SIViP (2012) 6:343–350
DOI 10.1007/s11760-012-0316-2

ORIGINAL PAPER

Introducing the fractional-order Darwinian PSO

Micael S. Couceiro · Rui P. Rocha · N. M. Fonseca Ferreira · J. A. Tenreiro Machado

Received: 16 September 2011 / Revised: 15 February 2012 / Accepted: 19 February 2012 / Published online: 5 May 2012

Abstract One of the most well-known bio-inspired algorithms used in optimization problems is the particle swarm optimization (PSO), which basically consists on a machine-learning technique loosely inspired by birds flocking in search of food. More specifically, it consists of a number of particles that collectively move on the search space in search of the global optimum. The Darwinian particle swarm optimization (DPSO) is an evolutionary algorithm that extends the PSO using natural selection, or survival of the fittest, to enhance the ability to escape from local optima. This paper firstly presents a survey on PSO algorithms mainly focusing on the DPSO. Afterward, a method for controlling the convergence rate of the DPSO using fractional calculus (FC) concepts is proposed. The fractional-order optimization algorithm, denoted as FO-DPSO, is tested using several well-known functions, and the relationship between the fractional-order velocity and the convergence of the algorithm is observed. Moreover, experimental results show that the FO-DPSO significantly outperforms the previously presented FO-PSO.

Keywords Fractional calculus · DPSO · Evolutionary algorithm

M. S. Couceiro (✉) · N. M. F. Ferreira
RoboCorp, Department of Electrotechnics Engineering,
Engineering Institute of Coimbra, Coimbra, Portugal
e-mail: micael@isec.pt

N. M. F. Ferreira
e-mail: nunomig@isec.pt

M. S. Couceiro · R. P. Rocha
Mobile Robotics Laboratory, Institute of Systems and Robotics,
University of Coimbra, Pólo 2, Portugal
e-mail: micaelcouceiro@isr.uc.pt

R. P. Rocha
e-mail: rprocha@isr.uc.pt

J. A. T. Machado
Department of Electrotechnics Engineering,
Engineering Institute of Porto, Porto, Portugal
e-mail: jtm@isep.ipp.pt

1 Introduction

Bio-inspired algorithms have been employed in situations where conventional optimization techniques cannot find a satisfactory solution, for example, when the function to be optimized is discontinuous, non-differentiable, and/or presents too many nonlinearly related parameters [1]. The Darwinian particle swarm optimization (DPSO), an evolutionary algorithm that extends the particle swarm optimization (PSO) using *natural selection*, was developed to enhance the ability of the PSO to escape from local optima [3].

The theory of fractional calculus (FC) is a useful mathematical tool for applied sciences [4]. In fact, FC has played a very important role increasing the performance of several algorithms used in modeling, curve fitting, filtering, pattern recognition, edge detection, identification, stability, controllability, observability, and robustness.

Therefore, this paper proposes a fractional-order (FO) DPSO using fractional calculus to control the convergence rate of the algorithm. Section two presents the state-of-the-art of the several PSO main variants mainly focusing on the DPSO developed by [3]. Section three generalizes the DPSO to a fractional order. Experimental results for the FO-DPSO are presented in section four. Finally, section five outlines the main conclusions.

2 A survey on PSO techniques

The original PSO was developed by Eberhart and Kennedy in 1995 [2], and it is based on social and computer science. The PSO basically takes advantages on the swarm intelligence concept, which is the property of a system, whereby the collective behaviors of unsophisticated agents that are interacting locally with their environment, create coherent global functional patterns [5]. Imagine a flock of birds where each bird cries at an intensity proportional to the amount of food that it finds at its current location. At the same time, each bird can perceive the position of neighboring birds and can tell which of the neighboring birds emits the loudest cry. There is a good chance that the flock will find a spot with the highest concentration of food if each bird follows a trajectory that combines three directions: (i) keep flying in the same direction; (ii) return to the location where it found the highest concentration of insects so far; and (iii) move toward the neighboring bird that cries the loudest [1]. In the traditional PSO, the candidate solutions are called particles. These particles travel through the search space to find an optimal solution, by interacting and sharing information with neighbor particles, namely their individual best solution (local best) and computing the neighborhood best. Also, in each step of the procedure, the global best solution obtained in the entire swarm is updated. Using all of this information, particles realize the locations of the search space where success was obtained and are guided by these successes. In each step of the algorithm (Algorithm 1), a fitness function is used to evaluate the particle success. To model the swarm, each particle n moves in a multidimensional space according to position (x_t^n) and velocity (v_t^n) values, which are highly dependent on local best ($\breve{x}_t^n$), neighborhood best ($\breve{n}_t^n$), and global best ($\breve{g}_t^n$) information:

$$v_{t+1}^n = wv_t^n + \rho_1 r_1 \left(\breve{g}_t^n - x_t^n\right) + \rho_2 r_2 \left(\breve{x}_t^n - x_t^n\right) + \rho_3 r_3 \left(\breve{n}_t^n - x_t^n\right) \quad (1)$$

$$x_{t+1}^n = x_t^n + v_{t+1}^n \quad (2)$$

The coefficients w, ρ_1, ρ_2, and ρ_3 assign weights to the inertial influence, the global best, the local best, and the neighborhood best when determining the new velocity, respectively. Typically, the inertial influence is set to a value slightly less than 1. ρ_1, ρ_2, and ρ_3 are constant integer values, which represent "cognitive" and "social" components. However, different results can be obtained by assigning different influences for each component. For example, several works do not consider the neighborhood best, and ρ_3 is set to zero. Depending on the application and the characteristics of the problem, tuning these parameters properly will lead to better results. The parameters r_1, r_2, and r_3 are random vectors with each component generally a uniform random number between 0 and 1. The intent is to multiply a new random component per velocity dimension, rather than multiplying the same component with each particle's velocity dimension.

In the beginning, the particles' velocities are set to zero, and their position is randomly set within the boundaries of the search space (Algorithm 1). The local, neighborhood, and global bests are initialized with the worst possible values, taking into account the nature of the problem. There are other few parameters that need to be adjusted: (i) population size—very important to optimize to get overall good solutions in acceptable time; and (ii) stopping criteria—it can be a predefined number of iterations without getting better results or other criteria, depending on the problem.

Algorithm 1 Traditional *PSO* Algorithm

```
Initialize swarm (Initialize $x_t^n$, $v_t^n$, $\breve{x}_t^n$, $\breve{n}_t^n$ and $\breve{g}_t^n$)
Loop:
    for all particles
        Evaluate the fitness of each particle
        Update $\breve{x}_t^n$, $\breve{n}_t^n$ and $\breve{g}_t^n$
        Update $v_{t+1}^n$ and $x_{t+1}^n$
    end
until stopping criteria (convergence)
```

PSO reveals an effect of implicit communication between particles (similar to broadcasting) by updating neighborhood and global information, which affects the velocity and consequent position of particles. Also, there is a stochastic exploration effect due to the introduction of the random multipliers (r_1, r_2, and r_3). The PSO has been successfully used in many applications such as robotics [6–9], electric systems [10], and sport sciences [11].

However, a general problem with the PSO and other optimization algorithms is that of becoming trapped in a local optimum such that it may work well on one problem, but may fail on another problem. In order to overcome this problem, many authors have suggested other adjustments to the parameters of the PSO algorithm combining Fuzzy logic (FA-PSO), where the inertia weight w is dynamically adjusted using fuzzy "IF-THEN" [12] rules, or Gaussian approaches (GPSO), where the inertia constant w is no longer needed, and the acceleration constants ρ_1, ρ_2, and ρ_3 are replaced by random numbers with Gaussian distributions [13].

More recently, Pires et al. used fractional calculus to control the convergence rate of the PSO [14]. The authors rearrange the original velocity Eq. (1) in order to modify the order of the velocity derivative. This paper tries to control the convergence rate of an evolutionary version of the PSO based on Pires et al. work, since the well-succeeded variants of the PSO are the ones based on evolutionary techniques [5].

Many authors have considered incorporating selection, mutation, and crossover, as well as the differential evolution (DE), into the PSO algorithm. The main goal is to increase the diversity of the population by either preventing the particles to move too close to each other and collide [15,16] or to self-adapt parameters such as the constriction factor, acceleration constants [17], or inertia weight [18]. The fusion between genetic algorithms (GA) and the PSO originated the GA-PSO [19], which combines the advantages of swarm intelligence and a natural selection mechanism, such as GA, in order to increase the number of highly evaluated agents, while decreasing the number of lowly evaluated agents at each iteration step. Similar to this last one, the EPSO is an evolutionary approach that incorporates a selection procedure to the original PSO algorithm, as well as self-adapting properties for its parameters. This algorithm adds a tournament selection method used in evolutionary programming (EP) [20]. Based on the EPSO, a differential evolution operator has been proposed to improve the performance of the algorithm in two different ways. The first one [21] applies the differential evolution operator to the particle's best position to eliminate the particles falling into local minima (DEPSO), while the second one [22] applies it to find the optimal parameters (inertia and acceleration constants) for the canonical PSO (C-PSO).

In search of a better model of natural selection using the PSO algorithm, the DPSO was formulated by [3], in which many swarms of test solutions may exist at any time. Each swarm individually performs just like an ordinary PSO algorithm with some rules governing the collection of swarms that are designed to simulate natural selection. Despite the similarities between the PSO and GAs like randomly generated population, fitness function evaluation, population update, search for optimality with random techniques, and not guaranteeing success, PSO does not use genetic operators like crossover and mutation, thus not being considered an evolutionary technique. On the other hand, the DPSO extends the PSO to determine whether natural selection (Darwinian principle of survival of the fittest) can enhance the ability of the PSO algorithm to escape from local optima. The idea is to run many simultaneous parallel PSO algorithms, each one a different swarm, on the same test problem, and a simple selection mechanism is applied. When a search tends to a local optimum, the search in that area is simply discarded and another area is searched instead. In this approach, at each step, swarms that get better are rewarded (extend particle life or spawn a new descendent) and swarms that stagnate are punished (reduce swarm life or delete particles). To analyze the general state of each swarm, the fitness of all particles is evaluated, and the neighborhood and individual best positions of each of the particles are updated. If a new global solution is found, a new particle is spawned. A particle is deleted if the swarm fails to find a fitter state in a defined number of steps (Algorithm 2).

Some simple rules are followed to delete a swarm, delete particles, and spawn a new swarm and a new particle: (i) when the swarm population falls below a minimum bound, the swarm is deleted; and (ii) the worst performing particle in the swarm is deleted when a maximum threshold number of steps (search counter) without improving the fitness function is reached. After the deletion of the particle, instead of being set to zero, the counter is reset to a value approaching the threshold number, according to:

$$SC_C\left(N_{\text{kill}}\right) = SC_C^{\max}\left[1 - \frac{1}{N_{\text{kill}} + 1}\right] \quad (3)$$

with being the number of particles deleted from the swarm over a period in which there was no improvement in fitness. To spawn a new swarm, a swarm must not have any particle ever deleted, and the maximum number of swarms must not be exceeded. Still, the new swarm is only created with a probability of $p = f/\text{NS}$, with f a random number in [0,1] and NS the number of swarms. This factor avoids the creation of newer swarms when there are large numbers of swarms in existence. The parent swarm is unaffected, and half of the parent's particles are selected at random for the child swarm, and half of the particles of a random member of the swarm collection are also selected. If the swarm initial population number is not obtained, the rest of the particles are randomly initialized and added to the new swarm. A particle is spawned whenever a swarm achieves a new global best and the maximum defined population of a swarm has not been reached.

Algorithm 2 *DPSO* Algorithm

```
Main Program Loop
For each swarm in the collection
    Evolve the swarm (Evolve Swarm Algorithm: right)
    Allow the swarm to spawn
    Delete "failed" swarms

Evolve Swarm Algorithm
  For each particle in the swarm
      Update Particles' Fitness
      Update Particles' Best
      Move Particle
      If swarm gets better
        Reward swarm: spawn particle: extend swarm life
      If swarm has not improved
        Punish swarm: possibly delete particle: reduce swarm life
```

Like the PSO, a few parameters also need to be adjusted to run the algorithm efficiently: (i) initial swarm population; (ii) maximum and minimum swarm population; (iii) initial number of swarms; (iv) maximum and minimum number of swarms; and (v) stagnancy threshold. In estimation problems previously studied in [11] or robotic exploration strategies proposed in [23], the DPSO has been successfully compared with the PSO showing a superior performance.

Later on, the results obtained using the proposed FO-DPSO will be compared with the FO-PSO developed by [14] and discussed. Next chapter presents the use of the FC to control the convergence rate of the DPSO.

3 Fractional-order Darwinian particle swarm optimization

In this section, a new method to control the DPSO algorithm based on Pires et al. approach to the traditional SO [14] is introduced and denoted as FO-DPSO.

Fractional calculus (FC) has attracted the attention of several researchers [4,24,25], being applied in various scientific fields such as engineering, computational mathematics, fluid mechanics, among others [26–29]. The *Grünwald–Letnikov* definition based on the concept of fractional differential with fractional coefficient $\alpha \in C$ of a general signal $x(t)$ is given by:

$$D^{\alpha}\left[x(t)\right] = \lim_{h\to 0}\left[\frac{1}{h^{\alpha}}\sum_{k=0}^{+\infty}\frac{(-1)^{k}\,\Gamma(\alpha+1)\,x(t-kh)}{\Gamma(k+1)\,\Gamma(\alpha-k+1)}\right] \tag{4}$$

where Γ is the gamma function.

An important property revealed by the *Grünwald–Letnikov* eqution (4) is that while an integer-order derivative just implies a finite series, the FO derivative requires an infinite number of terms. Therefore, integer derivatives are "local" operators, while fractional derivatives have, implicitly, a "memory" of all past events. However, the influence of past events decreases over time.

Based on Eq. (4), a discrete time implementations expression can be defined as:

$$D^{\alpha}\left[x(t)\right] = \frac{1}{T^{\alpha}}\sum_{k=0}^{r}\frac{(-1)^{k}\,\Gamma(\alpha+1)\,x(t-kT)}{\Gamma(k+1)\,\Gamma(\alpha-k+1)} \tag{5}$$

where T is the sampling period and r is the truncation order.

The characteristics revealed by fractional calculus make this mathematical tool well suited to describe phenomena such as irreversibility and chaos because of its inherent memory property. In this line of thought, the dynamic phenomena of particle's trajectory configure a case where fractional calculus tools fit adequately.

Considering the inertial influence of Eq. (1) $w = 1$, assuming $T = 1$ and based on [14] work, the following expression can be defined:

$$\begin{aligned} D^{\alpha}\left[v_{t+1}^{n}\right] &= \rho_1 r_1\left(\breve{g}_t^{n} - x_t^{n}\right) + \rho_2 r_2\left(\breve{x}_t^{n} - x_t^{n}\right) \\ &\quad + \rho_3 r_3\left(\breve{n}_t^{n} - x_t^{n}\right) \end{aligned} \tag{6}$$

Preliminary experimental tests on the algorithm presented similar results for $r \geq 4$. Furthermore, the computational requirements increase linearly with $r, thatis$, the FO-DPSO present a $\mathcal{O}(r)$ memory complexity. Hence, using only the first $r = 4$ terms of differential derivative given by (5)and Eq. (6) can be rewritten as (7):

$$\begin{aligned} v_{t+1}^{n} &= \alpha v_t^{n} + \frac{1}{2}\alpha v_{t-1}^{n} + \frac{1}{6}\alpha(1-\alpha)\,v_{t-2}^{n} \\ &\quad + \frac{1}{24}\alpha(1-\alpha)(2-\alpha)\,v_{t-3}^{n} + \rho_1 r_1\left(\breve{g}_t^{n} - x_t^{n}\right) \\ &\quad + \rho_2 r_2\left(\breve{x}_t^{n} - x_t^{n}\right) + \rho_3 r_3\left(\breve{n}_t^{n} - x_t^{n}\right) \end{aligned} \tag{7}$$

Observing Eq. (7), one can conclude that the DPSO is then considered as being a particular case of the FO-DPSO when $\alpha = 1$ (without "memory"). Moreover, the FO-DPSO may also be seen as a collection of FO-PSOs [14], in which each swarm individually performs with some natural selection rules (cf., Sect. 2).

Although this new equation incorporates the concept of FC, the difficulty to understand the influence inherent to the fractional coefficient α still remains: What should be the most adequate value for α?

As described in [30] and [31], a swarm behavior can be divided into two activities: (i) *exploitation* and (ii) *exploration*. The first one is related with the convergence of the algorithm, thus allowing a good short-term performance. However, if the exploitation level is too high, then the algorithm may be stuck on local solutions. The second one is related with the diversification of the algorithm, which allows exploring new solutions, thus improving the long-term performance. However, if the exploration level is too high, then the algorithm may take too much time to find the global solution. As first presented by Shi and Eberhart [12], the trade-off between exploitation and exploration in the classical PSO has been commonly handled by adjusting the inertia weight. A large inertia weight improves exploration activity, while exploitation is improved using a small inertia weight. Since the FO-DPSO presents a FC strategy to control the convergence of particles, the coefficient α needs to be defined in order to provide a high level of exploration while ensuring the global solution of the mission. Therefore, the FO-DPSO will be experimentally evaluated in next section using Eq. (7) for all particles in all swarms.

4 Experimental results

This section presents experimental results of the proposed FO-DPSO. Also, in order to compare this approach with Pires et al. approach [14], the same test functions and parameters are used as depicted in Table 1. Table 1 also shows the specific parameters of the DPSO algorithm.

The median of the fitness evolution of the best global particle is taken as the system output, for each value in the set $\alpha = \{0, 0.1, \ldots, 1\}$. In Figs. 1, 2, 3, 4 and 5, the results can be seen for the adopted optimization functions f_j, $j = \{1, \ldots, 5\}$.

Table 1 Specifications of the algorithm and optimization functions

	Min	Initial	Max
Number of simulations	–	201	–
Number of iterations	–	200	–
Coefficients $\rho_1 = \rho_2 = \rho_3$	–	0.8	–
Swarm population	3	4	5
Stagnancy threshold	–	10	–
Optimization functions f_j [32]	1- *Bohachevsky 1*		
	2- *Colville*		
	3- *Drop wave*		
	4- *Easom*		
	5- *Rastrigin*		

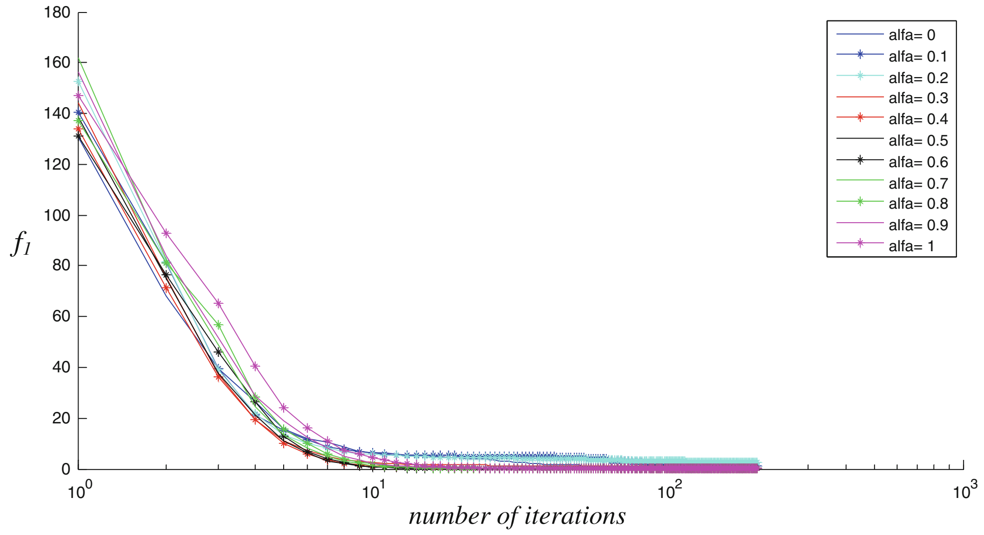

Fig. 1 Evolution of the *Bohachevsky 1* function changing α

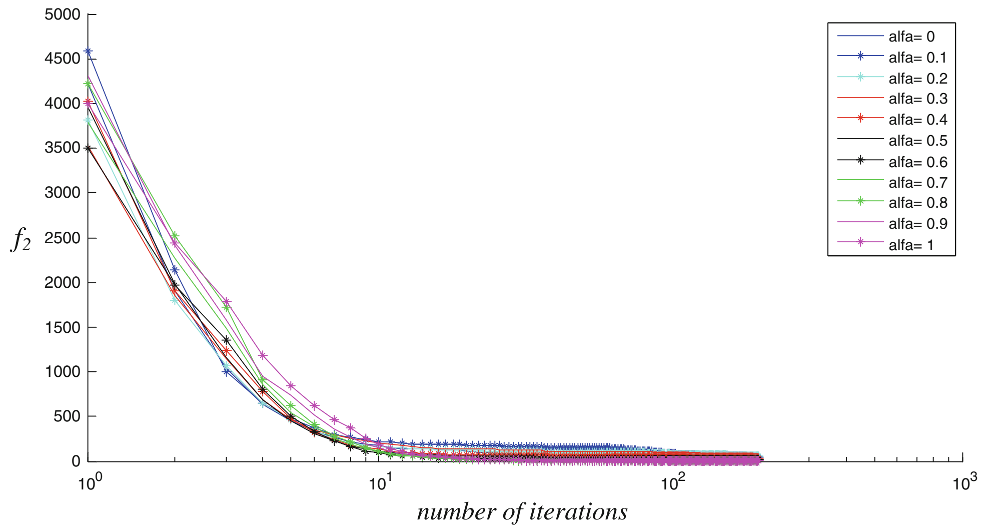

Fig. 2 Evolution of the *Colville* function changing α

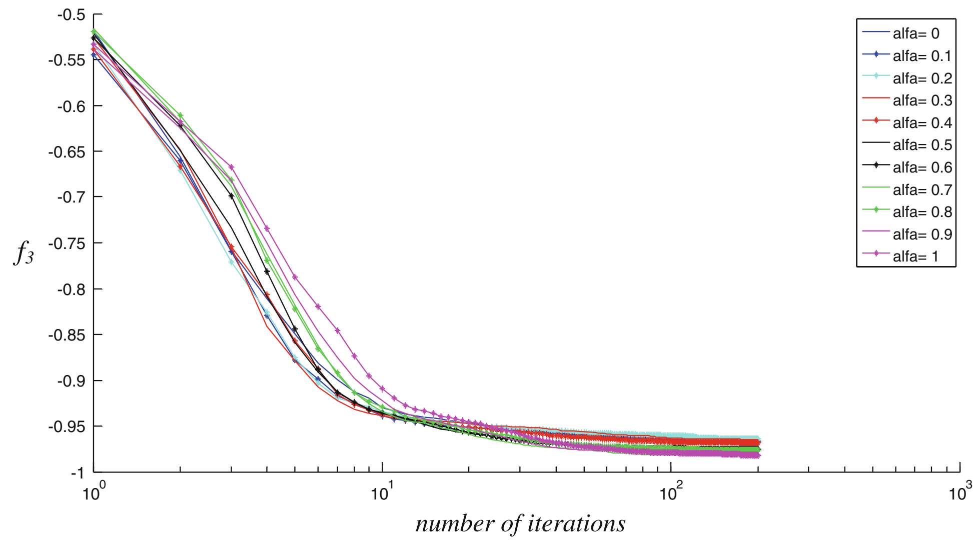

Fig. 3 Evolution of the *Drop wave* function changing α

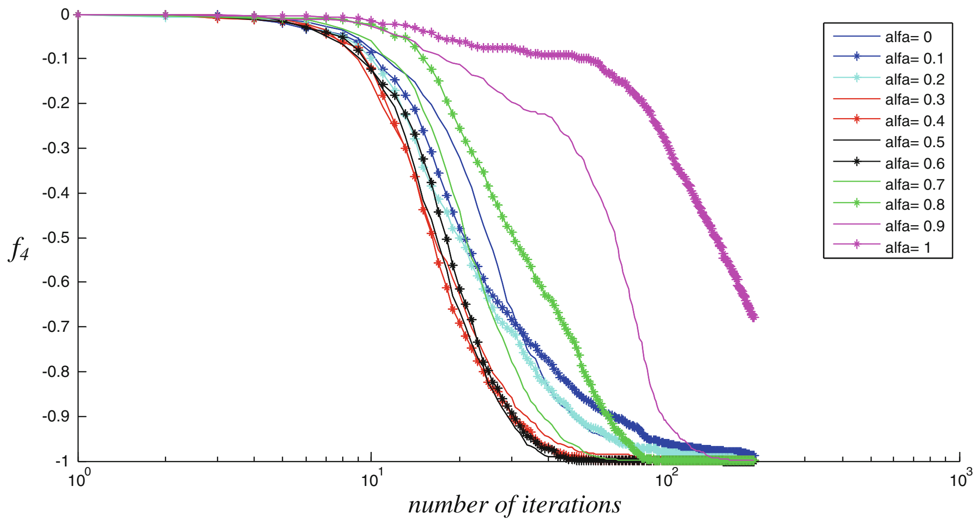

Fig. 4 Evolution of the *Easom* function changing α

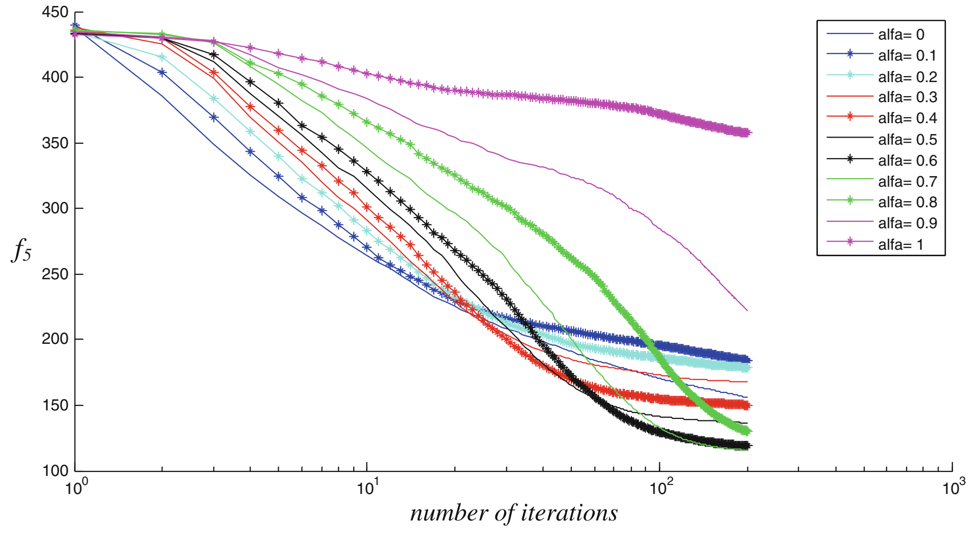

Fig. 5 Evolution of the *Rastrigin* function changing α

Experimental results show that the convergence of the algorithm depends upon the fractional order α. However, contrary to the FO-PSO presented in [14], the Darwinian algorithm easily avoids being stuck in local solutions independently on the value of α (since it is a particularity of the traditional DPSO). Moreover, one can observe that, in most situations, a faster optimization convergence is obtained for a fractional coefficient α in the range [0.5, 0.8]. Therefore, to further evaluate the FO-DPSO, let us then systematically adjust the fractional coefficient α between 0.5 and 0.8, according to the following expression:

$$\alpha(t) = 0.8 - 0.3\frac{t}{200} \tag{8}$$

Once again, the median of the fitness evolution of the best global particle is taken as the system output. In Fig. 6, the results can be seen for the adopted optimization functions f_j, $j = \{1, \ldots, 5\}$, while comparing the FO-DPSO with the FO-PSO proposed by Pires et al. [7] using Eq. (8). Observing Fig. 6, one can conclude that, despite both FO-PSO and FO-DPSO revealed a similar behavior, the combination of FC and Darwin's principles contributes to an improved convergence dynamics.

5 Conclusions

The search for an algorithm capable of dealing with most optimization problems without being very time-consuming and computationally demanding has been a subject of research in several scientific areas such as control engineering and applied mathematics. Fractional calculus has appeared as a tool to enhance the performance of conventional mathematical methods. This work proposed a new optimization algorithm based on the DPSO using the concept of fractional derivative to control the convergence rate.

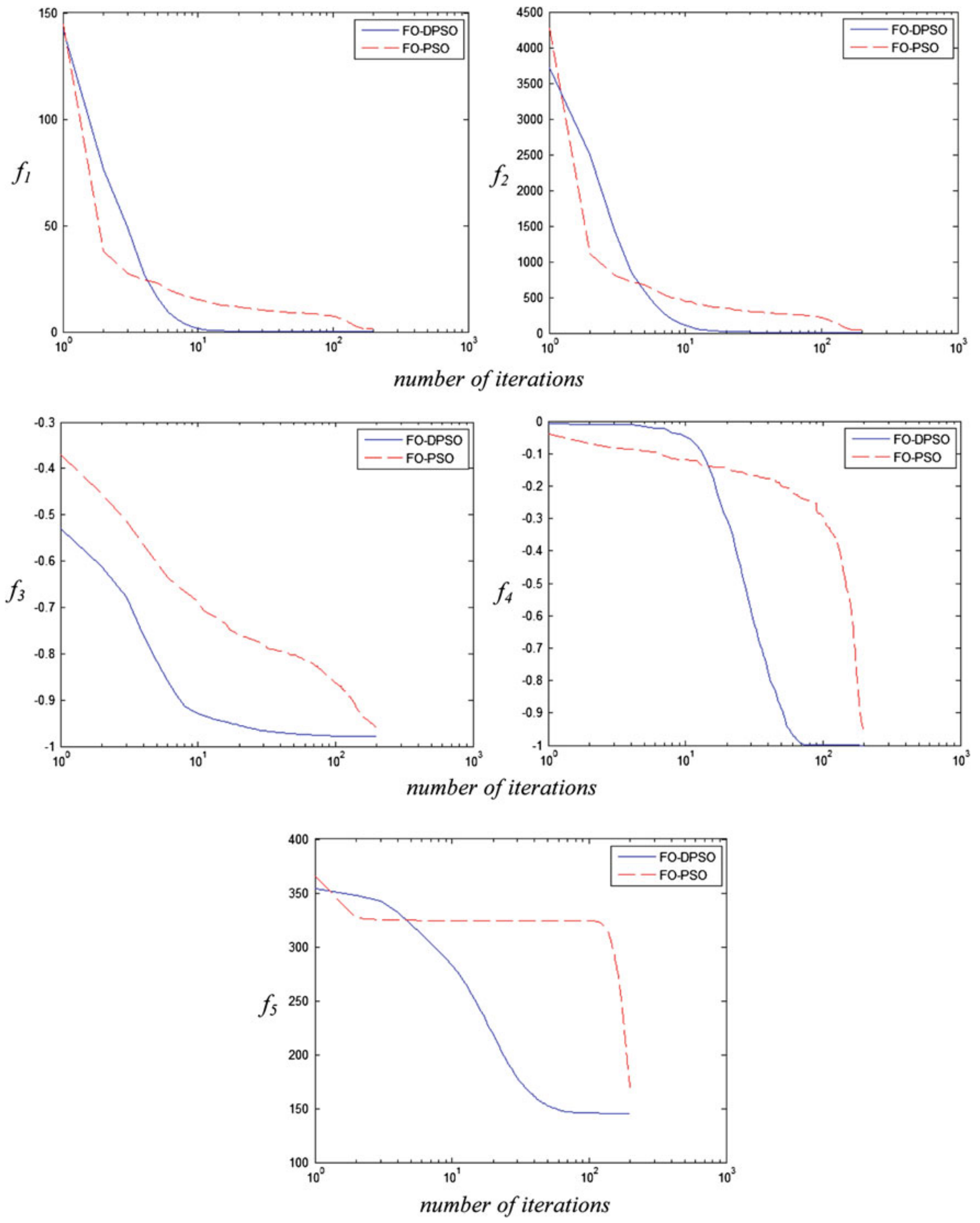

Fig. 6 Evolution of the fitness function, with variable α for FO-PSO and FO-DPSO

Experimental results show that, although the speed of convergence of the fractional-order DPSO (FO-DPSO) depends on the fractional order α, the herein proposed algorithm outperforms the traditional DPSO and PSO¸ as well as the FO-PSO previously presented in the literature. However, each optimization problem may have a slightly different optimal α. Therefore, as future work, we propose to extend the FO-DPSO with adaptive ability to tune the fractional order α based on the contextual information inherent to each problem.

Acknowledgments This work was supported by a PhD scholarship (SFRH/BD /73382/2010) granted to the first author by the Portuguese Foundation for Science and Technology (FCT), the Institute of Systems and Robotics and RoboCorp.

References

1. Floreano, D., Mattiussi, C.: Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies. MIT Press, Cambridge (2008)
2. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the 6th International Symposium Micro Machine and Human Science (MHS), 1995, pp. 39–43 (1995)
3. Tillett, T., Rao, T.M., Sahin, F., Rao, R.: Darwinian particle swarm optimization. In: Proceedings of the 2nd Indian international conference on artificial intelligence, Pune, Índia, pp. 1474–1487 (2005)
4. Sabatier, J., Agrawal, O.P., Tenreiro Machado, J.A. (eds.): Advances in Fractional Calculus—Theoretical Developments and Applications in Physics and Engineering. Springer, Berlin. ISBN:978-1-4020-6 -0. (2007)
5. del Valle , Y., Venayagamoorthy, G.K., Mohagheghi, S., Hernandez, J.C., Harley, R.: Particle swarm optimization: basic concepts, variants and applications in power systems. In: IEEE Trans. Evol. Comput. **12**(2), 171–195 (2008)
6. Tang, J., Zhu, J., Sun, Z.: A novel path panning approach based on appart and particle swarm optimization. In: Proceedings of the 2nd International Symposium on Neural Networks, LNCS 3498, pp. 253–258 (2005)
7. Pires, E.J.S., Oliveira, P.B.M., Machado, J.A.T., Cunha, J.B.: Particle swarm optimization versus genetic algorithm in manipulator trajectory planning. In: 7th Portuguese Conference on Automatic Control, September 11–13 (2006)
8. Couceiro, M.S., Mendes, R.M., Ferreira, N.M.F., Machado, J.A.T.: Control Optimization of a Robotic Bird. EWOMS '09, Lisbon, Portugal, June 4–6 (2009)
9. Couceiro, M.S., Luz, J.M.A., Figueiredo, C.M., Ferreira, N.M.F.: Modeling and control of biologically inspired flying robots. J. Robotica, Cambridge University Press (2011)
10. Alrashidi, M.R., El-Hawary, M.E.: A survey of particle swarm optimization applications in power system operations. Electr. Power Compon. Syst. **34**(12), 1349–1357 (2006)
11. Couceiro, M.S., Luz, J.M.A., Figueiredo, C.M., Ferreira, N.M.F. Dias, G.: Parameter estimation for a mathematical model of the golf putting. In: Marques, V.M., Pereira, C.S., Madureira, A. (eds.) Proceedings of WACI-Workshop Applications of Computational Intelligence. ISEC. IPC. Coimbra. 2 de Dezembro, pp. 1–8. ISSN 978-989-8331-10-6 (2010)
12. Shi, Y., Eberhart, R.: Fuzzy adaptive particle swarm optimization. In: Proceedings of the In: IEEE Congress Evolutionary Computation, vol. 1, pp. 101–106 (2001)
13. Secrest, B., Lamont, G.: Visualizing particle swarm optimization—Gaussian particle swarm optimization. In: Proceedings of the In: IEEE Swarm Intelligence Symposium, pp. 198–204 (2003)
14. Pires, E.J.S., Machado, J.A.T., Oliveira, P.B.M., Cunha, J.B., Mendes, L.: Particle swarm optimization with fractional-order velocity. J. Nonlinear Dyn. **61**, 295–301 (2010)
15. Blackwell, T., Bentley, P.: Don't push me! collision-avoiding swarms. In: Proceedings of the In: IEEE Congress on Evolutionary Computation, vol. 2, pp. 1691–1696 (2002)
16. Krink, T., Vesterstrom, J., Riget, J.: Particle swarm optimization with spatial particle extension. In: Proceedings of the In: IEEE Congress on Evolutionary Computation, vol. 2, pp. 1474–1479 (2002)
17. Miranda, V., Fonseca, N.: New evolutionary particle swarm algorithm (EPSO) applied to voltage/VAR control. In: Proceedings 14th Power Systems Computational Conference (2002)
18. Lovbjerg, M., Krink, T.: Extending particle swarms with self-organized criticality. In: Proceedings of the In: IEEE Congress on Evolutionary Computation, vol. 2, pp. 1588–1593 (2002)
19. Chia-Feng, J.: A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. In: IEEE Trans. Syst. Man Cybern. Part B Cybern. **34**(2), 997–1006 (2004)
20. Angeline, P.: Using selection to improve particle swarm optimization. In: Proceedings of the In: IEEE International Conference Evolutionary Computation, pp. 84–89 (1998)
21. Zhang, W., Xie, X.: DEPSO: hybrid particle swarm with differential evolution operator. In: Proceedings of the In: IEEE International Conference Systems, Man, Cybernetics, vol. 4, pp. 3816–3821 (2003)
22. Kannan, S., Slochanal, S., Padhy, N.: Application of particle swarm optimization technique and its variants to generation expansion problem. ELSERVIER Electr. Power Syst. Res. **70**(3), 203–210 (2004)
23. Couceiro, M.S., Rocha, R.P., Ferreira, N.M.F.: a novel multi-robot exploration approach based on particle swarm optimization algorithms. In: In: IEEE International Symposium on Safety, Security and Rescue Robotics, November 1–5, Kyoto, Japan (2011)
24. Ortigueira, M.D., Tenreiro Machado, J.A.: Special issue on fractional signal processing. Signal Process. **83**, 2285–2480 (2003)
25. Machado, J.A.T., Silva, M.F., Barbosa, R.S., Jesus, I.S., Reis, C.M., Marcos, M.G., Galhano, A.F.: Some Applications of Fractional Calculus in Engineering. Hindawi Publishing Corporation Mathematical Problems in Engineering, 2010, 1–34 (2010)
26. Podlubny, I.: Fractional Differential Equations Mathematics in Science and Engineering 198. Academic Press, San Diego (1999)
27. Debnath, L.: Recents applications of fractional calculus to science and engineering. Int. J. Math. Math. Sci. **54**, 3413–3442 (2003)
28. Elshehawey, E.F., Elbarbary, E.M.E., Afifi, N.A.S., El-Shahed, M.: On the solution of the endolymph equation using fractional calculus. Appl. Math. Comput. **124**, 337–341 (2001)
29. Camargo, R.F., Chiacchio, A.O., Oliveira, E.C.: Differentiation to fractional orders and the fractional telegraph equation. J. Math. Phys. **49**, 033–505 (2008)
30. Yasuda, K., Iwasaki, N., Ueno, G., Aiyoshi, E.: Particle swarm optimization: a numerical stability analysis and parameter adjustment based on swarm activity. In: IEEJ Transactions on Electrical and Electronic Engineering, vol. 3, pp. 642–659, Wiley InterScience (2008)
31. Wakasa, Y., Tanaka, K., Nishimura, Y.: Control-theoretic analysis of exploitation and exploration of the PSO algorithm. In: In: IEEE International Symposium on Computer-Aided Control System Design, In: IEEE Multi-Conference on Systems and Control, Yokohama, Japan (2010)
32. Bergh , F.V.den , Engelbrecht, A.P.: A study of particle swarm optimization particle trajectories. Inf. Sci. **176**(8), 937–971 (2006)