SPRINGER BRIEFS IN
APPLIED SCIENCES AND TECHNOLOGY

Micael Couceiro
Pedram Ghamisi

Fractional Order Darwinian Particle Swarm Optimization

Applications and Evaluation of an Evolutionary Algorithm

Springer

SpringerBriefs in Applied Sciences and Technology

More informations about this series at http://www.springer.com/series/8884

Micael Couceiro · Pedram Ghamisi

Fractional Order Darwinian Particle Swarm Optimization

Applications and Evaluation of an Evolutionary Algorithm

Springer

Micael Couceiro
Ingeniarius, Ltd.
Mealhada
Portugal

and

Institute of Systems and Robotics (ISR)
University of Coimbra
Coimbra
Portugal

Pedram Ghamisi
Faculty of Electrical and Computer Engineering
University of Iceland
Reykjavik
Iceland

ISSN 2191-530X ISSN 2191-5318 (electronic)
SpringerBriefs in Applied Sciences and Technology
ISBN 978-3-319-19634-3 ISBN 978-3-319-19635-0 (eBook)
DOI 10.1007/978-3-319-19635-0

Library of Congress Control Number: 2015940427

Springer Cham Heidelberg New York Dordrecht London
© The Author(s) 2016
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

Foreword

Biomimetics has been exploited in several research areas as a means to endow artificial systems with intelligence, resilience, adaptation, and natural selection typically exhibited by living organisms and biological ecosystems, in order to solve complex human problems through new technologies inspired by such biological systems at macro-and nanoscales.

Bioinspiration has received special attention in the robotics community for the past two decades in order to solve complex optimization problems through bioinspired algorithms. It has been mainly devoted to the study of robot swarms comprising many unsophisticated robots interacting locally with neighbor robots and the environment, which can exhibit useful collective patterns resembling the way swarms of biological species behave collectively to strive for survival in hostile environments against threats and predators. However, such techniques have a broader application range, not being confined only to swarm robotics.

One of the most well-known bioinspired optimization techniques is particle swarm optimization (PSO), which has demonstrated remarkably high potential in optimization problems wherein conventional optimization techniques cannot find a satisfactory solution, due to nonlinearities and discontinuities. The PSO technique consists of a number of particles whose collective dynamics, resembling a biological ecosystem, allows effectively exploring the search space to find the optimal solution. The Darwinian PSO (DPSO) is an evolutionary optimization algorithm and an extension of the original PSO that makes use of Darwin's theory of natural selection to regulate the evolution of the particles and of their collective dynamics, so that complex optimization of functions exhibiting many local maxima/minima can be successfully accomplished. The fractional order DPSO (FODPSO) incorporates in DPSO the notion of fractional-order derivatives to attain memory of past decisions and even better convergence properties.

This book is an important resource for researchers and practitioners on the use of PSO and related bioinspired techniques in complex optimization problems. The Chap. 1 is an excellent introduction to the basics of PSO, along with a presentation of its potential applications. The Chap. 2 presents how the original PSO can be extended to include Darwin's survival-of-the-fittest principle and fractional

calculus, thus presenting the DPSO and the FODPSO techniques. Chapters 3–5 illustrate how these techniques can be used to tackle three interesting case studies effectively: curve fitting, image segmentation, and swarm robotics. I am particularly close to the latter case study, as I had the privilege of supervising a Ph.D. thesis on the subject by one of this book's authors. But, nevertheless, all three case studies are equally relevant for the reader assessing the high potential of the bioinspired techniques addressed in the book. The final chapter summarizes conclusions and points out future research directions on the subject.

I hope the reader can have a pleasant and fruitful reading experience with this book as I had. Enjoy it!

February 2015

Rui P. Rocha
University of Coimbra, Portugal

Acknowledgements

The authors would like to thank Profs. Rui Paulo Pinto da Rocha, Nuno Miguel Fonseca Ferreira, José António Tenreiro Machado, and Jón Atli Benediktsson for their scientific and technical support.

Contents

1 Particle Swarm Optimization 1
1.1 The Traditional PSO. 1
1.1.1 Benefits 4
1.1.2 Drawbacks 4
1.2 PSO-Based Approaches 4
1.3 Applications: What Can We "Optimize"? 8
1.3.1 Curve Fitting. 8
1.3.2 Image Segmentation. 8
1.3.3 Swarm Robotics 9
1.4 Summary 9
References 9

2 Fractional-Order Darwinian PSO 11
2.1 Fractional Calculus. 11
2.2 FODPSO 13
2.2.1 Benefits 14
2.2.2 Drawbacks 14
2.3 Benchmarking Functions. 15
2.4 Summary 20
References 20

3 Case Study I: Curve Fitting 21
3.1 Traditional Optimization Methods 21
3.2 Mathematical Modeling of the Putting 22
3.3 Experimental Results 25
3.4 Summary 29
References 29

4 Case Study II: Image Segmentation 31
4.1 Thresholding-Based Image Segmentation 31
4.1.1 Problem Formulation 33

4.2 Experimental Results 35
4.3 Summary 38
References 38

5 Case Study III: Swarm Robotics 41
5.1 Robotic Darwinian PSO 41
5.1.1 Fractional-Order Swarm Robots 42
5.1.2 Obstacle Avoidance 45
5.1.3 Ensuring MANET Connectivity 47
5.1.4 "Punish–Reward" Mechanism 51
5.1.5 Outline 55
5.2 Experimental Results 60
5.2.1 Numerical Simulation 60
5.2.2 Laboratory Experiments 64
5.3 Summary 70
References 70

6 Conclusions 73
6.1 Main Contributions 73
6.2 Limitations and Further Research 74
References 75

Chapter 1
Particle Swarm Optimization

Abstract Bioinspired 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, nondifferentiable, and/or presents too many nonlinearly related parameters (Floreano and Mattiussi, Bio-inspired artificial intelligence: Theories, methods, and technologies, 2008). One of the most well-known bioinspired algorithms used in optimization problems is particle swarm optimization (PSO), which basically consists of 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. This beginning chapter aims to introduce the main mechanics behind the traditional PSO, outlining its advantages and disadvantages, as well as summarizing the several extensions proposed in the literature over the past almost 20 years.

Keywords PSO · Swarm intelligence · Optimization · Case studies

1.1 The Traditional PSO

The traditional PSO was developed by Kennedy and Eberhart in 1995, in which both researchers got inspired by social and computer sciences (Kennedy and Eberhart 1995). The PSO basically takes advantage of the swarm intelligence concept, which is the property of a system whereby the collective behavior of unsophisticated agents interacting locally with their environment, create coherent global functional patterns (Valle et al. 2008). Imagine a flock of birds wherein 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. Generally speaking, there is a good chance that the flock will find a spot with the highest concentration of food if each bird simply follows a trajectory that combines three directions (Floreano and Mattiussi 2008) as illustrated in Fig. 1.1.

© The Author(s) 2016

M. Couceiro and P. Ghamisi, *Fractional Order Darwinian Particle Swarm Optimization*, SpringerBriefs in Applied Sciences and Technology,
DOI 10.1007/978-3-319-19635-0_1

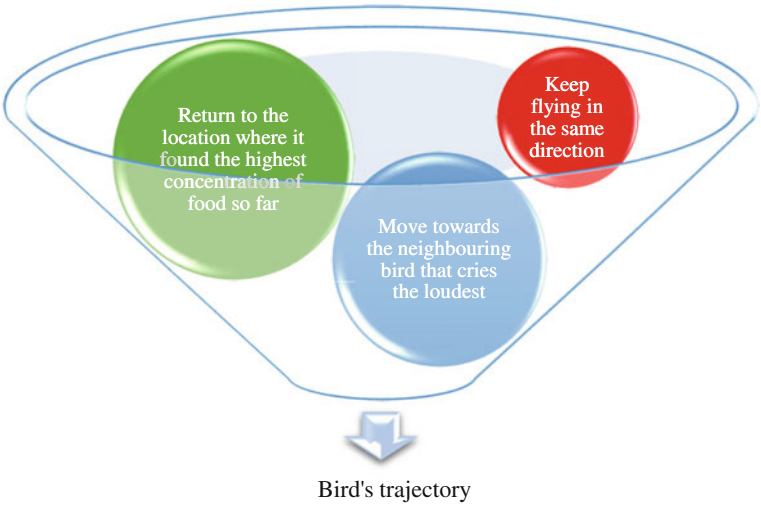

Fig. 1.1 Three main causes that may influence a bird's (particle) trajectory: (i) the previous direction it was traveling (*red ball*); (ii) the best location it personally found thus far (*green ball*); and (iii) the best location the flock collectively found thus far (*blue ball*)

In the traditional PSO, candidate solutions (e.g., birds) 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 1.1), a fitness function is used to evaluate the particle success. To model the swarm, in iteration t, each particle n moves in a multidimensional space according to position $(x_n[t])$ and velocity $(v_n[t])$ values which are highly dependent on local best $\left(\chi_{1_n}[t]\right)$, also known as the cognitive component, and global best $\left(\chi_{2_n}[t]\right)$, typically known as the social component, as follows.

$$v_n[t+1] = wv_n[t] + \sum_{i=1}^{2} \rho_i r_i \left(\chi_{i_n}[t] - x_n[t]\right), \tag{1.1}$$

$$x_n[t+1] = x_n[t] + v_n[t+1], \tag{1.2}$$

Coefficients w, ρ_1, and ρ_2 assign weights to the inertial influence, the local best, and the global best when determining the new velocity $v_n[t+1]$, respectively. Typically, the inertial influence is set to a value slightly less than 1. ρ_1 and ρ_2 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, some works additionally consider a component called neighborhood best (which in this case would be a ρ_3). The parameters r_1 and r_2 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.

Depending on the application and the characteristics of the problem, tuning all these parameters properly may lead to better results. However, this is a problem associated with the branch of parameterized complexity and is far from being completely solved (Downey and Fellows 1999). In fact, Fig. 1.1 clearly illustrates balls of different sizes. This is an analogy made to the influence of each component from Eq. (1.1), namely the previous velocity of the particle by means of w (red ball), its individuality by means of ρ_1 (green ball), and how much it cooperates with its society by means of ρ_2 (blue ball).

```
Initialize swarm (Initialize v_n[0], x_n[0], χ_1n[0], χ_2n[0])
Loop:
    for all particles n
        Evaluate the solution of each particle
        Update χ_1n[t], χ_2n[t]
        Update v_n[t + 1], x_n[t + 1]  // equations 1.1 and 1.2
    end
until stopping criteria (stagnation/performance/time/…)
```

Algorithm1.1. Traditional PSO algorithm.

In the beginning, particles' velocities are set to zero and their positions are randomly set within the boundaries of the search space (Algorithm 1.1). The local and global bests are initialized with the worst possible values, taking into account the nature of the problem. For instance, in a cost problem where the objective is to minimize the fitness function, particles are initialized with a large value (tending to infinity). There are a few other parameters that need to be adjusted:

- Population size: Very important to optimize to get overall good solutions in acceptable time (remember that each particle is a potential solution of the problem).
- Stopping criteria: It can be a predefined number of iterations without getting better results (known as stagnation), or other criteria depending on the problem (common examples are time or performance).

1.1.1 Benefits

The main advantage of using the PSO is its simple concept and ability to be implemented in a few lines of code. Furthermore, PSO also has a short-term memory, which helps the particles to fly over the local best and the global best positions. Alternatives, such genetic algorithms (GA), are more complex and, most of the time, they do not consider the previous iteration or the collective emergent performance. For instance, in GA, if a chromosome is not selected, the information contained by that individual is lost.

1.1.2 Drawbacks

Despite its features, a general problem with the PSO, similarly to other optimization algorithms that are not exhaustive methods, such as the brute-force search (Schaeffer et al. 1993), is that of becoming trapped in a local optimum, or sub-optimal solution, such that it may work well on one problem but yet fail on another problem.

In general, the main drawbacks of PSO can be summarized as follows.

(1) Premature convergence of a swarm: Particles try to converge to a single point, located on a line between the global best and the personal best positions (local best). This point is not guaranteed for a local optimum (Van den Bergh and Engelbrecht 2004). Another reason could be the fast rate of information flow between particles, which leads to the creation of similar particles. This results in a loss in diversity and the possibility of being trapped in local optima is increased (Premalatha and Natarajan 2009).

(2) Parameter settings dependence: This leads to the high performance variances for a stochastic search algorithm (Premalatha and Natarajan 2009). In general, there is not any specific set of parameters for different problems. As an example, and by simple observation of Eq. (1.1), increasing the inertia weight w will increase the speed of the particles $v[t+1]$ and cause more exploration (global search) and less exploitation (local search). As a result, finding the best set of parameters is not a trivial task, and it might be different from one problem to another (Premalatha and Natarajan 2009).

1.2 PSO-Based Approaches

In order to overcome the PSO tendency to get stuck in unwanted solutions and improve its convergence, many authors have suggested other adjustments to the parameters of the PSO algorithm combining, for instance, fuzzy logic, where the inertia weight w is dynamically adjusted using fuzzy IF-THEN-ELSE rules (Shi and

Eberhart 2001; Liu et al. 2007), or Gaussian approaches, where the inertia constant w is no longer needed and the acceleration constants ρ_1 and ρ_2 are replaced by random numbers with Gaussian distributions (Secrest and Lamont 2003). More recently, Pires et al. used fractional calculus to control the convergence rate of the PSO (Pires et al. 2010). The authors rearranged the original velocity Eq. (1.1) in order to modify the order of the velocity derivative. The work of Pires et al. (2010) was one of the foundations for the methodology presented in this book.

Many authors have considered incorporating evolutionary properties, such as selection, mutation, and crossover, as well as differential evolution, into the PSO algorithm (Valle et al. 2008). The main goal of those works was to increase the diversity of the population by either preventing the particles from moving too close to each other and colliding (Blackwell and Bentley 2002; Krink et al. 2002) or to self-adapt parameters, such as the constriction factor, acceleration constants (Miranda and Fonseca 2002), or inertia weight (Lovbjerg and Krink 2002). The fusion between GA and the PSO originated an improved version that combines the advantages of swarm intelligence and a natural selection mechanism in order to increase the number of highly evaluated agents, while decreasing the number of lowly evaluated agents at each iteration step (Chia-Feng 2004). Similar to this last one, a differential evolution operator has been proposed to improve the performance of the PSO algorithm in two different ways. The first one (Zhang and Xie 2003) applies the differential evolution operator to the particle's best position to eliminate the particles falling into a suboptimal solution, whereas the second one (Kannan et al. 2004) applies it to find the optimal parameters (inertia and acceleration constants).

In search of an ever-improving model of natural selection using the PSO algorithm, the Darwinian particle swarm optimization (DPSO) was first formulated by Tillett et al. (2005), in which many groups (or smaller swarms) of test solutions may exist at any time. Note that the word "group" is used throughout this book to identify clusters of particles within the whole swarm (i.e., population). In a few words, we could state that groups are a particular case of swarms, wherein, in contrast to swarms, there is a predefined maximum number of robots allowed to form each group.

In the DPSO, each group individually performs just like an ordinary PSO with some rules governing the collection of groups that are designed to simulate natural selection. This natural selection mechanism, or Darwinian principle of survival-of-the-fittest, enhances the ability of the PSO to escape from suboptimality. The idea is to run many simultaneous parallel PSO algorithms, each one being a different group, on the same test problem, and a simple selection coopetitive mechanism is applied. Note that the word "coopetition" is the neologism combining cooperation and competition (Tsai 2002) and nourishes the principles of natural selection (Darwin 1872).

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, groups that get better are rewarded (extend particle life or spawn a new descendant) and groups that stagnate are punished (reduce group life or delete particles). To analyze the general state of each group, the solution of all particles is evaluated and

the global best of a given group, as well as each individual local best positions of each of the particles within such group, are updated. If a new global solution is found, a new particle is spawned. A particle is deleted if the group fails to find a fitter state in a defined number of steps (Algorithm 1.2). In brief, particles within the same group cooperate, whereas different groups compete toward the same goal and with cross-beneficial properties.

Some simple rules are followed to delete a group, delete particles, and spawn a new group and a new particle, as described in Table 1.1.

After the deletion of a particle, instead of being set to zero, the counter resets to a value that tends to the threshold number, according to:

$$\mathrm{SC}_s = \mathrm{SC}_{\max}\left[1 - \frac{1}{N_s^{\mathrm{kill}} + 1}\right], \tag{1.3}$$

with N_s^{kill} being the number of particles deleted from group s over a period in which there was no improvement in the fitness. For a group to spawn a new group, the group must not have any particle ever deleted and the maximum number of groups must not be exceeded. Still, the new group is only created with a small probability of:

$$p_{sp} = \frac{r_{sp}}{\mathrm{NS}}, \tag{1.4}$$

wherein r_{sp} is a random number between 0 and 1 and NS the number of groups. This factor avoids the creation of newer groups when there are a large number of groups in existence. The parent group is not affected and half of the parent's particles are selected at random for the child group and the other half of the particles of a random member of the group collection are also selected. If the group initial population number is not obtained, the rest of the particles are randomly initialized and added to the new group. A particle is spawned whenever a group achieves a new global best and the maximum defined population of a group has not been

Table 1.1 "Punish–Reward" DPSO rules

Punish	Reward
If a group does not improve during a specific threshold $\mathrm{SC}_{\max}$ (stagnancy counter $\mathrm{SC}_s = \mathrm{SC}_{\max}$) and the number of particles is superior to $N_{\min}(N_s > N_{\min})$, then the group is punished by deleting the worst performing particle	If a group improves and its current number of particles is inferior to $N_{\max}(N_s < N_{\max})$, then it is rewarded with a new particle.
If a group does not improve during a specific threshold $\mathrm{SC}^{\max}$ (stagnancy counter $\mathrm{SC}_s = \mathrm{SC}^{\max}$) and the number of particles is $N_{\min}(N_s = N_{\min})$, then the group is deleted, that is, all particles from that group are deleted	If a group is not stagnant (stagnancy counter $\mathrm{SC}_s = 0$), then it has a small probability p_{sp} of spawning a new group

reached. All these rules belong to the "punish–reward" mechanism of the DPSO (Tillett et al. 2005).

```
Initialize all groups s (Initialize v_n^s[0], x_n^s[0], χ1_n^s[0], χ2_n^s[0])
Loop (Main Program) :
    For each group s in the collection
        Evolve the group (goto: Evolve Swarm Algorithm)
        Allow the group to spawn new group
        Delete "failed" groups
    end
until stopping criteria (convergence)

Function (Evolve Swarm Algorithm):
    for all particles n in the group s
        Evaluate the fitness of each particle
        Update χ1_n^s[t], χ2_n^s[t]
        Update v_n^s[t + 1], x_n^s[t + 1]
        If group s gets better
            Reward group: spawn particle: extend group life
        If group has not improved
            Punish group: possibly delete particle: reduce group life
    end
return
```

Algorithm 1.2. DPSO algorithm.

Despite the improvements of these new evolutionary solutions over the traditional PSO, they all present drawbacks. For instance, the DPSO presented above is considerably more computationally inefficient than the PSO, as it needs to compute several swarms at the same time. Moreover, it still shares the exact same parameter dependency problem depicted by the PSO algorithm, thus being far from having an optimal convergence. Hence, the need to find an improved PSO-based approach is still an open challenge. In this book, the DPSO algorithm is further extended afterwards, as the core of the fractional-order Darwinian particle swarm optimization (FODPSO) presented in this work. For now, let us wonder what problems we could solve if we had a "perfect" PSO-based version.

1.3 Applications: What Can We "Optimize"?

In general, an optimization problem is based on maximizing or minimizing a function by systematically choosing input values from within an allowed set and computing the value of the function. More generally, optimization includes finding "best available" values of some objective function, given a defined domain (or a set of constraints), including a variety of different types of objective functions and different types of domains. PSO-based approaches try to find the optimal solution often without making any assumptions about the problem being optimized and can search very large spaces of candidate solutions. Moreover, they do so iteratively by benefiting from interparticle interactions. Let us take a look at three case studies where one may exploit the performance of PSO-based approaches.

1.3.1 Curve Fitting

One may think that this may be not so much of a challenge. In fact, curve fitting is one of the most traditional optimization problems. As the name suggests, curve fitting is the process of finding the most fit mathematical function to represent a series of data points, possibly subject to some constraints (Lancaster and Salkauskas 1986). This problem has been tackled with many traditional approaches, including gradient descent (Lucey and Matthews 2006), pattern search (Momma and Bennett 2002), and downhill simplex (Emery et al. 2003). However, what all these methods have in common is the fact that they are time consuming in problems with multiple dimensions and relaxed restrictions, as is clear later on in Chap. 3. As a consequence, in this book the focus is shifted to an alternative PSO-based approach because of its popularity and performance.

1.3.2 Image Segmentation

Thresholding is one of the most commonly used methods for the segmentation of images into several objects. The use of the solution presented in this book for image segmentation is described in Chap. 4. In order to find the optimal set of thresholds, one way is by using exhaustive search. A commonly used exhaustive search is based on Otsu's criterion. However, this method is not suitable from a computational cost point of view and it takes a huge amount of time as the number of thresholds increases. The task of finding the optimal thresholds for image segmentation could be formulated as a multidimensional optimization problem. To do so, PSO-based approaches play a key role, generally due to their simple concept and capability for finding thresholds with the highest between-class variances.

1.3.3 Swarm Robotics

Inspired by the results of the existing cooperation in several biological societies (e.g., ants, bees, plants, humans), researchers have placed great emphasis on developing robots that can cooperate with each other and perform multiple tasks. Just as their virtual counterparts, swarm robots have the purpose of solving a complex problem through simplicity and emergent behavior. As such, they share exactly, or almost exactly, the same properties of simplicity, which is typically translated into low-cost units, and their capability of communicating[1] with each other in order to coordinate their actions and obtain the desired result. Nevertheless, contrary to virtual swarm agents, swarm robots are designed to act in the real world, where obstacles, communication problems, and many other interferences, need to be taken into account. The adaptation of the solution presented in this book to swarm robotics is exploited in Chap. 5.

1.4 Summary

In the first chapter of this book, we described the traditional PSO in detail. Then, due to the main shortcomings of the PSO, that is, stagnation into suboptimal solutions and premature convergence of the swarm, different modifications of the traditional PSO have been described. In particular, we put more emphasis on elaborating the DPSO inasmuch as that approach can be considered the backbone of FODPSO, which builds up the main motivation of this book. Finally, we summarized the problems that are solved by considering the FODPSO later in this book.

References

Blackwell, T., & Bentley, P. (2002). Don't push me! Collision-avoiding swarms. *Proceedings of the IEEE Congress on Evolutionary Computation* (Vol. 2, pp. 1691–1696). Honolulu, HI, USA.

Chia-Feng, J. (2004). A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Transactions on Systems*, *Man*, *and Cybernetics*, *Part B*: *Cybernetics, 34*(2), 997–1006.

Darwin, C. (1872). *On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life*. London: Public Domain Books.

Downey, R. G., & Fellows, M. R. (1999). *Parameterized Complexity.* Singapore: Springer.

Emery, L., Borland, M., & Shang, H. (2003). Use of a general-purpose optimization module in accelerator control. *Proceedings of the Particle Accelerator Conference (PAC)*, *IEEE* (Vol. 4, pp. 2330–2332).

[1]Note that communication does not need to be explicit. Robots can use implicit communication or simply observe their teammates' actions.

Floreano, D., & Mattiussi, C. (2008). *Bio-inspired artificial intelligence: Theories, methods, and technologies*. Cambridge: MIT Press.

Kannan, S., Slochanal, S., & Padhy, N. (2004). Application of particle swarm optimization technique and its variants to generation expansion problem. *Electric Power Systems Research, 70*(3), 203–210.

Kennedy, J., & Eberhart, R. (1995). A new optimizer using particle swarm theory. *Proceedings of the IEEE 6th International Symposium on Micro Machine and Human Science* (pp. 39–43). Nagoya, Japan.

Krink, T., Vesterstrom, J., & Riget, J. (2002). Particle swarm optimization with spatial particle extension. *Proceedings of the IEEE Congress on Evolutionary Computation, 2*, 1474–1479.

Lancaster, P., & Salkauskas, K. (1986). *Curve and surface fitting*. London: Academic Press.

Liu, H., Abraham, A., & Zhang, W. (2007). A fuzzy adaptive turbulent particle swarm optimisation. *International Journal of Innovative Computing and Applications, 1*(1), 39–47.

Lovbjerg, M., & Krink, T. (2002). Extending particle swarms with self-organized criticality. *Proceedings of the IEEE Congress on Evolutionary Computation* (pp. 1588–1593). Honolulu, HI, USA.

Lucey, S., & Matthews, I. (2006). Face refinement through a gradient descent alignment approach. *Proceedings of the HCSNet Workshop on Use of Vision in Human-Computer Interaction* (pp. 43–49). Australian Computer Society, Inc.

Miranda, V., & Fonseca, N. (2002). New evolutionary particle swarm algorithm (EPSO) applied to voltage/VAR control. *Proceedings of the 14th Power Systems Computation Conference* (pp. 1–6). Seville, Spain.

Momma, M., & Bennett, K. P. (2002). Pattern search method for model selection of support vector regression. *SDM*, *132*, 261–274.

Pires, E. J., Machado, J. A., Cunha, P. B., & Mendes, L. (2010). Particle swarm optimization with fractional-order velocity. *Journal on Nonlinear Dynamics, 61*(1–2), 295–301.

Premalatha, K., & Natarajan, A. M. (2009). Hybrid PSO and GA for global maximization. *International Journal of Open Problems in Computational Mathematics, 2*(4), 597–608.

Schaeffer, J., Lu, P., Szafron, D., & Lake, R. (1993). A re-examination of brute-force search. In *Proceedings of the AAAI Fall Symposium on Games: Planning and Learning* (pp. 51–58).

Secrest, B., & Lamont, G. (2003). Visualizing particle swarm optimization—Gaussian particle swarm optimization. *Proceedings of the IEEE Swarm Intelligence Symposium* (pp. 198–204). Indianapolis, Indiana, USA.

Shi, Y., & Eberhart, R. (2001). Fuzzy adaptive particle swarm optimization. *Proceedings of the IEEE Congress on Evolutionary Computation* (Vol. 1, pp. 101–106).

Tillett, J., Rao, T. M., Sahin, F., Rao, R., & Brockport, S. (2005). Darwinian particle swarm optimization. In B. Prasad (Ed.), *Proceedings of the 2nd Indian International Conference on Artificial Intelligence* (pp. 1474–1487). Pune, India.

Tsai, W. (2002). Social structure of "coopetition" within a multiunit organization: coordination, competition, and intraorganizational knowledge sharing. *Organization Science, 13*(2), 179–190.

Valle, Y. D., Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J. C., & Harley, R. (2008). Particle swarm optimization: Basic concepts, variants and applications in power systems. *IEEE Transactions on Evolutionary Computation*, *2*(2), 171–195.

Van den Bergh, F., & Engelbrecht, A. P. (2004). A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation, 8*(3), 225–239.

Zhang, W., & Xie, X. (2003). DEPSO: Hybrid particle swarm with differential evolution operator. *IEEE International Conference on Systems, Man and Cybernetics*, *4*, 3816–3821.

Chapter 2
Fractional-Order Darwinian PSO

Abstract As presented in Chap. 1, Darwinian particle swarm optimization (DPSO) presented by Tillett et al. (Darwinian particle swarm optimization, 2005) 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. Despite its superior performance when compared to its nonevolutionary counterpart, the DPSO also exhibits a key drawback: its computational complexity. This chapter proposes a method for controlling the convergence rate of the DPSO using fractional calculus (FC) concepts (Pires et al., Journal on Nonlinear Dynamics, 61(1–2), 295–301, 2010). The fractional-order optimization algorithm, denoted fractional-order Darwinian particle swarm optimization (FODPSO), is then tested using several well-known functions and the relationship between the fractional-order velocity and the convergence of the algorithm is observed.

Keywords FODPSO · Swarm intelligence · Optimization · Benchmark

2.1 Fractional Calculus

Fractional calculus (FC) has attracted the attention of several researchers (Sabatier et al. 2007), being applied in various scientific fields such as engineering, computational mathematics, and fluid mechanics, among others. FC can be considered as a generalization of integer-order calculus, thus accomplishing what integer-order calculus cannot. As a natural extension of the integer (i.e., classical) derivatives, fractional derivatives provide an excellent instrument for the description of memory and hereditary properties of processes.

The concept of the Grünwald–Letnikov fractional differential is presented by the following definition.

Definition 2.1 (Machado et al. 2010) Let Γ be the gamma function defined as

$$\Gamma(k) = (k-1)!. \tag{2.1}$$

© The Author(s) 2016

M. Couceiro and P. Ghamisi, *Fractional Order Darwinian Particle Swarm Optimization*, SpringerBriefs in Applied Sciences and Technology,
DOI 10.1007/978-3-319-19635-0_2

The signal $D^{\alpha}[x(t)]$ given by

$$D^{\alpha}[x(t)] = \lim_{h \to 0} \left[\frac{1}{h^{\alpha}} \sum_{k=0}^{+\infty} \frac{(-1)^k \Gamma(\alpha+1) x(t-kh)}{\Gamma(k+1)\Gamma(\alpha-k+1)} \right], \tag{2.2}$$

is said to be the Grünwald–Letnikov fractional derivative of order α, $\alpha \in \mathbb{C}$, of a generic signal $x(t)$.

An important property revealed by Eq. (2.2) is that although an integer-order derivative just implies a finite series, the fractional-order derivative requires an infinite number of terms. Therefore, integer derivatives are "local" operators whereas fractional derivatives have, implicitly, a "memory" of all past events. However, the influence of past events decreases over time.

The formulation in (2.2) inspires a discrete time calculation presented by the following definition.

Definition 2.2 (Machado et al. 2010) The signal $D^{\alpha}[x[t]]$ given by

$$D^{\alpha}[x[t]] = \frac{1}{T^{\alpha}} \sum_{k=0}^{r} \frac{(-1)^k \Gamma[\alpha+1] x[t-kT]}{\Gamma[k+1]\Gamma[\alpha-k+1]}, \tag{2.3}$$

where T is the sampling period and r is the truncation order, is the approximate discrete time Grünwald–Letnikov fractional difference of order α, $\alpha \in \mathbb{C}$, of the generic discrete signal $x[t]$.

The series presented in Eq. (2.3) can be implemented by a rational fraction expansion that leads to a superior compromise in what concerns the number of terms versus the quality of the approximation. Nevertheless, because this study focuses on the convergence of robots toward a given solution considering past events, the simple series approximation is adopted.

That being said, it is possible to extend an integer discrete difference, that is, classical discrete difference, to a fractional-order one, using the following definition.

Definition 2.3 (Ostalczyk 2009) The classical integer "direct" discrete difference of signal $x[t]$ is defined:

$$\Delta^d x[t] = \begin{cases} x[t], & d = 0 \\ x[t] - x[t-1], & d = 1 \\ \Delta^{d-1} x[t] - \Delta^{d-1} x[t-1], & d > 1 \end{cases} \tag{2.4}$$

where $d \in \mathbb{N}_0$ is the order of the integer discrete difference. Hence, one can extend the integer-order $\Delta^d x[t]$ assuming that the fractional discrete difference satisfies the following inequalities,

$$d - 1 < \alpha < d. \tag{2.5}$$

The features inherent to fractional calculus make this mathematical tool well suited to describe many phenomena, such as irreversibility and chaos, because of its inherent memory property. In this line of thought, the dynamic phenomena of particles' trajectories configure a case where fractional calculus tools fit adequately.

2.2 FODPSO

Considering the inertial influence of Eq. (1.1) as $w = 1$, for a specific swarm s, one would obtain:

$$v_n^s[t+1] = v_n^s[t] + \sum_{i=1}^{2} \rho_i r_i \left(\chi_{\text{in}}^s[t] - x_n^s[t]\right). \tag{2.6}$$

This expression can be rewritten:

$$v_n^s[t+1] - v_n^s[t] = \sum_{i=1}^{2} \rho_i r_i \left(\chi_{\text{in}}^s[t] - x_n^s[t]\right). \tag{2.7}$$

Hence, $v_n^s[t+1] - v_n^s[t]$ corresponds to the discrete version of the fractional difference of order $\alpha = 1$, that is, the first-order integer difference $\Delta^d v_n^s[t+1]$. Assuming $T = 1$ and based on Definition 2.2, yields the equation:

$$D^\alpha \left[v_n^s[t+1]\right] = \sum_{i=1}^{2} \rho_i r_i \left(\chi_{\text{in}}^s[t] - x_n^s[t]\right). \tag{2.8}$$

Based on the FC concept and Definition 2.3, the order of the velocity derivative can be generalized to a real number $0 < \alpha < 1$, thus leading to a smoother variation and a longer memory effect. Therefore, considering the discrete-time fractional differential presented in Definition 2.2, one can rewrite Eq. (2.8) as

$$\begin{aligned} v_n^s[t+1] = &- \sum_{k=1}^{r} \frac{(-1)^k \Gamma[\alpha+1] v_n^s[t+1-kT]}{\Gamma[k+1]\Gamma[\alpha-k+1]} \\ &+ \sum_{i=1}^{2} \rho_i r_i \left(\chi_{\text{in}}^s[t] - x_n^s[t]\right). \end{aligned} \tag{2.9}$$

The DPSO is, therefore, a particular case of the FODPSO for $\alpha = 1$ (without "memory").

2.2.1 Benefits

The FODPSO is, in simple terms, the same as having multiple PSOs, wherein particles strive to find the best solution for their own "survival", with the perk of intrinsically having a memory of past decisions. This new architecture handles the first drawback pointed out for the traditional PSO: the premature convergence of a swarm. The FODPSO, as does the traditional DPSO (Tillett et al. 2005), discards swarms that prematurely converge toward a solution that may, or may not, be optimal. At the same time, it fosters the creation of new swarms formed by particles that may "genetically" share some of the knowledge already retrieved by other particles. Additionally to this, each FODPSO particle is considerably "smarter" than PSO and DPSO particles due to the fractional-order extension that improves the balance between exploration and exploitation (see Sect. 2.3). This allows running the FODPSO algorithm with a smaller population when compared to the DPSO algorithm, thus reducing the computational complexity, and still expecting the same end result. This book repeatedly compares the FODPSO with alternatives, including the PSO and DPSO, and clearly depicts its superiority in every aspect.

2.2.2 Drawbacks

Compared to the other PSO-based alternatives, from which the traditional PSO and the DPSO algorithms are used as references, the FODPSO presents two drawbacks: (i) its memory complexity and (ii) the addition of a new coefficient α.

As opposed to most PSO-based approaches that only require memorizing the previous iteration, as one can observe from Eq. (2.9), computing a new velocity at time $t + 1$ requires memorizing the previous r iterations. Therefore, one needs to ensure a proper r, being large enough to ensure an improved convergence of particles toward the solution when compared to the alternatives, while at the same time small enough not to increase the cost of the algorithm significantly.

Preliminary experimental tests on the algorithm presented similar results for ≥ 4 (Couceiro et al. 2012). Furthermore, the computational requirements increase linearly with r; that is, the FODPSO presents an $\mathcal{O}(r)$ memory complexity. Hence, using only the first $r = 4$ terms of differential derivative given by (2.3), Eq. (2.9) can be rewritten as (2.10):

$$\begin{aligned} v_n^s[t+1] =& \alpha v_n^s[t] + \frac{1}{2}\alpha(1-\alpha)v_n^s[t-1] \\ &+ \frac{1}{6}\alpha(1-\alpha)(2-\alpha)v_n^s[t-2] \\ &+ \frac{1}{24}\alpha(1-\alpha)(2-\alpha)(3-\alpha)v_n^s[t-3] \\ &+ \sum_{i=1}^{2} \rho_i r_i \left(\chi_{in}^s[t] - x_n^s[t]\right) \end{aligned} \tag{2.10}$$

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

As described in Yasuda et al. (2008) and Wakasa et al. (2010), a swarm behavior can be divided into two activities: (i) exploitation and (ii) exploration. The first one is related to 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 to 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 (2001), 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 whereas exploitation is improved using a small inertia weight. Because the FODPSO 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 FODPSO is experimentally evaluated in the next section using Eq. (2.10) for all particles in all swarms.

2.3 Benchmarking Functions

This section presents experimental results of the proposed FODPSO. In order to compare this approach with the Pires et al. approach (Pires et al. 2010), the same test functions and parameters are considered as depicted in Table 2.1. Table 2.1 also shows the specific parameters of the FODPSO 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. 2.1, 2.2, 2.3, 2.4 and 2.5, the results can be seen for the adopted optimization functions $f_j, j = \{1, \ldots, 5\}$.

Experimental results show that the convergence of the algorithm depends upon the fractional order α. However, contrary to the FOPSO presented in Pires et al. (2010), the Darwinian algorithm easily avoids being stuck in local solutions

Table 2.1 Parameters of the algorithm and optimization functions

	Min	Initial	Max
Number of simulations	–	201	–
Number of iterations	–	200	–
Coefficients $\rho_1 = \rho_2$	–	0.8	–
Swarm population	3	4	5
Number of swarms	1	2	3
Stagnancy threshold	–	10	–
Optimization functions f_j (Van Den Bergh and Engelbrecht 2006)	Bohachevsky 1		
	Colville		
	Drop wave		
	Easom		
	Rastrigin		

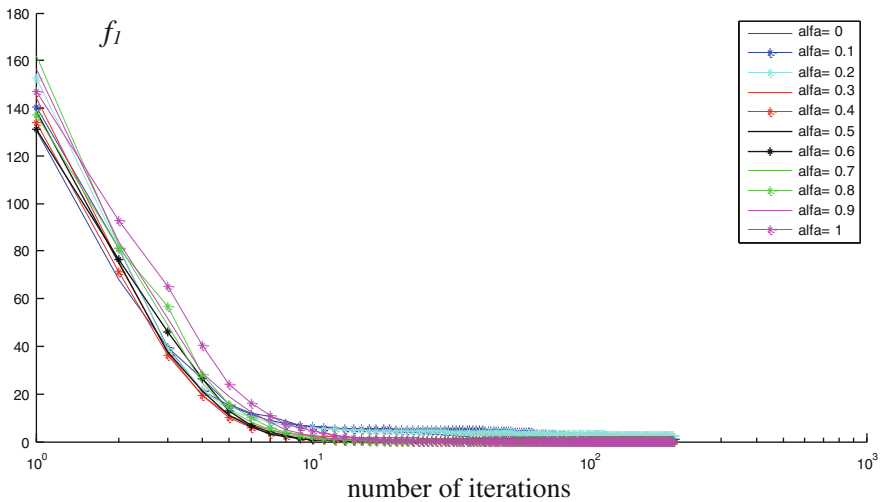

Fig. 2.1 Evolution of the Bohachevsky 1 function changing α

independently of the value of α (because 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 evaluate the FODPSO further, let us then systematically adjust the fractional coefficient α between 0.5 and 0.8, according to the expression:

$$\alpha(t) = 0.8 - 0.3\frac{t}{200}. \tag{2.11}$$

Once again, the median of the fitness evolution of the best global particle is taken as the system output. In Fig. 2.6, the results can be seen for the adopted optimization

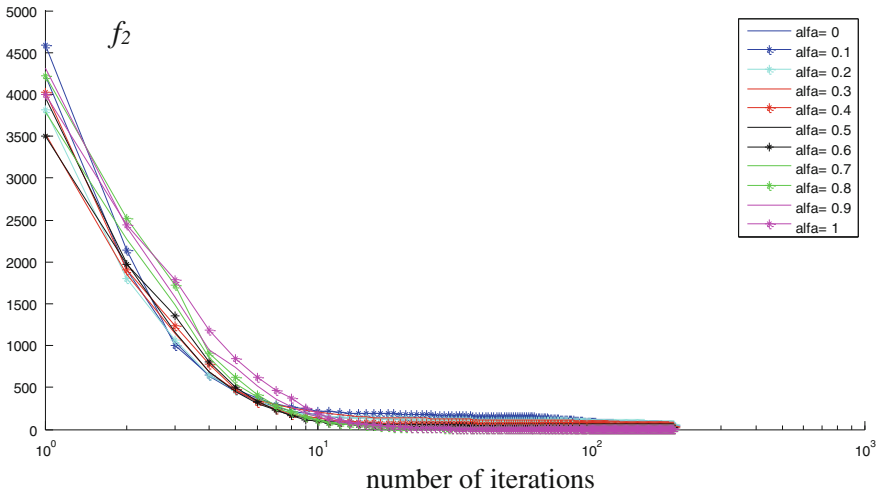

Fig. 2.2 Evolution of the Colville function changing α

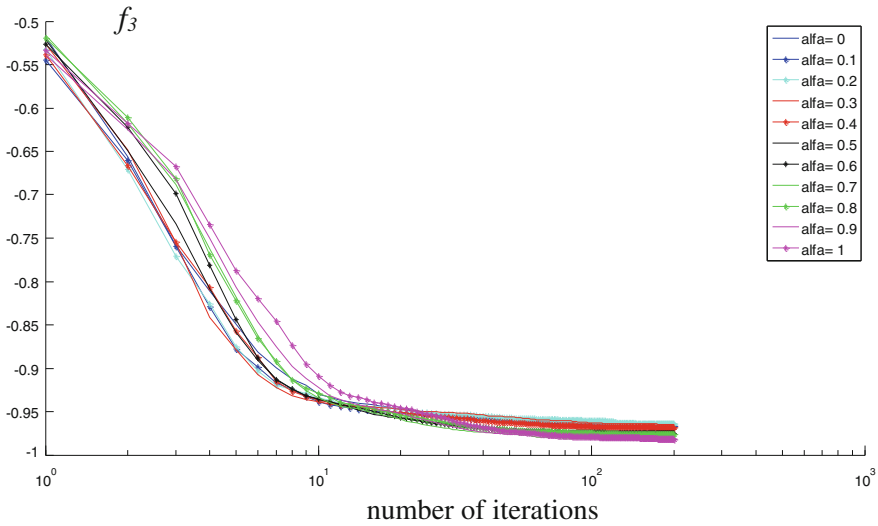

Fig. 2.3 Evolution of the drop wave function changing α

functions $f_j, j = \{1, \ldots, 5\}$, while comparing the FODPSO with the FOPSO proposed by Pires et al. (2010) using Eq. (2.11). Observing Fig. 2.6, one can conclude that, despite both FOPSO and FODPSO revealing a similar behavior, the combination of FC and Darwin's principles contributes to an improved convergence dynamics.

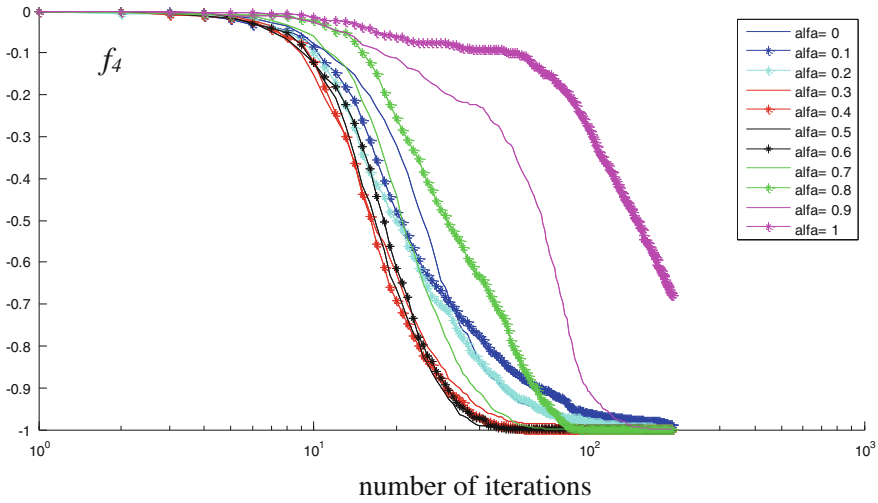

Fig. 2.4 Evolution of the Easom function changing α

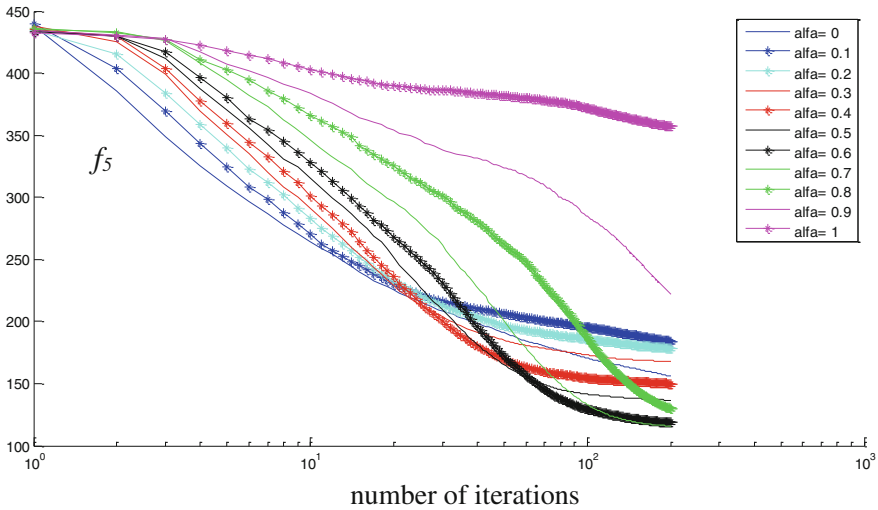

Fig. 2.5 Evolution of the Rastrigin function changing α

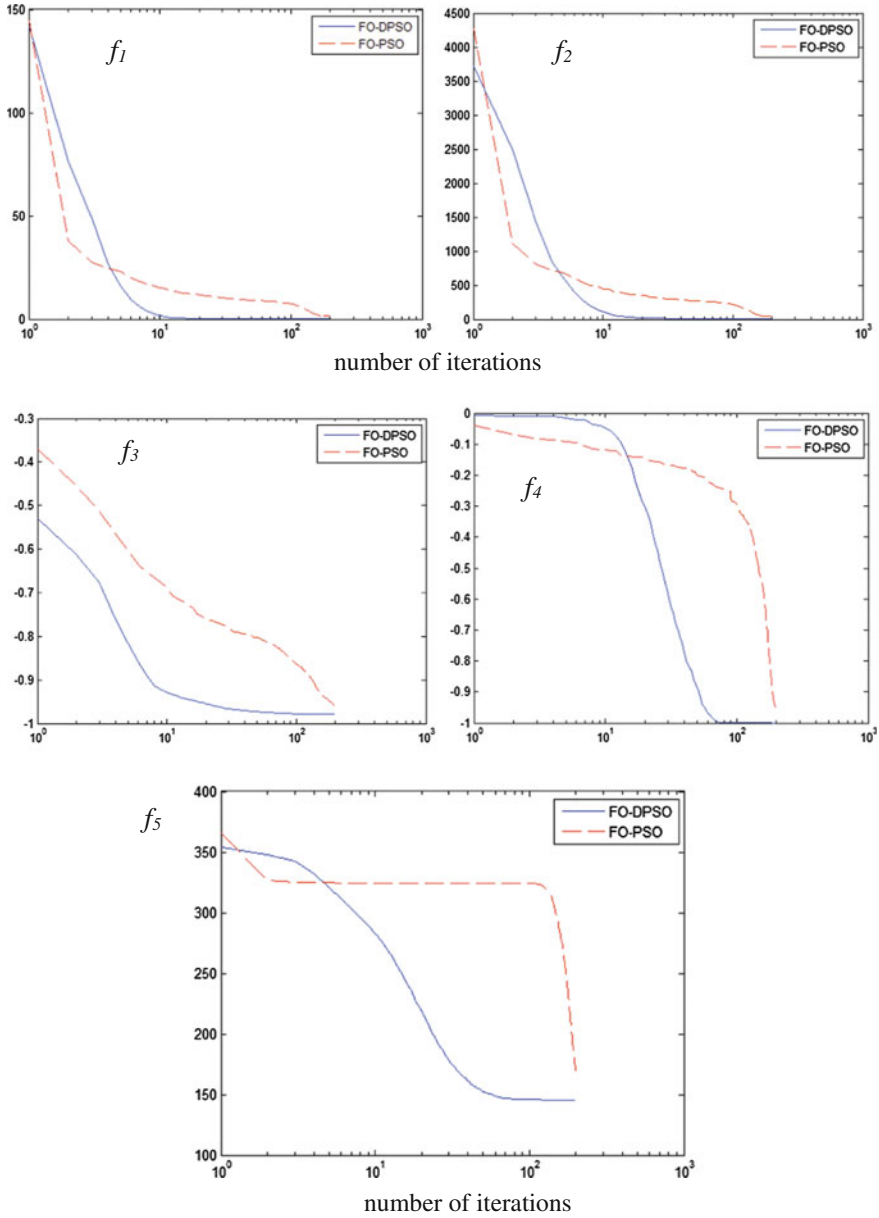

Fig. 2.6 Evolution of the fitness function, with variable α for FOPSO and FODPSO

2.4 Summary

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 chapter presented a new optimization algorithm based on the Darwinian particle swarm optimization (DPSO) using the concept of the fractional derivative to control the convergence rate.

Experimental results show that, although the speed of convergence of the fractional order DPSO (FODPSO) depends on the fractional order α, the herein presented algorithm outperforms the traditional DPSO and PSO¸ as well as the FOPSO previously presented in the literature.

References

Couceiro, M. S., Martins, F. M., Rocha, R. P., & Ferreira, N. M. (2012). Introducing the fractional order robotic Darwinian PSO. *Proceedings of the 9th International Conference on Mathematical Problems in Engineering, Aerospace and Sciences (ICNPAA2012)* (pp. 242–251). Vienna, Austria.

Machado, J. T., Silva, M. F., Barbosa, R. S., Jesus, I. S., Reis, C. M., Marcos, M. G., et al. (2010). Some applications of fractional calculus in engineering, Hindawi Publishing Corporation *Mathematical Problems in Engineering*, *1*(34), 174–183.

Ostalczyk, P. W. (2009). A note on the Grünwald-Letnikov fractional-order backward-difference. *Physica Scripta, 136*, 014036.

Pires, E. J., Machado, J. A., Cunha, P. B., & Mendes, L. (2010). Particle swarm optimization with fractional-order velocity. *Journal on Nonlinear Dynamics, 61*(1–2), 295–301.

Sabatier, J., Agrawal, O. P., & Machado, J. A. (2007). *Advances in fractional calculus—theoretical developments and applications in physics and engineering*. Berlin: Springer.

Shi, Y., & Eberhart, R. (2001). Fuzzy adaptive particle swarm optimization. *Proceedings of the IEEE Congress on Evolutionary Computation, 1*, 101–106.

Tillett, J., Rao, T. M., Sahin, F., Rao, R., & Brockport, S. (2005). Darwinian particle swarm optimization. In B. Prasad (Ed.), *Proceedings of the 2nd Indian International Conference on Artificial Intelligence* (pp. 1474–1487). Pune, India.

Van Den Bergh, F., & Engelbrecht, A. P. (2006). *A study of particle swarm optimization particle trajectories, 176*(8), 937–971.

Wakasa, Y., Tanaka, K., & Nishimura, Y. (2010). Control-theoretic analysis of exploitation and exploration of the PSO algorithm. *IEEE International Symposium on Computer-Aided Control System Design, IEEE Multi-Conference on Systems and Control* (pp. 1807–1812). Yokohama, Japan.

Yasuda, K., Iwasaki, N., Ueno, G., & Aiyoshi, E. (2008). Particle swarm optimization: A numerical stability analysis and parameter adjustment based on swarm activity. *IEEJ Transactions on Electrical and Electronic Engineering, Wiley InterScience, 3*, 642–659.

Chapter 3
Case Study I: Curve Fitting

Abstract Curve fitting may be considered as one of the most traditional problems where one may benefit from optimization methods. However, its real applicability goes way beyond solving theoretical mathematical problems. Curve fitting may be used, for instance, to infer values of a function where no data are available, or obtain a mathematical function describing a physical, chemical, or biological state or process (Bishop, Pattern Recognition and Machine Learning, 2006). In this chapter, we explore the performance of the FODPSO side by side with four other alternatives, including the traditional PSO, in order to solve the same problem: find the set of parameters of a mathematical function that has the best fit to a series of data points representing the trajectory of a golf club. This is a rather interesting curve-fitting problem as it represents a real case study wherein one wants to obtain a mathematical representation of a given putting execution in order to compare process variables among different golfers (Couceiro et al. Proceedings of WACI-Workshop Applications of Computational Intelligence, 2010). In other words, the optimization algorithms are employed to obtain a kinematical analysis and a characterization of each golfer's putting technique.

Keywords FODPSO · Swarm intelligence · Curve fitting

3.1 Traditional Optimization Methods

Five different estimation techniques were studied, applied, and compared in a curve-fitting problem in this chapter. One popular first-order optimization algorithm used in many engineering related works is gradient descent. In this method, the search is carried through proportional steps in the direction of the negative of the gradient, or the approximate gradient, of the function at the current point in order to find a local minimum. It has been applied in the literature, for example, for face alignment in computer recognition systems (Lucey and Matthews 2006).

Another vastly used method is pattern search, which is a similar approach to gradient descent. However, it does not compute the gradient, meaning that it can be

© The Author(s) 2016

M. Couceiro and P. Ghamisi, *Fractional Order Darwinian Particle Swarm Optimization*, SpringerBriefs in Applied Sciences and Technology,
DOI 10.1007/978-3-319-19635-0_3

used with nondifferentiable functions. In this case, a descent search direction is produced by varying the parameters of the problem with different step sizes, aiming to obtain a fit to the experimental data. It has been successfully applied in model selection of support vector machines and for transformation function search in automatic image registration among others (Momma and Bennett 2002). Both the gradient descent and the pattern search methods are more suitable for low-dimensional optimization problems.

In addition, the simplex method was also studied. This is an optimization method to solve linear programming problems numerically by searching optimal solutions in the vertices of the admissible region of the space, considering all constraints, and iteratively improving the objective function. It is perhaps the most popular optimization algorithm for linear problems with low dimensions, being applied previously in contexts such as particle accelerator control (Emery 2003). A nonlinear heuristic version of the simplex method called the downhill simplex or Nelder–Mead algorithm was utilized. This algorithm is more suitable for minimizing objective functions in a many-dimensional space, such as the problem presented in this chapter. Among previous works, Miura et al. (2006) used a modified version of the Nelder–Mead algorithm to minimize an objective function related to the displacement of the joints of a robotic manipulator by visual servoing (Miura 2006).

All the methods referred to are part of the traditional approaches adopted to solve the most traditional problems, such as curve fitting. One may find them in most optimization or curve-fitting software tools, including MATLAB®'s Optimization Toolbox. Yet, the performance of such methods starts dropping, especially in terms of computational complexity, as the number of dimensions of the problem, in this case the representative mathematical model, increase.

In this chapter, we compare the performance of these three methods against the traditional PSO (Chap. 1) and the FODPSO (Chap. 2), to find the optimal set of parameters defining a mathematical model that may more accurately represent the trajectory of a golf club during the execution of putting. To start with, let us first present the mathematical function used to represent the trajectory of the golf putter.

3.2 Mathematical Modeling of the Putting

Mathematically modeling a given phenomenon may be considered as the ultimate way to understand it fully. One of the oldest attempts to model a considerably complex phenomenon was carried out by Lorenz, with the help of physicists Howard and Markus, who formulated the mathematics behind a water mill's rotation. Through experimentation, Lorenz was able to conclude that the mill is very susceptible to initial conditions and, therefore, he classified it as a chaotic system (Gleick 1987).

This was an important step in mathematical modeling and any mathematics applied to the study of human movement for that matter. Human movement is inherently a biological process and, as such, hardly periodic (Harbourne and

Stergiou 2009). In this section we focus on presenting a mathematical model that may completely represent the horizontal trajectory of golf putting. Note that other process variables around golf putting (e.g., vertical trajectory, linear acceleration, and others) could as well be represented by the exact same mathematical model. However, and for the sake of simplicity, we focus on that one.

To build a mathematically representative model of the putter's horizontal trajectory, several putting trials were recorded using a frontal camera and a simple color-detection algorithm. Please refer to our previous work for a detailed description of the detection method (Couceiro 2010). Figure 3.1 shows a sequence recorded using the camera.

The chart presented in Fig. 3.2 shows an example of a point cloud that represents the detected position, in the horizontal plane, of a golf club during putting execution of an expert subject.

As is clear in Fig. 3.2, the detection algorithm's output has some data lacking. This happens mostly due to lighting conditions, in which the detection algorithm fails. In order to classify the point cloud, curve-fitting techniques are then required to fit the acquired points.

Fig. 3.1 Sequence depicting the putter's movement

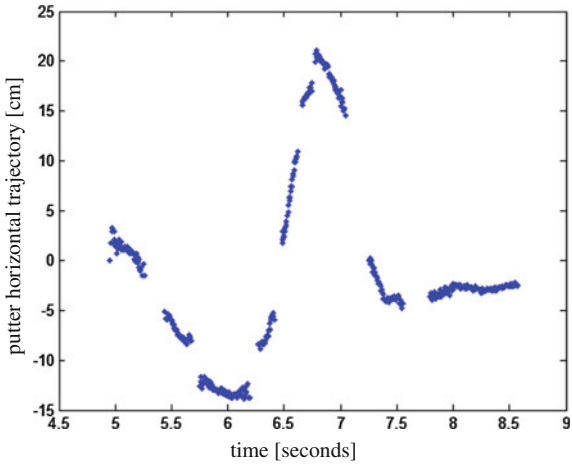

Fig. 3.2 Example of a point cloud obtained with a vision-based detection algorithm

In spite of this, by analyzing the typical shape of putting trials from Fig. 3.2, it is clear that to model the putter's horizontal position in time, one ought to use a sinusoidal-like function. Nevertheless, a function composed of only one sinusoid was not precise enough to describe the movement, as is clear in f_1 of Fig. 3.3, which results, in this case, in a mean squared error (MSE) of 2.6568 units. This happens because the amplitude, angular frequency, and phase of the descending half-wave, which correspond to the player's backswing and downswing, are usually different than the ascending half-wave, which corresponds to the ball's impact and follow-through (Dias and Couceiro 2015). These disparities could not be represented using solely one sinusoid wave. Therefore, to obtain a more precise model a sum of sinusoid waves was employed. However, a compromise between precision and complexity of the problem had to be assumed, because each sinusoid adds three more dimensions to the estimation problem (amplitude, angular frequency, and phase of the corresponding sine wave). In order not to let the complexity of the problem grow inappropriately, a function composed of the sum of three sinusoids was used (f_3 of Fig. 3.3), due to its precision, with a MSE of 0.6926, when compared to using solely a sum of two sinusoids, with a MSE of 0.7124 (f_2 of Fig. 3.3).

Thus, having the estimation function defined as a sum of three sine waves, each of the three parameters of each wave needs to be estimated, resulting in a nine-dimension estimation problem, which attempts to minimize the mean-squared estimation error for every experiment, in order to obtain an accurate mathematical function that describes the horizontal position of the golf club during putting execution (Couceiro et al. 2013).

Overall, in the end, we have an optimization method that shall find the optimal solution that may fit some data points, which is reflected upon a set of parameters evolving over time. For the specific case of golf putting, and considering the mathematical model chosen (see Fig. 3.3), the mathematical function at stake is:

$$f(t) = a_1 \sin(b_1 t + c_1) + a_2 \sin(b_2 t + c_2) + a_3 \sin(b_3 t + c_3), \tag{3.1}$$

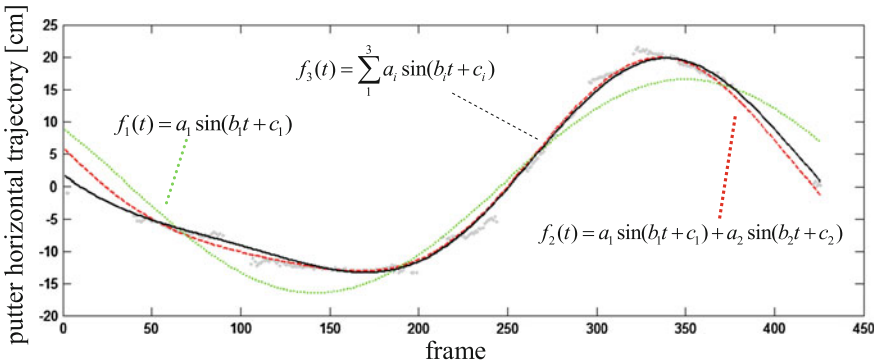

Fig. 3.3 Fitting sinusoidal functions to a point cloud, representing the position of a golf club during putting execution

being represented by nine parameters (a_i, b_i, c_i), $i = \{1, 2, 3\}$, which yields a given "solution". Such a solution can be represented by the MSE, as previously considered, or any other measure of error, such as the sum of squared errors (SSE).

The next section evaluates the FODPSO algorithm over the four alternatives previously described under this specific optimization problem.

3.3 Experimental Results

A first analysis of all five optimization methods previously mentioned was conducted, to verify the one better suited for the problem in hand. Three distinct trials of three different subjects were used in this first study. The goal was to check and compare the performance of each of the optimization techniques.

All algorithms were run under the same conditions, with the same restrictions regarding the parameters of the model from Eq. (3.1), namely $a_i \in [0, 35]$, $i = \{1, 2, 3\}$, $b_1 \in [0, 0.03]$, $b_i \in [0, 0.05], i = \{1, 2\}$, $c_i \in [-\pi, \pi], i = \{1, 2, 3\}$, and with random initialization of all variables. The stopping criterion was defined as an MSE of 0.75 or a maximum running time of 1,000 s, whichever comes first. Table 3.1 depicts the parameters of both PSO and FODPSO algorithms.

These parameters guaranteed the fine results below. Note that the velocity limits of ±0.5 were imposed in both cases to prevent explosion of particle velocities and each particle's position is verified in every iteration to guarantee that it moves inside the boundaries imposed by the constraints of all dimensions of the problem.

Figures 3.4, 3.5 and 3.6, as well as Table 3.2, clearly show the superior performance obtained by the FODPSO algorithm for all three cases, followed by PSO, which was only slightly inferior in the second dataset, when compared to pattern search and achieved second best results in the first and third dataset. The other three

Table 3.1 Parameters of the PSO and FODPSO for the curve-fitting problem

Parameter	PSO	FODPSO
Number of particles N	500	20
Cognitive coefficient ρ_1	1.2	1.2
Social coefficient ρ_2	0.8	0.8
Inertial weight w	0.8	–
Maximum velocity Δv	±0.5	±0.5
Minimum number of particles in swarm $N_{\min}$	–	15
Maximum number of particles in swarm $N_{\max}$	–	100
Initial number of swarms N^s	–	10
Minimum number of swarms $N^s_{\min}$	–	1
Maximum number of swarms $N^s_{\max}$	–	100
Stagnancy threshold I_{kill}	–	10
Fractional coefficient α	–	0.6

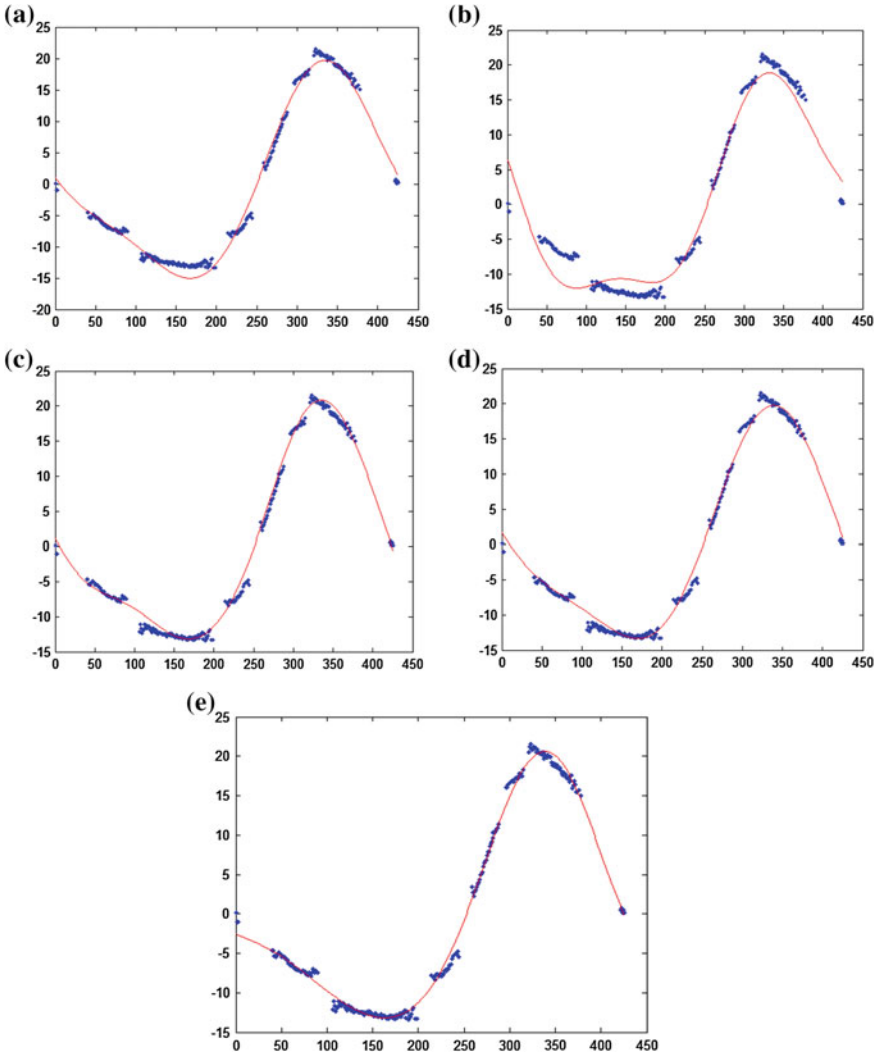

Fig. 3.4 Results for the first dataset, frame versus putter trajectory [cm]: **a** gradient descent; **b** pattern search; **c** downhill simplex; **d** PSO; **e** FODPSO

algorithms generally obtained inferior results. These results depict that population-based and evolutionary algorithms are more suited for complex multi-dimensional optimization problems. However, both the PSO and the FODPSO need some predefined parameters, as opposed to gradient descent (GD), pattern search (PS), and downhill simplex (DS).

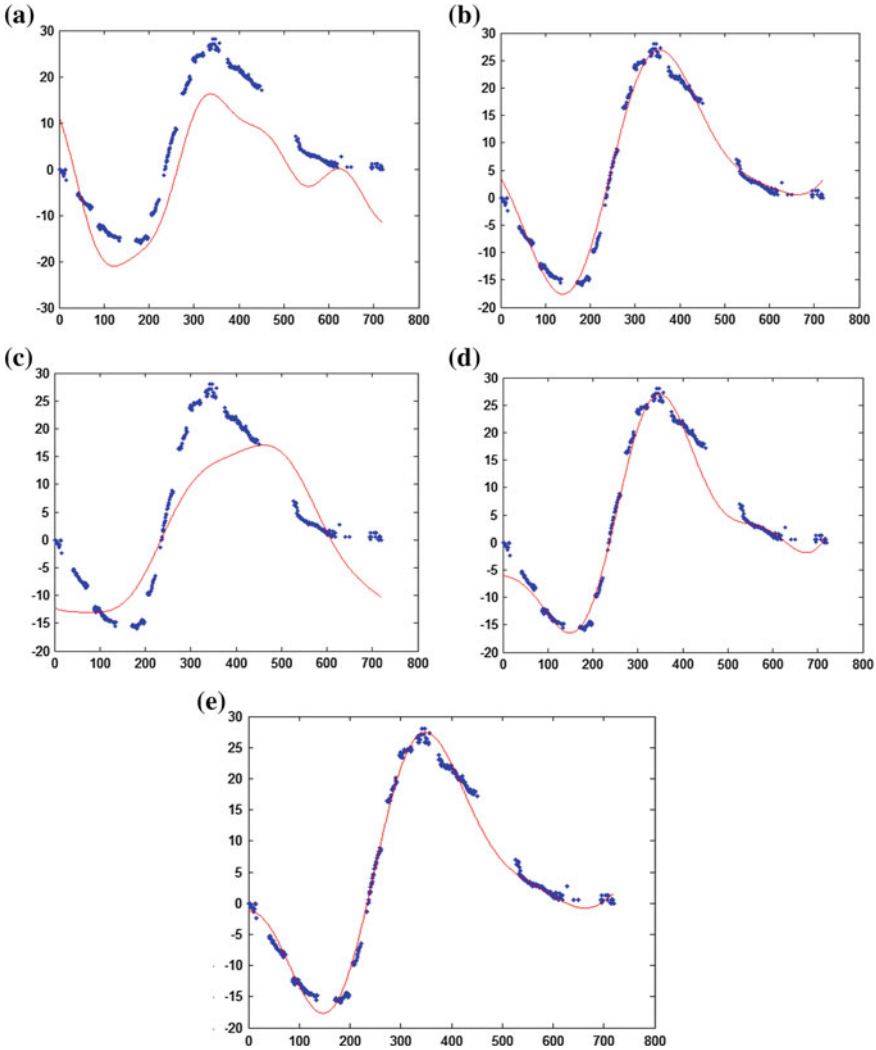

Fig. 3.5 Results for the second dataset, frame versus putter trajectory [cm]: **a** gradient descent; **b** pattern search; **c** downhill simplex; **d** PSO; **e** FODPSO

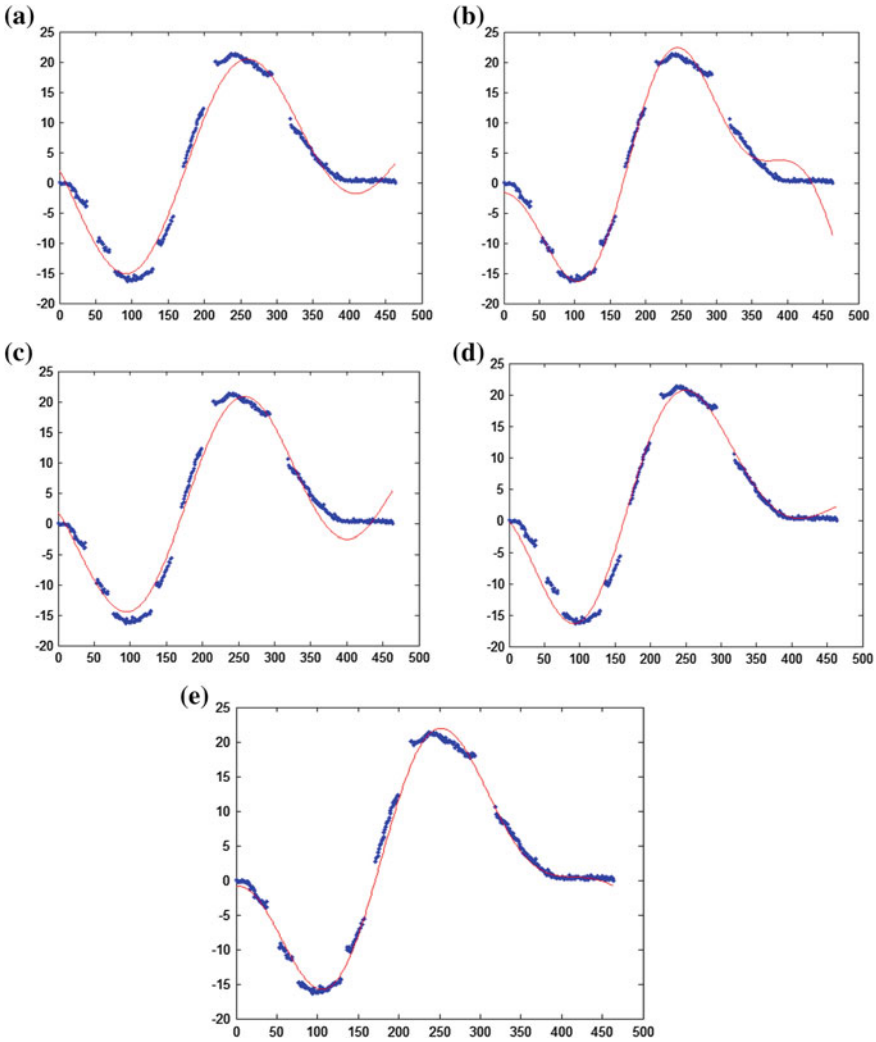

Fig. 3.6 Results for the third dataset, frame versus putter trajectory [cm]: **a** gradient descent; **b** pattern search; **c** downhill simplex; **d** PSO; **e** FODPSO

Table 3.2 Comparative results for all three datasets

	First dataset		Second dataset		Third dataset	
	Time (s)	MSE	Time (s)	MSE	Time (s)	MSE
GD	1,000	0.9270	1,000	7.2550	1,000	1.4989
PS	1,000	1.7989	1,000	1.4315	1,000	1.4087
DS	952	0.7147	1,000	5.7457	1,000	1.6770
PSO	312	0.6926	1,000	1.4391	1,000	0.9789
FODPSO	102	0.5658	1,000	1.0001	8	0.7368

3.4 Summary

This chapter compared the FODPSO on a curve-fitting real problem, side by side with other traditional optimization methods, including the PSO, gradient descent, pattern search, and simplex. Results confirmed the superior performance of the FODPSO method.

Nevertheless, generally speaking, at this point, both PSO and FODPSO algorithms have shown excellent search abilities for the considered curve-fitting problem. Yet, as occurring with other algorithms, they may lose their efficacy when applied to large and complex problems, for example, problem instances with high dimensionality. The next chapter presents a methodology to segment hyperspectral images using the FODPSO algorithm.

References

Bishop, C. M. (2006). *Pattern recognition and machine learning* (Vol. 1). New York: Springer.

Couceiro, M. S., Luz, J. M., Figueiredo, C. M., Ferreira, N. M., & Dias, G. (2010). Parameter estimation for a mathematical model of the golf putting. In V. M. Marques, C. S. Pereira, & A. Madureira (Eds.), *Proceedings of WACI-Workshop Applications of Computational Intelligence* (pp. 1–8). Coimbra, Portugal: ISEC—IPC.

Couceiro, M. S., Portugal, D., Gonçalves, N., Rocha, R., Luz, J. M., Figueiredo, C. M., & Dias, G. (2013). A methodology for detection and estimation in the analysis of the golf putting. *Pattern Analysis and Applications, 16*(3), 459–474.

Dias, G., & Couceiro, M. S. (2015). *The science of golf putting—a complete guide for researchers, players and coaches* (SpringerBriefs in Applied Sciences and Technology ed.). Springer.

Emery, L., Borland, M., & Shang, H. (2003). Use of a general-purpose optimization module in accelerator control. in *Proceedings of the Particle Accelerator Conference (PAC), IEEE* (Vol. 4), pp. 2330–2332.

Gleick, J. (1987). *Chaos: Making a new science.* Viking Penguin.

Harbourne, R. T., & Stergiou, N. (2009). Movement variability and the use of nonlinear tools: principles to guide physical therapist practice. *Journal of Neurologic Physical Therapy, 89*(3), 267–282.

Lucey, S., & Matthews, I. (2006). Face refinement through a gradient descent alignment approach. in *Proceedings of the HCSNet Workshop on Use of Vision in Human-Computer Interaction.* Australian Computer Society Inc, pp. 43–49.

Miura, K., Hashimoto, K., Inooka, H., Gangloff, J. A., & Mathelin, M. F. (2006). Model-less visual servoing using modified simplex optimization. *Artificial Life and Robotics, 10*(2), 131–135.

Momma, M., Bennett, K. P. (2002). Pattern search method for model selection of support vector regression. (pp. 261–274), *SDM.*

Chapter 4
Case Study II: Image Segmentation

Abstract Image segmentation has been investigated as a vital task in a wide variety of applications including (but not limited to): document image analysis for extraction of printed characters; map processing in order to find lines, legends, and characters; topological feature extraction for extraction of geographical information; remote sensing image analysis; and quality inspection of materials where defective parts must be delineated among many other applications (Ghamisi et al. IEEE International Geoscience Remote Sensing Symposium (IGARSS) 2012). In addition, for the purpose of image classification and object detection, the use of an efficient segmentation technique plays a key role. This chapter is devoted to one of the important application of FODPSO, which is related to introducing a novel thresholding-based segmentation method based on FODPSO for determining the $n-1$ optimal n-level threshold on a given image. This approach has been widely used in the literature for the segmentation of benchmark images, remote sensing data, and medical images. This chapter first, elaborates the mathematical formulation of thresholding-based image segmentation. Then, some well-known thresholding segmentation techniques such as genetic algorithm (GA)-, bacteria foraging (BF)-, PSO-, DPSO-, and FODPSO-based thresholding-based segmentation techniques are compared in terms of accuracy and CPU processing time. Experimental results demonstrate the efficiency of the FODPSO-based segmentation method compared to other optimization-based segmentation methods when considering a number of different measures.

Keywords FODPSO · Swarm intelligence · Image segmentation · Hyperspectral imaging

4.1 Thresholding-Based Image Segmentation

Image segmentation is regarded as the process of partitioning a digital image into multiple regions or objects. In more detail, in image segmentation, a label is assigned to each pixel in the image such that pixels with the same label share certain

© The Author(s) 2016

M. Couceiro and P. Ghamisi, *Fractional Order Darwinian Particle Swarm Optimization*, SpringerBriefs in Applied Sciences and Technology,
DOI 10.1007/978-3-319-19635-0_4

visual characteristics (Sezgin and Sankur 2004). Undoubtedly, these objects provide more information than individual pixels because the interpretation of images based on objects is more meaningful than that based on individual pixels. Image segmentation has been widely used as an important fundamental task in the analysis, interpretation, and understanding of images, consequently being widely used for further image processing purposes such as classification and object recognition (Ghamisi et al. 2014).

From one point of view, image segmentation can be split into four different types including texture-analysis–based methods, histogram-thresholding–based methods, clustering-based methods, and region-based split and merging methods (Brink 1995). In Kulkarni and Venayagamoorthy (2010), it was mentioned that one of the most common methods for image segmentation is the thresholding method, which is commonly used for segmentation of an image into two or more regions (objects).

The problem of bilevel thresholding can be reduced to an optimization problem in order to search for the threshold t that maximizes the σ_B^2 (between − class variance) and minimizes σ_W^2 (within − class variance) (Kulkarni and Venayagamoorthy 2010). For two-level thresholding, the problem is solved by finding the value T^* that satisfies max $\left(\sigma_B^2\left(T^*\right)\right)$, where $0 \leq T^* < L$ and L is the maximum intensity value. This problem could be extended to n-level thresholding through satisfying max $\sigma_B^2(T_1^*, T_2^*, \ldots, T_{n-1}^*)$ that $0 \leq T_1^* < T_2^* <, \ldots, < T_{n-1}^* < L$. Traditionally, in order to find the optimal set of thresholds, exhaustive search techniques have been taken into account. Among all exhaustive search methods, the one based on the Otsu criterion has gained a lot of attention from researchers (Otsu 1979). That approach is easy to implement, but it has the disadvantage of being computationally expensive (Kulkarni and Venayagamoorthy 2010). Exhaustive search for $n-1$ optimal thresholds involves evaluations of fitness of $n(L-n+1)^{n-1}$ combinations of thresholds (Kulkarni and Venayagamoorthy 2010) so this method is not suitable from a computational cost point of view. In other words, when the number of demanded thresholds increases, the computational complexity of the exhaustive techniques becomes unwieldy. The task of determining $n - 1$ optimal thresholds for n-level image thresholding can be formulated as a multidimensional optimization problem. To solve such a problem, several biologically inspired algorithms have been introduced in image segmentation (e.g., Fogel 2000; Kulkarni and Venayagamoorthy 2010; Ghamisi 2011). Bioinspired algorithms have been used in situations where conventional optimization techniques cannot find a satisfactory solution or they take too much time to find it (e.g., when the function to be optimized is discontinuous, nondifferentiable, and/or presents too many nonlinearly related parameters (Floreano and Mattiussi 2008)).

4.1.1 Problem Formulation

Let the term L be intensity levels in each component, for example, three color components for RGB images of a given image and these levels are in the range {0, 1, 2, …, $L - 1$}. For example, for an 8-bit image, L is equal to 256. Then one can define:

$$p_i^C = \frac{h_i^C}{N}, \qquad \sum_{\substack{i=1 \\ C=\{R,G,B\}}}^{N} p_i^C = 1, \tag{4.1}$$

where i represents a specific intensity level; that is, $0 \le i \le L - 1$. The term C represents the component of the image; for example, $C = \{R, G, B\}$ for RGB images and N represents the total number of pixels in the image and h_i^C denotes the number of pixels for the corresponding intensity level i in the component C. In this manner, h_i^C represents an image histogram for each component C, which can be normalized and regarded as the probability distribution p_i^C. The total mean (i.e., combined mean) of each component of the image can be easily simply calculated as follows.

$$\mu_T^C = \sum_{\substack{i=1 \\ C=\{R,G,B\}}}^{L} i p_i^C. \tag{4.2}$$

The n-level thresholding presents $n - 1$ threshold levels $t_j^C, j = 1, \ldots, n - 1$, and the operation is performed as expressed below:

$$F^C(w,h) = \begin{cases} 0, & f^C(w,h) \le t_1^C \\ \frac{1}{2}\left(t_1^C + t_2^C\right), & t_1^C < f^C(w,h) \le t_2^C \\ \vdots & \\ \frac{1}{2}\left(t_{n-2}^C + t_{n-1}^C\right), & t_{n-2}^C < f^C(w,h) \le t_{n-1}^C \\ L, & {}^C(w,h) > t_{n-1}^C \end{cases}, \tag{4.3}$$

where w and h are the width (W) and height (H) pixel, that is, the location of the pixel of the image of size $H \times W$ denoted by $f^C(w,h)$ with L intensity levels for each component. In this situation, the pixels of a given image will be divided into n classes $D_1^C, \ldots, D_n^C$, which may represent multiple objects or even specific features on such objects (e.g., topological features).

The simplest and computationally most efficient method of obtaining the optimal threshold is the one that maximizes the between-class variance of each component which can be generally defined by:

$$\sigma_B^{c2} = \sum_{\substack{j=1 \\ C=\{R,G,B\}}}^{n} w_j^C \left(\mu_j^C - \mu_T^C\right)^2, \tag{4.4}$$

where j represents a specific class in such a way that w_j^C and μ_j^C are the probability of occurrence and mean of class j, respectively. The probabilities of occurrence w_j^C of classes $D_1^C, \ldots, D_n^C$ are given by:

$$w_j^C = \begin{cases} \sum\limits_{\substack{i=1 \\ C=\{R,G,B\}}}^{t_j^C} p_i^C \ , j=1 \\ \sum\limits_{\substack{i=t_{j-1}^C+1 \\ C=\{R,G,B\}}}^{t_j^C} p_i^C, \ 1<j<n \\ \sum\limits_{\substack{i=t_{j-1}^C+1 \\ C=\{R,G,B\}}}^{L} p_i^C \ , j=n \end{cases}, \tag{4.5}$$

The mean of each class μ_j^C can then be calculated as

$$\mu_j^C = \begin{cases} \sum\limits_{\substack{i=1 \\ C=\{R,G,B\}}}^{t_j^C} \frac{ip_i^C}{w_j^C} \ , j=1 \\ \sum\limits_{\substack{i=t_{j-1}^C+1 \\ C=\{R,G,B\}}}^{t_j^C} \frac{ip_i^C}{w_j^C} \ , 1<j<n \\ \sum\limits_{\substack{i=t_{j-1}^C+1 \\ C=\{R,G,B\}}}^{L} \frac{ip_i^C}{w_j^C} \ , j=n \end{cases}. \tag{4.6}$$

In other words, the problem of n-level thresholding can be considered as an optimization problem to look for the thresholds t_j^C that maximize the objective functions (i.e., fitness function) of each image component C, which can generally be defined as

$$\varphi^C = \max_{\substack{1 < t_1^C < \cdots < t_{n-1}^C < L \\ C}} \sigma_B^{c2}\left(t_j^C\right). \tag{4.7}$$

The computation of this optimization problem will lead to a much larger computational effort when the number of threshold levels and image components increases. In order to address this issue, recently, biologically inspired methods, such as the well-known PSO, have been investigated as a computationally efficient alternative to analytical methods for solving such optimization problems (Del Valle et al. 2008). In this line of thought, the efficiency of each particle is evaluated with respect to the between-class variance σ_B^2 of the image-intensity distributions estimated by Eq. (4.4).

For the sake of image segmentation, at the beginning of the procedure, the particles' velocities are set to zero and their corresponding positions are randomly set within the boundaries of the search space. The search space will depend on the number of intensity levels L; that is, for instance, if the input image is of 8 bits then the particles will be deployed between 0 and 255.

The personal and global bests are initialized with the worst possible values with respect to the nature of the problem. The other parameters that need to be adjusted are population size and stopping criteria. The population size is very important in the sense of getting an overall good solution in an acceptable time limit (Ghamisi et al. 2014). Stopping criteria can be a predefined number of iterations without getting better results or other criteria, depending on the problem.

PSO reveals an important drawback: the susceptibility to local solutions (Ghamisi et al. 2014). As such, under complex problems with multiple local optima, the PSO may fail and the output result may not be satisfactory. To address this issue, different modifications of the PSO have been introduced to adjust the traditional PSO algorithm such as the FODPSO algorithm (Ghamisi et al. 2012a, 2012b, 2014). In order to find comprehensive information regarding PSO-, DPSO-, and FODPSO-based segmentation approaches, we refer readers to (Ghamisi et al. 2012b, 2012c, 2014a, 2014b and Kulkarni and Venayagamoorthy 2010). In addition, the concept of PSO and FODPSO has been used for other applications such as image clustering, classification, data compression, and feature selection in the remote-sensing community. For more information on other applications of the PSO family, we refer interested readers to (Ghamisi et al. 2013, 2015a, 2015b; Kargozar Nahavandi et al. 2015 and Ghamisi and Benediktsson 2015)

4.2 Experimental Results

In the experiments reported in (Ghamisi et al. 2012), different thresholding-based segmentation techniques including GA, BF, PSO, DPSO, and FODPSO have been applied on a few common images such as: airplane, hunter, butterfly, road, and map.

With respect to the fitness values (between-class variance of different thresholds) obtained by GA, BF, PSO, DPSO, and FODPSO, it can be inferred that despite small differences in between-class variances obtained by different techniques, all algorithms seem to reach the vicinities of the optimal solution, that is, higher between-class variance. However, those differences are more significant when the number of thresholds increases. The FODPSO leads to higher fitness values than the DPSO. The main reason beyond this outperformance is that although both techniques use natural selection in order to avoid stagnation, the FODPSO takes advantage of the fractional-order mechanism that allows controlling the convergence rate of particles, thus presenting a more exploitative behavior when near the solution vicinities.

Originally, PSO was introduced to solve continuous problems, whereas GA was originally introduced for solving discrete problems. Based on experiments conducted by (Veeramachaneni et al. 2003), it was shown that a PSO performed better than GAs when applied on some continuous optimization problems. In addition, based on Kennedy and Spears (1998) and Sathya and Kayalvizhi (2011), PSO is generally faster and more robust to local solutions than GAs, in particular when the dimension of a problem increases. Therefore, for the purpose of thresholding-based segmentation, when the number of dimensions increases, a significant difference between the fitness values of the PSO family (PSO-, DPSO-, and FODPSO-based segmentation techniques) and GA happens, and the PSO family shows better results than GA in higher dimensions. However, the traditional PSO suffers from premature convergence and as a result, BF acts better than PSO in most cases. However, FODPSO-based segmentation outperforms all the above-mentioned techniques in terms of finding the highest between-class variances.

In terms of CPU processing time, PSO has been shown to be faster than GA and BF for finding thresholds (Sathya and Kayalvizhi 2011). With reference to (Ghamisi et al. 2012), FODPSO-based segmentation is able to reach its solution in a shorter period of time than PSO and DPSO. The DPSO still presents a lower CPU time than the PSO especially when the number of thresholds increases. This is a small repercussion of having an exploitation activity when near the solution: a high level of exploitation allows a good short-term performance but slows down the convergence in order to reach a more feasible solution.

To compare visually the segmented results of different test cases by FODPSO, the segmented images with various threshold levels are given in Fig. 4.1. As can be seen, images with a higher level of segmentation have more detail than the other images. In contrast, the 3-level segmented image is considered the roughest image in different test cases. It is easy to conclude that by increasing the level of segmentation, the segmented image includes more detail. Therefore, the 6-level segmented image in different test cases is smoother than the 3-level one.

As a conclusion, FODPSO-based image segmentation is able to reach a slightly better fitness solution in less CPU processing time than its alternatives (DPSO and PSO). This should be highly appreciated in the many applications for which real-time segmentation is required, such as the autonomous deployment of sensor nodes in a given environment or the detection of flaws in quality inspection of

Fig. 4.1 The result of segmentation with 2, 3, 4, and 5 thresholds, respectively (from *left* to *right*). Illustration taken from (Ghamisi et al. 2012)

materials. Moreover, FODPSO is slightly faster than DPSO because its fractional calculus is able to control the convergence rate of the algorithm. A swarm behavior can be divided into two main activities: exploitation and exploration. The former controls the convergence of the algorithm, thus allowing a good short-term performance. However, if the exploitation level is too high, then the algorithm may get stuck on local solutions. The latter, however, controls 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. In the DPSO, the trade-off between exploitation and exploration can only be controlled by adjusting the inertia weight. A large inertia weight improves exploration activity, however, the exploitation may be improved using a small inertia weight. Because the FODPSO presents a fractional-calculus strategy to control the convergence of particles with memory effect, the coefficient α allows providing a higher level of exploration while ensuring the global solution of the algorithm (Ghamisi et al. 2012).

4.3 Summary

In this chapter, the concept of FODPSO has been used for image segmentation. In more detail, in order to address the main drawback of the exhaustive Otsu thresholding-based image segmentation, which is computationally extensive, FODPSO has been taken into account for determining the $n - 1$ optimal n-level threshold on a given image. Then, a few well-known thresholding-based segmentation techniques such as GA, BF, PSO, DPSO, and FODPSO have been compared in terms of their accuracy and consecutive CPU processing time. The FODPSO leads to a higher fitness value (between-class variance) than other thresholding-based segmentation techniques, especially when the number of thresholds increases. In terms of consecutive CPU processing time, again FODPSO leads to the best fitness value in the least CPU processing time. This outperformance encourages one to consider the FODPSO for the segmentation of volumetric data where a rapid and accurate response is needed.

References

Brink, A. D. (1995). Minimum spatial entropy threshold selection. *IEE Proceedings on Vision Image and Signal Processing, 142*(1995), 128–132.

Del Valle, Y., Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J. C., & Harley, R. G. (2008). Particle swarm optimization: basic concepts, variants and applications in power systems". *IEEE Transactions on Evolutionary Computation, 12*(2), 171–195.

Floreano, D., & Mattiussi, C. (2008). *Bio-inspired artificial intelligence: theories, methods, and technologies*. Cambridge, MA: MIT Press.

Fogel, D. B. (2000). *Evolutionary computation: toward a new philosophy of machine intelligence* (2nd ed.). Piscataway, NJ: IEEE Press.

Ghamisi, P. (2011). A novel method for segmentation of remote sensing images based on hybrid GA-PSO. *International Journal of Computer Applications, 29*(2), 7–14.

Ghamisi, P., & Benediktsson, J. A. (2015). Feature selection based on hybridization of genetic algorithm and particle swarm optimization. *IEEE Geoscience and Remote Sensing Letter, 12*(2), 309–313.

Ghamisi, P., Couceiro, M. S., Benediktsson, J. A., & Ferreira, N. M. F. (2012a). An efficient method for segmentation of images based on fractional calculus and natural selection. *Expert System with Applications, 39*(2012), 12407–12417.

Ghamisi, P., Couceiro, M. S., Ferreira, N. M. F., & Kumar, L. (2012b). Use of Darwinian particle swarm optimization technique for the segmentation of remote sensing images. IEEE International Geoscience Remote Sensing Symposium (IGARSS), pp. 4295–4298.

Ghamisi, P., Couceiro, M. S., & Benediktsson, J. A. (2012c). Extending the fractional order Darwinian particle swarm optimization to segmentation of hyperspectral images. In *Proceeding of SPIE 8537, Image and Signal Processing for Remote Sensing XVIII*, 85370F, pp. 85370F–85370F–11.

Ghamisi, P., Couceiro, M. S., & Benediktsson, J. A. (2013). Classification of hyperspectral images with binary fractional order darwinian pso and random forests. In *Proceeding of SPIE, Image and Signal Processing for Remote Sensing XIX*, 88920S88920S-8.

Ghamisi, P., & Couceiro, M. S., & Martins, F. M. L., & Benediktsson, J. A. (2014a). Multilevel image segmentation based on fractional-order darwinian particle swarm optimization. *IEEE Transactions on Geoscience and Remote Sensing*, 52(5), 2382–2394.

Ghamisi, P., & Couceiro, M. S., & Fauvel, M., & Benediktsson, J. A. (2014b). Integration of segmentation techniques for classification of hyperspectral images. *IEEE Geoscience and Remote Sensing Letters*, 11(1), pp. 342–346.

Ghamisi, P., ALi, A., Couceiro, M. S., & Benediktsson, J. A. (2015a). A Novel Evolutionary Swarm Fuzzy Clustering Approach for Hyperspectral Imagery. *IEEE Journal of Selected. Topics in Applied Earth Observations and Remote Sensing*, accepted. pp. 1–10.

Ghamisi, P., Couceiro, M. S., and Benediktsson, J. A., (2015b). A novel feature selection approach based on FODPSO and SVM. *IEEE Transactions on Geoscience and Remote Sensing*, *53*(5), 2935–2947.

Kargozar Nahavandi, S., Ghamisi, P., Kumar, L., & Couceiro, M. S. (2015). A novel adaptive compression technique for dealing with corrupt bands and high levels of band correlations in hyperspectral images based on binary hybrid GA-PSO for big data compression. *International Journal of Computer Applications*, *109*(8), 18–25.

Kennedy, J., & Spears, W. (1998). Matching Algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. *IEEE International Conference on Evolutionary Computation*, Achorage, Alaska, USA.

Kulkarni, R. V., & Venayagamoorthy, G. K. (2010). Bio-inspired algorithms for autonomous deployment and localization of sensor. *IEEE Transactions on System, Man, and Cybernetics, 40*(6), 663–675.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on System, Man, and Cybernetics*, SMC-9, 62–66.

Sathya, P. D., & Kayalvizhi, R. (2011). Modified bacterial foraging algorithm based multilevel thresholding for image segmentation. *Journal Engineering Applications of Artificial Intelligence, 24*(4), 595–615.

Sezgin, M., & Sankur, B. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronics Imaging, 13*(1), 146–168.

Veeramachaneni, K., Peram, T., Mohan, C., & Osadciw, L. (2003). Optimization using particle swarm with near neighbor interactions. Lecture Notes Computer Science, vol. 2723, Springer Verlag, Berlin.

Chapter 5
Case Study III: Swarm Robotics

Abstract The navigation of groups of robots, especially multirobot systems (MRS) or even swarm robots, has been one of the fields that has benefited from biological inspiration (Bonabeau et al., Proceedings of the 2000 IEEE International Conference on Robotics, 1999). As with many other things in robotics, the advances were first introduced and evaluated in the context of computer agents. One of the first applications of those methods started with optimization tools with the well-known particle swarm optimization (PSO) previously mentioned. In this chapter, we try to go a step forward by adapting the version of the FODPSO to MRS, denoting it robotic DPSO (RDPSO). Any other PSO variant could be adapted to MRS exploration. However, the FODPSO was chosen because it is an evolutionary algorithm that extends the PSO using natural selection to enhance the ability to escape from suboptimal solutions and employs fractional calculus to improve the convergence of particles.

Keywords FODPSO · Swarm intelligence · Multirobot systems · Swarm robotics · RDPSO

5.1 Robotic Darwinian PSO

In brief, just as in MRS where groups of robots interact to accomplish their goals, FODPSO uses groups of interacting virtual agents in order to achieve optimization. However, real MRS present several constraints that need to be considered. Contrary to virtual agents, robots are designed to act in the real world where obstacles need to be taken into account. Also, and because in certain environments or applications the communication infrastructure may be damaged or missing, the self-spreading of autonomous mobile nodes of a mobile ad hoc network (MANET) over a geographical area needs to be considered. For instance, the development of robot teams for surveillance or rescue missions in unstructured and unknown environments requires that robots have to be able to maintain communication among them without the aid of a communication infrastructure.

© The Author(s) 2016

M. Couceiro and P. Ghamisi, *Fractional Order Darwinian Particle Swarm Optimization*, SpringerBriefs in Applied Sciences and Technology,
DOI 10.1007/978-3-319-19635-0_5

Inasmuch as the herein-presented RDPSO approach is an adaptation of the FODPSO to real mobile robots, four general features are proposed:

1. An improved inertial influence adapted from the fractional calculus extension taking into account robots' dynamics
2. Integration of an obstacle avoidance behavior to avoid collisions
3. A way to enforce multihop network connectivity to ensure that the MANET remains connected throughout the mission
4. An adaptation of the "punish–reward" mechanism to emulate social exclusion and inclusion of robots

Therefore, to model a robotic swarm, each robot n moves in a multidimensional space. For the sake of simplicity, let us rearrange the previous equations from (5.1) to (5.2), describing the RDPSO with the discrete equation (DE) system:

$$v_n[t+1] = w_n[t] + \sum_{i=1}^{4} \rho_i r_i (\chi_i[t] - x_n[t]), \tag{5.1}$$

$$x_n[t+1] = x_n[t] + v_n[t+1], \tag{5.2}$$

$$w_n[t] = -\sum_{k=1}^{r} \frac{(-1)^k \Gamma[\alpha+1] v_n^s[t+1-kT]}{\Gamma[k+1]\Gamma[\alpha-k+1]}, \tag{5.3}$$

wherein the only new parameters are the ones with suffix $i = 3$ and $i = 4$, related to both the obstacle avoidance component and enforcing communication component, respectively. In this situation, $v_n[t]$ and $x_n[t]$ represent the velocity and position vector of robot n, respectively. Whereas $\|v_n[t]\|$ is limited to the maximum allowed velocity of $v_{\max}$ for robots, that is,$\|v_n[t]\| \leq v_{\max}$, $x_n[t]$ depends on the scenario dimensions. The cognitive $\chi_1[t]$ and social components $\chi_2[t]$ are the same as previously discussed (cf., Eq. (1.1)). $\chi_1[t]$ represents the local best position of robot n and $\chi_2[t]$ represents the global best position of robot n. The size of the vectors (ϖ) depends on the dimensionality $\mathbb{R}^{\varpi}$ of the physical space being explored, for example, $\varpi = 2$ for planar problems.

Because the other features are novel, they are further explored in the following sections. Let us start by exploiting the inertial component $w_n[t]$ in the context of robotics, in which the RDPSO benefits from fractional calculus (Podlubny 1999), to describe the dynamic phenomenon of a robot's trajectory that depends on past events (Couceiro et al. 2012).

5.1.1 *Fractional-Order Swarm Robots*

As previously presented in Sect. 2.1, fractional calculus (FC) has been drawing researchers' attention for the last two decades, being rediscovered and applied in an

increasing number of fields. This section further explores the extension applied in the FODPSO, so as to control the convergence rate of robots.

Based on Eq. (5.3), the next position of a given robot depends, not only on its current position, but also on the previous r positions. This "memory" of past events allows fitting the dynamic phenomena of a robot's trajectory, thus controlling robot convergence. In other words, adding memory to the swarm algorithm allows improving the convergence rate of robots because each robot will have the information about its preceding actions. Nevertheless, the computational requirements increase linearly with r; that is, the RDPSO present an $\mathcal{O}(r)$ memory complexity per robot. Moreover, it is noteworthy that this kind of optimization or foraging algorithm presents a higher performance as the number of robots increases. Hence, robots should be as simple and low cost as possible (i.e., swarm robots) which are usually memory limited (Couceiro et al. 2011).

Therefore, the truncation of Eq. (5.3) will depend on the requirements of the application and the features of the robot. For instance, for the eSwarBot platforms developed by Couceiro et al. (2011), and presented in Fig. 5.1, an $r = 4$ leads to results of the same type as for $r > 4$. Although one could consider processing power as the main reason for such a limited number of terms, the kinematical features of the platform and the mission requirements also need to be considered in such a way that one can present the following result.

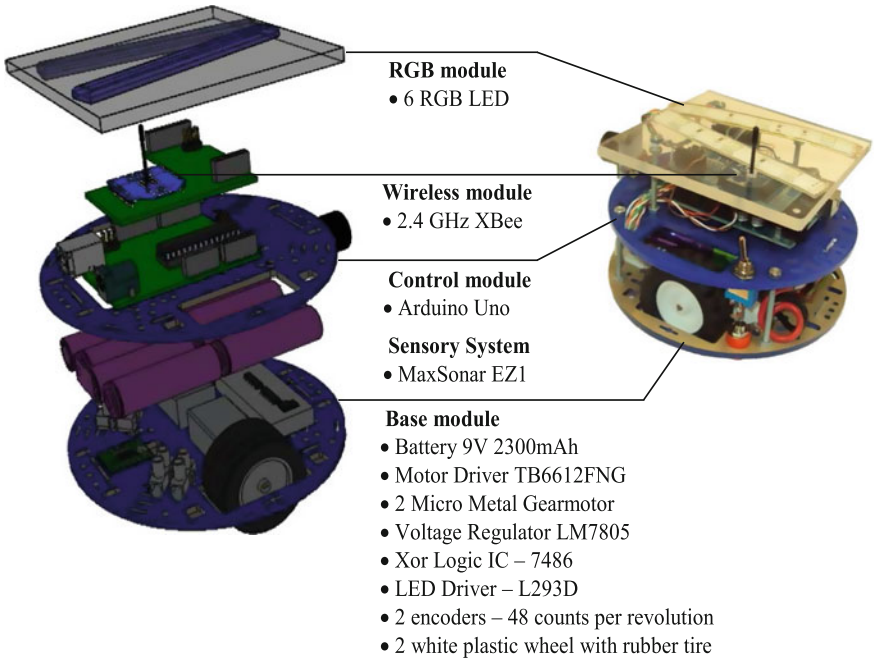

Fig. 5.1 The eSwarBot robot (Couceiro et al. 2011)

Proposition 5.1 *Let δ and $\Delta x_{\max}$ be the encoders-wheel resolution of robots and the maximum travelled distance allowed between iterations; that is, $\Delta x_{\max} = x_n[t-1] - x_n[t]$, respectively. If τ is the minimum natural number that verifies the inequality*:

$$-\frac{(-1)^{\tau}\Gamma[\alpha+1]\Delta x_{\max}}{\Gamma[\tau+1]\Gamma[\alpha-\tau+1]} < \delta, \tag{5.4}$$

then the RDPSO Eq. (5.3) *should be truncated based on δ and $\Delta x_{\max}$, in $r = \tau - 1$.*

Proof Let us consider the example of a differential drive robot (e.g., eSwarBot). A differential drive robot consists of two independently driven wheels and, usually, a free wheel for stability (e.g., caster wheel). For navigation purposes, the driven wheels are usually equipped with encoders that provide odometry measures. Hence, the major odometry parameter of such a mobile robot to drive forward is the radius of the wheels R_{wheel} and the number of pulses from revolution of the wheel τ_{rev}. The kinematical equation of a differential drive robot, while moving forward, can be defined as

$$\text{pulses} = \tau_{\text{rev}} \times \frac{\text{dist}}{2\pi \times R_{\text{wheel}}}, \tag{5.5}$$

where pulses is the number of pulses necessary for the robot to travel a distance of dist. Defining pulses = 1 we can obtain the minimum distance that a robot can travel at each iteration, that is, the resolution δ. Hence, an increment of the distance lower than δ would be infeasible for the robot to travel. Also, one may observe through Eq. (5.3), that the relevance of past events, that is, the $v[t+1-kT]$ term, reduces over time. In other words, from a given term $r = \tau - 1$, the relevance of all previous events before it would be irrelevant as the robot would be unable to travel with such accuracy. □

To clarify the previous result, let us consider the following example.

Example 5.1 Considering the eSwarBot platform, a resolution of $\delta = 2.76$ mm is obtained for a single pulse, taking into account that $R_{\text{wheel}} = 21.09$ mm and the combination between encoders–wheel provides $\tau_{\text{rev}} = 48$ pulses/revolution. Let us consider a maximum travelled distance between two iterations of $\Delta x_{\max} = 0.1$ m; that is, the robot cannot travel more than 0.1 m without any update of the information. Figure 5.2 presents the computation of each term of Eq. (5.10). As one may observe, a term of $r = 4$ would be enough to represent the RDPSO dynamics in such conditions as the fifth term returns an increment of 2.73 mm. In other words, the algorithm would present similar results for $r \geq 4$.

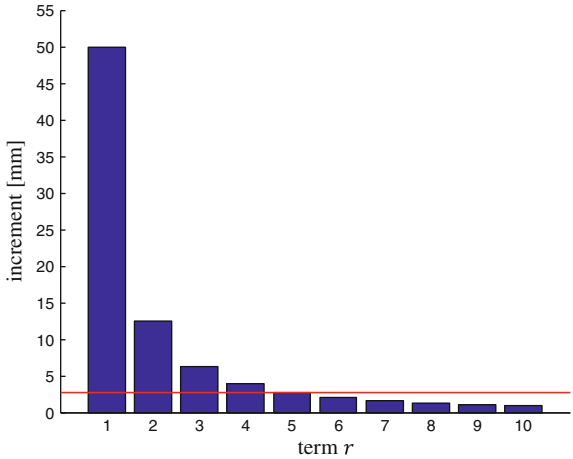

Fig. 5.2 Convergence of the robot toward the solution changing the differential derivative $\delta\, r$

As eSwarBots are the robotic platforms most widely used throughout this work, consider only the first $r = 4$ terms of the fractional discrete difference in (5.3), yielding:

$$v_n[t+1] = \alpha v_t^n + \frac{1}{2}\alpha(1-\alpha)v_{t-1}^n + \frac{1}{6}\alpha(1-\alpha)(2-\alpha)v_{t-2}^n + \frac{1}{24}\alpha(1-\alpha)(2-\alpha)(3-\alpha)v_{t-3}^n + \sum_{i=1}^{4}\rho_i r_i(\chi_i[t] - x_n[t]) \tag{5.6}$$

It is, however, noteworthy that a similar analysis should be conducted if one intends to use different robotic platforms.

5.1.2 Obstacle Avoidance

In short, the RDPSO algorithm tries to minimize a cost function, or maximize a fitness function depending on the mission objective. For instance, if we have a gas leak, robots running the RDPSO algorithm try to maximize the sensed gas at each iteration. The approach presented in this chapter seeks to create a new cost or fitness function in such a way that it would guide the robot to perform the main mission (e.g., find the gas leak) while avoiding obstacles.

When a robot needs to move from any arbitrary start position to any target position in the environment, it must be able to avoid both static and dynamic obstacles (Williams and Wu 2010). For this purpose we assume that each robot is equipped with sensors capable of sensing the environment for obstacle detection within a finite sensing radius R_w. A monotonic and positive sensing function $g(x_n[t])$ is then defined. This function depends on the sensing information, that is, the distance from the robot to the obstacle. Note that in most situations, as can be

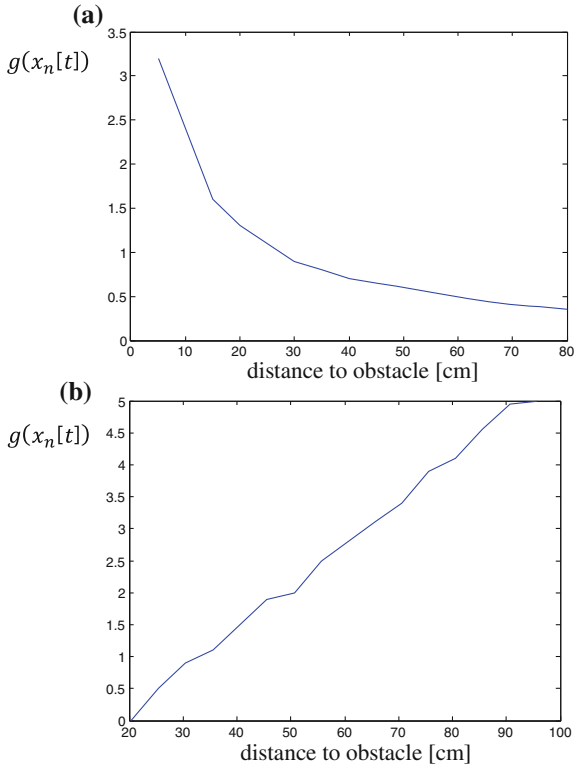

Fig. 5.3 Sensing function $g(x_n[t])$ represented as the relation between the analog output voltage of distance sensors and the distance to the obstacle. **a** Sensor Sharp GP2Y0A21YK IR: monotonically decreasing sensing function $g(x_n[t])$. **b** Sensor Sonaswitch Ultrasound: monotonically increasing sensing function $g(x_n[t])$

observed in Fig. 5.3, the sensing function $g(x_n[t])$ can be represented as the relation between the analog output voltage of distance sensors (e.g., sonars) and the distance to the detected object.

Variable $\chi_3[t]$ is then represented by the position of each robot that optimizes the monotonically decreasing, or increasing, $g(x_n[t])$. In other words, when a robot does not sense any obstacle at time t, the best position that optimizes $g(x_n[t])$ is constantly updated and equal to the current position $x_n[t]$. Afterward, if the robot detects an obstacle inside its sensing range, the best position that optimizes $g(x_n[t+1])$ is not updated, thus creating an attractive force toward its last best position $x_n[t]$. Although in a free-obstacle environment ρ_3 can be set to zero, in real-world scenarios, obstacles need to be taken into account and the value of ρ_3 depends on several conditions related to the main objective (i.e., minimizing a cost function or maximizing a fitness function) and the sensing information (i.e., monotonicity of the sensing function $g(x_n[t])$). Furthermore, the relation between ρ_3 and the other weights, namely ρ_1, ρ_2, and ρ_4, depends on the susceptibility of each robot to the main objective and the obstacle avoidance behavior (Fig. 5.4). For instance, $\rho_3 \ll \min(\rho_1, \rho_2)$ may lead to a faster convergence to the solution allowing the team of robots to accomplish the main objective faster, but it may also lead to obstacle collisions (Fig. 5.4a) whereas $\rho_3 \gg \min(\rho_1, \rho_2)$ may increase the performance of

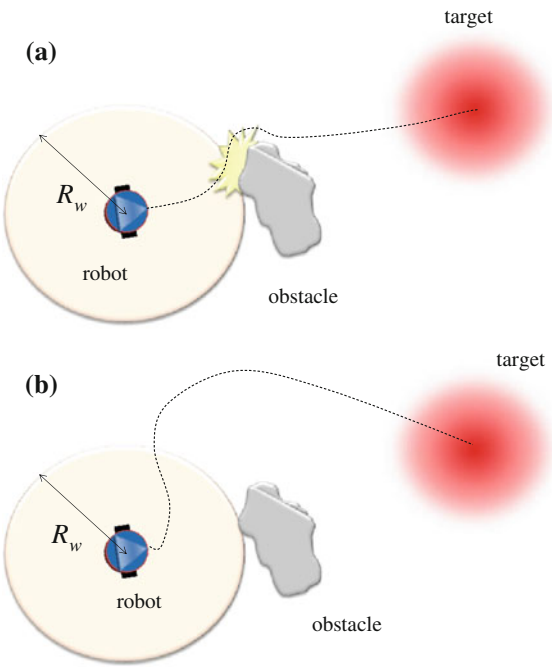

Fig. 5.4 Illustrative example of obstacle avoidance behavior of a robot: **a** $\rho_3 \ll \min(\rho_1, \rho_2)$; **b** $\rho_3 \gg \min(\rho_1, \rho_2)$

obstacle avoidance but may lead to a slower convergence of the main objective (Fig. 5.4b).

In addition to considering obstacle avoidance in real-world applications, communication constraints also need to be considered.

5.1.3 Ensuring MANET Connectivity

It has generally been assumed in MRS that each robot has the ability to communicate with any other robot with small consideration for the quality and performance of the wireless communication network. Although being valid in some particular situations, the assumption does not generally hold. Because robots may move apart to farther areas, it is important to have a pervasive networking environment for communications among robots. Furthermore, without a pre-existent infrastructure, robots need to be able to act as intermediate nodes, that is, routers, in order to relay information from one point to another, thus supporting multihop communication in a MANET (Miller 2001). One way of ensuring that is by controlling robots' positions in order to maintain the communication based on constraints such as maximum distance $d_{\max}$ or minimum signal quality $q_{\min}$. The way the network will be forced to preserve connectivity depends on communication characteristics (e.g., multihop, biconnectivity) (Crispin 2009).

Consider a group (or smaller swarm) of N_S robots where each robot is both an exploring agent of the environment and a mobile node of a MANET that performs packet forwarding, according to a paradigm of multihop communication. The goal is to ensure that robots explore an unknown environment, while ensuring that the MANET remains connected throughout the mission.

Assuming that the network supports multihop connectivity, the communication between two end nodes (i.e., robots) is carried out through a number of intermediate nodes whose function is to relay information from one point to another. Considering that nodes are mobile, it is necessary to guarantee the communication between all nodes. In the case where each robot corresponds to a node, in order to overcome the nonconnectivity between them, the desired position $x_n[t+1]$ must be controlled.

The connectivity between robots can be described by means of a link matrix $L = [l_{ij}] \in \mathbb{R}^{N_s \times N_s}$ for an N_s-robot network, wherein each entry represents the link between robot i and j. The link is defined according to the users' preferences. The most common approaches include:

1. Calculating the l_{ij} values as functions of the distance between pairs of robots indicating the link distance between them (Sheng et al. 2006)
2. Calculating the l_{ij} values as functions of the radio quality signal between pairs of robots indicating the link quality between them (Tardioli and Villarroel 2007)

Trying to maintain network connectivity by only taking into account the communication range $d_{\max}$ (approach 1) does not match reality because the propagation model is more complex: the signal depends not only on the distance but also on the multiple paths from walls and other obstacles (approach 2). However, in simulation, the communication distance is a good approach and it is easier to implement.

Depending on the chosen approach (1 or 2), an adjacency matrix $A_c = [a_i j] \in \mathbb{B}^{N_s \times N_s}$, in which $\mathbb{B}$ represents the set of binary numbers (i.e., 0 or 1), can be defined based on the maximum distance or minimum radio quality signal between robots, respectively (Miller 2001). The adjacency matrix, that is, one-hop connectivity matrix, where a 1 entry at (i,j) indicates a connection between robot i and j and a 0 entry at (i,j) indicates no connection between robot i and j, represents the neighbors of each node, that is, direct connection between robots.

$$a_{ij} = \begin{cases} 1\,, & \text{connection between nodes } i \text{ and } j \\ 0\,, & \text{no connection between nodes } i \text{ and } j \end{cases}. \tag{5.7}$$

Note that the diagonal elements (i.e., when $i = j$) of the adjacency matrix are set equal to 0. If the communication system supports the relay of messages to distant robots via intermediate robots, then multihop connections can be made. Using the hop distances, that is, the smallest numbers of hops to connect nonadjacent robots, the zero-valued off-diagonal entries in the adjacency matrix can be manipulated in order to create a multihop connectivity matrix $C^{(N_k)} = \left[c_{ij}^{(N_k)}\right] \in \mathbb{R}^{N_s \times N_s}$, for which the entry at (i,j) represents the minimum number of hops necessary to connect

robots i and j, and N_k represents the iteration that varies with the number of hops the network can handle. $N_s - 1$ is the maximum number of possible hops. The connectivity matrix can then be defined:

$$0$$

$$c_{ij}^{(N_k)} = \begin{cases} h, \text{node } i \text{ connected to } j \text{ by } h \leq N_k \text{hops} \\ 0, \text{otherwise} \end{cases}. \tag{5.8}$$

Note that the diagonal elements of the connectivity matrix ($i = j$) are set equal to 0. Furthermore, the adjacency matrix is the first iteration in calculating the connectivity matrix ($C^{(1)} = A_c$). When $N_k > 1$ (i.e., for multihop connections) an auxiliary matrix $B^{(N_k)} = \left[b_{ij}^{(N_k)}\right] \in \mathbb{R}^{N_s \times N_s}$ is then calculated based on the iteration (number of hops):

$$b_{ij}^{(N_k)} = \begin{cases} 0, c_{ij}^{(N_k-1)} > 0 \\ K, \sum_{v=1}^{N_s} c_{iv}^{(N_k-1)} b_{vj}^{(N_k-1)} > 0 \text{ and } c_{ij}^{(N_k-1)} = 0 \end{cases} \tag{5.9}$$

Note that the diagonal elements ($i = j$) of the auxiliary matrix are set equal to 0 and ($B^{(1)} = A_c$). The connectivity matrix can now be calculated using the equation:

$$C^{(N_k)} = C^{(N_k-1)} + B^{(N_k)}. \tag{5.10}$$

After $N_s - 1$ iterations, $C^{(N_s-1)}$ represents the multihop network connectivity. The existence of zero elements (except diagonal elements) indicates no connection between robot i and j even using multihop. In this case it is necessary to implement an algorithm to ensure the complete connectivity of the network. One strategy is to define a binary connectivity matrix $C_B = \left[c_{B_{ij}}\right] \in \mathbb{B}^{N_s \times N_s}$ wherein each nonzero element of the connectivity matrix matches the logic value 1.

$$c_{B_{ij}} = \begin{cases} 1, c_{ij}^{(N_s-1)} \neq 0 \\ 0, c_{ij}^{(N_s-1)} = 0 \end{cases}. \tag{5.11}$$

Performing an element-by-element multiplication between the link matrix and the logical inverse (binary NOT) of the binary connectivity matrix, yields a break matrix $C_{\text{break}} = \left[c_{\text{break}_{ij}}\right] \in \mathbb{B}^{N_s \times N_s}$ containing the values that represent the break of connection between the robots. In the case where each robot corresponds to a robot, in order to overcome the nonconnectivity between them, the desired position $x_n[t+1]$ of each robot from Eq. (5.2) must be controlled because it influences the link matrix. In this chapter, the multihop connectivity matrix $C^{(N_s-1)}$ and auxiliary matrices (C_B and C_{break}) are only used as information about the network topology (Algorithm 5.1).

```
Determines N_s × N_s link matrix L
Calculates N_s × N_s adjacency matrix A_c
Initialize N_s × N_s connectivity and auxiliary matrix C^(1) = A, B^(1) = A_c
For N_k = 2 to the longest hop (N_k = N_s − 1)
    For all robot pairs (i, j) = (1,1) to (N_s, N_s)
        If i = j OR c_ij^(N_k) > 0
            skip to the next robot pair
        If Σ_{v=1}^{N_s} c_iv^(N_k−1) b_vj^(N_k−1) > 0 AND c_ij^(N_k−1) = 0
            b_ij^(N_k) = N_k
    C^(N_k) = C^(N_k−1) + B^(N_k)
Calculates N_s × N_s binary connectivity and break matrix C_B, C_break
If the connectivity depends on the distance/quality, find the minimum/maximum value of each line
of link matrix L, excluding zeros and (i, j) pairs previously chosen
Computes Equation (1.1)
```

Algorithm 5.1. Ensuring group s network connectivity.

As Algorithm 5.1 describes, one way to ensure the full connectivity of the MANET is to "force" each robot to communicate with its nearest neighbor that has not chosen it as its nearest neighbor. Inasmuch as the connectivity depends on the distance/signal quality, connectivity between robots may be ensured by computing the minimum/maximum value of each line of link matrix L, after excluding zeros and (i,j) pairs previously chosen. Therefore, the MANET component $\chi_4[t]$ is represented by the position of the nearest neighbor increased by the maximum communication range $d_{\max}$ toward the robot's current position. A larger ρ_4 may enhance the ability to maintain the network connected ensuring a specific range or signal quality between robots. To better understand how the MANET connectivity principle works, please consider the topology depicted in Fig. 5.5.

As may be perceived, robot 2 is the nearest neighbor of robot 1 and is at the correct distance $d_{\max}$ resulting in a null force connectivity vector. The nearest neighbor of robot 2 is robot 3 which is too close, thus resulting in a repulsive force at robot 2 in order to ensure $d_{\max}$. Finally, the nearest neighbor of robot 3, that was not previously chosen, is robot 4 which is too far away, thus being affected by an attractive force toward robot 4.

Note that having multiple groups, which is inherent to the proposed FODPSO, and inevitably RDPSO, algorithms, enables a distributed approach because the network that was previously defined by the whole population of robots (swarm) is

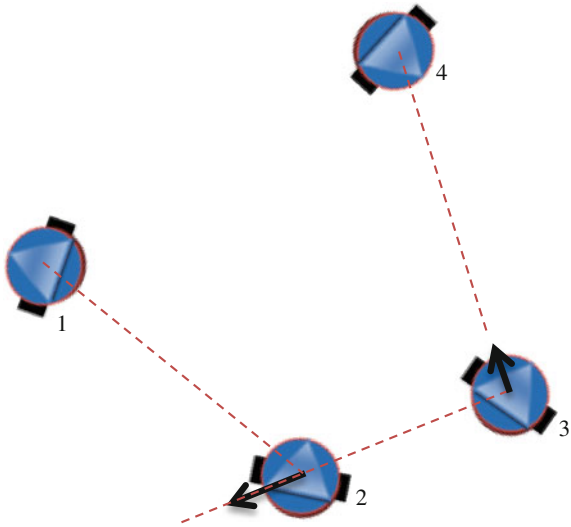

Fig. 5.5 Illustration of a MANET topology of a group. *Dashed lines* represent the maximum distance d_{max} between each pair of robots and the *bold arrows* represent the force vectors that ensure MANET connectivity

now divided into multiple smaller MANETs (one for each group), thus decreasing the number of robots and the information exchanged between robots of the same network. In other words, robot interaction with other robots through communication is confined to local interactions inside the same group, thus making RDPSO scalable to large populations of robots. The exchanged data concerning the signal quality or robot's position allow the implicit processing of the RDPSO algorithm by the team in a distributed way. In other words, every robot needs to be aware of the position or signal quality of all other robots in the same group in order to proceed to the next iteration of the algorithm. This is a limitation of the algorithm because all robots need to be equipped with a good odometry or localization system (e.g., GPS). An alternative to it would be extending the GPS capabilities of some robots to non-GPS robots using strategies to find the teammates' positions under their visual range (Kulkarni and Venayagamoorthy 2010). For instance, if robots are equipped with LRFs, retroreflective markers can be used for recognition. Inasmuch as the implementation of such strategies is beyond the scope of this book, they are not taken into account and it is considered that each robot knows its own position.

5.1.4 "Punish–Reward" Mechanism

In addition to the biological inspiration that makes roboticists capture the underlying principles of living organisms and assimilate them into ever-improving robot forms, the social behaviors inherent to the several biological societies have been studied and applied to MRS showing interesting and promising results (Balch and Hybinette 2000).

This section addresses one of the most relevant social processes of human society: social exclusion. The formal concept of social exclusion appeared in the 1970s in France referring to the "rupture of the social bond," where socially excluded individuals or groups were unprotected by the French social security system, thus resulting in a lack of resources and inadequate access to services making it difficult to participate in society (Scutella et al. 2009). This concept has since grown, being taken up by most of Europe, and is currently used to refer to the range of dimensions that marginalize people and reduce their opportunities to engage in social or political life.

Burchardt proposes a more precise, and also closely related with our approach, definition of social exclusion (Burchardt 2000): "An individual is socially excluded if he or she does not participate to a reasonable degree over time in certain activities of his or her society, and (a) this is for reasons beyond his or her control, and (b) he or she would like to participate." On the other hand, for a socially excluded individual or group to be accepted back in the society, aka social inclusion, the Brotherhood of St Laurence's Executive Director Tony Nicholson has proposed the following definition (Nicholson 2008): "A social inclusion approach involves the building of personal capacities and material resources, in order to fulfil one's potential for economic and social participation, and thereby a life of common dignity." In other words, social inclusion, the converse of social exclusion, is the affirmative action to change the circumstances and habits that lead to (or have led to) social exclusion.

However, this is not an exclusive concept of the human race. These concepts of social exclusion and inclusion may also be found in nonhuman animals through stigmatization[1] processes such as

- Territoriality (e.g., fish, birds, reptiles, and mammals): Exclusion of other members of the same species (e.g., certain sex) from an area.
- Status hierarchies (e.g., some bird species, lions, baboons, chimps): Individual at the top of the hierarchy excludes others from resources (e.g., food, territory, mates).
- Social ostracism (e.g., some fish species, lemurs, baboons, chimps): Prevention of others from joining social group or forcing expulsion.

For instance, Three-Spined Sticklebacks (fish) avoid others of the same species with parasites whereas Grizzlies (bear) present a hierarchy-related behavior that provides a mechanism which mutes the potential social costs of membership in stable aggregations (Craighead et al. 1995).

How this concept of social exclusion (and inclusion) is used in this book and the advantages inherent to it reside in the "punish–reward" mechanism previously described in the DPSO (Table 1.1) and is now adapted to MRS.

As the original DPSO, the number of times a group s is evolved without finding an improved objective was tracked with a search counter defined in Eq. (1.3), SC_s.

[1]Process where certain individuals are excluded from particular sorts of social interactions.

In the proposed approach, N_s^{kill} represents the number of robots excluded from group s over a period of time in which there was no improvement in the group's objective function. If the group's search counter exceeds a maximum critical threshold, $SC_{\max}$, the group is punished by excluding the worst-performing robot, which is added to a socially excluded group. The worst-performing robot is evaluated by the value of its objective function compared to other members in the same group: that is, if the objective is to maximize the fitness function the robot to be excluded will be the one with the higher fitness value. If the number of robots falls below the minimum acceptable number of robots to form a group $N_{\min}$, the group is punished by being dismantled and all the robots belonging to that group are added to the socially excluded group. On the other hand, if the group improves its objective function, then it is rewarded with the best-performing robot in the socially excluded group. If a group has been more often rewarded than punished, it has a small probability p_{sp} of spawning a new group with the predefined number of robots N_I. However, contrary to the DPSO in which p_{sp} depended on the current number of active groups, because the RDPSO does not allow intergroup communication (unless to forward messages between robots from the same group), p_{sp} is rewritten:

$$p_{\text{sp}} = r_{\text{sp}} \frac{N_s}{N_{\max}}, \tag{5.12}$$

with r_{sp} being a random number between 0 and 1, N_s is the number of robots within group s and $N_{\max}$ is the maximum number of allowed robots in a group. In other words, if a socially active group is constantly improving, it is able to create another socially active group with an expected value of 0.5 when it is at full capacity; that is, $N_s = N_{\max}$. Moreover, the group of robots forming this new group will be the best-performing robots within the socially excluded group.

The key issue in this novel approach is the answer to the question:

What do the robots of the socially excluded group do?

In fact, the answer is the same that we would give if asked about a group excluded from our society: they do not follow the rules imposed by society and, henceforth, they do not directly contribute to it. In the context of this work, instead of searching for the objective function's optimality (i.e., the main activity of the society) as other robots in the active groups do, socially excluded robots basically randomly wander in the scenario. Note, however, that they are always aware of their individual solution and the global solution of the socially excluded group.

Nevertheless, the dynamic partitioning of the whole population of robots into multiple groups, despite the advantages inherent to it (i.e., avoid suboptimal solutions and improve scalability), still yields several issues that need to be solved.

When a robot is excluded from a group, it needs to be able to find the other excluded robots. However, it may be unable to communicate with them. As a socially excluded robot, it will randomly wander in the scenario. As it moves, the

robot will broadcast a message containing its current position. At this point, three possible situations may occur:

i. No robot ear the message and the socially excluded robot will continue to randomly wander broadcasting the same message.
ii. If a robot from an active group receives the message, it will forward it to any available robots in the vicinity.
iii. If a robot from the socially excluded group receives the message, it will answer with a message containing its current position.

If the second situation is verified and an excluded robot receives the forwarded message, it will answer with a message containing its current position. The active group will then receive the answer, thus forwarding it to the robot that first broadcast its position. If the third situation is verified, then the recently excluded robot is included in the group of excluded robots that replied to its message. However, this group may not contain all excluded robots. Therefore, the behavior of excluded robots will follow this cycle inasmuch as they are unable to identify if all other excluded robots are "connected;" they always randomly wander, considering the position of the other nearby excluded robots, while broadcasting their current position.

The opposite situation also needs to be considered: when a socially active group fulfills the necessary conditions to be rewarded with a robot or spawn a new group (Table 1.1), it broadcasts its award. Similarly as before, three situations may occur:

i. No robot ears the message and robots within the socially active group continue their mission while broadcasting their award.
ii. If a robot from an active group receives the message, it will broadcast it.
iii. If a robot from the socially excluded group receives the message, it will communicate it with the other connected excluded robots.

If the second situation is verified and an excluded robot receives the forwarded message, then the third situation occurs. Therefore, depending on the active group reward, the best-performing robot from this socially excluded group will be added to the active group or the best N_I robots from this socially excluded group will form a new group (note that this will only happen if the number of robots within this socially excluded group is equal or superior to N_I). It is noteworthy that the best-performing robots from this socially excluded group may not be the best-performing robots from all socially excluded robots inasmuch as this group may not encompass all excluded robots. Also, if a group is unable to be rewarded with a new robot before being punished, then both reward and punishment are cancelled out.

5.1.5 Outline

Let us represent each RDPSO component from Eqs. (5.1)–(5.3), by considering the geometrical illustration depicted in Fig. 5.6.

Considering a group of two robots as Fig. 5.6 depicts, the cognitive coefficient ρ_1 influences robots to improve their own individual solutions. In the case of robot 2, both cognitive ρ_1 and social coefficients ρ_2 influence toward the same position because it is the best-performing robot of the group. On the other hand, the social coefficient ρ_2 of robot 1 attracts it to the global best position found thus far by robot 2. The obstacle susceptibility weight ρ_3 influences robots to move to a previous position in which obstacles were not detected within the robot's range. As for the enforcing communication component ρ_4, because robot 2 was the first to choose robot 1 as its nearest neighbor, being at a distance inferior to $d_{\max}$, it is slightly repelled by it. It is noteworthy that the fractional coefficient α influences the next position of both robots $x_n[t+1]$, $n = 1, 2$, with an inertial factor that considers their trajectory.

To further understand RDPSO social dynamics, let us suppose a population divided into three groups of 3 robots each as Fig. 5.7a depicts. If groups 1 and 2 (red and green robots, respectively) cannot improve their objective for $SC_{\max}$ iterations, they are punished by excluding the worst-performing robot of each group and adding them to the socially excluded group (Fig. 5.7b).

The socially excluded robots randomly wander in the scenario memorizing their individual best solution and the global best solution of the socially excluded group

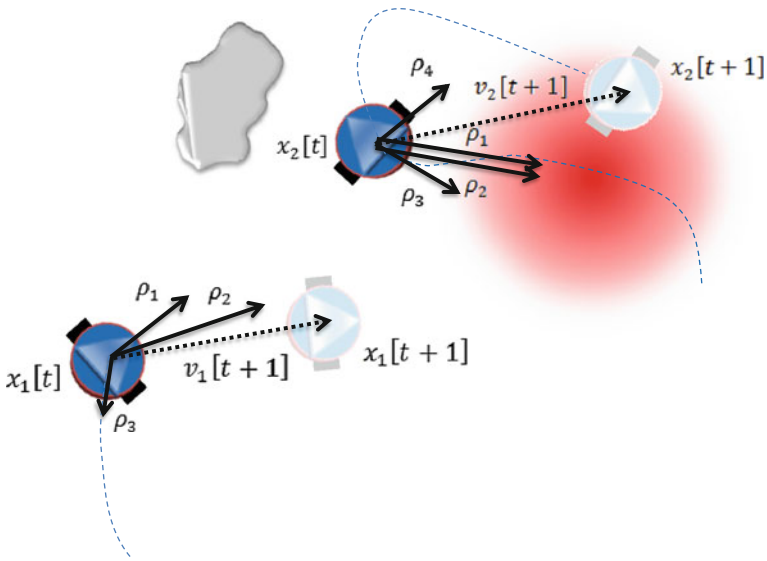

Fig. 5.6 Geometrical illustration of the RDPSO using a group of two robots

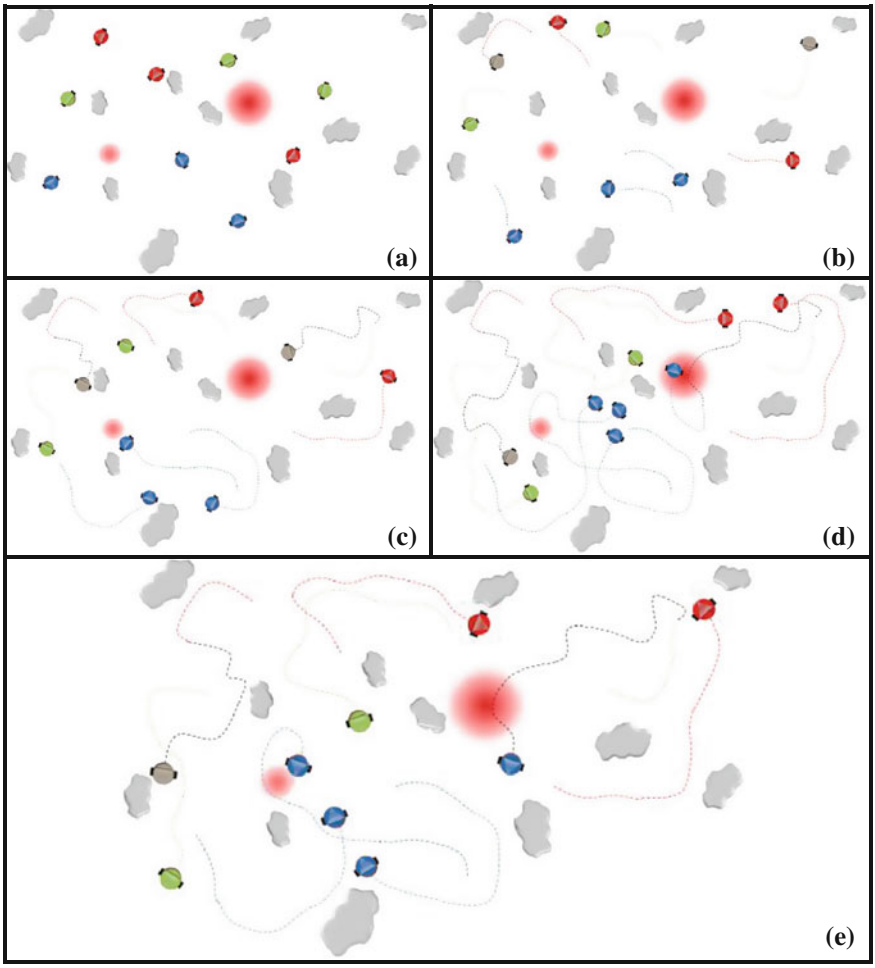

Fig. 5.7 Sequence of a MRS exploration using the RDPSO algorithm (Couceiro et al. 2013)

(Fig. 5.7c). Group 3 improves its solution, because it finds a local optimum, and it is rewarded with the best-performing robot in the socially excluded group (Fig. 5.7d). Finally, the new member of group 3 communicates its best individual solution to the other members that is better than their best global solution inducing them to move toward this new solution (Fig. 5.7e).

At least, and to complete the description of the RDPSO base mechanisms, let us summarize it with the following algorithm (Algorithm 5.2).

Wait for information about initial pose $\langle x_n[0], \phi_n[0] \rangle$ and swarmID

Loop:

- If swarmID $\neq 0$ // it is not an excluded robot
  - Evaluate its individual solution $h_n[t]$
  - If $h(x_n[t]) > h_{best}$ // robot has improved
    - $h_{best} = h(x_n[t])$
    - $\chi_1[t] = x_n[t]$
  - Exchange information with teammates about the individual solution $h_n[t]$ and current position $x_n[t]$
  - Build a vector $H[t]$ containing the individual solution of all robots within swarmID
  - If $\max H[t] > H_{best}$ // group has improved
    - $H_{best} = \max H[t]$
    - $\chi_2[t] = x_n[t]$
    - If $SC_s > 0$
      - $SC_s = SC_s - 1$ // stagnancy counter
    - If $SC_s = 0$ // the group can be rewarded
      - If $N_S < N_{max}$ and $\text{rand}(\,)\frac{1}{N_s^{kill}+1} > (\,)$ // small probability of calling a new robot
        - Broadcast the need of a new robot to any available excluded robot // Table 1.1
        - If $N_s^{kill} > 0$
          - $N_s^{kill} = N_s^{kill} - 1$ // excluded robots counter
      - If $\text{rand}(\,)\frac{N_s}{N_{max}} > (\,)$ // small probability of creating a new group
        - Broadcast the possibility of creating a new group to any available excluded robot // Table 1.1
        - If $N_s^{kill} > 0$
          - $N_s^{kill} = N_s^{kill} - 1$ // excluded robots counter
  - Else // group has not improved
    - $SC_s = SC_s + 1$ // stagnancy counter
    - If $SC_s = SC_{max}$ // punish group

If $N_S > N_{min}$ // it is possible to exclude the worst performing robot

$N_s^{kill} = N_s^{kill} + 1$ // excluded robots counter

$SC_s = SC_{max}\left[1 - \frac{1}{N_s^{kill}+1}\right]$ // reset search counter

If $h_{best} = \min H[t]$ // this is the worst performing robot

swarmID = 0 // exclude this robot

Else // delete the entire group

swarmID = 0 // exclude this robot

If $g(x_n[t]) \geq g_{best}$ // maximize distance to obstacles

$g_{best} = g(x_n[t])$

$\chi_3[t] = x_n[t]$

$[L_n, index_n] = sort_ascending(L_{n,1:N_S})$ // sort the elements of line n from link matrix L in ascending order

For $i = 1: N_S$

If $index_n(i)$ has not yet chosen it as its nearest neighbor

$\chi_4[t] = x_i[t] + d_{max}\frac{x_i[t]-x_n[t]}{\|x_i[t]-x_n[t]\|}$ // the position of the nearest neighbor increased by d_{max} toward $x_n[t]$

Communicate to robot i that it was chosen by robot n

break from For

$v_n[t+1] = w_n[t] + \sum_{i=1}^{4} \rho_i r_i\left(\chi_i[t] - x_n[t]\right)$ // Equation (5.1)

$x_n[t+1] = x_n[t] + v_n[t+1]$ // Equation (5.2)

Else // it is an excluded robot

```
Wandering algorithm // e.g., (Bräunl, 2008)
Evaluate its individual solution h_n[t]
If h(x_n[t]) > h_best // robot has improved
    h_best = h(x_n[t])
Exchange information with teammates about the individual solution h_n[t] and current position
x_n[t]
Build a vector H[t] containing the individual solution of all N_X robots within the excluded group
(swarmID = 0)
If max H[t] > H_best
    H_best = max H[t]
    If h_best = max_{N_I} H[t] // this is one of the best N_I performing robot of the excluded
    group
        If N_X ≥ N_I and rand( ) N_X/N_T > ( ) // small probability of creating a new group
            swarmID = swarmID_new // include this robot in the new active group
            Broadcast the need of N_I − 1 robots to any available excluded robot // Table
            1.1
        Else
            If receives information about the need of a new robot
                swarmID = swarmID_received // include this robot in the active group
                N_S = N_S + 1
                Exchange information with teammates about N_S
            If receives information about the need of creating a new group
                swarmID = swarmID_new // include this robot in a new active group
                N_S = N_I // reset number of robots in the group
                N_s^kill = 0 // reset number of excluded robots
                SC_s = 0 // reset search counter
until stopping criteria (convergence/time)
```

Algorithm 5.2. *RDPSO* Algorithm.

As an evaluation of the scientific content provided in this chapter, the next section provides experimental results retrieved from both simulation and laboratory experiments with physical mobile robots.

5.2 Experimental Results

5.2.1 Numerical Simulation

This section carries out a preliminary statistical analysis of the previously presented algorithm in order to evaluate the relationship between the two vital variables within swarm systems with explicit communication: the population of robots and communication constraints (Mohan and Ponnambalam 2009). In this section, the use of virtual agents in a numerical context instead of realistic robots (i.e., without considering the robots' dynamic and radio frequency propagation) was necessary to evaluate the RDPSO using statistically significant samples.

Robots were randomly deployed in the search space of 300 × 300 m (area of A = 90,000 m^2) with obstacles randomly deployed at each trial (Fig. 5.8). A Gaussian cost function $F(x, y)$ was defined where the x- and y-axis represent the planar coordinates in meters (Molga and Smutnicki 2005).

In order to improve the interpretation of the algorithm performance, results were normalized in a way that the objective of robotic teams was to maximize $f(x, y)$, that is, minimize the original benchmark functions $F(x, y)$, thus finding the optimal solution of $f(x, y) = 1$, while avoiding obstacles and ensuring the MANET connectivity:

$$f(x, y) = \frac{F(x, y) - \max F(x, y)}{\min F(x, y) - \max F(x, y)}. \tag{5.13}$$

Because the RDPSO is a stochastic algorithm, every time it is executed it may lead to a different trajectory convergence. Therefore, multiple test groups of 100 trials of 300 iterations each were considered. Independently of the population of robots, it will be used a minimum, initial, and maximum number of $N_s^{\max} = 0$ (i.e.,

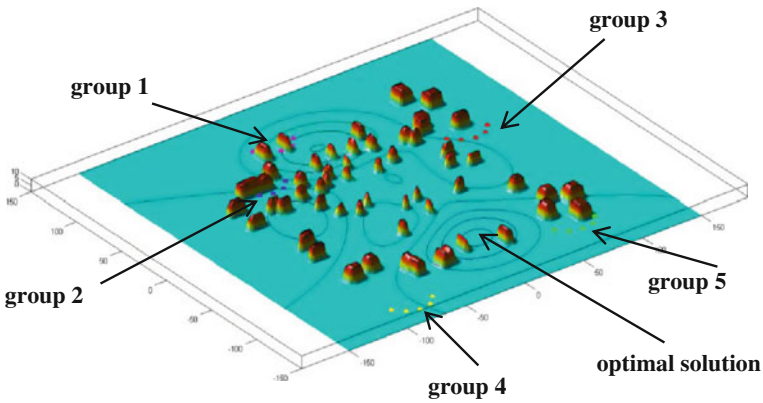

Fig. 5.8 Virtual scenario with obstacles and robots divided into five groups

all robots socially excluded), $N_s^I = 3$, and $N_s^{\max} = 6$ socially active groups (represented by different colors in Fig. 5.8), respectively. The maximum traveled distance between iterations was set as 0.5 meter; that is, $\Delta x_{\max} = \max \| x_n[t+1] - x_n[t] = 0.5$. Thus, robots moved in the 300 × 300 m environment where their solution depended on the intensity of the Gaussian function at each (x, y) position. Note that while the robots moved they needed to consider all the components within the RDPSO algorithm from the DE in Eqs. (5.1)–(5.3). Although those preliminary experiments did not consider realistic robot dynamics, they still considered the fractional-order convergence from Sect. 5.1.1. Moreover, they needed to avoid obstacles according to Sect. 5.1.2 and maintain communication with teammates from the same group according to Sect. 5.1.3.

Regarding this last point, it is important to note that trying to maintain network connectivity by only taking into account the communication range does not match reality inasmuch as the propagation model is more complex: the signal depends not only on the distance but also on the multiple paths from walls and other obstacles. However, in simulation, the communication distance is a good approach and it is easier to implement. Therefore, for the sake of simplicity and without the lack of generality, the maximum communication range was considered at this point. The maximum communication distance $d_{\max}$ will then vary depending on the chosen wireless protocol. Four conditions were described: (i) existence of a communication infrastructure (i.e., without communication constraints $\equiv d_{\max} \to \infty$), (ii) WiFi, (iii) ZigBee, and (iv) Bluetooth. Table 5.1 depicts the maximum communication distance adapted from a comparison between the key characteristics of each wireless protocol in (Lee et al. 2007). The mean between the minimum and maximum range shown in (Lee et al. 2007) was considered the maximum communication distance $d_{\max} = \{10, 55, 100, \infty\}$.

The number of robots in the swarm varied from 3 to 33 robots with incremental steps of 6 robots; that is, $N_T = \{3, 9, 15, 21, 27, 33\}$, in order to understand the performance of the algorithm while changing the population size and the maximum communication distance.

Table 5.2 summarizes the whole RDPSO configuration. Note that, thus far, we do not hold any sort of knowledge regarding the RDPSO parameters and, as such, the parameters presented in Table 5.2, namely α and ρ_i, $i = 1, 2, 3, 4$, were retrieved by trial and error based on exhaustive numerical simulations. Also note that the number of trials is considered for each different configuration, thus resulting in 2,400 trials for the 24 pairwise combinations (swarm size and communication range).

Because these simulation experiments represent a search task, it is necessary to evaluate not only the completeness of the mission but also the speed. Therefore, the

Table 5.1 Typical maximum communication distances of WiFi, ZigBee, and Bluetooth

	No Limit	WiFi	ZibBee	Bluetooth
$d_{\max}$ [m]	∞	100	55	10

Table 5.2 RDPSO parameters obtained by trial and error and used in numerical simulations

RDPSO parameter	Value
Number of trials	100
Time per trial [iterations]	300
N_T	$\{3, 9, 15, 21, 27, 33\}$
$N_s^{\min}$	0
N_s^{I}	3
$N_s^{\max}$	6
$SC_{\max}$	30
$d_{\max}$ [m]	$\{10, 55, 100, \infty\}$
$\Delta x_{\max}$ [m]	0.5
α	0.5
ρ_1	0.2
ρ_2	0.4
ρ_3	0.8
ρ_4	0.8

performance of the algorithm was evaluated through the analysis of the final global solution of the population and the runtime of the simulation. If the swarm could not find the optimal solution, the runtime was considered to be the simulation time (i.e., 300 iterations).

The significance of the maximum communication distance and the number of robots (independent variables) on the global solution and the runtime (dependent variables) was analyzed using a two-way multivariate analysis of variance analysis (MANOVA) after checking the assumptions of multivariate normality and homogeneity of variance/covariance. The assumption of normality of each of the univariate dependent variables was examined using univariate tests of Kolmogorov–Smirnov (p-value < 0.05). Although the univariate normality of each dependent variable was not verified, because the number of trials was over 30 (100), based on the central limit theorem (CLT) (Pedrosa and Gama 2004), the assumption of multivariate normality was validated (Pestana and Gageiro 2008; Maroco 2010). The assumption about the homogeneity of the variance/covariance matrix in each group was examined with Box's M test (M = 6465.13, F(69; 5368369.62) = 92.98; p-value = 0.001). Although the homogeneity of the variance/covariance matrices was not verified, the MANOVA technique is robust to this violation because all the samples have the same size (Pallant 2011). The classification of the effect size, that is, measure of the proportion of the total variation in the dependent variable explained by the independent variable, was done according to Pallant (2011). This analysis was performed using IBM SPSS Statistics for a significance level of 5 %.

The MANOVA revealed that the maximum communication distance had a small effect and significant on the multivariate composite (Pillai's trace = 0.75; F(6; 4752) = 30.974; p-value = 0.001; partial eta squared η_p^2 = 0.038; power = 1.0). The number of robots also had a small effect and significant on the multivariate composite (Pillai's trace = 0.080; F(10; 4752) = 19.706; p-value = 0.001; η_p^2 = 0.04;

Table 5.3 Tukey's HSD post hoc test to the maximum communication distance $d_{\max}$

$d_{\max}$	Final solution	Runtime
No Limit vs. WiFi	0.002*	0.854
No Limit vs. ZigBee	0.001*	0.001*
No Limit vs. Bluetooth	0.001*	0.001*
WiFi vs. ZigBee	0.207	0.019*
WiFi vs. Bluetooth	0.001*	0.001*
ZigBee vs. Bluetooth	0.001*	0.001*

*The corresponding p-value for mean difference when it is significant at the 0.05 level

power = 1.0). Finally, the interaction between the two independent variables had a small statistically significant effect on the multivariate composite (Pillai's trace = 0.032; $F(30; 4752) = 2.55$; p-value = 0.001; $\eta_p^2 = 0.016$; power = 1.0).

After observing the multivariate significance in the maximum communication distance and the number of robots, a univariate ANOVA for each dependent variable followed by Tukey's HSD test was carried out. For the maximum communication distance, the dependent variable final global solution presented statistically significant differences $\left(F(3, 2376) = 45.185; p\text{-value} = 0.001; \eta_p^2 = 0.054; \text{Power} = 10.0\right)$ and the dependent variable runtime presented statistically significant differences $(F(3, 2376) = 53.683; p\text{-value} = 0.001; \eta_p^2 = 0.063; \text{power} = 10.0)$. For the number of robots, the dependent variable final global solution also presented statistically significant differences $(F(5, 2376) = 23.347; p\text{-value} = 0.001; \eta_p^2 = 0.047; \text{power} = 10.0)$ and also the dependent variable runtime showed statistically significant differences $(F(5, 2376) = 39.816; p\text{-value} = 0.001; \eta_p^2 = 0.077, \text{power} = 10.0)$.

Using Tukey's HSD post hoc, it was possible to verify where the differences between maximum distances of communication lay. Analyzing the swarm's final solution and the runtime variables, it appears that there were statistically significant differences between experiments without communication constraints and experiments using the WiFi protocol, the ZigBee protocol, and the Bluetooth protocol (Table 5.3).

It is noteworthy that the algorithm produces better solutions without communication constraints. Also, using the WiFi protocol produces better solutions than using the ZigBee protocol and, on the other hand, this last one produces better solutions than the Bluetooth protocol as expected. In fact, using the Bluetooth protocol proves to be the worst communication protocol to employ.

Analyzing both the final global solution of the team and the runtime variables, it appears that there were statistically significant differences between a population inferior to 15 robots and a population superior to 21 robots, not showing statistically significant differences for a population between 3 to 15 robots and 21 to 33 robots (Table 5.4). Note that the worst result was obtained using 3 robots, which cannot be considered significantly worse than using 9 or even 15 robots. This may be relevant

Table 5.4 Tukey's HSD post hoc test to the total number of robots N_T

N	Final solution	Runtime
3 vs. 9	1.000	0.861
3 vs. 15	0.151	0.182
3 vs. 21	0.001*	0.001*
3 vs. 27	0.001*	0.001*
3 vs. 33	0.001*	0.001*
9 vs. 15	0.249	0.844
9 vs. 21	0.001*	0.001*
9 vs. 27	0.001*	0.001*
9 vs. 33	0.001*	0.001*
15 vs. 21	0.004*	0.001*
15 vs. 27	0.001*	0.001*
15 vs. 33	0.001*	0.001*
21 vs. 27	0.842	0.654
21 vs. 33	0.785	0.076
27 vs. 33	1.000	0.845

*The corresponding p-value for mean difference when it is significant at the 0.05 level

inasmuch as the increase in the number of robots results in an increase in the cost of the solution.

To strengthen the conclusions from Tables 5.3 and 5.4, Figs. 5.9 and 5.10 depict the estimated marginal means for both the final global solution and runtime, respectively. These figures illustrate how the performance of the RDPSO is affected under pairwise combinations between the swarm population and communication technology.

A video of these numerical experiments is provided to have a general overview of the RDPSO dynamics.[2]

Having studied the RDPSO main mechanisms in theory and numerically evaluated them, let us present experiments carried out with physical mobile robots.

5.2.2 *Laboratory Experiments*

This section explores the effectiveness of using the RDPSO on swarms of real robots, while performing a collective foraging task with local and global information under communication constraints. Multiple test groups of 20 trials of 180 s each were considered.

The eSwarBot was the platform used to evaluate the algorithm (cf., Fig. 5.1). Its low cost and high energy autonomy allowed us to perform experiments with up to

[2]http://www2.isr.uc.pt/~micaelcouceiro/media/RDPSO_numerical.wmv.

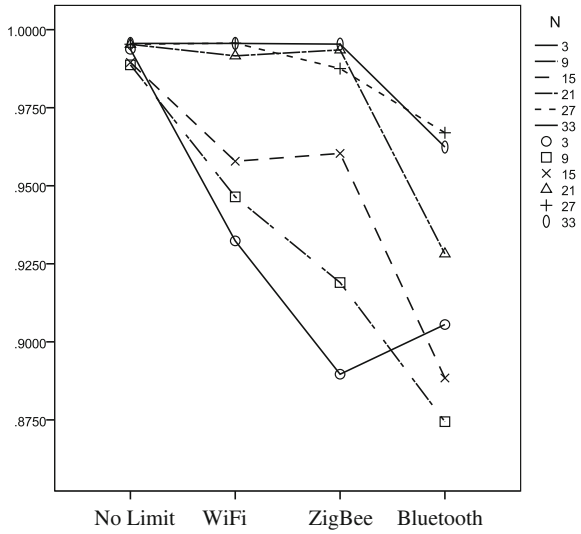

Fig. 5.9 Estimated marginal means of the final global solution

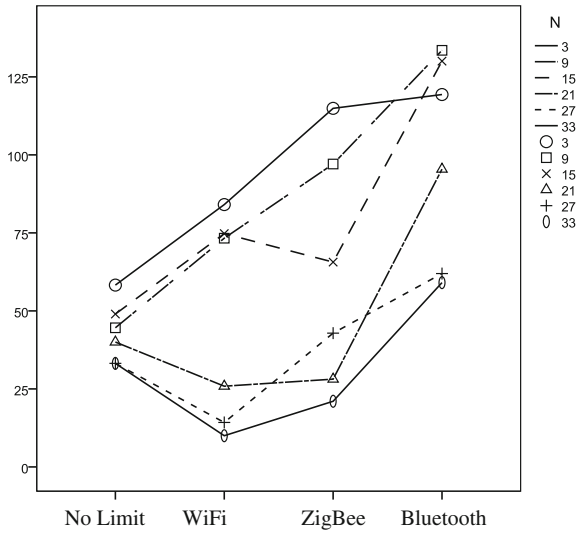

Fig. 5.10 Estimated marginal means of the runtime (number of iterations)

12 robots, with $N_T = \{4, 8, 12\}$. The RGB–LEDs on top of the eSwarBots were used to identify their group using different colors. Independently of the population of robots, a minimum, initial, and maximum number of $N_s^{\min} = 1$, $N_s^I = 3$, and $N_s^{\max} = 6$ socially active groups (represented by different colors in Fig. 5.11a), respectively, were used. The maximum traveled distance between iterations was set to 0.1 m; that is, $\Delta x_{\max} = 0.1$.

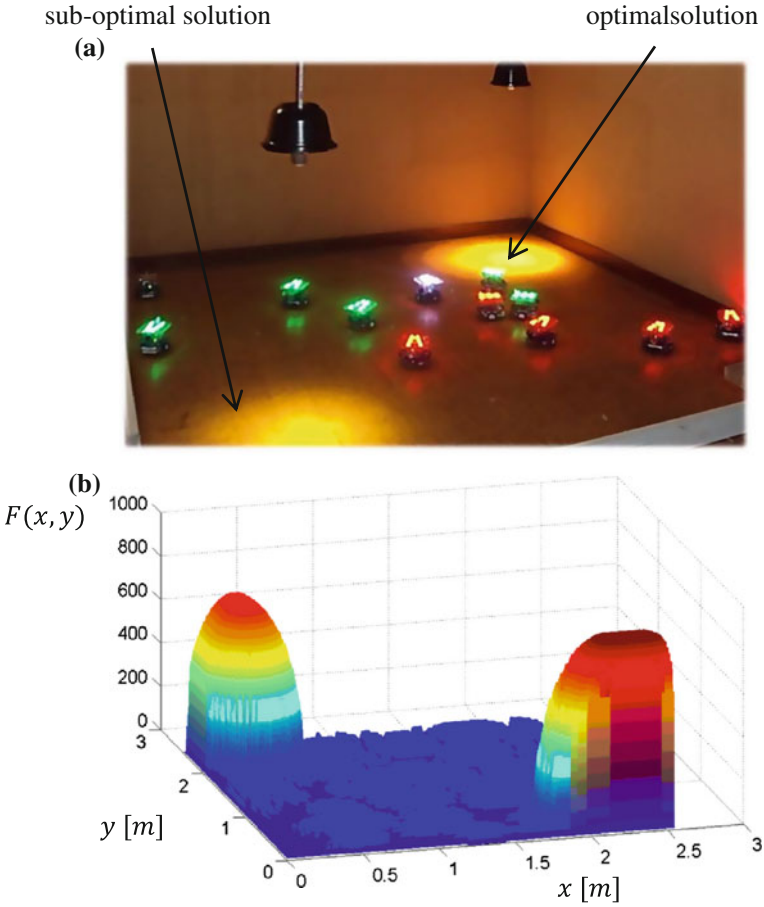

Fig. 5.11 Experimental setup: **a** enclosed arena with two groups (different colors); **b** virtual representation of the target distribution

All of the experiments were carried out in a 2.55 × 2.45 m scenario ($A \approx 6.25$ m^2). The experimental environment (Fig. 5.11a) was an enclosed arena containing two sites represented by illuminated spots uniquely identifiable by controlling the brightness of the light. Despite being an obstacle-free scenario, the robots themselves act as dynamic obstacles: a maximum number of 12 robots correspond to a population density of approximately 2 robots. m-2. Each eSwarBot possesses overhead light sensors (LDR) that allow it to find candidate sites and measure their quality. The brighter site (optimal solution) was considered better than the dimmer one (suboptimal solution), and so the goal of the swarm robots was collectively to choose the brighter site. The intensity values $F(x, y)$ represented in Fig. 5.11b were obtained sweeping the whole scenario with a single robot in which

Table 5.5 RDPSO parameters obtained by trial and error and used in its first evaluation with physical robots

RDPSO parameter	Value
Number of trials	20
Time per trial [s]	180
N_T	$\{4, 8, 12\}$
$N_s^{\min}$	0
N_s^{I}	2
$N_s^{\max}$	3
$SC_{\max}$	30
$d_{\max}$ [m]	$\{0.5, 1.5\}$
$\Delta x_{\max}$ [m]	0.1
α	0.5
ρ_1	0.2
ρ_2	0.4
ρ_3	0.8
ρ_4	0.8

the light sensor was connected to a 10-bit analog input. The maximum intensity values obtained (hot color in Fig. 5.11b) was found between 800 and 860.

Interrobot communication to share positions and local solutions was carried out using the ZigBee 802.15.4 wireless protocol. Robots were equipped with XBee modules allowing a maximum communication range larger than the whole scenario, therefore robots were provided with a list of their teammates' addresses in order to simulate MANET communication with limited range. The maximum communication distance between robots $d_{\max}$ was varied between 0.5 and 1.5 m. At each trial, robots were manually deployed in the scenario while preserving the maximum communication distance $d_{\max}$.

Table 5.5 summarizes the whole RDPSO configuration. Note that, once again, we still do not have any sort of knowledge regarding the RDPSO parameters and, as such, the parameters presented in Table 5.5 were the same as previously retrieved by trial and error (as in the previous section).

The previously described conditions give a total of 120 experimental trials, thus leading to a runtime of 6 h. The sequence of frames in Fig. 5.12 presents a trial of the team's performance with $N = 12$ and $d_{\max} = 1.50$ m.

Once again, let us evaluate both the completeness of the mission and the time needed to complete it. Figure 5.13 depicts the convergence of the RDPSO for the several proposed conditions. The median of the best solution in the 20 experiments was taken as the final output in the set $N = \{4, 8, 12\}$ for each $d_{\max}$.

Analyzing Fig. 5.13, it is clear that the proposed mission could be accomplished by any number of robots between 4 and 12. In fact, independently of the number of robots, the swarm converged to the solution in approximately 90 % of the experiments. The charts also show that increasing the number of robots slightly increased the runtime needed to accomplish the mission. A population of 4, 8, and 12 robots took an average of 77, 106, and 112 s to converge to the optimal solution,

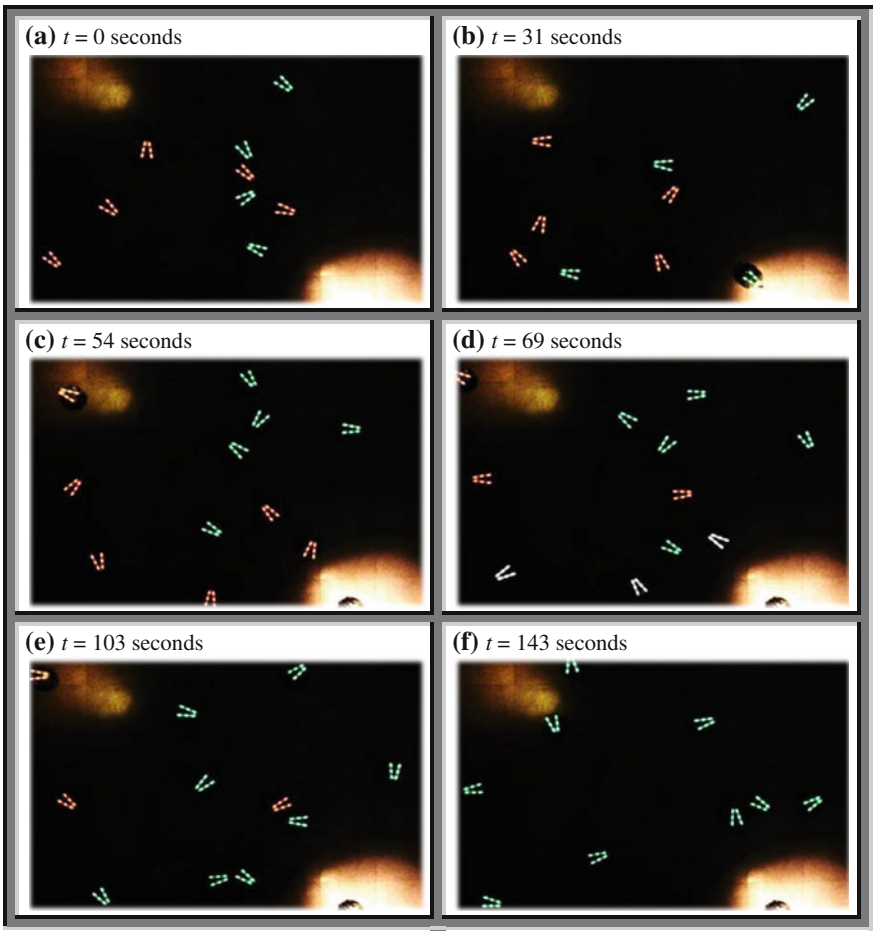

Fig. 5.12 Frame sequence showing the RDPSO performance on a population of 12 robots (some robots may be outside camera range). **a** The population is initially divided into two swarms—*green* and *red*—deployed in a spiral manner; **b** the swarms independently search for the brighter site taking into account a maximum communication distance of 1.5 m between robots of the same swarm; **c** one robot from the *red* and *green* swarm finds the suboptimal and optimal solution, respectively; **d** as the *red* swarm does not improve, some robots are excluded, thus being added to the socially excluded group (*white* swarm); **e** because the *green* swarm has improved, it is able to call new members from the socially excluded group; **f** finally, the *green* swarm proliferates, calling all the previously excluded robots that were unable to improve their solution. Note that robots do not all converge the optimal solution as they try to maintain a distance of $d_{\max}$ between them

respectively. This is a consequence of having more robots inside the same arena: the number of dynamic obstacles is higher. As expected, increasing the maximum communication distance generally resulted in a faster convergence to the optimal solution. However, this relationship was found to be not linear and varied depending on the number of robots in the population. For instance, for a swarm

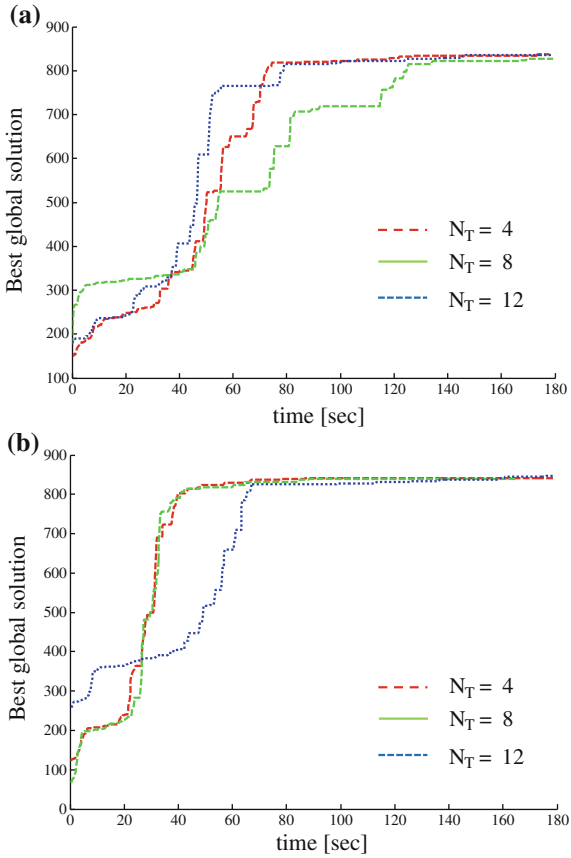

Fig. 5.13 Performance of the algorithms changing the number of robots N in the population: **a** $d_{\max} = 0.5$ m; **b** $d_{\max} = 1.5$ m

population of 12 robots, the performance slightly decreased for a communication distance of $d_{\max} = 1.5$ m. This is an expected factor inasmuch as the communication interference between robots increases with the population, and, as a consequence, the existence of repulsive forces to maintain ideal interrobot distances of 1.5 m constrains the robots' motion ($\chi_4[t]$ from Eq. (5.1)). Adding this to the fact that the scenario was considerably small, yielded a considerably large amount of constraints thus jeopardizing the success of the mission.

Another important factor is that some robots of a given swarm were unable to converge to the final solution when one robot of the same swarm found it. This issue was related to odometry limitations of the platforms that resulted in the accumulation of positioning errors. The use of encoders such as the ones used in these robots is a classical method, being of low cost and simple use. However, this yielded to the existence of cumulative errors inherent to their use, which makes it difficult for the robots to complete the proposed odometry objectives accurately.

A video of the real experiments is provided to better understand how real robots behave under the RDPSO algorithm.[3]

5.3 Summary

This chapter described the extension of the FODPSO to swarm robotics by presenting the key points behind the development of the robotic Darwinian particle swarm optimization (RDPSO). The RDPSO integrates insights from the FODPSO presented in Chap. 2, by adapting them to real-world multirobot system (MRS) features and a more realistic, or robotic, natural selection mechanism. This last feature is achieved by creating some social exclusion and inclusion rules to manage the whole swarm into clusters, herein denoted groups. The dependency between groups gives rise to a competitive evolutionary process mimicking animal nature following the Darwinian survival-of-the-fittest principle. On the other hand, as with many other biological societies involved in diverse survival conditions, the outcome of this competitive evolutionary process is reflected in social cooperation among members from the same group. This is a highly recurrent process in nature denoted coopetition (Tsai 2002). For instance, certain birds are unable to reach parasites on some parts of their bodies, thus benefiting from preening one another. Hence, there is an entire flock of potential preeners that compete in hopes of establishing a beneficial cooperative relationship. To the similarity of the RDPSO, birds that try to be preened without preening others are excluded from these relationships as they do not compete.

The features presented in this chapter were first implemented in a numerical simulation environment in MATLAB® and afterwards further validated using real eSwarBot platforms. Experimental results show the performance of a MRS with biologically inspired behavior based on natural selection and social exclusion. As expected, the influence inherent to communication's limitations can be attenuated as the number of robots or the communication range/quality increases. This is a promising result for communities of swarm robots with many individuals because they can develop efficient coordination techniques, just like natural swarm agents, allowing cooperative and competitive work in large and superlarge societies.

References

Balch, T., & Hybinette, M. (2000). Social Potentials for Scalable Multi-Robot Formations. In *Proceedings of the 2000 IEEE International Conference on Robotics, ICRA*. San Francisco, CA, USA.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. New York: Oxford University Press.

[3] http://www2.isr.uc.pt/~micaelcouceiro/media/RDPSO_initial.mp4.

Burchardt, T. (2000). Social exclusion: Concepts and evidence. In D. G. P. Townsend (Ed.), *Breadline Europe: The measurement of poverty*. Bristol: The Policy.

Couceiro, M. S., Figueiredo, C. M., Luz, J. A., Ferreira, N. M., & Rocha, R. P. (2011). A low-cost educational platform for swarm robotics. *International Journal of Robots, Education and Art, 2* (1), 1–15.

Couceiro, M. S., Martins, F. M., Rocha, R. P., & Ferreira, N. M. (2012). Introducing the fractional order robotic Darwinian PSO. In *Proceedings of the 9th International Conference on Mathematical Problems in Engineering, Aerospace and Sciences (ICNPAA2012)* (pp. 242–251). Vienna, Austria.

Couceiro, M. S., Rocha, R. P., & Ferreira, N. M. (2013). A PSO multi-robot exploration approach over unreliable MANETs. *Advanced Robotics, 27*(16), 1–14.

Craighead, J. J., Sumner, J. S., & Mitchell, J. A. (1995). *The grizzly bears of Yellowstone: Their ecology in the Yellowstone ecosystem*. Washington, D.C., USA: Island Press.

Crispin, Y. J. (2009). Cooperative control of multiple swarms of mobile robots with communication constraints. In LNCIS, & M. Hirsch (Ed.), *Optimization and cooperative control* (pp. 207–220). Berlin, Heidelberg: Springer.

Kulkarni, R. V., & Venayagamoorthy, G. K. (2010). Bio-inspired algorithms for autonomous deployment and localization of sensor nodes. *IEEE Transactions on Systems, Man and Cybernetics, 40*(6), 663–675.

Lee, J. S., Su, Y. W., & Shen, C. C. (2007). A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi. In *Proceedings of the 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON'07)* (pp. 46–51).

Maroco, J. (2010). *Análise Estatística com utilização do SPSS*. Lisboa, Portugal: Edições Silabo.

Miller, L. E. (2001). Multihop connectivity of arbitrary networks. In *Wireless Communication Technologies Group, NIST*.

Mohan, Y., & Ponnambalam, S. G. (2009). An extensive review of research in swarm robotics. *World Congress on Nature and Biologically Inspired Computing (NaBIC'09)* (pp. 140–145). Coimbatore, India.

Molga, M., & Smutnicki, C. (2005). Test functions for optimization needs. *Engineering Optimization: An Introduction with Metaheuristic Applications*, 1–43.

Nicholson, T. (2008). *Symposium on social inclusion down under* (pp. 1–5). Melbourne, Australia: Brotherhood of St Laurence.

Pallant, J. (2011). *SPSS Survival Manual* (4th ed.). New York: Open University Press.

Pedrosa, A. C., & Gama, S. M. (2004). *Introdução Computacional à Probabilidade e Estatística*. Portugal: Porto Editora.

Pestana, M. H., & Gageiro, J. N. (2008). *Análise de dados para Ciências Sociais - A complementaridade do SPSS* (5th ed.). Lisboa, Portugal: Edições Sílabo.

Podlubny, I. (1999). *Fractional differential equations* (198 ed., Vol. 198). San Diego, California: Academic Press.

Scutella, R., Wilkins, R., & Horn, M. (2009). *Measuring poverty and social exclusion in australia: A proposed multidimensional framework for identifying socio-economic disadvantage* (Vol. 4). Victoria, Australia: Melbourne Institute Working Paper Series.

Sheng, W., Yang, Q., Tan, J., & Xi, N. (2006). Distributed multi-robot coordination in area exploration. *Robotics and Autonomous Systems, 54*(2006), 945–955.

Tardioli, D., & Villarroel, J. L. (2007). Real time communications over 802.11: RT-WMP. *IEEE International Conference on Mobile Adhoc and Sensor Systems* (pp. 1–11). Pisa, Italy.

Tsai, W. (2002). Social structure of "Coopetition" within a multiunit organization: Coordination, competition, and intraorganizational knowledge sharing. *Organization Science, 13*(2), 179–190.

Williams, R. L., & Wu, J. (2010). Dynamic obstacle avoidance for an omnidirectional mobile robot (S. Arimoto, Ed.). *Journal of Robotics, 2010*(901365), 1–15.

Chapter 6
Conclusions

Abstract Narrowing down all that was previously presented to a sentence, the focus of this short book was the bottom-up applicability of swarm intelligence to solve multiple different problems, such as typical curve fitting, the relevant image segmentation process, and even the more technologically oriented swarm robotics. This final chapter summarizes the research covered around a novel PSO-based algorithm, denoted fractional-order Darwinian particle swarm optimization (FODPSO). After discussing the presented contributions, and considering their advantages and limitations, it points out perspectives on future research.

Keywords FODPSO · Contributions · Discussion · Future work

6.1 Main Contributions

In this book, we have contributed to the state of the art on the use of swarm intelligence to solve different real-world problems. We presented the development of a novel algorithm, denoted FODPSO, and its applicability considering multiple approaches. All these were validated numerically, analytically, or experimentally. These and other examples considering the FODPSO algorithm are provided to the community at the MATLAB® Central File Exchange.[1]

The FODPSO was adapted and applied to three key problems: (i) curve fitting, (ii) image segmentation, and (iii) swarm robotics. In these three case studies, the results proved the superiority of the FODPSO over other PSO-based approaches

[1] http://www.mathworks.com/matlabcentral/fileexchange/49148-fodpso-and-svm-based-feature-selection-approach
http://www.mathworks.com/matlabcentral/fileexchange/47149-fodpso-for-fitting
http://www.mathworks.com/matlabcentral/fileexchange/46473-fractional-order-darwinian-particle-swarm-optimization
http://www.mathworks.com/matlabcentral/fileexchange/29517-segmentation
http://www.mathworks.com/matlabcentral/fileexchange/38409-mrsim-multi-robot-simulator–v1-0-.

© The Author(s) 2016

M. Couceiro and P. Ghamisi, *Fractional Order Darwinian Particle Swarm Optimization*, SpringerBriefs in Applied Sciences and Technology,
DOI 10.1007/978-3-319-19635-0_6

and even traditional exhaustive and nonexhaustive optimization methods, such as gradient descent or simplex. In every case study, the key advantage of using the FODPSO algorithm was identified as its ability to provide improved convergence toward the solution, while avoiding suboptimality.

Yet, despite the main contributions, this book also has some limitations. In the following section, and in spite of these limitations, we briefly discuss possible extensions of the presented work and future lines of research.

6.2 Limitations and Further Research

Despite the positive results achieved using the FODPSO algorithm, its memory complexity is larger than the alternatives because it intrinsically has memory properties related to the fractional extension. In other words, even considering the truncation order of the approximate fractional difference, it needs to track the last four steps of each particle's velocity. Although this may not be relevant in most problems, the last case study presents a valid example wherein this may be seen as such: swarm robots have limited processing and memory power due to their simplicity. Therefore, a potential topic deserving future research is the development of parallel implementations of FODPSO in a distributed fashion, either considering high-performance computing architectures, or throughout multiple robotic units.

Additionally, the FODPSO may be more acutely adapted to each case study, depending on the specifications at stake. For instance, conventional best-known clustering techniques, such as fuzzy C-means (FCM) and K-means, have been widely used in image processing and computer vision. However, they are known for being very sensitive to their initial cluster configuration and may fall into suboptimal solutions (Wang et al. 2006). In order to address this issue, researchers have tried to improve the resilience of the K-means and FCM approaches by optimizing them with bioinspired optimization techniques (e.g., PSO (Kennedy and Eberhart 1995)). As mentioned in this book, a general problem with these techniques, such as the PSO algorithm, is that they may get trapped in local optimum points in such a way that they may be successful in some problems, but fail in others (Couceiro et al. 2012). As a result, one may consider optimizing those clustering techniques with FODPSO. Inasmuch as FODPSO benefits from a coopetition paradigm, in which particles within each swarm cooperate with one another, whereas multiple swarms compete to find the optimal solution, FODPSO might outperform the remaining alternatives in terms of improved convergence toward better solutions, for example, the best cluster center found among all best cluster centers provided by each particle.

References

Couceiro, M. S., Rocha, R. P., Ferreira, N. M. F., & Machado, J. A. T. (2012). Introducing the fractional order Darwinian PSO. *Signal, Image and Video Processing*, *6*(3), 343–350 (2012). doi: 10.1007/s11760-012-0316-2

Kennedy, J., & Eberhart, R. (1995). A new optimizer using particle swarm theory. In *Proceedings of IEEE Sixth International Symposium on Micro Machine Human Science* (Vol. 34, Issue 2008, pp. 39–43).

Wang, W., Zhang, Y., Li, Y., & Zhang, X. (2006). The global fuzzy c-means clustering algorithm. In *Intelligent Control and Automation*, pp. 3604–3607.