A Particle Swarm Optimization Method for Multimodal Optimization Based on Electrostatic Interaction

Julio Barrera and Carlos A. Coello Coello*

CINVESTAV-IPN
Departamento de Computación
Evolutionary Computation Group
Av. IPN No. 2508, Col. San Pedro Zacatenco
México, D.F. 07360, MEXICO
julio.barrera@gmail.com, ccoello@cs.cinvestav.mx

Abstract. The problem of finding more than one optimum of a fitness function has been addressed in evolutionary computation using a wide variety of algorithms, including particle swarm optimization (PSO). Several variants of the PSO algorithm have been developed to deal with this sort of problem with different degrees of success, but a common drawback of such approaches is that they normally add new parameters that need to be properly tuned, and whose values usually rely on previous knowledge of the fitness function being analyzed. In this paper, we present a PSO algorithm based on electrostatic interaction, which does not need any additional parameters besides those of the original PSO. We show that our proposed approach is able to converge to all the optima of several test functions commonly adopted in the specialized literature, consuming less evaluations of the fitness function than other previously reported PSO methods.

1 Introduction

There exist certain applications in which the objective function to be optimized has more than one optimum. Such problems are called *multimodal* and may have several local optima and only one global optimum that we want to find, or may have several optima, all of which we aim to locate. This latter problem is probably the most challenging for traditional mathematical programming techniques, which normally are unable to work properly in such problems. When using metaheuristics, multimodal functions are also challenging, because stochastic noise tends to make population-based metaheuristics (e.g., genetic algorithms) to converge to a single solution if run for a sufficiently large number of generations [1]. The currently available metaheuristics designed to deal with multimodal problems, normally require several additional parameters that must be set by the user. More recently, some authors have proposed adaptive procedures for setting

* The second author is also affiliated to the UMI-LAFMIA 3175 CNRS.

up the parameters of some metaheuristics designed for multimodal optimization. However, these adaptive procedures also tend to require additional parameters, and normally do not perform better than other (less elaborate) available methods.

In this paper, we present a method for locating more that one optimum of a fitness function in a given search region. The proposed method does not introduce new parameters into our baseline metaheuristic (particle swarm optimization) other than those originally required by such technique. The method is based on the electrostatic interaction between charged point particles. In our model, we consider the fitness value of a particle to be its charge and then, we compute the magnitude of the force that appears between two charged particles (considering that both have a positive charge). For updating a particle, the particle with the maximum force value in the swarm, replaces the particle with the best global fitness value recorded so far; that is, the particle will follow the particle with which has the maximum electrostatic attraction.

The remainder of the paper is organized as follows. Section 2 reviews some basic notions of the PSO algorithm. In Section 3, we describe some of the available methods for finding more than one optimum using PSO. Section 4 presents our proposed method based on electrostatic interaction. Section 5 shows the test functions used, the parameters setup for the experiments and the results obtained. Finally, in Section 6, we provide our conclusions and some possible paths for future work.

2 Particle Swarm Optimization

The PSO algorithm was originally proposed by Kennedy and Eberhart in the mid-1990s [2] and has been successfully applied to a wide variety of problems. Originally, it was only used for dealing with unimodal problems, but more recently was extended to solve both multimodal and multiobjective optimization problems [3]. One of the reasons for the success of the PSO algorithm is its simplicity and its ease of use. It starts with a population of particles randomly positioned in the search space (such population is called *swarm*). Each particle has a position, a velocity and a fitness function value (evaluated at its current position). The best position obtained by each particle so far (i.e., the position corresponding to the best fitness value obtained so far for that particle), is also stored. At each iteration, the position and velocity of a particle are updated following two simple rules shown in equations (1) and (2).

$$v_{t+1} = v_t + R_1 \cdot C_1 \cdot (g - x_t) + R_2 \cdot C_2 \cdot (p - x_t) \tag{1}$$

$$x_{t+1} = x_t + v_{t+1} \tag{2}$$

where R_1 and R_2 are randomly generated numbers in the range $[0, 1]$ using a uniform distribution, C_1 and C_2 are the "learning" constants, p is the position where the particle reached its best fitness value through the iterations, g is the

position with the best fitness value of all p positions in the swarm, x_t and v_t are the position and velocity of the particle at iteration t, respectively.

A number of modifications have been proposed to improve the convergence of the PSO algorithm. The most commonly used are the modifications to the velocity update equation (1) proposed by Clerc [4] (called the Constriction Factor model), and by Shi [5] (called the Inertia Weight model). Under the Constriction Factor model, the velocity is updated using equation (3).

$$v_{t+1} = \chi \left[v_t + R_1 \cdot C_1 \cdot (g - x_t) + R_2 \cdot C_2 \cdot (p - x_t) \right] \tag{3}$$

with

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

where $\phi = C_1 + C_2$, $\phi > 4$, and κ is an arbitrary value in the range $(0, 1]$. Analogously, the Inertia Weight model uses equation (5) for updating the velocity.

$$v_{t+1} = \omega v_t + R_1 \cdot C_1 \cdot (g - x_t) + R_2 \cdot C_2 \cdot (p - x_t) \tag{5}$$

where ω is an arbitrary value. In contrast with the Constriction factor model, in this case the value of ω does not depend on the value of C_1 nor C_2. The modification of the update equation is complemented confining the particles to the search space using a X_{max} and a X_{min} values for each coordinate. It is also common to limit the velocity of the particles using a value V_{max} for each coordinate, usually $V_{max} = X_{max}$. The Constriction Factor and Inertia Weight model are the two PSO variants most commonly adopted in the literature.

3 Finding more than One Optimum

Like any other metaheuristic, the PSO algorithm requires certain modifications in order to make it capable of dealing with multimodal problems. In this section, we will review the most relevant previous work in that direction that has appeared in the specialized literature.

Some of the existing proposals borrow ideas from the genetic algorithms literature, such as the species-based PSO introduced by Li [6], and later modified by Iwamatsu [7, 8]. This method is a PSO adaptation of the Species Conserving Genetic Algorithm (SCGA), originally proposed by Jiang-Ping Li [9]. It requires the setup of a parameter value called "radius" in order to determine when a particle belongs or not to a certain species. The proper choice of this value is essential for obtaining good results and, in most cases, the optimal value for this parameter depends on the objective function being optimized.

The use of niching (which has been a popular diversity maintenance mechanism in the genetic algorithms literature [10–12]) has also been proposed for PSO. Brits et al. [13] proposed a niching PSO algorithm in which the radius of a niche is computed using the mean of the distance among the particles in a sub-swarm. This approach requires the initialization of several parameters. To create a sub-swarm, the variance of the fitness of a particle through the iterations of the algorithm must be measured. To compute the variance, the fitness of the particles is observed during e iterations, and if such fitness variation is less than a δ threshold, a sub-swarm is created. The approach also requires a user-defined threshold μ for preventing the collision of two sub-swarms.

To avoid setting multiple parameters, an adaptive method was introduced by Bird et al. [14]. In this case, the radius is also computed as the mean of the distances among the particles of a swarm, but they use a graph that stores the information of the particles that are at a distance which is smaller than the radius. If two particles are close (according to the computed radius) for a number e of iterations, then a sub-swarm is formed. In this case, there is also a limit m of particles that are allowed per sub-swarm.

Passaro and Sarita [15] present a follow-up of one of Kennedy's papers [16] and use clustering to divide the main swarm into sub-swarms. In order to avoid setting up the number k of clusters, they adopt the x-means algorithm from Pelleg et al. [17] and choose an optimal value for k. A maximum and minimum value for k must be set because the x-means algorithm computes a statistical value for each k in a given range in order to determine the optimum value.

An attempt to develop a PSO method capable of locating more that one optimum, and which does not require any additional parameters was presented by Li [18]. The basis of this work is the proposal of Peram et al. [19] in which a new component is added to the equation for updating the velocity. This component consists of the difference between the position x_t of the particle at iteration t and a computed position p_n for each particle. Each coordinate of the p_n vector is computed separately, and might result in unstable convergence. Li proposed that, instead of computing each coordinate separately at the position p_n, they are selected from the p vectors of the particles in the swarm. Such selection is based on the computation of the ratio of the difference between the fitness value of two particles being compared and their distance. Thus, the p_n position for a particle i is the p vector of the particle j that maximizes the computed ratio. Also, the p_n position, replaces the g position in the update equation of the velocity rather than being added in an additional component. With these modifications, the method performs better, but still shows unstable convergence in some cases.

As we have shown in this section, there are several PSO methods that have been designed to find more than one optimum. Their nature is also varied, but one important problem that is common in almost all of them is that they require additional parameters that have an impact on performance, and whose definition is normally not trivial. In the next section, we describe a PSO method that does not need any additional parameters, besides those of the original PSO algorithm.

As we will see later on, the proposed method is also able to find better results than those reported by other PSO methods previously proposed for multimodal optimization.

4 The Electrostatic Interaction Method

As indicated in the previous section, one possible method for finding more than one optimum of a function consists of the selection of a particle's g position from the p positions of the other particles in the swarm. In the case of the FER-PSO of Li [18], the selection method consists of computing and maximizing a ratio for all the particles in the swarm with respect to the particle being updated.

In our proposed method, we follow a similar approach. When the velocity of a particle is being updated, we select the particle's g position by computing and maximizing a value for each particle in the swarm. However, in this case, we take inspiration on electrostatics: according to Coulomb's law, the force between two charged particles can be computed using equation (6).

$$F = \frac{1}{4\pi\varepsilon_0}\frac{Q_1 Q_2}{R^2} \tag{6}$$

where F is the force between the two particles, Q_1 and Q_2 are the charges of the particles, ε_0 is the electric constant (vacuum permitivity), and R is the distance between the two particles. For our proposed method we compute a "force" F_{ij} between a particle being updated i and the rest of the particles in the swarm by replacing the charge Q of a particle by the best fitness value recorded by the particle $f(p)$ so far. Also, we replace the constant $1/(4\pi\varepsilon_0)$ with a scaling constant α computed in the same way as in [18]. Thus, to compute the F_{ij} between two particles i and j, we use equation (7).

$$F_{ij} = \alpha\frac{f(p_i)f(p_j)}{||p_i - p_j||^2} \tag{7}$$

In order to prevent numerical errors, if the distance between the particles i and j is zero, the F_{ij} value is not computed and the particle j is not considered for selection. The best position recorded for the particle j with the maximum value of F_{ij} is used to replace the best position of the particle with the current best global fitness (i.e., the g position). The equations for updating the velocity and position of a particle remain unchanged. For each particle i in a swarm, the values of F_{ij} are computed for the rest of the swarm, that is, this operation has a complexity $O(N^2)$ for each generation, with N being the number of particles in the swarm. In this work, we use a different PSO algorithm that does not modify the update equations like in the Constriction Factor or the Inertia Weight models. Instead, the maximum value of the velocity is decreased in a nonlinear way. For iteration t, the value of the maximum velocity V_{max} is computed according to equation (8).

$$V_{max} = |X_{max} - X_{min}| \cdot \omega^t \tag{8}$$

with ω an arbitrary number in the range $(0, 1)$. The outline of the selection method for the g position is presented in Algorithm 1. In Algorithm 2 we show the procedure to compute the velocity of the particles, and in Algorithm 3, we outline our proposed PSO algorithm for finding more than one optimum: the Electrostatic Particle Swarm Optimization (EPSO).

Algorithm 1: computeIndexMaximum(index) Algorithm for selecting the position p_n with the maximum electrostatic force F for a particle

```
input  : Index of a particle
output: Index j of the recorded position p with the maximum value of F for
        the particle at the input index
indexMaximum = 0;
FMaximum = 0;
for j ← 0 to N-1 do
    if distance(i,j) > 0 then
        F = alpha * f(p_i) * f(p_j)/distance(i, j) **2;
        if i=0 then
            FMaximum = F;
        end
        if F > FMaximum then
            FMaximum = F;
            indexMaximum = j;
        end
    end
end
return indexMaximum;
```

Algorithm 2: computeVelocities(void) Algorithm to compute the velocity of the particles in the swarm

```
indexMaximum = 0;
for i ← 0 to N-1 do
    indexMaximum = computeIndexMaximum(i);
    velocity = R_1 * C_1 * (position(i) - bestPosition(i)) + R_2 * C_2 * (position(i)
    - bestPosition(indexMaximum);
    setVelocity(i, velocity);
end
```

Algorithm 3: EPSO() Algorithm to find more than one optimum

```
generateSwarm();
evaluateSwarm();
for i ← 0 to iterations do
    computeVelocities();
    computeVelocityLimits(i);
    applyVelocityLimits();
    updatePositions();
    applyPositionLimits();
    evaluateSwarm();
    updateBestPositions();
    countOptimaFound();
end
```

5 Experiments and Results

For testing and comparison purposes, we use the same set of functions that appeared in the works of Passaro et al. [15] and Bird et al. [14]. We also compare our results with those reported by these authors.

5.1 Test Functions

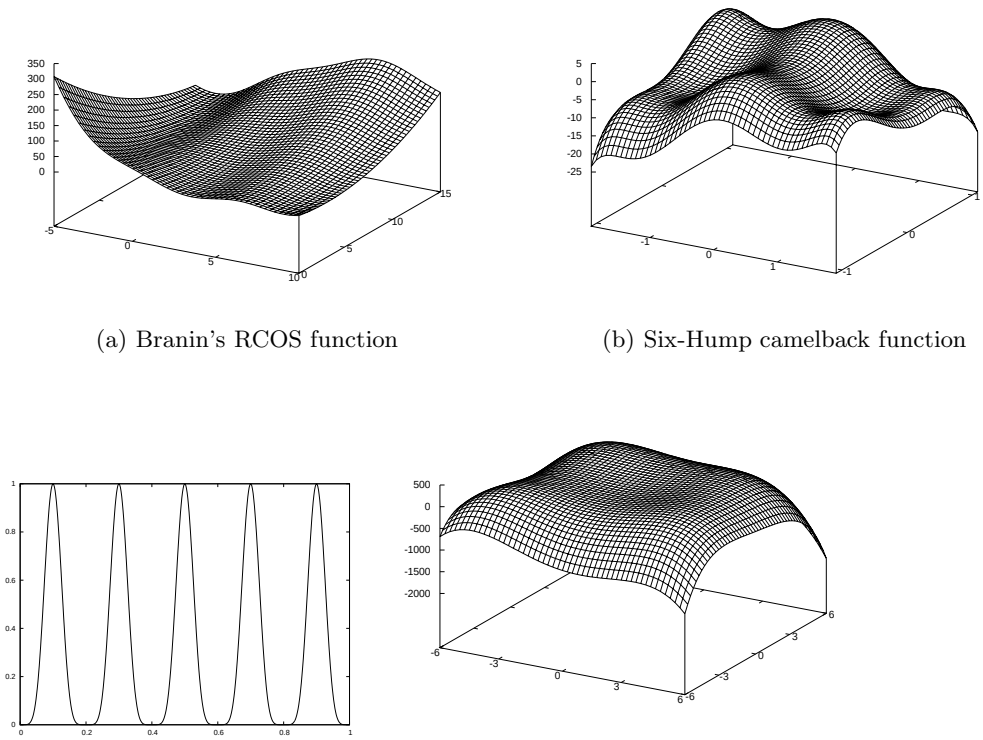
The set of test functions that we adopted to validate our proposed approach is sumarized in Table 1. For each of them, we show their corresponding equations and the allowable ranges for their decision variables. Next, we describe in more detail each of these test functions.

Table 1. Test functions adopted for our experiments.

Function	Equation	Search range
F1	$f(x,y) = \left(y - \frac{5.1x^2}{4\pi} + \frac{5x}{\pi} - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x) + 10$	$-5 \leq x \leq 10$ $0 \leq y \leq 15$
F2	$f(x,y) = -4\left[\left(4 - 2.1x^2 + \frac{x^4}{3}\right)x^2 + xy + (-4 + 4y^2)y^2\right]$	$-1.9 \leq x \leq 1.9$ $-1.1 \leq y \leq 1.1$
F3	$f(x) = \sin^6(5\pi x)$	$0 \leq x \leq 1$
F4	$f(x,y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2$	$-6 \leq x, y \leq 6$
F5	$f(x,y) = \sum_{i=1}^{5} i\cos[(i+1)x + i]\sum_{i=1}^{5}[(i+1)y + i]$	$-10 \leq x, y \leq 10$

F1 is Branin's RCOS function and has 3 global minima. Its plot is shown in Figure 1(a). F2 is the Six-hump camel back function, which has 2 global and 4 local maxima. Its plot is shown in Figure 1(b). The F3 function is also known as Deb's 1st function. This is the only one-dimensional test function that we

adopted. It has 5 global maxima uniformly distributed (see Figure 1(c)). The F4 function is Himmelblau's function with 4 global optima shown in Figure 1(d). The last test function is the two-dimensional Shubert's function. This function is well known for its complexity in the given search range. It has 760 optima, including 18 global minima. The plot of Shubert's function is shown in Figure 2.



(a) Branin's RCOS function

(b) Six-Hump camelback function

(c) Deb's function

(d) Himmelblau's function

Fig. 1. Test functions used for assessing performance of our proposed electrostatic inspired PSO.

In order to allow a fair comparison, we perform experiments similar to those reported by Passaro [15] and Bird [14]. We repeat each experiment 50 times in order to gather statistics and we adopt a maximum of 500 iterations per run. Also, for each test function, we adopt two different sizes for the swarm, 30 and 60 particles. For Shubert's function, we adopt 300 and 500 particles per swarm. In all our experiments, we use a threshold value $\epsilon = 0.00001$ in order to determine if a particle has reached an optimum.

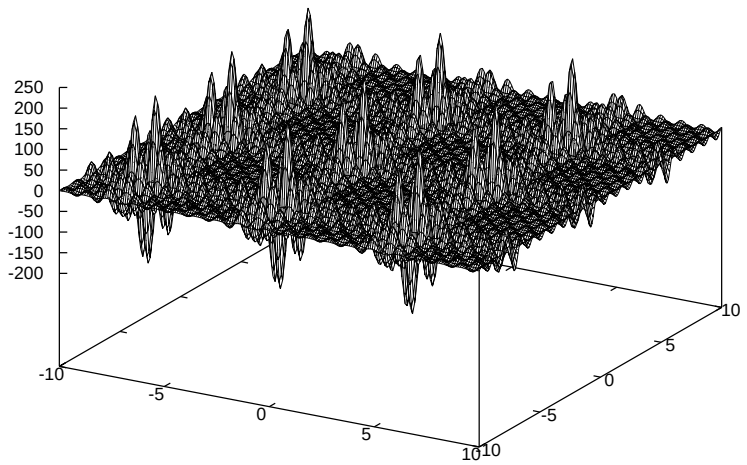


Fig. 2. The two-dimensional Shubert's function.

5.2 Results

Table 2 summarizes the results of our experiments. In the first column, we show the name of the test function. The second column indicates the size of the swarm, and the rest of the columns indicate the mean and standard deviation, of the number of evaluations required to find all the optima of each test function. The last column corresponds to the proposed EPSO method. It is worth mentioning that with the EPSO method, all the global optima are found in all experiments.

Table 2. Comparison with the results reported for other PSO methods

Function	Particles	SPSO	ANPSO	kPSO	EPSO
F1	30	3169±692	5220±3323	2084±440	**1581±79**
	60	6226±1707	6927±2034	3688±717	**2961±133**
F2	30	2872±827	2798±857	1124±216	**888±78**
	60	5820±1469	4569±1316	2127±341	**1735±128**
F3	30	2007±703	6124±2465	1207±688	**889±93**
	60	4848±2092	8665±2974	1654±705	**1529±174**
F4	30	4096±731	16308±13157	2259±539	**1669±56**
	60	7590±2018	17168±12006	3713±570	**2523±111**
F5	300	166050±42214	82248±10605	81194±45646	**33093±607**
	500	219420±80179	114580±18392	117503±77451	**54010±1111**

From the results shown in Table 2, we can observe that the proposed method needs less function evaluations than any of the other methods with respect to which it was compared. Also, the proposed approach has a smaller standard deviation, which means that it has better stability. We found particularly remarkable the results produced by our proposed approach in the case of the two-dimensional Shubert's function. In that case, our proposed approach requires less than half of the evaluations required by the best approach previously reported for this problem. Additionally, the standard deviation of our proposed approach is, in this case, smaller than that of the best approach previously reported for this problem, by two orders of magnitude in the case of 300 particles, and in one order of magnitude for the case of 500 particles per swarm.

6 Conclusions and Future Work

We have presented a new multimodal PSO method that does not require any additional parameters besides those that are inherent to the original PSO algorithm. We argue that our proposed approach is very easy to implement, and we have shown that keeps an $O(N^2)$ complexity (N is the number of particles in the swarm) when computing a numerical value for each pair of particles at each generation. However, in spite of its simplicity, our proposed approach was capable of obtaining better results than other previously reported PSO proposals, in terms of the number of iterations required to locate all the global optima of a function. Our results also show that the standard deviations achieved are small, which is a clear indication of better stability than previously reported PSO methods used for multimodal optimization. Additionally, it is worth mentioning that in the most complex test function adopted (Schubert's function), our proposed approach found all the global optima with less than half of the number of evaluations needed by the best PSO method previously reported for this problem.

As part of our future work, we are interested in applying our proposed approach to problems of higher dimensionality. However, since the current test problems available are normally of very low dimensionality (one or two decision variables), we are currently developing a framework that provides a simple and flexible way to increment the number of variables and optima of the test functions.

The proposed method does not fully model the interaction of a set of charged particles. We only consider the interaction between pairs of particles and select the particle's "best" as the particle with the maximum electrostatic force in the swarm. Thus, a more detailed model of electrostatic interactions would be quite interesting, and is part of our ongoing research. For example, we believe that the use of the electrostatic force to compute the acceleration of the particles might provide an improved accuracy and stability of our proposed approach, when searching for several optima.

References

1. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Reading, Massachusetts (1989)
2. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks. Volume 4. (December 1995) 1942–1948
3. Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems. Second edn. Springer, New York (September 2007) ISBN 978-0-387-33254-3.
4. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation **6**(1) (February 2002) 58–73
5. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: Proceedings of the 1998 IEEE International Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence. (May 1998) 69–73
6. Li, X.: Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization. In: Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO '04), Springer-Verlag (2004) 105–116
7. Iwamatsu, M.: Multi-species particle swarm optimizer for multimodal function optimization. Transactions on Information and Systems **E89-D**(3) (2006) 1181–1187
8. Iwamatsu, M.: Locating all the global minima using multi-species particle swarm optimizer: The inertia weight and the constriction factor variants. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06). (2006) 816–822
9. Li, J.P., Balazs, M.E., Parks, G.T., Clarkson, P.J.: A species conserving genetic algorithm for multimodal function optimization. Evolutionary Computation **10**(3) (2002) 207–234
10. Goldberg, D.E., Richardson, J.: Genetic algorithm with sharing for multimodal function optimization. In Grefenstette, J.J., ed.: Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, Hillsdale, New Jersey, Lawrence Erlbaum (1987) 41–49
11. Deb, K., Goldberg, D.E.: An Investigation of Niche and Species Formation in Genetic Function Optimization. In Schaffer, J.D., ed.: Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, California, George Mason University, Morgan Kaufmann Publishers (June 1989) 42–50
12. Mahfoud, S.W.: Niching methods for genetic algorithms. PhD thesis, University of Illinois at Urbana-Champaign, Champaign, IL, USA (1995)
13. Brits, R., Engelbrecht, A.P., Bergh, F.V.D.: A niching particle swarm optimizer. In: Proceedings of the Conference on Simulated Evolution and Learning. (2002) 692–696
14. Bird, S., Li, X.: Adaptively choosing niching parameters in a pso. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation (GECCO '06), New York, NY, USA, ACM (2006) 3–10
15. Passaro, A., Starita, A.: Particle swarm optimization for multimodal functions: a clustering approach. Journal Artificial Evolution and Applications **8**(2) (2008) 1–15
16. Kennedy, J.: Stereotyping: improving particle swarm performance with cluster analysis. In: Proceedings of the 2000 Congress on Evolutionary Computation. Volume 2. (2000) 1507–1512

17. Pelleg, D., Moore, A.: X-means: Extending k-means with efficient estimation of the number of clusters. In: Proceedings of the 17th International Conference on Machine Learning, Morgan Kaufmann (2000) 727–734
18. Li, X.: A multimodal particle swarm optimizer based on fitness euclidean-distance ratio. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation (GECCO '07), New York, NY, USA, ACM (2007) 78–85
19. Peram, T., Veeramachaneni, K., Mohan, C.: Fitness-distance-ratio based particle swarm optimization. In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium (SIS '03). (April 2003) 174–181