The particle swarm optimization algorithm: convergence analysis and parameter selection

Ioan Cristian Trelea

INA P-G, UMR Génie et Microbiologie des Procédés Alimentaires, BP 01, 78850 Thiverval-Grignon, France

Abstract

The particle swarm optimization algorithm is analyzed using standard results from the dynamic system theory. Graphical parameter selection guidelines are derived. The exploration–exploitation tradeoff is discussed and illustrated. Examples of performance on benchmark functions superior to previously published results are given.
© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Particle swarm optimization; Stochastic optimization; Analysis of algorithms; Parallel algorithms

1. Introduction

The particle swarm optimization (PSO) is a parallel evolutionary computation technique developed by Kennedy and Eberhart [4] based on the social behavior metaphor. A standard textbook on PSO, treating both the social and computational paradigms, is [5]. The PSO algorithm is initialized with a population of random candidate solutions, conceptualized as particles. Each particle is assigned a randomized velocity and is iteratively moved through the problem space. It is attracted towards the location of the best fitness achieved so far by the particle itself and by the location of the best fitness achieved so far across the whole population (global version of the algorithm).

The PSO algorithm includes some tuning parameters that greatly influence the algorithm performance,

often stated as the exploration–exploitation tradeoff: Exploration is the ability to test various regions in the problem space in order to locate a good optimum, hopefully the global one. Exploitation is the ability to concentrate the search around a promising candidate solution in order to locate the optimum precisely. Despite recent research efforts, the selection of the algorithm parameters remains empirical to a large extent. A complete theoretical analysis of the algorithm has been given by Clerc and Kennedy [2]. Based on this analysis, the authors derived a reasonable set of tuning parameters, as confirmed by [3]. The reference [2] contains a good deal of mathematical complexity, however, and deriving from it simple user-oriented guidelines for the parameter selection in a specific problem is not straightforward.

The present work gives some additional insight into the PSO parameter selection topic. It is established that some of the parameters add no flexibility to the algorithm and can be discarded without

E-mail address: trelea@grignon.inra.fr (I.C. Trelea).

loss of generality. Results from the dynamic system theory are used for a relatively simple theoretical analysis of the algorithm which results in graphical guidelines for parameter selection. The user can thus take well-informed decisions according to the desired exploration–exploitation tradeoff: either favor exploration by a thorough sampling of the solution space for a robust location of the global optimum at the expense of a large number of objective function evaluations or, on the contrary, favor exploitation resulting in a quick convergence but to a possibly non-optimal solution. Non surprisingly, the best choice appears to depend on the form of the objective function. The newly established parameter selection guidelines are applied to standard benchmark functions. Examples of performance superior to previously published results are reported.

2. Particle swarm optimization algorithm

2.1. Standard algorithm

The basic PSO algorithm can be described in vector notation as follows:

$$v_{k+1} = \vec{a} \otimes \vec{v}_k + \vec{b}_1 \otimes \vec{r}_1 \otimes (\vec{p}_1 - \vec{x}_k) + \vec{b}_2 \otimes \vec{r}_2 \otimes (\vec{p}_2 - \vec{x}_k), \quad (1)$$

$$\vec{x}_{k+1} = \vec{c} \otimes \vec{x}_k + \vec{d} \otimes \vec{v}_{k+1}. \quad (2)$$

The symbol $\otimes$ denotes element-by-element vector multiplication. At iteration k, the velocity $\vec{v}_k$ is updated based on its current value affected by a momentum factor ($\vec{a}$) and on a term which attracts the particle towards previously found best positions: its own previous best position ($\vec{p}_1$) and globally best position in the whole swarm ($\vec{p}_2$). The strength of attraction is given by the coefficients $\vec{b}_1$ and $\vec{b}_2$. The particle position $\vec{x}_k$ is updated using its current value and the newly computed velocity $\vec{v}_{k+1}$, affected by coefficients $\vec{c}$ and $\vec{d}$, respectively. It is shown later that $\vec{c}$ and $\vec{d}$ can be set to unity without loss of generality. Randomness useful for good state space exploration is introduced via the vectors of random numbers $\vec{r}_1$ and $\vec{r}_2$. They are usually selected as uniform random numbers in the range [0, 1]:

$$\vec{r}_1, \vec{r}_2 \in Uniform[0, 1]. \quad (3)$$

2.2. One-dimensional algorithm

It appears from Eqs. (1) and (2) that each dimension is updated independently from the others. The only link between the dimensions of the problem space is introduced via the objective function, i.e., through the locations of the best positions found so far $\vec{p}_1$ and $\vec{p}_2$. Thus, without loss of generality, the algorithm description can be reduced for analysis purposes to the one-dimensional case:

$$v_{k+1} = av_k + b_1 r_1 (p_1 - x_k) + b_2 r_2 (p_2 - x_k), \quad (4)$$

$$x_{k+1} = cx_k + dv_{k+1}. \quad (5)$$

2.3. Deterministic algorithm

For the theoretical analysis of the PSO, the deterministic version will be considered. The exact relationship between the random and the deterministic versions of the algorithm was not yet rigorously established, but a qualitative discussion is given in Section 4 below. The deterministic version is obtained by setting the random numbers to their expected values:

$$r_1 = r_2 = \tfrac{1}{2}. \quad (6)$$

Eq. (4) can be simplified using the notation:

$$b = \frac{b_1 + b_2}{2}, \quad (7)$$

$$p = \frac{b_1}{b_1 + b_2} p_1 + \frac{b_2}{b_1 + b_2} p_2. \quad (8)$$

Using this notation, the deterministic PSO algorithm can be expressed as:

$$v_{k+1} = av_k + b(p - x_k), \quad (9)$$

$$x_{k+1} = cx_k + dv_{k+1}. \quad (10)$$

The newly introduced attraction coefficient b is thus the average of the "own" and "social" attraction coefficients b_1 and b_2. The attraction point p is the weighted average of p_1 and p_2.

The algorithm described by Eqs. (9) and (10) contains four tuning parameters a, b, c and d. It will be now shown that only two of them are truly useful. The other two can be fixed arbitrarily without loss of generality.

2.4. Algorithm with $d = 1$

Using Eqs. (9) and (10) the velocity can be eliminated from the description of the algorithm yielding the following second order recursion formula involving only successive particle positions:

$$x_{k+1} + (bd - a - c)x_k + acx_{x-1} = bdp. \qquad (11)$$

It appears that individual values of coefficients b and d are not important; the only important quantity is the product bd. Without any loss of generality, one can always select, for example,

$$d \equiv 1. \qquad (12)$$

In other words, any sequence of particle positions $\{x_k\}$ generated by the PSO algorithm described by Eqs. (9) and (10) can also be generated with $d = 1$ and a suitably chosen value of b. The sequence of $\{v_k\}$ will be different, however, but this has no impact on the optimization algorithm since the objective function only depends on x, with v being just an auxiliary variable.

2.5. Algorithm with $c = 1$

For optimization purposes it is desired that, in the long run, the population of particles converges to the optimum location found so far:

$$\lim_{k \to \infty} x_k = p. \qquad (13)$$

Taking into account Eqs. (11) and (12), a necessary condition is:

$$(a - 1)(c - 1) = 0. \qquad (14)$$

The choices $a = 1$ or $c = 1$ are equivalent as far as the sequence of particle positions $\{x_k\}$ is concerned because Eq. (11) is symmetric with respect to a and c. The case

$$c \equiv 1 \qquad (15)$$

will be considered in the rest of the paper. The choice $c = d = 1$ has the nice feature that the variable v can be interpreted as a true "velocity", i.e., the difference between two successive particle positions (Eq. (10)).

3. Dynamic analysis

The theory of linear, discrete-time dynamic systems provides a powerful set of tools and results for assessing the dynamic behavior of a particle. Some of the following results have been proved specifically for the PSO algorithm in [2].

3.1. Matrix form

Eqs. (9), (10), (12) and (15) can be combined and written in compact matrix form as follows:

$$y_{k+1} = Ay_k + Bp \quad \text{with}$$

$$y_k = \begin{bmatrix} x_k \\ v_k \end{bmatrix}, \quad A = \begin{bmatrix} 1-b & a \\ -b & a \end{bmatrix}, \quad B = \begin{bmatrix} b \\ b \end{bmatrix}. \quad (16)$$

In the context of the dynamic system theory, y_k is the particle *state* made up of its current position and velocity, A is the *dynamic matrix* whose properties determine the time behavior of the particle (asymptotic or cyclic behavior, convergence, etc.), p is the *external input* used to drive the particle towards a specified position and B is the *input matrix* that gives the effect of the external input on the particle state.

3.2. Equilibrium point

An equilibrium point is a state maintained by the dynamic system in absence of external excitation (i.e., $p = $ constant). If one exists, any equilibrium point must satisfy the obvious condition $y_{k+1}^{eq} = y_k^{eq}$ for any k. For the particle in a deterministic swarm a straightforward calculation using Eq. (16) gives:

$$y^{eq} = [p \quad 0]^{\mathrm{T}}, \quad \text{that is}$$
$$x^{eq} = p \quad \text{and} \quad v^{eq} = 0 \qquad (17)$$

which is intuitively reasonable: At equilibrium (provided other particles do not find better positions, i.e., p_2 and hence p do not change) the considered particle must have zero velocity and must be positioned at the attraction point p given by Eq. (8).

3.3. Convergence

In general, the initial particle state is not at equilibrium. It is of highest practical importance to determine
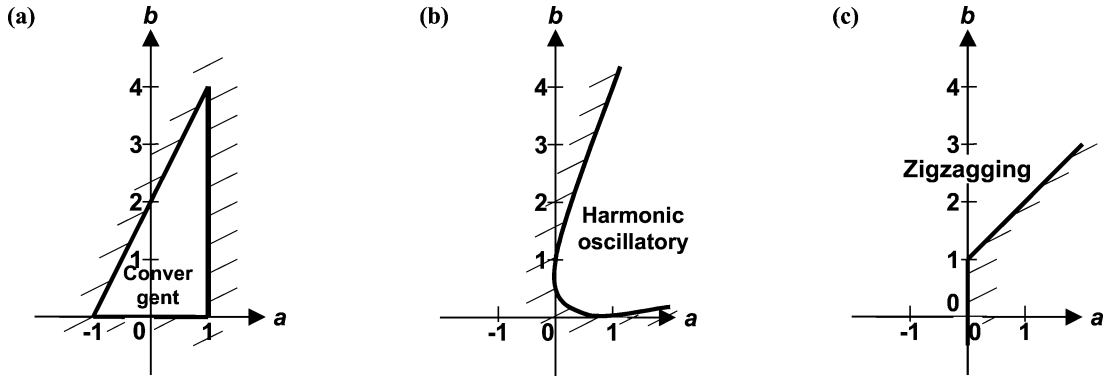
Fig. 1. Domains of dynamic behavior in the (a, b) parameter space. (a) Convergence domain. (b) Domain of harmonic oscillatory behavior. (c) Domain of zigzagging behavior.

whether the particle will eventually settle at the equilibrium (that is if the optimization algorithm will converge) and how the particle will move in the state space (that is how the particle will sample the state space in search of better points). Standard results from dynamic system theory say that the time behavior of the particle depends on the eigenvalues of the dynamic matrix A. The eigenvalues λ_1 and λ_2 (either real or complex) are the solutions of the equation:

$$\lambda^2 - (a - b + 1)\lambda + a = 0. \tag{18}$$

The necessary and sufficient condition for the equilibrium point given by Eq. (17) to be stable (an attractor) is that both eigenvalues of the matrix A (whether real or complex) have magnitude less than 1. In this case the particle will eventually settle at the equilibrium and the PSO algorithm will converge. The analysis of the roots of Eq. (18) leads to the following set of conditions:

$$a < 1, \quad b > 0, \qquad 2a - b + 2 > 0. \tag{19}$$

The convergence domain in the (a, b) plane is the triangle shown in Fig. 1(a). For any initial position and velocity, the particle will converge to its equilibrium position given by Eqs. (8) and (17) if and only if the algorithm parameters are selected inside this triangle.

3.4. Harmonic oscillations

Before convergence, the particle exhibits harmonic oscillations around the equilibrium point when the eigenvalues of the matrix A, which are also the roots of Eq. (18), are complex. This is equivalent to:

$$a^2 + b^2 - 2ab - 2a - 2b + 1 < 0. \tag{20}$$

The corresponding domain in the (a, b) plane is shown in Fig. 1(b).

3.5. Zigzagging

The particle may also exhibit zigzagging behavior around the equilibrium point when at least one of the eigenvalues of the matrix A, whether real or complex, has negative real part. This is equivalent to:

$$a < 0 \quad \text{or} \quad a - b + 1 < 0. \tag{21}$$

The corresponding domain in the (a, b) plane is shown in Fig. 1(c). Zigzagging may be combined with harmonic oscillations. Several examples of particle dynamics, corresponding to various choices of the algorithm parameters, are given in the next section.

4. Parameter selection guidelines

4.1. Examples of dynamic behavior

Several typical choices of the algorithm parameters a and b are reported in Fig. 2 (left). Simulations of particle behavior for these parameter couples are given in Fig. 3. All simulations were performed with:

$$x_0 = 2, \quad v_0 = -0.1, \quad p = 0, \quad m = 50 \text{ iterations.} \tag{22}$$

Slowly convergent harmonic oscillations are shown in Fig. 3(a) (complex eigenvalues with positive real part).
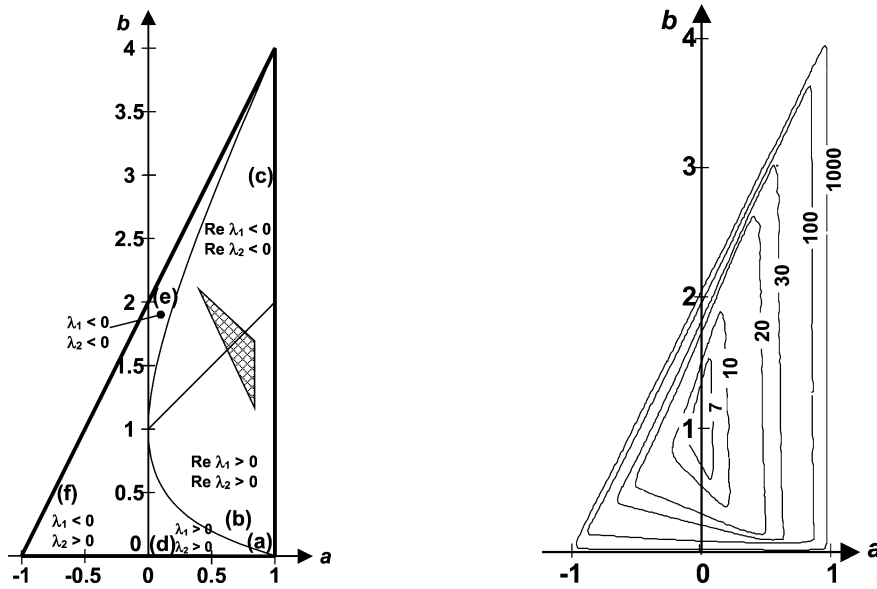
Fig. 2. Left: Position of several typical parameter couples in the parameter space. Points marked (a) to (f) correspond to simulations in Fig. 3. The hatched triangle corresponds to parameter couples that provided good results for the test functions listed in the "Optimization experiments" section. Right: Contours of equal convergence rate. The contours are labeled with the number of iterations necessary to reduce the distance to the equilibrium point by a factor of 1000.

The particle samples its state space relatively well. The exploration of state space and the exploitation of the current optimum are balanced. In contrast, the oscillations shown in Fig. 3(b) decay quickly. The exploitation is favored compared to exploration. As a general rule, parameter couples close to the center of the stability triangle induce quick convergence, while parameter couples close to its borders require many iterations to converge, as illustrated in Fig. 2 (right). The terms "slow" and "quick" convergence should be related to the allowed number of iterations (m). If, for example, $m = 1000$ iterations were allowed instead of 50, then the parameters used in Fig. 3(a) should be interpreted as inducing a "quick" convergence, since most of the particle positions (≈ 900) would be quite close to the equilibrium. In real-life problems the number of allowed iterations is a function of the admissible computation time and of the complexity of the cost function. Harmonic oscillations can be combined with zigzagging as in Fig. 3(c) (complex eigenvalues with negative real part).

An example of non-oscillatory convergence is given in Fig. 3(d) (real positive eigenvalues). For optimization purposes this behavior is not recommended in general, since the state space is only sampled on the one side of the current optimum. In special cases, however, this might be a useful option. For example, negative values of x might not make sense in a given optimization problem while the optimum is suspected to lie at or near zero.

Symmetric zigzagging convergence is shown in Fig. 3(e) (real negative eigenvalues). The parameters a and b can be tuned to make the convergence either slow or fast as in the case of harmonic oscillations. Asymmetric zigzagging is illustrated in Fig. 3(f) (real eigenvalues with opposite signs).

4.2. Effect of the random numbers

The rigorous analysis of the optimization algorithm with random numbers described by Eqs. (4) and (5) is beyond the scope of this paper. Qualitatively, the considerations presented in the previous paragraphs remain valid, however, as shown by extensive simulation studies. The presence of random numbers enhances the zigzagging tendency and slows down convergence, thus improving the state space exploration and preventing premature convergence to non-optimal
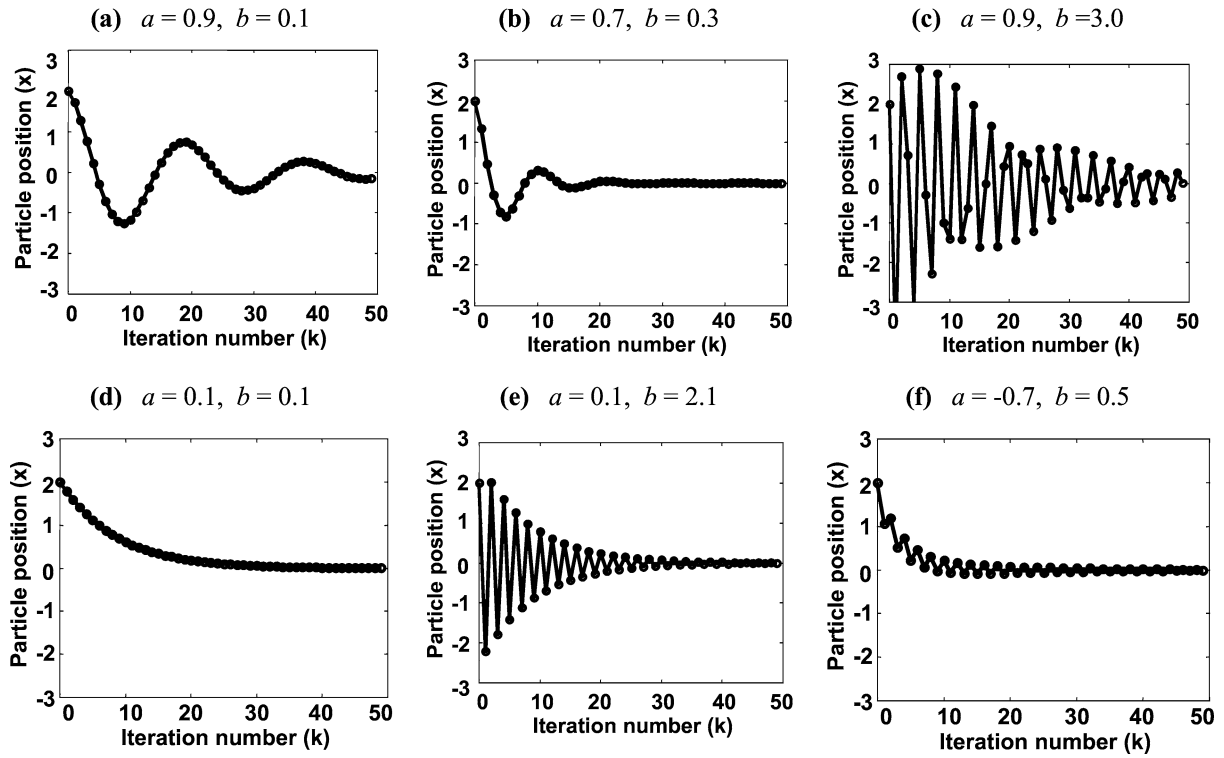
(a) $a = 0.9$, $b = 0.1$

(b) $a = 0.7$, $b = 0.3$

(c) $a = 0.9$, $b = 3.0$

(d) $a = 0.1$, $b = 0.1$

(e) $a = 0.1$, $b = 2.1$

(f) $a = -0.7$, $b = 0.5$

Fig. 3. Examples of dynamic behavior of a single particle for various choices of the parameters a and b. (a) Harmonic oscillations with slow convergence. (b) Harmonic oscillations with quick convergence. (c) Harmonic oscillations with zigzagging. (d) Non-oscillatory convergence. (e) Symmetric zigzagging. (f) Asymmetric zigzagging.

points. This is especially true when the particle's own attraction point p_1 is situated far from the population attraction point p_2. The equivalent attraction point p, is, in the case of the random algorithm, given by:

$$p = \frac{b_1 r_1}{b_1 r_1 + b_2 r_2} p_1 + \frac{b_2 r_2}{b_1 r_1 + b_2 r_2} p_2. \qquad (23)$$

If $p_1 \neq p_2$, it changes from iteration to iteration even if no better solutions are discovered, i.e., p_1 and p_2 remain constant. In the long run, however, it is expected that $p_1 = p_2$ as all the particles in the population "agree" upon a single best point which becomes the unique attractor. In this case, Eq. (23) says that $p = p_1 = p_2$ irrespective of the generated random numbers.

4.3. Parameter tuning heuristic

In addition to parameters a and b and to the effect of the random numbers discussed above, the conver-

gence of the algorithm is influenced by the number particles in the swarm N (larger swarms need more iterations to converge) and by the topology: strongly connected swarms (e.g., with large neighborhoods) converge faster than loosely connected ones. The best rate of convergence, i.e., the best tradeoff between exploration and exploitation, strongly depends of the function being optimized: number of local optima and distance to the global one, position of the global optimum in the search domain (near center, near border), size of the search domain, required accuracy in the location of the optimum, etc. It is probably impossible to find a unique set of algorithm parameters that work well in all cases but the following empirical procedure, based on the above considerations, was found to work in practice.

Start with a relatively quickly convergent parameter set, like those mentioned in the "Optimization experiments" section and run the algorithm several times until convergence. If different results are obtained in

most runs, the convergence rate is too high: the swarm converges prematurely to non-optimal points. A more slowly convergent parameter set should be selected (Fig. 2), a larger number of particles in the swarm and/or a less connected topology should be used. If the same result is obtained consistently, but during a large fraction of the algorithm iterations no better points are found, the convergence rate is too low: particles in the swarm do not focus the search around promising candidate solutions quickly enough. A more quickly convergent parameter set should be selected (Fig. 2), a smaller number of particles in the swarm and/or a more strongly connected topology could be used. If consistent results are obtained in most runs and continuous improvement over iterations is observed, then the convergence rate is adequate and the same parameter set, swarm size and topology should be used to solve similar problems. Monitoring the dispersion of the particles in the swarm over iterations also helps assessing the adequacy of the convergence rate.
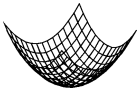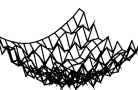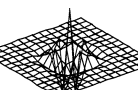
The ideas upon which this procedure is based are illustrated numerically in the following section.

5. Optimization experiments

5.1. Test conditions

The particle swarm optimization algorithm (Eqs. (1) and (2)) was used for the optimization of five benchmark functions, also used in [2] and [3]. Vector parameter $\vec{a}$ had all elements equal to a. Similarly, $\vec{b} = \vec{b}_1 = \vec{b}_2$ had all elements equal to b. Two sets of parameters a and b were used. Parameter set 1 ($a = 0.6$ and $b = 1.7$) was selected by the author in the algorithm convergence domain (Fig. 2) after a large number of simulation experiments. Parameter set 2 ($a = 0.729$ and $b = 1.494$) was recommended by Clerc [1] and also tested in [3] giving the best results published so far known to the author. All elements of $\vec{c}$ and $\vec{d}$ were set to 1 as explained above. The functions, the number of dimensions (n), the admissible range of the variable (x), the optimum and the goal values are summarized in Table 1. The number of iterations required to reach the goal was recorded. The maximum iteration number was fixed to 10000. Each optimization

Table 1
Optimization test functions

Name	Formula	Dim. n	Range $[x_{\min}, x_{\max}]$	Optimal f	Goal for f	Sketch in 2D
Sphere	$f_0(\vec{x}) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]^n$	0	0.01	
Rosenbrock	$f_1(\vec{x}) = \sum_{i=1}^{n-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$	30	$[-30, 30]^n$	0	100	
Rastrigin	$f_2(\vec{x}) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	30	$[-5.12, 5.12]^n$	0	100	
Griewank	$f_3(\vec{x}) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]^n$	0	0.1	
Schaffer's f6	$f_6(\vec{x}) = 0.5 - \frac{(\sin\sqrt{x_1^2 + x_2^2})^2 - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2}$	2	$[-100, 100]^2$	0	10^{-5}	

experiment was run 20 times with random initial values of x and v in the range $[x_{min}, x_{max}]$ indicated in Table 1. A fully connected topology (all particles being neighbors) was used in all cases. Population sizes of $N = 15$, 30 and 60 particles were tested. During the optimization process the particles were allowed to "fly" outside the region defined by $[x_{min}, x_{max}]$ and the velocity was not restricted.

5.2. Optimization results and discussion

5.2.1. Effect of the number of particles in the swarm (N)

In most cases, increasing the number of particles decreased the number of required algorithm iterations, as indicated by the average, median, minimum and maximum values reported in Table 2. The success rate

of the algorithm (fraction of the number of runs it reached the goal) was also increased significantly. This was expected, since more particles sample the state space more thoroughly. More particles require more function evaluations, however. Since in real-life applications the optimization cost is usually dominated by the evaluations of the objective function, the *expected number of function evaluations* was retained as the main algorithm performance criterion. It takes into account the number of particles, the number of algorithm iterations and the success rate (Table 2). Best results were usually obtained with a medium number of particles (except for the Rosenbrock function). Using too few particles gave a very low success rate and required more iterations. Using too many particles required too many function evaluations per iteration.

Table 2
PSO algorithm performance

Function	Number of particles (N)	Number of algorithm iterations to achieve the goal								Success rate[a]		Expected number of function evaluations[b]	
		Average		Median		Minimum		Maximum					
		Set 1	Set 2	Set 1	Set 2	Set 1	Set 2	Set 1	Set 2	Set 1	Set 2	Set 1	Set 2
Sphere	15	769	764	722	731	523	586	1377	1275	0.40	1	28 838	11 460
	30	344	395	333	395	266	330	457	572	1	1	**10 320**	11 850
	30[c]	–	530	–	525	–	495	–	573	–	1	–	15 900
	60	252	314	252	313	214	269	309	368	1	1	15 120	18 840
Rosenbrock	15	531	1430	523	729	413	452	695	9476	0.50	1	**15 930**	21 450
	30	614	900	383	408	239	298	3718	4642	1	1	18 420	27 000
	30[c]	–	669	–	621	–	402	–	1394	–	1	–	20 070
	60	337	611	284	311	189	219	916	4450	1	1	20 220	36 660
Rastrigin	15	172	299	147	292	102	152	308	523	0.35	0.80	7371	5606
	30	140	182	128	174	104	123	208	299	0.90	0.95	**4667**	5747
	30[c]	–	213	–	200	–	161	–	336	–	1	–	6390
	60	122	166	116	164	84	119	168	214	0.95	1	7705	9960
Griewank	15	689	755	580	608	443	470	1589	1755	0.35	0.60	29 529	18 875
	30	313	365	304	361	257	319	401	455	0.90	0.90	10 433	12 167
	30[c]	–	313	–	308	–	282	–	366	–	1	–	**9390**
	60	226	287	224	280	202	266	250	328	0.95	1	14 274	17 220
Schaffer	15	583	1203	138	126	63	91	3706	5853	0.45	0.40	19 433	45 112
	30	161	350	120	157	74	102	595	1264	0.75	0.60	**6440**	17 500
	30[c]	–	532	–	321	–	94	–	2046	–	1	–	15 960
	60	169	319	91	119	40	83	854	2361	0.90	0.95	11 267	20 147

Parameter set 1: $a = 0.6$ and $b = 1.7$. Parameter set 2: $a = 0.729$, $b = 1.494$.

[a] Fraction of the number of optimization experiments in which the goal was reached.
[b] (Number of particles in the swarm) × (Average number of iterations)/(Success rate).
[c] Best results reported by Eberhart and Shi [3]. Obtained with $N = 30$, $a = 0.729$, $b = 1.494$ and velocity limited to $[x_{min}, x_{max}]$.

5.2.2. Effect of the parameters a and b

Parameter set 1 has a higher convergence rate than set 2 (Fig. 2, right). For set 1, the exploitation is thus favored compared to the exploration of the state space. The number of algorithm iterations for set 1 was generally smaller (Table 2) but the risk of premature convergence to non-optimal points was higher, as indicated in Table 2 by the lower success rate. The former did or did not outweigh the latter depending on the function and on the number or particles. Generally, the smallest expected number of function evaluations (among all values of N) was achieved for parameter set 1, except for the Griewank function. The convergence rate of a parameter set is not the only important factor, though. Moving along the curves of equal convergence rate (Fig. 2, right) does not at all give equally good results. The convergence trajectory (Fig. 3) is also important. For $N = 30$, the author obtained good results for parameter sets in the small hatched triangle in Fig. 2, left. The exact meaning of this triangle is not clear yet.

5.2.3. Effect of the objective function

In all cases, the global minimum (the only that achieved the goal) was situated at or near the center of the search domain. The Sphere and the Rosenbrock functions have a single minimum, while the other functions have multiple local minima (Table 1). Rastrigin and Griewank functions have a "large scale" curvature which guides the search towards the global minimum, while the Schaffer function is essentially flat except near the global minimum. Therefore, it is not surprising that the largest success rates were achieved for the Sphere and Rosenbrock functions and the smallest one for the Schaffer function. Most of the times, failure to achieve the goal was due to a premature convergence to a local minimum. For the combination of a small number of particles ($N = 15$) and a quickly convergent parameter set (set 1) convergence was also ob-served to completely non-optimal points, as illustrated by a low success rate for the Sphere and Rosenbrock functions which have no local minima.

6. Summary

The dynamic behavior and the convergence of the simplified (deterministic) PSO algorithm was analyzed using tools from the discrete-time dynamic system theory. The analysis provided qualitative guidelines for the general (random) algorithm parameter selection. Simulation experiments with two parameter sets, three numbers of particles in the swarm and five benchmark functions were carried out. The speed of convergence—robustness tradeoff was discussed. For four benchmark functions, better than previously published results were obtained in terms if the expected number of objective function evaluations.

Further research is needed, however, to clarify the effect of the randomness, of the swarm topology and of their interaction with the function being minimized. Better parameter sets probably await discovery in the outlined algorithm convergence domain.

References

[1] M. Clerc, The swarm and the queen: towards a deterministic and adaptive particle swarm optimization, in: Proc. ICEC, Washington, DC, 1999, pp. 1951–1957.

[2] M. Clerc, J. Kennedy, The particle swarm: explosion stability and convergence in a multi-dimensional complex space, IEEE Trans. Evolution. Comput. 6 (1) (2002) 58–73.

[3] R.C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Proc. CEC, San Diego, CA, 2000, pp. 84–88.

[4] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proc. IEEE Conf. on Neural Networks, IV, Piscataway, NJ, 1995, pp. 1942–1948.

[5] J. Kennedy, R.C. Eberhart, Y. Shi, Swarm Intelligence, Morgan Kaufmann Publishers, San Francisco, CA, 2001.