

A New Library of Bio-Inspired Algorithms

Natiele Lucca and Claudio Schepke(✉)

Laboratory of Advanced Studies, Postgraduate Program in Software Engineering, Federal University of Pampa (UNIPAMPA), Alegrete, RS, Brazil
natielelucca@gmail.com, claudioschepke@unipampa.edu.br

Abstract. Real engineering, science, and economics problems cannot be ever solved exactly due to the high computation time to find the optimal solution. One way to solve such problems is to apply bio-inspired algorithms, to minimize the time to search for potential solutions. Bio-inspired algorithms are based on the collective behavior of social organisms and are used to solve optimization problems. This article presents a new library of bio-inspired algorithms. The library offers the implementation of some algorithms, being easily extensible through interfaces. An evaluation was made using 7 test functions applied to each of the implemented algorithms. The tests have shown that the ABC algorithm obtained the best convergence results in 5 tests and the ACO algorithm in 2 tests.

Keywords: Bio-inspired algorithms · Library · Artificial bee colony · Ant colony optimization · Particle swarm optimization

1 Introduction

The resolution of a problem may not be achieved exactly due to the complexity given by a large number of variables and/or potential solutions. An algorithm that describes a problem sequentially may require a large processing capacity to find the optimal solution, that is, the best solution to the problem [12]. The time required for exact and sequential computing can vary from hours to years, which highlights the importance of approaches and techniques that optimize the execution of algorithms [8]. The problems mentioned above can be identified in several research areas, for example, in simulations of real activities involving engineering, science, and economics. Traveling salesman and vehicle routing are examples of classic problems.

In this context, bio-inspired algorithms are an strategy to reduce the execution time, so that the solution is improved. These algorithms apply swarm intelligence or colony intelligence, or collective intelligence, which simulate the collective behavior of self-organized, distributed, autonomous, flexible, and dynamic systems [9]. The elements that integrate the swarm or colony can optimize a global objective through the collaborative search in space following specific rules

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12249, pp. 474–484, 2020.
https://doi.org/10.1007/978-3-030-58799-4_35

[7]. According to [6], algorithms of flocks of birds and the colony of social insects, such as ants and bees, are known examples in the literature.

This article features a new library[1] for problem-solving using bio-inspired algorithms. The library implements the classic algorithms: Particle Swarm Optimization (PSO); Artificial Bee Colony (ABC); and Ant Colony Optimization (ACO), in addition to being extensible to other algorithms through interfaces created in C++ language.

The purpose of this article is to evaluate the functioning of the proposed bio-inspired algorithm library through test cases. With this, we want to measure the quality of convergence of numerical solutions. The contribution of this article is (a) to provide a new extensible library of bio-inspired algorithms; (b) evaluate the quality of the solutions.

The paper is organized as follows. Section 2 presents the related works in a systematic way. Section 4 describes the development and testing methodology used to carry out this work. In the Sect. 5 the results are presented. Finally, in Sect. 6, the final considerations about the work are highlighted.

2 Related Works

A set of 25 optimization libraries available in the literature are present in [10]. The authors perform a comparative analysis, identifying the strengths and weaknesses of each library. The article looks at possible shortcomings and possibilities for future work since it seeks to provide complete material on the optimization tools. Of the tools evaluated, only six have bio-inspired implementations: ECJ[2], EvA2[3], FOM[4], ParadisEO[5], MALLBA[6], and Opt4[7]. In addition to these libraries, we also found new implementations: PaGMO[8] and PyGMO[9] libraries.

Table 1 characterizes the libraries of bio-inspired algorithms found in the literature. For each library, the following were identified: the programming language, the algorithms implemented, and whether licenses are required. All libraries have some selected algorithms implemented and the mostly use open source license.

In our work, we propose a library implemented in C++ that provides the implementations of the PSO, ABC, and ACO algorithms, in addition to enabling the extension of new classes of bio-inspired algorithms.

[1] Available at https://github.com/NatiLucca/Bio-inspired-library.
[2] Available at https://cs.gmu.edu/~eclab/projects/ecj/.
[3] Available at http://www.ra.cs.unituebingen.de/software/EvA2/.
[4] Available at http://www.isa.us.es/fom.
[5] Available at http://paradiseo.gforge.inria.fr/.
[6] Available at http://neo.lcc.uma.es/mallba/easy-mallba/.
[7] Available at http://opt4j.sourceforge.net/.
[8] Available at: http://esa.github.io/pagmo/index.html.
[9] Available at: http://esa.github.io/pygmo/index.html.

Table 1. Natural computing libraries

Libraries	Language	Algorithms	Licence
ECJ	Java	GA, PSO and DE	Open Source
Eva2	Java	ES, DE and PSO	LGPL
FOM	Java	SA, ACO and ES	LGPL
MALLBA	C++	SA, GA, PSO and ACO	Open Source
Opt4	Java	DE, PSO and SA	LGPL
ParadisEO	C++	PSO	CECILL
PaGMO	C++	ABC, PSO, SA and GE	Open Source
PyGMO	Python	PSO, ABC, GA and ACO	Open Source

3 Swarm Intelligence

Swarm intelligence is a research area that explores the characteristics of swarms in the face of the way they seek food and interact. This approach allows real problems in different areas to be solved [6].

The Artificial Bee Colony (ABC) algorithm simulates the behavior of bees in collecting nectar. The colony is formed by three groups of bees: employed bees who explore and seek food sources that are possible solutions to the problem; onlooker bees that are recruited by the employed according to the quality of the source found; and scout bees that are employed bees that have found a bad source and abandoned them.

The Ant Colony algorithm was proposed by [4] and inspired by the behavior of ants in the search for food. Ants leave the anthill in search of food sources, that is, they explore the search space by releasing pheromone along the way. During a certain period of time the ants search for a source of food, but those that find a source closer to the anthill concentrate a greater amount of pheromone, as they travel the path more than once. Therefore, the more chemistry a trail is, that is, the higher the concentration of pheromone, the more ants will be attracted to it. Thus, most ants tend to follow the same path [9].

The Particle Swarm optimization algorithm (PSO) was modeled after the social behavior of the flock of birds [5]. In PSO, the population of individuals or particles is grouped into a cluster (set of solutions). These particles simulate the behavior of birds through self-learning (cognitive component) and flock learning (social component), that is, they imitate their own success and that of neighboring individuals. From this behavior the particles can define new positions that direct them towards optimal solutions.

Other examples of algorithms in this area of bio-inspired computing are the Genetic Algorithms (GS), Differential Evolution (DE), Simulated Annealing (SA), Evolution Strategies (ES) and Bee colonies Optimization (BCO).

4 Methodological Aspects

In this section, we present the methodological aspects related to the development, tests, and execution of the library.

4.1 Library Implementation

The bio-inspired library was developed in the C++ programming language, to facilitate programming and have resources that allow solving problems efficiently. This language is oriented to objects so it allows abstraction, encapsulation, inheritance, and polymorphism, which is essential for good code generation practices. The language also offers libraries for manipulating vectors and matrices and for generating random values. These contribute to the implementation of swarm intelligence algorithms.

The bio-inspired computation algorithms selected for this study belong to the swarm intelligence class, thus simulating the behavior of agents before the colony. The ABC, ACO, and PSO algorithms were chosen, as they have desirable characteristics for the optimization of problems that cannot be solved or solved in polynomial time.

The source code of the ABC, ACO and PSO algorithms is organized in 3 directories: *include*, *main* and *src*. The first directory contains the files of extension *hpp*, that is, the header files that define the classes and methods. The *main* directory contains the execution initialization file, the implementation of the benchmark equations used in the tests, and an auxiliary file that contains the methods for executing. The *src* directory contains the implemented source codes. In addition to the directories, the library also contains a script for executing the tests, called *run.sh* and a file *README*, a text document with information and usage details for the library. The Table 2 shows the related file structure for each algorithm.

Table 2. File structure implementation

Directory	ABC	ACO	PSO
include	abc.hpp	aco.hpp	pso.hpp
	bee.hpp	ant.hpp	particle.hpp
	benchmarking.hpp	benchmarking.hpp	benchmarking.hpp
main	main.cpp	main.cpp	main.cpp
	benchmarking.cpp	benchmarking.cpp	benchmarking.cpp
	auxiliar.cpp	auxiliar.cpp	auxiliar.cpp
src	abc.cpp	aco.cpp	pso.cpp
	bee.cpp	ant.cpp	particle.cpp
.	run.sh	run.sh	run.sh
	README.md	README.md	README.md

To use the library, the user must instantiate an object that corresponds to the class of the chosen algorithm.

4.2 Test Functions

To validate the implementation, it is necessary to verify the characteristics of the algorithm through tests. In this work, the tests were done by functions that have different properties, to ensure that the tested algorithm can or cannot solve certain types of optimization efficiently [13]. The quality of the optimization functions can be assessed by the properties: unimodal and multimodal; two-dimensional and N-dimensional; continuous and non-continuous; and convex, and non-convex [13]. A function that has only one optimal location is called a unimodal and functions with two or more optimal locations are called multimodal. Functions over two-dimensional space are classified as two-dimensional. The other functions are classified in this work as multidimensional or N-dimensional. A function is continuous when all points p in your domain D follow Eq. 1.

$$\lim_{x \to p} f(x) = f(p) \tag{1}$$

A function is convex when the secant line joining two points is above the graph or if any point $m \in D$ follows Eq. 2.

$$f''(m) > 0 \tag{2}$$

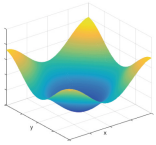
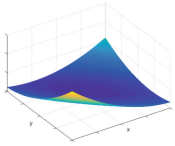
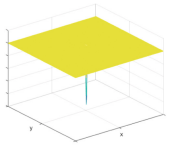
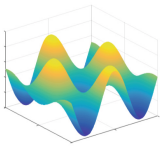
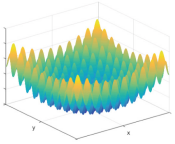
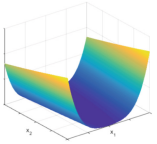
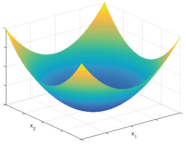
Thus, to test the algorithms implemented in the bio-inspired library, a *benchmark* was developed as a base application composed of seven test functions. They are: *Alpine*, *Booth*, *Easom*, *Griewank*, *Rastrigin*, *Rosenbrock* and *Sphere*. These functions are presented in Table 3. The characteristics of the functions such as agent size, search space, maximum number of iterations and minimum global were established according to [2,9], being described in Table 4. All the minimization functions used in the tests are continuous.

4.3 Execution of the Tests

The final results comprise the average of 30 executions, as adopted in the work of [3]. An execution comprises the generation of the input files and execution. Input files are generated using the *runDetails* method. This method creates a folder called *Inputs*, which stores the input files generated during the method's execution. Then the sequential version is executed by the *run* method, and the population and the other random values of the code are initialized with the values of the files previously generated in the *runDetails* method. The *run* method return the best solution found by the algorithm and the execution time in seconds.

The tests were performed on a workstation composed of 2 Intel® Xeon® Silver 4116 processors CPU. Each processor has 12 physical cores (24 threads) operating at the standard 2.1 GHz frequency and 3 GHz turbo frequency. The

Table 3. Benchmark functions [2]

Benchmark	Function	Graphic
Alpine	$f(\mathbf{x}) = \sum i = 1^n \lvert x_i sin(x_i) + 0.1 x_i \rvert$	
Booth	$f(x, y) = (x + 2y - 7)^2 + (2x + y - 5)^2$	
Easom	$f(x, y) = (-1) * \cos(x) * \cos(y) * e^{-[(x-\pi)^2-(y-\pi)^2]}$	
Griewank	$f(\mathbf{x}) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})$	
Rastrigin	$f(\mathbf{x}) = 10n + \sum_{i=1}^{n} (x_i^2 - 10cos(2\pi x_i))$	
Rosenbrock	$f(\mathbf{x}) = \sum_{i=1}^{n} [b(x_{i+1} - x_i^2)^2 + (a - x_i)^2]$	
Sphere	$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$	

main memory (RAM) is 96 GB in size and DDR4 technology. The operating system is Linux Debian kernel version 4.19.0-8 using GNU GCC 9.3 compiler with `-O3` optimization flag and standard `c++11`.

Table 4. Parameters of the benchmark functions [2]

Benchmark	Dimensions	Limit of search	Iterations	Minimal global	Multimodal	Convex
Alpine	10	[0, 10]	3000	f(0,...,0) = 0	Yes	No
Booth	2	[−10, 10]	1000	f(1,3) = 0	No	Yes
Easom	2	[−100, 100]	1000	$f(\pi, \pi) = -1$	Yes	No
Griewank	50	[−100, 100]	5000	f(0,...,0) = 0	No	No
Rastrigin	50	[−5.12, 5.12]	5000	f(0,...,0) = 0	Yes	Yes
Rosenbrock	50	[−5, 10]	5000	f(1,...,1) = 0	Yes	No
Sphere	6	[−5.12, 5.12]	3000	f(0,...,0) = 0	No	Yes

4.4 Test Automation

Test launches are automated through instructions in *shell script*. This language allows the creation of folders to store the output files of each test and the creation of *loops* on a command line, allowing the launch of 30 repetitions on each step of the execution. The language also makes it possible to redirect the output of methods to a file. In this way, the results of the executions are stored in a file of type `.csv`, with the name of the function, where each line of the file corresponds to the solution and the measured execution time.

According to the studies of [1,11] the number of agents responsible for determining the optimal global of a function corresponds to the range of 20 to 60 agents. Thus, the functions were tested for a population of 10 to 100 agents, varying from 10 to 10 units.

5 Experimental Results

Table 5 respectively show the results obtained in the tests for the PSO, ABC, and ACO algorithms, applying each of the 7 test functions and varying the size of the population. The tables show the numerical average of the solutions and average execution time in seconds.

As shown in Table 4, all tests have a global minimum value to be found equal to 0, with the exception of Easom, which has a minimum value of −1. Therefore, it is expected that the algorithms reach this value as the closest possible. Most of the algorithms evaluated obtained optimal solutions for the test functions analyzed, especially the ABC algorithm. However, the functions *Rastrigin* and *Rosenbrock* obtained, on average, less precise solutions, even using a larger number of iterations, as they are complex functions, since they have greater dimensionality.

For the *Alpine* function, the ABC provided the best solution of all algorithms and ACO has close results to them. However, PSO has the best execution time for all populations evaluated and ABC has close results to them. ACO has the worst execution time. In *Booth* function tests, ABC has the best solution. PSO and ACO are close results with each other. PSO has the best execution time, followed by ABC and ACO, respectively. In the *Easom* function, all algorithms present results close to the ideal −1. However, ABC found the exact result in 7

Table 5. PSO, ABC and ACO test results

		Solutions			Time (s)		
Function	Pop.	ABC	ACO	PSO	ABC	ACO	PSO
Alpine	10	0,00E+00	4,07E-07	7,99E+00	0,079604	0,799142	0,062815
	20	0,00E+00	2,37E-15	5,56E+00	0,156575	1,680333	0,123000
	30	0,00E+00	9,01E-12	4,85E+00	0,233024	2,679626	0,167739
	40	0,00E+00	1,99E-19	3,47E+00	0,300065	3,697194	0,213916
	50	0,00E+00	3,49E-01	3,03E+00	0,378131	4,773935	0,268318
	60	0,00E+00	2,94E-01	1,95E+00	0,448755	5,854582	0,318456
	70	0,00E+00	5,88E-03	3,44E+00	0,525266	6,971155	0,370068
	80	0,00E+00	2,07E-27	4,02E+00	0,604550	8,100047	0,420565
	90	0,00E+00	3,13E-20	3,51E+00	0,689727	9,370560	0,469736
	100	0,00E+00	3,43E-25	2,09E+00	0,753482	10,468643	0,520913
Booth	10	1,51E-16	7,83E-01	1,16E-02	0,030539	0,566618	0,012690
	20	4,14E-17	6,50E-01	1,82E-02	0,044914	1,216361	0,020079
	30	3,85E-17	6,24E-01	1,47E-05	0,083222	1,920431	0,028284
	40	3,53E-17	6,63E-01	6,99E-04	0,102553	2,742401	0,037351
	50	2,26E-17	6,03E-01	7,98E-05	0,107827	3,457020	0,045551
	60	1,39E-17	4,26E-01	2,28E-03	0,125350	4,303888	0,054218
	70	1,34E-17	8,20E-01	7,74E-03	0,146475	5,077780	0,061906
	80	1,30E-17	4,31E-01	1,88E-03	0,164568	5,840491	0,072140
	90	1,40E-17	5,82E-01	1,93E-03	0,183986	6,744297	0,079553
	100	1,40E-17	7,62E-01	7,59E-03	0,206001	7,566422	0,089137
Easom	10	−8,32E-01	−8,28E-01	−2,33E-01	0,033830	0,541508	0,014166
	20	−9,62E-01	−8,56E-01	−4,33E-01	0,066373	1,181808	0,026534
	30	−1,00E+00	−8,82E-01	−5,22E-01	0,093602	1,866469	0,035950
	40	−1,00E+00	−9,03E-01	−6,96E-01	0,097185	2,584197	0,044718
	50	−9,99E-01	−9,39E-01	−6,78E-01	0,116278	3,405775	0,055735
	60	−1,00E+00	−9,17E-01	−8,98E-01	0,138095	4,143588	0,065220
	70	−1,00E+00	−9,03E-01	−9,33E-01	0,157819	4,994868	0,066310
	80	−1,00E+00	−9,06E-01	−9,33E-01	0,181854	5,755332	0,085933
	90	−1,00E+00	−8,61E-01	−9,00E-01	0,206434	6,669711	0,094607
	100	−1,00E+00	−9,24E-01	−9,64E-01	0,227616	7,523491	0,104991
Griewank	10	7,17E-03	6,90E-15	3,21E+02	0,260855	0,946143	0,444829
	20	3,86E-04	0,00E+00	1,63E+02	0,510473	1,849194	0,868826
	30	3,02E-04	0,00E+00	2,47E+02	0,771970	2,745251	1,293660
	40	7,24E-06	0,00E+00	2,02E+02	1,026819	3,698099	1,699266
	50	2,79E-10	0,00E+00	1,79E+02	1,274411	4,519503	2,149955
	60	6,42E-12	0,00E+00	1,57E+02	1,533927	5,502049	2,515776
	70	1,71E-12	0,00E+00	1,48E+02	1,805498	6,512473	2,970469
	80	2,60E-13	6,96E-03	2,05E+02	2,041599	7,428721	3,335748
	90	1,11E-13	0,00E+00	1,20E+02	2,306967	8,467161	3,792227
	100	1,55E-12	0,00E+00	1,89E+02	2,547220	9,511772	4,142624

(*continued*)

Table 5. (*continued*)

		Solutions			Time (s)		
Function	Pop.	ABC	ACO	PSO	ABC	ACO	PSO
Rastrigin	10	2,80E-01	4,35E+02	4,82E+02	0,239999	0,725733	0,405736
	20	3,06E-04	4,53E+02	4,04E+02	0,467220	1,533267	0,781653
	30	2,51E-06	3,58E+02	3,93E+02	0,699583	2,616785	1,158362
	40	3,48E-09	4,25E+02	3,93E+02	0,931612	3,378083	1,555668
	50	8,23E-11	2,06E+02	3,88E+02	1,165836	4,696016	1,930535
	60	5,84E-12	1,88E+02	3,58E+02	1,401252	5,768915	2,309815
	70	1,88E-11	1,99E+02	3,99E+02	1,637102	6,787756	2,688676
	80	7,59E-13	2,05E+02	3,45E+02	1,854091	7,775599	3,025114
	90	5,35E-13	2,01E+02	3,05E+02	2,097354	8,923411	3,478108
	100	2,39E-12	1,90E+02	3,25E+02	2,320607	10,013488	3,947903
Rosenbrock	10	4,02E+01	5,10E+03	9,52E+05	0,212855	0,731430	0,355297
	20	2,56E+01	6,51E+03	4,62E+05	0,423005	1,523693	0,732902
	30	1,75E+01	7,68E+03	3,56E+05	0,618770	2,412705	1,065611
	40	1,40E+01	8,25E+03	3,35E+05	0,834661	3,310153	1,349149
	50	1,13E+01	9,62E+03	1,30E+05	1,033823	4,269835	1,709407
	60	1,12E+01	9,40E+03	3,95E+05	1,234289	5,192442	2,031221
	70	1,05E+01	9,31E+03	2,18E+05	1,436416	6,246066	2,393251
	80	1,07E+01	9,29E+03	2,60E+05	1,644925	7,263775	2,737766
	90	9,50E+00	9,16E+03	1,31E+05	1,837177	8,264279	3,202591
	100	9,16E+00	9,19E+03	2,14E+05	2,042155	9,346481	3,466199
Sphere	10	3,15E-17	4,71E-19	4,69E-01	0,075137	0,783766	0,050192
	20	1,83E-17	1,51E-32	5,89E-02	0,131975	1,658922	0,085171
	30	2,13E-17	1,29E-41	1,41E-01	0,195781	2,646467	0,108543
	40	1,60E-17	4,47E-48	9,39E-02	0,259194	3,664331	0,137414
	50	1,53E-17	7,07E-54	1,70E-01	0,323400	4,738335	0,160632
	60	1,38E-17	1,41E-56	9,21E-01	0,382747	5,829655	0,188244
	70	1,04E-17	1,48E-58	8,92E-02	0,448285	6,931901	0,227250
	80	1,54E-17	5,18E-60	3,92E-01	0,504095	8,013681	0,248711
	90	1,20E-17	3,90E-61	7,18E-02	0,564699	9,201093	0,279173
	100	1,08E-17	3,03E-62	1,31E-02	0,630266	10,331690	0,314831

population sizes. PSO achieved the best execution time. In Alpine, Booth, and Easom, the best execution time results for PSO are around two times faster than ABC, and ACO a dozen more than ABC.

ACO presented the best solution for the *Griewank* function, reaching the exact solution in 8 of the 10 population size evaluated. PSO appearing to be stuck in a local minimum, presenting the worst solution. The best execution time for *Griewank* was obtained by the ABC algorithm and ACO the worst.

In the *Rastrigin* and *Rosenbrock* function none of the algorithms achieved the exact results. However, the ABC algorithm presents the best solution in both

functions. In *Rastrigin* ACO and PSO are close results. In terms of performance, ABC has also the best execution time for both cases.

In the *Sphere* test, the ACO achieved results close to the exact solution. However, the best execution time was obtained for the PSO algorithm.

When the solution was evaluated, the ABC algorithm obtained the best solutions for five of the algorithms: *Alpine*, *Booth*, *Easom*, *Rastrigin* and *Rosenbrock*, while the algorithm ACO obtained the best solutions for the *Griewank* and *Sphere* algorithms. PSO resulted in worse solutions than the ABC and ACO algorithms.

The PSO algorithm was more efficient in terms of performance for the *Alpine*, *Booth*, *Easom*, and *Sphere* algorithms, while the ABC algorithm obtained the better performance rates for the *Griewank* and *Rastrigin* and *Rosenbrock* algorithms. The ACO algorithm obtained the lowest performance when compared to the other algorithms.

6 Conclusions and Future Work

The bio-inspired algorithms correspond to a range of efficient strategies for solving problems in several areas, from simple data analysis to complex problems such as distributed power generation or engineering simulations. The contribution of this work was to provide a library that allows the use of bio-inspired algorithms to solve problems.

In this article, the numerical evaluation and performance of the proposed library were assessed. Tests of the PSO, ABC, and ACO algorithms were performed using 7 functions and 10 different population sizes for each function. The ABC algorithm was the best in most cases to find the optimal solution and with less execution time in other cases.

As future work, it is intended to include new bio-inspired algorithms to the library. We also want to validate it by testing the algorithms on a real problem, such as that of distributed energy production. We also expect to reduce the execution time in a new version of the library using parallelization techniques.

References

1. Abraham, S., Sanyal, S., Sanglikar, M.: Particle swarm optimization based Diophantine equation solver. Int. J. Bio Inspired Comput. **2**(2), 100–114 (2010)
2. Ardeh, M.A.: BenchmarkFcns toolbox: a collection of benchmark functions for optimization (2016). http://benchmarkfcns.xyz/
3. Couto, D.C., Silva, C.A., Barsante, L.S.: Otimização de funções multimodais via técnica de inteligência computacional baseada em colônia de vaga-lumes. In: Proceedings of the XXXVI Iberian Latin American Congress on Computational Methods in Engineering. CILAMCE, Rio de Janeiro, RJ (2015)
4. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. IEEE Comput. Intell. Mag. **1**(4), 28–39 (2006)
5. Engelbrecht, A.P.: Computational Intelligence: An Introduction. Wiley, Chichester (2007)

6. Karaboga, D., Gorkemli, B., Ozturk, C., Karaboga, N.: A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif. Intell. Rev. **42**(1), 21–57 (2014)
7. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publishers Inc., San Francisco (2001)
8. Nievergelt, J., Gasser, R., Mäser, F., Wirth, C.: All the needles in a haystack: can exhaustive search overcome combinatorial chaos? In: van Leeuwen, J. (ed.) Computer Science Today. LNCS, vol. 1000, pp. 254–274. Springer, Heidelberg (1995). https://doi.org/10.1007/BFb0015248
9. Serapiao, A.: Fundamentos de Otimização por Inteligência de enxames: Uma Visão Geral. Controle y Automacao **20**, 271–304 (2009). https://doi.org/10.1590/S0103-17592009000300002
10. Silva, M.A.L., de Souza, S.R., Souza, M.J.F., de Franca Filho, M.F.: Hybrid metaheuristics and multi-agent systems for solving optimization problems: a review of frameworks and a comparative analysis. Appl. Soft Comput. **71**, 433–459 (2018)
11. Talukder, S.: Mathematicle modelling and applications of particle swarm optimization (2011)
12. Wilkinson, B., Allen, M.: Parallel Programming-Techniques and Applications using Networked Workstations and Parallel Computers, 2nd edn. Prentice Hall, Newark (1998)
13. Yang, X.S.: Test problems in optimization (2010). arXiv preprint arXiv:1008.0549