.

The results presented in this section indicate that the MPSO family of algorithms have the ability

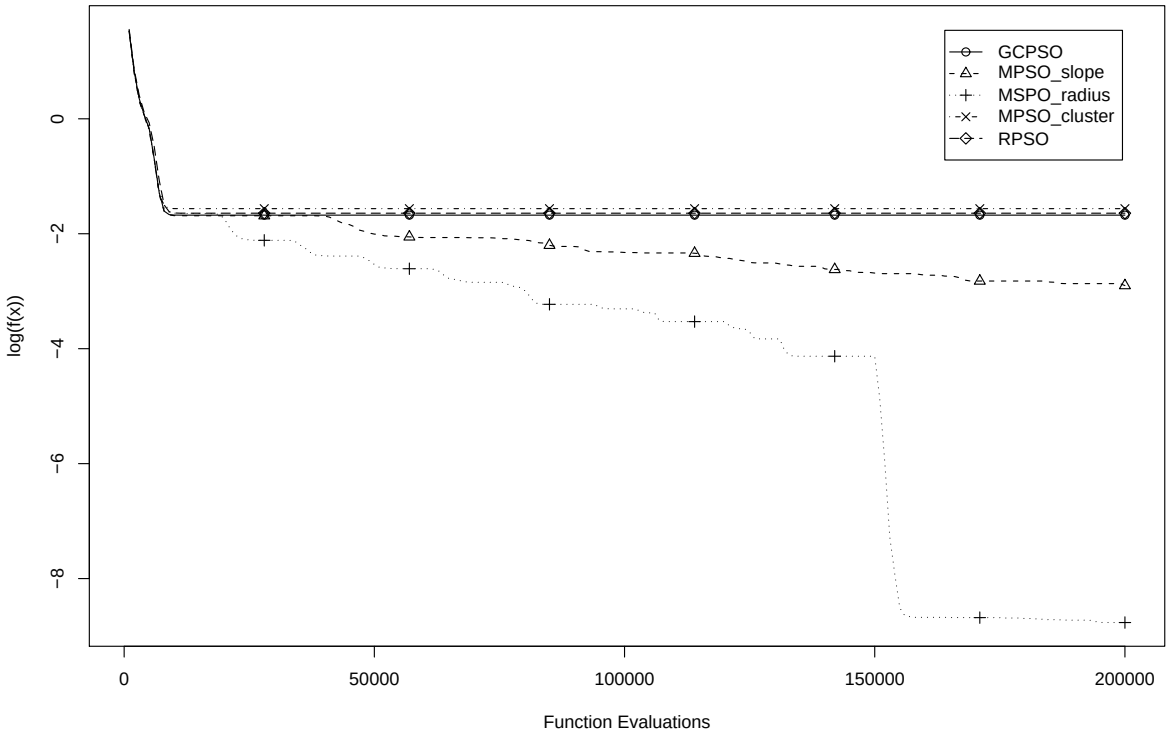


Figure 11. The Griewank function error profile, obtained using the various global PSO algorithms.

to improve the performance of the GCPSO on functions containing multiple minima. The effectiveness of the MPSO algorithm depends on the algorithm used to determine when to re-initialise the swarm. From the results it is clear that the MPSO$_{slope}$ and MPSO$_{radius}$ algorithms were able to improve on the performance of the GCPSO consistently, while the MPSO$_{cluster}$ algorithm sometimes failed to deliver improved performance.

10. Conclusions

The main objective of this paper was to provide convergence proofs for the particle swarm optimiser (PSO). It was formally proven that the original PSO is not a local or global optimiser. A flaw in the PSO was identified and addressed in the new, guaranteed convergence PSO (GCPSO). A proof was given to show that the GCPSO is a local minimiser. Several versions of the PSO with global convergence were presented. Rates of convergence and stopping conditions for the global PSO versions have been explored.

Experiments on unimodal functions show that the theoretical guaranteed local convergence property of the GCPSO does indeed translate into improved performance compared to the original PSO algorithm. The difference in performance between the GCPSO and the original PSO is more pronounced in configurations with smaller swarm sizes. This indicates that stagnation of the original PSO is less of a problem if the swarm is sufficiently large. There are certain configurations, such as the cooperative PSO algorithms, where smaller swarm sizes are desirable. The GCPSO can thus be used to reduce the number of function evaluations required to solve a problem, while maintaining the solution quality.

The experiments on multi-modal functions have shown that the GCPSO does not offer such a clear advantage over the original PSO. This is not completely unexpected, because the guaranteed convergence property of the GCPSO does not improve the global search behaviour of the algorithm explicitly. The RPSO and MPSO variants do have a mechanism that ensures that they will locate the global minimiser, but convergence onto a global minimiser is only guaranteed with the number of iterations approaching infinity. The results obtained with the RPSO and MPSO variants show that the theoretical convergence does not readily translate into improved performance over a finite number of iterations. In particular, the RPSO variant does not appear to offer any improvement over the GCPSO. The MPSO variants do appear

to offer some improvement, but it is clear that a much more sophisticated strategy will be required to improve the rate of convergence onto global minimisers.

References

[1] van den Bergh, F.: *An Analysis of Particle Swarm Optimizers*, Ph.D. Thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[2] van den Bergh, F., Engelbrecht, A.: A Cooperative approach to particle swarm optimization, *Evolutionary Computation, IEEE Transactions on*, **8**(3), June 2004, 225–239.

[3] van den Bergh, F., Engelbrecht, A. P.: A New Locally Convergent Particle Swarm Optimizer, *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Hammamet, Tunisia, 2002.

[4] van den Bergh, F., Engelbrecht, A. P.: A Study of Particle Swarm Optimization Particle Trajectories, *Information Sciences*, **176**(8), 2006, 937–971.

[5] Clerc, M.: The Swarm and the Queen: Towards a Deterministic and Adaptive Particl e Swarm Optimization, *Proceedings of the IEEE Congress on Evolutionary Computation*, 1999.

[6] Clerc, M.: Think Locally, Act Locally: The Way of Life of Cheap-PSO, an Adaptive PSO, 2001.

[7] Clerc, M., Kennedy, J.: The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space, *IEEE Transactions on Evolutionary Computation*, **6**(1), 2002, 58–73.

[8] Eberhart, R. C., Kennedy, J.: A New Optimizer using Particle Swarm Theory, *Proceedings of the Sixth International Symposium on Micromachine and Human Science*, Nagoya, Japan, 1995.

[9] Eberhart, R. C., Shi, Y.: Comparing Inertia Weights and Constriction Factors in Particle Swarm O ptimization, *Proceedings of the IEEE Congress on Evolutionary Computation*, San Diego, USA, 2000.

[10] Eberhart, R. C., Shi, Y.: Particle Swarm Optimization: Developments, Applications and Resources, *Proceedings of the IEEE Congress on Evolutionary Computation*, IEEE Press, Seoul, Korea, 2001.

[11] Eberhart, R. C., Simpson, P. K., Dobbins, R. W.: *Computational Intelligence PC Tools*, First edition, Academic Press Professional, 1996.

[12] Kennedy, J.: Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance, *Proceedings of the IEEE Congress on Evolutionary Computation*, 1999.

[13] Kennedy, J., Eberhart, R. C.: Particle Swarm Optimization, *Proceedings of the IEEE International Joint Conference on Neural Networks*, IEEE Press, 1995.

[14] Kennedy, J., Mendes, R.: Population Structure and Particle Performance, *Proceedings of the IEEE Congress on Evolutionary Computation*, IEEE Press, Honolulu, Hawaii, 2002.

[15] Mendes, R., Cortez, P., Rocha, M., Neves, J.: Particle Swarms for Feedforward Neural Network Training, *Proceedings of the International Joint Conference on Neural Networks*, 2002.

[16] Ozcan, E., Mohan, C. K.: Analysis of a Simple Particle Swarm Optimization System, *Intelligent Engineering Systems through Artificial Neural Networks*, 1998.

[17] Ozcan, E., Mohan, C. K.: Particle Swarm Optimization: Surfing the Waves, *Proceedings of the IEEE Congress on Evolutionary Computation*, Washington D.C., USA, 1999.

[18] Parsopoulos, K. E., Plagianakos, V. P., Magoulas, G. D., Vrahitis, M. N.: Stretching Technique for Obtaining Global Minimizers Through Particle Swarm Optimization, *Proceedings of the Particle Swarm Optimization Workshop*, 2001, Indianapolis, USA.

[19] Poli, R.: *The Sampling Distribution of Particle Swarm Optimisers and their Stability*, Technical Report CSM-465, Department of Computer Science, University of Essex, mar 2007.

[20] Potter, M. A., de Jong, K. A.: A Cooperative Coevolutionary Approach to Function Optimization, *The Third Parallel Problem Solving from Nature*, Springer-Verlag, Jerusalem, Israel, 1994.

[21] Shi, Y., Eberhart, R. C.: A Modified Particle Swarm Optimizer, *Proceedings of the IEEE Congress on Evolutionary Computation*, Piscataway, USA, 1998.

[22] Shi, Y., Eberhart, R. C.: Parameter Selection in Particle Swarm Optimization, *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, New York, USA, 1998.

[23] Shi, Y., Eberhart, R. C.: Fuzzy Adaptive Particle Swarm Optimization, *Proceedings of the IEEE Congress on Evolutionary Computation*, IEEE Press, Seoul, Korea, 2001.

[24] Solis, F., Wets, R.: Minimization by Random Search Techniques, *Mathematics of Operations Research*, **6**, 1981, 19–30.

[25] Suganthan, P. N.: Particle Swarm Optimiser with Neighborhood Operator, *Proceedings of the IEEE Congress on Evolutionary Computation*, IEEE Press, Piscataway, USA, 1999.

[26] Trelea, I. C.: The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection, *Information Processing Letters*, **85**(6), 2003, 317–325.

[27] Venter, G., Sobieszczanski-Sobieski, J.: Multidisciplinary Optimization of a Transport Aircraft Wing using Particle Swarm Optimization, *Ninth AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, USA, 2002.

[28] Xie, X., Zhang, W., Yang, Z.: Adaptive Particle Swarm Optimization on Individual Level, *Proceedings of the Sixth International Conference on Signal Processing*, Beijing, China, 2002.

[29] Yasuda, K., Ide, A., Iwasaki, N.: Adaptive Particle Swarm Optimization, *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 2003.

[30] Yoshida, H., Fukuyama, Y., Takayama, S., Nakanishi, Y.: A Particle Swarm Optimization for Reactive Power and Voltage Control in Electric Power Systems Considering Voltage Security Assessment, *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 1999.

A. Convergence of particles onto global best particle

Using a technique similar to that used in Section 6, it will now be shown that all the particles have a non-zero probability of moving closer to $\hat{\mathbf{y}}$ by at least distance ρ, if it is not already within distance ρ from $\hat{\mathbf{y}}$.

Lemma A.1. Every particle $1 \leq i \leq h$ has a non-zero probability of moving closer to the global best particle position, $\hat{\mathbf{y}}$, by at least a distance $\rho > 0$.

Proof: Let B' be a ball of radius ρ, centered at $\hat{\mathbf{y}}$. Pick the point $\mathbf{x}' \in \text{argmax}_{\mathbf{x}}\{\text{dist}(\hat{\mathbf{y}}, \mathbf{x}) | \mathbf{x} \in L_0\}$, as illustrated in Figure 12. Let B be the hypercube centered at $\hat{\mathbf{y}}$, with sides of length $2(\text{dist}(\hat{\mathbf{y}}, \mathbf{x}') - 0.5\rho)$.

Let C be the convex hull of $\mathbf{x}'$ and B' (see Figure 12). Consider a line tangent to B', passing through $\mathbf{x}'$ (i.e., one of the edges of C). This line is the longest such line, for $\mathbf{x}'$ is the point furthest from B'. This implies that the angle subtended by $\mathbf{x}'$ is the smallest such angle of any point in L_0. In turn, this

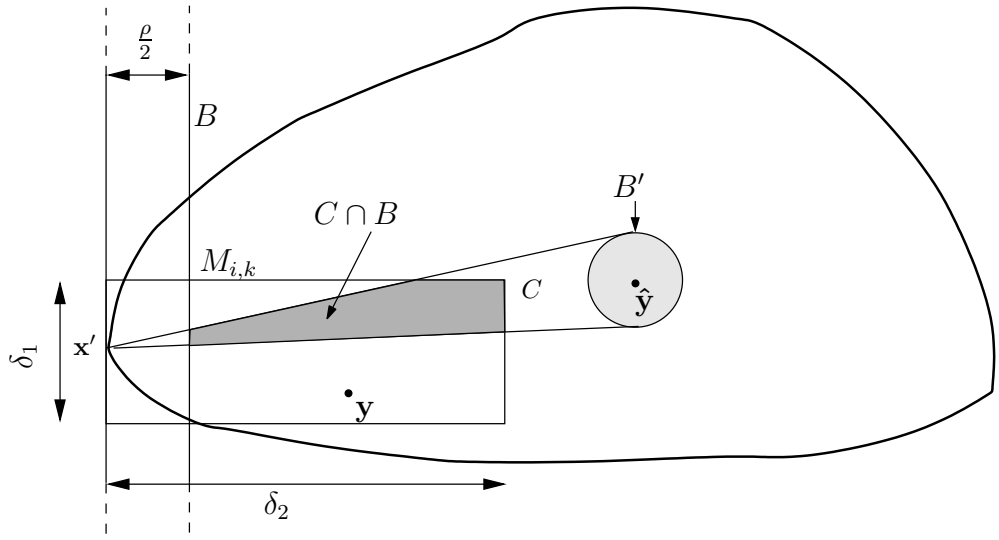


Figure 12. The intersection $C \cap B$, with B centered on $\hat{\mathbf{y}}$.

implies that the volume $C \cap B$ is smaller than that of $C' \cap B$ for any other convex hull C' defined by any arbitrary point $\mathbf{x} \in L_0$.

Let $M_{i,k}$ be the support of the hyper-rectangle from which particle i will sample its next position. This hyper-rectangle is defined by its two corners, $\mathbf{p}_i$ and $\mathbf{q}_i$:

$$p_{i,j} = v_{ij,k} + c_1(y_{ij,k} - x_{ij,k}) + c_2(\hat{y}_{ij,t} - x_{ij,t}), \quad 1 \leq j \leq n \tag{46}$$

$$\mathbf{q}_i = \mathbf{x}_i + \mathbf{v}_i \tag{47}$$

Note that the distribution $\mu_{i,k}$ over $M_{i,k}$ is the convolution of the uniform distributions of its two components, which assume values in the ranges $[0, c_1(y_{ij,k} - x_{ij,k})]$ and $[0, c_2(\hat{y}_{ij,t} - x_{ij,t})]$. Observe that $M_{i,k}$ will typically contain $\hat{\mathbf{y}}$, although this is not required at every step to ensure convergence.

Then we have that for all $\mathbf{x}$ in L_0, and thus all particle positions $\mathbf{x}_{i,t}$,

$$\mu_{i,k}[\text{dist}(\mathbf{g}(\mathbf{x}_{i,t}), \hat{\mathbf{y}}) < \text{dist}(\mathbf{x}_{i,t}, \hat{\mathbf{y}}) - 0.5\rho] \geq \eta = \mu[C \cap B] > 0 \tag{48}$$

If a particle is thus further than ρ from $\hat{\mathbf{y}}$ in any dimension j, then Equation (48) guarantees that the particle will be able to move closer to $\hat{\mathbf{y}}$ by at least distance ρ. ■