Evolutionary Bacterial Foraging Algorithm to solve constraint numerical optimization problems

Betania Hernández-Ocaña[1], Efrén Mezura-Montes[2], and Ma. Del Pilar Pozos-Parra[1]

[1] Juárez Autonomous University of Tabasco, Cunduacán, Tabasco, México
[2] Department of Artificial Intelligence, University of Veracruz, Xalapa, Veracruz, México {betania.hernandez@ujat.mx, pilar.pozos@ujat.mx, emezura@uv.mx}

Abstract. A version of Modified Bacterial Foraging Optimization Algorithm to solve Constraints Numerical Optimization is tested. The proposal uses mutation operator, skew mechanism and local search operator. To prove the effectiveness of the mechanism and adaptations proposed, 24 well-known test problems are solved along set experiments. Performance measures are used for validating results obtained by the proposal and they are compared against state-of-the-art algorithms. The results show that the proposed algorithm is able to generate feasible solutions within of feasible region with few evaluations and improves them over the generations. Moreover, the results are competitive against the comparison algorithms based on performance measures found in the literature.
Bacterial foraging optimization, mutation operator, swarm intelligence, Constrained optimization, premature convergence.

1 Introduction

Recently nature-inspired meta-heuristic algorithms have gained popularity solving Constrained Numerical Optimization Problems (CNOPs). These algorithms were created in order to solve unconstrained optimization problems, but nevertheless the main feature of the real world problems are the constraints. In answer to this, many constraints-handling techniques have emerged, and have been added to nature-inspired algorithms in several proposals. In [20] the different techniques found in the specialized literature were grouped in twelve groups, which are based on Penalty functions, Decoders, Special Operators and Separation of objective function and constraints, Feasibility rules, Stochastic ranking, ϵ-constrained method, Novel penalty functions, Novel special operators, Multi-objective concepts, and Ensemble of constraint-handling techniques.
The nature-inspired meta-heuristic algorithms have been divided in two classes 1) the Evolutionary Algorithms (EAs) that emulate the evolution of species and the survival of the fittest, and 2) the Swarm Intelligence Algorithms (SIAs) which have the capability of emulate the collaborative behavior of some simple species searching for food or shelter [8]. Some well-known EAs are Genetic Algorithms (GAs) [6], Evolution Strategies (ES) [27], Evolutionary Programming [9], Genetic Programming (GP) [14] and Differential Evolution (DE) [25]. Some SIAs are the Particle Swarm Optimization (PSO) [13] and the Ant Colony Optimization (ACO) [5], these last algorithms have gained popularity for their great performance in solving CNOP. In 2002 other SIA

was proposed by Passino [23] called Bacterial Foraging Optimization (BFOA), which emulates the behavior of bacteria E.Coli in the search of nutrients in its environment. This behavior is summarize in four process (1) chemotaxis (swim-tumble movements), (2) swarming (communication between bacteria), (3) reproduction (cloning of the best bacteria) and (4) elimination-dispersal (replacement of the worst bacteria). In BFOA each bacterium tries to maximize its obtained energy per each unit of time spent on the foraging process while avoiding noxious substances. BFOA was used initially to solve unconstrained optimization problems, however, more recently proposals add some constraints-handling technique to solve CNOPs, where the penalty function is the most used [10].
Further investigations have address the fact that BFOA is particularly sensitive to the step size parameter, a recent study of the step size was reported in [11], where different approaches were compared and the dynamic control mechanism was found to be slightly superior to static, random, and adaptive versions. In [21] BFOA was adapted in the proposal called Modified Bacterial Foraging Optimization Algorithm (MBFOA) to solve CNOPs. The proposal inherits the four processes of BFOA, however, them are divided as independent processes that interact sequentially, therefore the parameters that determine the number of swim, number of tumble and the swarming loop were eliminated. Moreover, The feasible rules, proposed by Deb [3], are used as a constraint-handling technique. Unlike of BFOA where the step size is static, in MBFOA the step size (used in the swim movements) was adapted according the boundary of the decision variables.
The aim of this paper is tested a version of MBFOA which used mechanisms based on evolutionary operators, Skew mechanism for the initial swarm and Local Search Operator called Evolutionary MBFOA (EMBFOA) in 24 well-known test problems. The results obtained will be compared against what is currently the most successful and well-known algorithm of the state of the art: Memetic-SAMSDE [7]. We evaluate our results with performance measures found in the literature. An advantage of the new proposal is that it uses the same parameter values for all test problems. It is the first time that the mutation is used as swim operator.
The document is organized as follows: Section 2 briefly describes the EMBFOA. Section 3 presents and compares the results obtained by proposal and another state-of-the-art algorithms. Finally, Section 4 presents the general conclusions and future works.

2 Evolutionary Modified Bacterial Foraging Algorithm

Evolutionary Modified Bacterial Foraging Algorithm is an algorithm derived from MBFOA in order to improve the performance in constrained spaces. Four modifications are made: (1) a skew mechanism for the initial swarm of bacteria is applied, (2) two swim operators are applied in the chemotaxis process to improve the exploration and exploitation of the bacteria; one of them uses a random step size easy to implement and other one use mutation, (3) the elimination-dispersal process is adapted to preserve the diversity of the swarm, and (4) a local search operator is added.

In BFOA, MBFOA and EMBFOA a bacterium i represents a potential solution to the CNOP (i.e., a n-dimensional real-value vector identified as $\boldsymbol{x}$), and it is defined as $\theta^i(j,G)$, where j is its chemotaxis loop index and G is the current cycle of the algorithm.
Skew mechanism for the initial swarm: The initial swarm of S_b bacteria is formed from three groups. The first group is formed of bacteria randomly skewed towards the

lower limit of the decision variables. The second group is formed of bacteria randomly skewed towards the upper limit of the decision variables. Finally, a group of randomly located bacteria without skew, as in the original MBFOA, is used. The formulas to set the limits for the first and second group per variable are presented in Equations 1 and 2.

$$[L_i, L_i + ((U_i - Li)/ss)] \tag{1}$$

$$[U_i - ((U_i - Li)/ss), U_i] \tag{2}$$

where ss is the skew size ($ss > 1$), for which large values increase the skew effect and small values decrease the skew effect. The aim of this skew in the initial swarm, combined with the two swim operators and the random stepsize control, is to avoid the swarm of bacteria converging prematurely (behavior observed in the original MBFOA caused by its swarming , reproduction process and the fixed stepsize), and improve the exploration and exploitation of the search space in the initial phase of the search.

Chemotaxis: In this process two swims are interleaved, in each cycle only one of the exploitation swim or exploration swim is performance. The process starts with the exploitation swim (classical swim). However, a bacterium will not necessarily interleave exploration and exploitation swims, because if the new position of a given swim, $\theta^i(j+1, G)$ has a better fitness (based on the feasibility rules) than the original position $\theta^i(j, G)$, another similar swim in the same direction will be carried out in the next cycle. Otherwise, a new tumble for the other swim will be computed. The process stops after N_c attempts. The exploration swim uses the mutation between bacteria and is computed as indicated in Equation 3:

$$\theta^i(j+1, G) = \theta^i(j, G) + (\beta - 1)(\theta_1^r(j, G) - \theta_2^r(j, G)) \tag{3}$$

where $\theta_1^r(j, G)$ and $\theta_2^r(j, G)$ are two different randomly selected bacteria from the swarm. β is an user-defined parameter used in the swarming that defines the closeness of the new position of a bacterium with respect to the position of the best bacterium, in this operator, $\beta - 1$ is a positive control parameter for scaling the different vectors into (0,1], i.e., it scales the area where a bacterium can move.
The exploitation swim is computed as indicated in Equation 4:

$$\theta^i(j+1, G) = \theta^i(j, G) + C(i, G)\phi(i) \tag{4}$$

where $\phi(i)$ is calculated with the original BFOA Equation 5 and $C(i, G)$ is the random stepsize of each bacterium update in each generation with the Equation 6.

$$\phi(i) = \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \tag{5}$$

where $\Delta(i)$ is a uniformly distributed random vector of size n with elements within the [-1,1].

$$C(i, G) = R * \Theta(i) \tag{6}$$

where $\Theta(i)$ is a randomly generated vector of size n with elements within of the range of the each decision variable: $[Upper_k, Lower_k]$, $k = 1, ...n$, and R is an user-defined parameter to scale the stepsize, this value should be close to zero, eg. *5.00E-04*. The initial $C(i, 0)$ is generated using $\Theta(i)$. This random stepsize allows that the bacteria can move in different direction into of the search space and avoids the premature convergence as is suggest in [12].
It is important to remark that the exploration swim (Equation 3) performs larger

movements due the mutation operator which used bacteria randomly. On the other hand, the exploitation swim (Equation 4) generates small movements using the random stepsize in the search process.
Swarming: In the half cycle of the chemotaxis process the swarming operator is applied with the Equation 7, where β is an user-defined parameter positive into (1, 2]. However, in this proposal if a solution violates the boundary of decision variables, unlike of MBFOA, a new solution x_i is generated randomly which is bounded by lower and upper limits $L_i \leq x_i \leq U_i$.

$$\theta^i(j+1, G) = \theta^i(j, G) + \beta(\theta^B(G) - \theta^i(j, G)) \quad (7)$$

where $\theta^i(j+1, G)$ is the new position of bacterium i, $\theta^i(j, G)$ is the current position of bacterium i, $\theta^B(G)$ is the current position of the best bacterium in the swarm so far at generation G, and β defines the closeness of the new position of bacterium i with respect to the position of the best bacterium $\theta^B(G)$. The attractor movement applies twice in a chemotaxis loop, while in the remaining steps the tumble-swim movement is carried out.
Reproduction: To reduce premature convergence due to bacteria duplication, The reproduction takes place only at certain cycles of the algorithm (defined by the *RepCycle* parameter) deleting the S_r worst bacteria and duplicating the remaining S_b-S_r.
Elimination-dispersal: To preserver and increase the diversity of swarm, the elimination-dispersal process takes place at certain cycles of the search process defined by the *EliCycle* user-defined parameter. The number of bacteria to eliminate-dispersal is defined by the user in the parameter S_e. In this proposal, the diversity of swarm is necessary because the mutation is effective when there is already some level of diversity in the existing swam. Moreover, the generation of new bacteria avoids the premature convergence.
Local Search Operator: Sequential Quadratic Programming (SQP) [24] is incorporated into EMBFOA as a local search operator in order to help the algorithm to generate better results and based on the improved behavior observed by memetic algorithms [22]. SQP is also used once in the middle of the search process. However, the user can define the frequency of usage of the local search operator by defining the parameter local search frequency LS_G. The structure of EMBFOA is showed in the Figure 1.

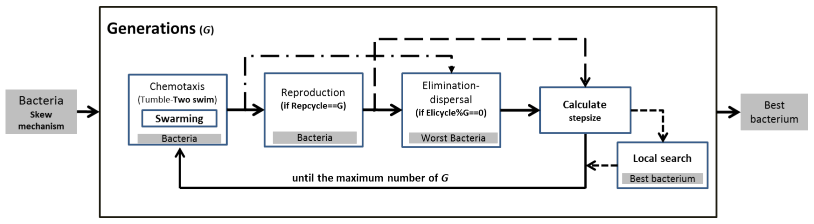

Fig. 1. EMBFOA general process.

3 Results and analysis

In this section are presented and compared the results obtained by EMBFOA in the CEC2006 benchmark against MBFOA and Memetic-SAMSDE which is one the best state-of-the-art algorithms for solving this benchmark. EMBFOA was coded in Matlab R2009b, and run on a PC with a 3.5 Core 2 Duo Processor, 4GB RAM, and Windows 7.
The 95%-confidence Wilcoxon Signed Rank Test (WSRT) [2] suggested for nature-inspired algorithms comparison in [4] is used as the statistical test to validate the differences in the samples of runs. The scores are based on the best fitness value of each algorithm in each test problem.
The input parameter values used by MBFOA are: S_b=40, N_c=20, S_r=2, R=1.20E-03 and β=1.5. For EMBFOA the parameters are: S_b= 40, N_c= 20, S_r= 2, R= 1.20E-03, β= 1.5, *RepCycle*= 100, *EliCycle*= 50, LS_G= 1 and $(GMAX/2)$ generations, Max_LS= 5,000 Fes and ss= 8; they were fine-tuned by the iRace tool [16]. The iRace obtained five possible values for each parameter and the average of the five values was used. In this case, the only fixed parameter were $GMAX$ according to the termination condition of the maximum number of evaluations (Max_FEs) of CEC2006 benchmark, and the maximum number of evaluations for the local search operator Max_LS which is $5,000$ evaluations.
24 well-known CNOPs found in [15] were used in the experiments. Max_Fes allowed for each one of the 24 test problems was 240,000 evaluations. The tolerance for equality constraints was set to $\varepsilon = 1E-04$, this value is proposed in the specialized literature of nature-inspired algorithms to solve CNOPs [1, 19, 20].

In this experiment the performance of EMBFOA is compared against MBFOA and some EAs, such as Mememtic Self-Adaptive Multi-Strategy Differential Evolution (Memetic-SAMSDE) which is currently one of the best state-of-the-art algorithm for solving the set of test problems used in this paper. This algorithm also uses SQP as local search. Unlike the EMBFOA that uses SPQ only twice, in Memetic-SAMSDE is used each 50 generations. The Adaptive Penalty Formulation with GA (APF-GA) [28], the Modified Differential Evolution (MDE) [18], the Adaptive Tradeoff Model with evolution strategy (ATMES) [29], the Multimembered evolution strategy (SMES) [17], and the Stochastic ranking with evolution strategy (SR+ES) [26]. Even when ATMES, SMES and SR solved solely the first 13 test problems a computation against them is shown. The number of evaluations computed by APF-GA and MDE was 500,000. SR+ES used 350,000 evaluations while for Memetic-SAMSDE, EMBFOA, ATMES and SMES used 240,000 evaluations. The number of independent runs used by EMBFOA, Memetic-SAMSDE, ATMES, SMES and SR+ES was 30, and APF-GA and MDE used 25 independent runs. In the Table 1 the results of all algorithms for the 24 test problems are present whit the best solutions of each test problem is highlighted in bold. From the results obtained by each algorithm, the Wilcoxon Sign-Rank test suggest that there is significant difference between EMBFOA and MBFOA and there is no significant difference between EMBFOA, Memetic-SAMSDE and MDE. However the last algorithm uses 500,000 evaluations which are the double evaluation used by EMBFOA. There is a significant difference in favor of EMBFOA when it is compared against APF-GA, ATMES, SME and SR+ES. Moreover APF-GA and SR+ES used more number of evaluations than EMBFOA.
In this experiment the performance of EMBFOA was compared against five state-of-the-art nature-inspired algorithms used to solve CNOPs successfully, with the results

Table 1. Statistical comparison of MBFOA, EMBFOA, Memetic-SAMSDE, APF-GA, MDE, ATMES, SMES and SR+ES. '-' means that the value is not proportionate by the algorithm

Prob.	$f(x^*)$	Criteria	MBFOA	EMBFOA	Memetic-SAMSDE	APF-GA	MDE	ATMES	SMES	SR+ES
g01	-15	Best	-9.88536	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**
		Average	-4.91338	-14.921	**-15**	**-15**	**-15**	**-15**	**-15**	**-15**
		St.d	2.11E+00	2.97E-01	**0**	**0**	**0**	1.60E-14	**0**	**0**
g02	-0.8036191	Best	-0.44923	**-0.8036191**	**-0.8036191**	-0.803601	**-0.8036191**	-0.803339	-0.803601	-0.803515
		Average	-0.33277	-0.679341365	**-0.8036191**	-0.803518	-0.78616	-0.790148	-0.785238	-0.781975
		St.d	4.70E-02	6.23E-02	**0**	1.00E-04	1.20E-02	1.30E-02	1.67E-02	2.00E-02
g03	-1.0005001	Best	-1.0004	**-1.001**	-1.0005	**-1.001**	-1.0005	-1	-1	-1
		Average	**-1.0004**	-0.993	-1.0005	**-1.001**	-1.0005	-1	-1	-1
		St.d	**2.30E-05**	4.16E-02	**0**	**0**	**0**	5.90E-05	0.0000209	0.0000209
g04	-30665.5387	Best	-30665.1	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.5386**	**-30665.539**	**-30665.539**	**-30665.539**
		Average	-30663.3	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.5386**	**-30665.539**	**-30665.539**	**-30665.539**
		St.d	4.19E+00	1.17E-11	**0**	1.00E-04	1.00E-04	7.40E-12	**0**	2.00E-05
g05	5126.49671	Best	-	**5126.497**	**5126.497**	**5126.497**	**5126.497**	5126.498	5126.599	**5126.497**
		Average	-	**5126.497**	**5126.497**	5127.5423	**5126.497**	5127.648	5174.492	5128.881
		St.d	-	2.78E-08	**0**	1.43E+00	**0**	1.80E+00	5.01E+01	3.50E+00
g06	-6961.81388	Best	-6960.83	**-6961.813875**	**-6961.813875**	**-6961.814**	**-6961.814**	**-6961.814**	**-6961.814**	**-6961.814**
		Average	-6950.8	**-6961.813875**	**-6961.813875**	**-6961.814**	**-6961.814**	**-6961.814**	-6961.284	-6875.94
		St.d	1.21E+01	4.63E-12	**0**	**0**	**0**	4.60E-12	1.85E+00	1.60E+02
g07	24.3062091	Best	24.72605	**24.3062**	**24.3062**	**24.3062**	**24.3062**	**24.306**	24.327	24.307
		Average	25.91026	**24.3062**	**24.3062**	**24.3062**	**24.3062**	24.316	24.475	24.374
		St.d	8.36E-01	1.20E-07	**0**	**0**	**0**	1.10E-02	1.32E-01	6.60E-02
g08	-0.095825	Best	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**
		Average	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**
		St.d	1.60E-12	1.79E-17	**0**	**0**	**0**	2.80E-16	**0**	**0**
g09	680.630057	Best	680.6534	**680.63**	**680.63**	**680.63**	**680.63**	**680.63**	680.632	**680.63**
		Average	680.706	**680.63**	**680.63**	**680.63**	**680.63**	680.639	680.643	680.656
		St.d	4.25E-02	5.24E-13	**0**	**0**	**0**	1.00E-02	1.55E-02	3.40E-02
g10	7049.24802	Best	7082.964	**7049.24802**	**7049.24802**	**7049.24802**	**7049.24802**	7052.253	7051.903	7051.903
		Average	7356.791	**7049.24802**	**7049.24802**	7077.6821	**7049.24802**	7250.437	7253.047	7253.047
		St.d	4.89E+02	1.27E-04	**0**	5.12E+01	**0**	1.20E+02	1.36E+02	1.36E+02
g11	0.7499	Best	**0.7499**	**0.7499**	**0.7499**	**0.7499**	**0.7499**	**0.75**	**0.75**	**0.75**
		Average	**0.7499**	**0.7499**	**0.7499**	**0.7499**	**0.7499**	**0.75**	**0.75**	**0.75**
		St.d	2.12E-06	1.86E-07	**0**	**0**	**0**	3.40E-04	1.52E-04	8.00E-05
g12	-1	Best	-0.9999	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**
		Average	-0.9992	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**
		St.d	1.95E-03	**0**	**0**	**0**	**0**	1.00E-03	**0**	**0**
g13	0.053941	Best	-	**0.053942**	**0.053942**	**0.053942**	**0.053942**	0.05395	0.053986	0.067543
		Average	-	0.358400084	**0.053942**	**0.053942**	**0.053942**	0.053959	0.166385	0.067543
		St.d	-	2.81E-01	**0**	**0**	**0**	1.30E-05	1.77E-01	3.10E-02
g14	-47.764888	Best	-42.5345	**-47.764888**	**-47.764888**	-47.76479	-47.764887	-	-	-
		Average	-38.6844	**-47.764888**	**-47.764888**	-47.76479	-47.764874	-	-	-
		St.d	2.52E+00	1.26E-13	**0**	1.00E-04	1.40E-05	-	-	-
g15	961.715022	Best	961.7153	**961.71502**	**961.71502**	**961.71502**	**961.71502**	-	-	-
		Average	961.7177	**961.71502**	**961.71502**	**961.71502**	**961.71502**	-	-	-
		St.d	1.57E-03	2.87E-05	**0**	**0**	**0**	-	-	-
g16	-1.905155	Best	-1.90335	**-1.905155**	**-1.905155**	**-1.905155**	**-1.905155**	-	-	-
		Average	-1.88754	**-1.905155**	**-1.905155**	**-1.905155**	**-1.905155**	-	-	-
		St.d	5.64E-02	2.06E-15	**0**	**0**	**0**	-	-	-
g17	8853.53967	Best	-	8912.914588	**8853.5397**	8853.5398	**8853.5397**	-	-	-
		Average	-	8927.107321	**8823.5397**	8888.4876	**8853.5397**	-	-	-
		St.d	-	2.68E+00	**0**	2.90E+01	**0**	-	-	-
g18	-0.866025	Best	-0.85966	**-0.866025**	**-0.866025**	**-0.866025**	**-0.866025**	-	-	-
		Average	-0.73024	**-0.866025**	**-0.866025**	**-0.865925**	**-0.866025**	-	-	-
		St.d	1.18E-01	5.08E-05	**0**	1.00E-04	**0**	-	-	-
g19	32.655592	Best	49.47301	**32.655592**	**32.655593**	**32.655593**	**32.64827**	-	-	-
		Average	117.2929	32.65581447	**32.655593**	**32.655593**	33.34125	-	-	-
		St.d	7.33E+01	6.40E-04	**0**	**0**	8.47E-01	-	-	-
g20	0.2049794	Best	-	-	-	-	-	-	-	-
		Average	-	-	-	-	-	-	-	-
		St.d	-	-	-	-	-	-	-	-
g21	193.72451	Best	-	**193.72451**	**193.72451**	196.63301	**193.72451**	-	-	-
		Average	-	245.010861	**193.72451**	199.51581	**193.72451**	-	-	-
		St.d	-	5.27E+01	**0**	2.36E+00	**0**	-	-	-
g22	236.430976	Best	-	-	**236.370313**	-	-	-	-	-
		Average	-	-	**245.738829**	-	-	-	-	-
		St.d	-	-	**9.05E+00**	-	-	-	-	-
g23	-400.0551	Best	-	-400.0544473	**-400.0551**	-399.7624	**-400.0551**	-	-	-
		Average	-	-377.0888655	**-400.0551**	-394.7627	**-400.0551**	-	-	-
		St.d	-	6.61E+01	**0**	3.87E+00	**0**	-	-	-
g24	-5.50801327	Best	**-5.508013**	**-5.508013**	**-5.508013**	**-5.508013**	**-5.508013**	-	-	-
		Average	-5.50768	**-5.508013**	**-5.508013**	**-5.508013**	**-5.508013**	-	-	-
		St.d	2.83E-04	1.81E-15	**0**	**0**	**0**	-	-	-

of the comparison it is prove that EMBFOA is a competitive algorithm. Moreover, the superiority is evident of EMBFOA over MBFOA to find the feasible region and improve the feasible solutions found.

4 Conclusion

In this paper, a Swarm Intelligence algorithm called Evolutionary Modified Bacterial Foraging Optimization Algorithm (EMBFOA) was proposed to solve constrained numerical optimization problems. This proposal is an improved version of the algorithm Modified Bacterial Foraging Optimization. The performance of EMBFOA was tested in 24 well-known test problems using a set of performance measures found in the literature and the results obtained were compared against state-of-the-art algorithms. According to the results, EMBFOA is better than the original algorithm MBFOA and competitive against state-of-the-art algorithms. To the best of the authors knowledge, this is the first attempt to design a BFOA-based algorithm that uses mutation as swim operator. As future work, EMBFOA will be tested in more CNOPs and a study on the impact of the population size and the way it is generated over the performance of the algorithm, will be performed.

References

1. Carlos A. Coello Coello. Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, January 2002.
2. G.W. Corde and D.I. Foreman. *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach.* John Wiley, Hoboken, NJ, 2009.
3. Kalyanmoy Deb. An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2/4):311–338, 2000.
4. J. Derrac, S. García, D. Molina, and F. Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.
5. M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions of Systems, Man and Cybernetics-Part B*, 26(1):29–41, 1996.
6. A.E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing.* Natural Computing Series. Springer Verlag, 2003.
7. Saber M. Elsayed, Ruhul A. Sarker, and Daryl L. Essam. On an evolutionary approach for constrained optimization problem solving. *Applied soft computing*, 12(0):3208–3227, 2012.
8. Andries P. Engelbrecht. *Computational Intelligence. An Introduction.* John Wiley & Sons, 2nd edition, 2007.
9. Lawrence J. Fogel. *Intelligence Through Simulated Evolution. Forty years of Evolutionary Programming.* John Wiley & Sons, New York, 1999.
10. Betania Hernández-Ocaña, Efrén Mezura-Montes, and Pilar Pozos-Parra. A review of the bacterial foraging algorithm in constrained numerical optimization. In *Proccedings of the Congress on Evolutionary Computation (CEC'2013)*, pages 2695–2702. IEEE, 2013.
11. Betania Hernández-Ocaña, Pilar Pozos-Parra, and Efrén Mezura-Montes. Stepsize control on the modified bacterial foraging algorithm for constrained numerical optimization. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 25–32, New York, NY, USA, 2014. ACM.
12. Alireza Kasaiezadeh, Amir Khajepour, and Steven L. Waslander. Spiral bacterial foraging optimization method: Algorithm, evaluation and convergence analysis. *Engineering Optimization*, 46(4):439–464, 2014.
13. James Kennedy and Russell C. Eberhart. *Swarm Intelligence.* Morgan Kaufmann, UK, 2001.
14. John R. Koza, Martin A. Keane, Matthew J. Streeter, William Mydlowec, Jessen Yu, and Guido Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence.* Kluwer Academic Publishers, Hingham, MA, USA, 2003.

15. J.J. Liang, Thomas Philip Runarsson, Efrn Mezura-Montes, Maurice Clerc, P.N. Suganthan, Carlos A. Coello Coello, and K. Deb. Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. Technical report, School of EEE Nanyang Technological University, Singapore, September 2006.
16. Manuel López-Ibáñez, Jéremie Dubois-Lacoste, Thomas Sttzle, and Mauro Birattari. The irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.
17. E. Mezura-Montes and Carlos A. Coello Coello. A simple multimembered evolution strategy to solve constrained optimization problems. *Evolutionary Computation, IEEE Transactions on*, 9(1):1–17, Feb 2005.
18. E. Mezura-Montes, J. Velazquez-Reyes, and C.A. Coello Coello. Modified differential evolution for constrained optimization. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 25–32, 2006.
19. Efrén Mezura-Montes, editor. *Constraint-Handling in Evolutionary Optimization*, volume 198 of *Studies in Computational Intelligence*. Springer-Verlag, 2009.
20. Efrén Mezura-Montes and Carlos A. Coello Coello. Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194, 2011.
21. Efrén Mezura-Montes and Betania Hernández-Ocaña. Modified bacterial foraging optimization for engineering design. In Cihan H. Dagli and et al., editors, *Proceedings of the Artificial Neural Networks in Enginnering Conference (ANNIE'2009)*, volume 19 of *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 357–364, St. Louis, MO, USA, November 2009. ASME Press.
22. Ferrante Neri and Carlos Cotta. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1–14, 2012.
23. Kevin M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, 22(3):52–67, 2002.
24. M. J. D. Powell. Algorithms for Nonlinear Constraints that use Lagrangian Functions. *Mathematical Programming*, 14:224–248, 1978.
25. K. Price, R. Storn, and J. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer-Verlag, 2005.
26. T.P. Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *Evolutionary Computation, IEEE Transactions on*, 4(3):284–294, Sep 2000.
27. Hans-Paul Schwefel, editor. *Evolution and Optimization Seeking*. Wiley, New York, 1995.
28. B. Tessema and G.G. Yen. An adaptive penalty formulation for constrained evolutionary optimization. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 39(3):565–578, May 2009.
29. Yong Wang, Zixing Cai, Yuren Zhou, and Wei Zeng. An adaptive tradeoff model for constrained evolutionary optimization. *Evolutionary Computation, IEEE Transactions on*, 12(1):80–92, Feb 2008.