Firefly Algorithm with a Feasibility-Based Rules for Constrained Optimization

Ivona BRAJEVIC, Milan TUBA, Nebojsa BACANIN
Faculty of Computer Science
University Megatrend Belgrade
Bulevar umetnosti 29, N. Belgrade
SERBIA
ivona.brajevic@googlemail.com, tuba@ieee.org, nbacanin@megatrend.edu.rs

Abstract: - Firefly algorithm (FA) is recently developed nature-inspired metaheuristic based on the flashing patterns and behaviour of fireflies. Original FA was successfully applied to solve unconstrained optimization problems. This paper presents firefly algorithm to solve constrained optimization problems. For constraint handling, firefly algorithm uses certain feasibility-based rules in order to guide the search to the feasible region. Our proposed approach is tested on nine well-known benchmark functions. Obtained results are compared to those of the state-of-the-art algorithms.

Key-Words: - Firefly algorithm, Metaheuristics, Constrained optimization, Swarm intelligence

1 Introduction

As many optimization problems involve a number of constraints that the decision solutions need to satisfy, the aim of constrained optimization is to search for feasible solutions with better objective values. Generally, a constrained optimization problem is to find *x* so as to

$$\text{minimize } f(x), \quad x=(x_1,...,x_n)\in R^n \qquad (1)$$
$$\text{where } x\in F\subseteq S.$$

The objective function f is defined on the search space $S\subseteq R^n$ and the set $F\subseteq S$ defines the feasible region. The search space S is defined as an n-dimensional rectangle in R^n. The variable domains are limited by their lower and upper bounds:

$$l_i \le x_i \le u_i,\ 1\le i\le n \qquad (2)$$

whereas the feasible region $F\subseteq S$ is defined by a set of m additional constraints ($m\ge 0$):

$$g_j(x)\le 0,\ \text{for } j=1,...,q \qquad (3)$$
$$h_j(x)=0,\ \text{for } j=q+1,..m \qquad (4)$$

For an inequality constraint that satisfies $g_j(x)=0$, we will say that is active at *x*. All equality constraints h_j (regardless of the value of *x* used) are considered active at all points of *F*. Both the objective function and the constraints can be linear or nonlinear.

Finding optimal solutions to these problems requires efficient optimisation algorithms. Applicability of the deterministic methods is limited, since they use a variety of assumptions (such as continuity) about the search space before they actually start the search [1]. Having regard to the fact that they are quite efficient in finding local optima, a common practice is to introduce a stochastic component to an algorithm in order to avoid the possibility for the algorithm to be trapped at local optima.

Different stochastic optimization algorithms have been developed [2], [3] and later improved for solving constrained optimization problems [4], [5], [6], [7], [8], [9]. Among the most popular metaheuristics are those based on mimicking different natural phenomena and biological models [10], [11]. Some of the most popular nature inspired metaheuristics are genetic algorithm (GA), inspired by Darwin's theory about evolution, particle swarm optimization (PSO) algorithm, inspired by the social behavior of birds or fishes and artificial bee colony (ABC) algorithm, based on honey bee foraging behavior.

Firefly Algorithm (FA) proposed by Yang is one of the new metaheuristic techniques inspired by the flashing behaviour of fireflies. FA was proposed to solve unconstrained optimization problems [12]. Simulation results indicate that FA is superior over GA and PSO. Also, an object-oriented software implementation of FA [13], as well as implemen-

This research is supported by Ministry of Education and Science, Republic of Serbia, Project No. III-44006

tation of the parallelized FA was provided [14]. In [15] FA was applied to solve mixed variable structural optimization problems. Furthermore, different variants of FA have been developed and used in many practical fields [16], [17].

Since most of the basic versions of nature-inspired search algorithms lack a mechanism to deal with the constraints of a numerical optimization problem, constraint handling techniques are usually incorporated in the algorithms in order to direct the search towards the feasible regions of the search space [18]. The most common approach adopted to deal with constrained search spaces is the use of penalty functions. In these methods, a constrained problem is solved as an unconstrained one, where the objective function is designed such that infeasible solutions are characterized by high function values (in minimization cases). Despite the popularity of penalty functions, regard its simplicity and direct applicability, they have several drawbacks. The main one is that they require a careful fine tuning of the penalty factors that estimate the degree of penalization to be applied.

The constraint-handling approach used in this paper is based on three feasibility rules, also called Deb's rules, proposed for binary tournaments in [19]. The main advantage of this constraint-handling scheme is in the lack of user-defined parameters, although it may also lead to premature convergence. In this work, we incorporated three feasibility rules into the FA in order to solve constrained optimization problems. The performance of the proposed firefly algorithm for constrained optimization, called CFA, has been tested on nine well-known benchmark functions taken from the literature and compared with GA, PSO and ABC.

This paper is organized as follows. Section 2 presents firefly algorithm. A detailed description of our approach is provided in Section 3. Section 4 presents benchmark functions, test results and discussion.

2 Firefly algorithm

In order to develop firefly algorithm, some of the flashing characteristics of fireflies were idealized. Yang formulated this firefly algorithm by assuming three simplification rules [12].

1. All fireflies are unisex so that one firefly will be attracted at other fireflies regardless of their sex.
2. Attractiveness is proportional to firefly brightness. For any couple of flashing fireflies, the less bright one will move towards the brighter one. The brightness decreases when the distance between fireflies is increased. The brightest firefly moves randomly, because there is no other bug to attract it.
3. The brightness of a firefly is somehow related with the analytical form of the cost function. For a maximization problem, brightness can simply be proportional to the value of the cost function. Other forms of brightness can be defined in a similar way to the fitness function in genetic algorithms.

Similarily to other metaheuristics optimization methods, firefly algorithm generates random initial population of feasible candidate solutions. All fireflies of the population are handled in the solution search space with the aim that knowledge is collectively shared among fireflies to guide the search to the best location in the search space. Each individual of the population is called a firefly. At each iterative step, the brightness and the attractiveness of each firefly is calculated. The brightness of each firefly is compared with all other fireflies and the positions of the fireflies are dynamically updated based on the knowledge of the firefly and its neighbors.

According to above rules there are two main points in firefly algorithm, the attractiveness of the firefly and the movement towards the attractive firefly.

2.1 The attractiveness of the firefly

At the source, the brightness is higher than at some distant point. Also, the brightness decreases while environment absorbs the light while it is travelling. It can be concluded that the attractiveness of firefly β is relative. It is known that the light intensity $I(r)$ varies following the inverse square law:

$$I(r) = \frac{I_0}{r^2} \tag{5}$$

where I_0 represents the light intensity at the source.

Suppose there are n fireflies and that x_i corresponds to the solution for firefly i. The brightness of the firefly i, is associated with the objective function $f(x_i)$. The brightness (the attractiveness) I of a firefly is chosen to reveal its recent position of its fitness value or objective function $f(x)$:

$$I_i = f(x_i) \tag{6}$$

The less bright (attractive) firefly is attracted and moved to the brighter one and each firefly has a certain attractiveness value β. However, the

attractiveness value β is relative based on the distance between fireflies. The attractiveness function of the firefly is established by:

$$\beta(r) = \beta_0 \cdot e^{-\gamma \cdot r^2} \quad (7)$$

where β_0 is the firefly attractiveness value at $r = 0$ and γ is the media light absorption coefficient.

2.2 The movement towards attractive firefly

Fireflies movement is based on the principles of attractiveness: when firefly *j* is more attractive than firefly *i* the movement is determined by the following equation:

$$x_i = x_i + \beta_0 \cdot e^{-\gamma \cdot r_{ij}^2} \cdot (x_i - x_j) + \alpha \cdot \left(rand - \frac{1}{2} \right) \quad (8)$$

In Eq. (8) third term is randomization term where $\alpha \in [0,1]$, *rand* is random number between 0 and 1. Distance r_{ij} between fireflies *i* and *j* is obtained by Cartesian distance form by:

$$r_{ij} = \sqrt{\sum_{k=1}^{D} \left(x_{i,k} - x_{j,k} \right)^2} \quad (9)$$

Therefore, the algorithm compares the attractiveness of the firefly *i* with the attractiveness of the firefly *j*. If the firefly *j* is more attractive than firefly *i*, then the firefly *i* is moved to the new position; otherwise the firefly *i* will remain in the current position. The termination criterion of the FA is based on an arbitrary predefined number of iterations or predefined fitness value.

3 Our proposed approach: CFA

In order to solve constrained optimization problems, we incorporated the three simple selection criteria based on feasibility into the firefly algorithm to guide the search to the feasible region. This constraint-handling technique, originally proposed by Deb, uses a tournament selection operator, where two solutions are compared at a time, and the following rules are always enforced [19]:

1. When comparing two feasible solutions, the one with the better objective function is chosen.
2. When comparing a feasible and an infeasible solution, the feasible one is chosen.
3. When comparing two infeasible solutions, the one with the lower sum of constraint violation is chosen.

The sum of constraint violation for a solution x is given by:

$$CV(x) = \sum_{j=1}^{q} \max(0, g_j(x)) + \sum_{j=q+1}^{m} h_j(x) \quad (10)$$

In CFA, Deb's rules are used instead of the greedy selection in order to decide what solution will be updated. Hence, the decision what firefly is more attractive is made according these feasibility rules. The CFA does not start with the feasible initial population, since initialization with feasible solutions is hard and in some cases impossible to achieve randomly. During running process of CFA, the feasibility rules direct the solutions to feasible region.

As in the version of FA proposed to solve structural optimization problems [15], it was found that the solution quality can be improved by reducing the randomization parameter α with a geometric progression reduction scheme similar to the cooling schedule of simulated annealing which can be described by:

$$\alpha = \alpha_0 \cdot \theta^t \quad (11)$$

where $0 < \theta < 1$ is the reduction factor of randomization. In CFA, described reduction scheme was followed by reducing α from 0.5 to 0.0001. In our implementation we set parameter γ, which characterizes the variation of attractiveness to the value 1. The initial value of attractiveness $\beta_0 = 1$ was utilized. The pseudo code for the CFA is:

Objective function *f(x)*, $x = (x_1, ..., x_D)^T$
Initialize a population of fireflies x_i *(i = 1, 2, ..., n)* randomly
Initialize algorithm's parameters α, β_0, γ
while *(t <MaxGeneration)*
  for *i* = 1 : *n*
    for *j* = 1 : *n*
      if (x_j is chosen according to Deb's rules when we compare x_i and x_j)
      Obtain attractiveness which varies with distance *r* via *exp[- γ r]*
      Move firefly *i* towards *j* in all *D* dimensions
      Evaluate new solution
      end if
    end for *j*
  end for *i*
Reduce the randomization parameter α by (11)
Find the current best solution
end while

ISBN: 978-1-61804-126-5

4 Experimental study

To test the performance of our proposed approach CFA, we used 9 well-known benchmark problems which can be found in [2]. This set of 9 benchmark problems includes various forms of objective functions such as linear, nonlinear and quadratic. Type of objective function, the optimal solution, the number of linear equalities (LE), nonlinear equalities (NE), linear inequalities (LI), nonlinear inequalities (NI) and the number of optimization parameters (D) are given in Table 1. In Table 1, ρ is an estimate of the ratio between the feasible region and the entire search space computed by $\rho = |F|/|S|$ where $|F|$ is the number of feasible solutions and $|S|$ is the total number of solutions randomly generated.

Our proposed approach has been implemented in Java programming language. Tests were done on a PC with Intel® Core™ i3-2310M processor @2,10 GHz with 2GB of RAM and Windows 7 x64 Professional operating system. The performance of the CFA is compared with the performance of genetic algorithm (GA) [5], particle swarm optimization (PSO) [6] and artificial bee colony (ABC) algorithm [2].

Function	Optimal	D	Type of Fun.	ρ (%)	LI	NI	LE	NE
g01	-15.000	13	quadratic	0.0003	9	0	0	0
g03	1.000	10	nonlinear	0.0026	0	0	0	1
g06	-6961.814	2	nonlinear	0.0057	0	2	0	0
g07	24.306	10	quadratic	0.0000	3	5	0	0
g08	0.095825	2	nonlinear	0.8581	0	2	0	0
g09	680.630	7	nonlinear	0.5199	0	4	0	0
g11	0.75	2	quadratic	0.0973	0	0	0	1
g12	1.000	3	quadratic	4.7697	0	9	0	0
g13	0.053950	5	nonlinear	0.0000	0	0	1	2

Table 1. Summary of main properties of the benchmark functions

4.1 Parameter Settings

In GA population size is 200, maximum number of generations is 1200, crossover rate is 0.8, mutation rate is 0.6 and the number of objective function evaluations is 240000. All equality constraints have been converted into inequality constraints, $|h_j| \le \varepsilon$, with ε varying dynamically.

In PSO algorithm the swarm size is 50 and the generation number is 7000. Hence, PSO performs 350000 objective function evaluations. Cognitive and social components are both set to 1, while inertia weight is uniform random real number in the range [0.5,1].

In ABC algorithm, the value of modification rate (*MR*) is 0.8, colony size (*SN*) is 40 and the maximum cycle number is 6000. Therefore, ABC performs 240000 objective function evaluations. The value of *limit* and *SPP* is equal to *SN*D*0.5*, where *D* is the dimension of the problem and *SN* is the number of solutions in the population.

In CFA population size is set to 35, maximum number of iterations is 3000 and therefore the number of objective function evaluations is 105000. Each of the experiments was conducted 30 times using different random seeds.

In PSO, ABC and CFA all equality constraints have been converted into inequality constraints, $|h_j| \le \varepsilon$, with $\varepsilon = 0.001$. As in our proposed approach, Deb's rules are used for constraint handling in GA, PSO and ABC.

4.2 Results and Discussion

The statistical results of CFA when it was applied to 9 benchmark problems are presented in Table 2. From Table 1 it can be seen that our approach was able to find the global optimum in 6 out of 9 benchmarks (g01, g03, g08, g09, g11, g12) and it found solutions very close to the global optimum in the remaining test functions, with the exception of g13.

Comparative results of the best and mean solutions of the GA, PSO, ABC and our proposed algorithm are presented in Table 3 and Table 4.

ISBN: 978-1-61804-126-5

F.	Best	Mean	Worst	S.D.
g01	-15.000	-14.563	-12.453	0.881
g03	1.005	1.005	1.005	0.000
g06	-6961.813	-6961.812	-6961.809	0.001
g07	24.308	24.315	24.328	0.005
g08	0.095825	0.095825	0.095825	0.000
g09	680.630	680.631	680.632	0.000
g11	0.749	0.749	0.749	0.000
g12	1.000	1.000	1.000	0.000
g13	0.177	0.455	1.263	0.216

Table 2. Statistical results obtained by CFA for 9 test functions over 30 independent runs

F.	GA	PSO	ABC	CFA
g01	-14.440	-15.000	-15.000	-15.000
g03	0.990	0.993930	1.000	1.005
g06	-6952.472	-6961.814	-6961.814	-6961.813
g07	31.097	24.370153	24.330	24.308
g08	0.095825	0.095825	0.095825	0.095825
g09	685.994	680.630	680.634	680.630
g11	0.75	0.749	0.750	0.749
g12	1.000	1.000	1.000	1.000
g13	0.134057	0.085655	0.760	0.177

Table 3. The best solutions obtained by GA, PSO, ABC and CFA for 9 test functions over 30 independent runs

F.	GA	PSO	ABC	CFA
g01	-14.236	-14.710	-15.000	-14.563
g03	0.976	0.764813	1.000	1.005
g06	-6872.204	-6961.814	-6961.813	-6961.812
g07	34.980	32.407	24.473	24.315
g08	0.095799	0.095825	0.095825	0.095825
g09	692.064	680.630	680.640	680.631
g11	0.75	0.749	0.750	0.749
g12	1.000	0.998875	1.000	1.000
g13	-	0.569358	0.968	0.455

Table 4. The mean solutions obtained by GA, PSO, ABC and CFA for 9 test functions over 30 independent runs (- means that no feasible solutions were found)

If we compare the performance of CFA with the performance of GA algorithm we can see that CFA performs better, since it reached better best and mean results for 8 out of 9 benchmark problems.

When comparing our approach with respect to PSO algorithm, we can see that CFA found a better best solution for 2 benchmarks (g03 and g07) and a worse best result for 2 test functions (g06 and g13). From the mean results, CFA outperforms PSO on 4 benchmarks (g03, g07, g12, g13) and performs worse on 3 problems (g01, g06, g09). We can conclude that CFA and PSO show similar performances according to the best and mean results.

Compared with ABC, our approach found a better best results for 3 benchmarks (g07, g09, g13) and similar best result for the remaining benchmarks, with the exception of g06 and g11 where CFA performs slightly worse. From the mean results, CFA shows a better performance on 3 problems (g07, g09, g13) and also a worse performance on 3 problems (g01, g06, g11). According to the best and mean results we can conclude that CFA and ABC show similar performances.

5 Conclusion

In this paper, firefly algorithm for constrained problems (CFA) is presented. The basic firefly algorithm was first developed to solve unconstrained optimization problems and showed that it has superior performance over GA and PSO. Our approach incorporates constraint handling technique based on three feasibility rules into the basic firefly algorithm in order to prefer feasible solutions to infeasible ones. CFA has been tested on nine well-known benchmark functions. Comparisons show that CFA outperforms or performs similarly to three other state-of-the-art algorithms such as GA, PSO and ABC.

Future work will include investigation to a more detailed performance analysis of CFA. Also, the effect of constraint handling methods on the performance of firefly algorithm can be considered.

References:

[1] O. Yeniay: A comparative study on optimization methods for the constrained nonlinear programming problems, Mathematical Problems in Engineering Hindawi Publishing Corporation, 2005, pp. 165-173

[2] Raka Jovanovic, Milan Tuba: An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem,

Applied Soft Computing, Vol. 11, Issue 8, Dec. 2011, pp. 5360–5366

[3] Milan Tuba, Milos Subotic, Nadezda Stanarevic : Performance of a modified cuckoo search algorithm for unconstrained optimization problems, WSEAS Transactions on Systems, Volume 11, Issue 2, February 2012, pp. 62-74

[4] D. Karaboga, B. Akay, A Modified Artificial Bee Colony (ABC) Algorithm for Constrained Optimization Problems, Applied Soft Computing, Volume 11, Issue 3, 2011, pp. 3021-3031

[5] Efrén Mezura-Montes, Carlos A. Coello Coello, A Simple Multimembered Evolution Strategy to Solve Constrained Optimization Problems, IEEE transactions on evolutionary computation, Vol. 9, No. 1, 2005, pp. 1-17

[6] Zavala A, Aguirre A, Diharce E, Constrained optimization via particle evolutionary swarm optimization algorithm (PESO), In Proceedings of the 2005 conference on Genetic and evolutionary computation (GECCO'05), No. 1-59593-010-8, 2005, pp. 209–216

[7] Milan Tuba, Nebojsa Bacanin, Nadezda Stanarevic : Adjusted artificial bee colony (ABC) algorithm for engineering problems, WSEAS Transaction on Computers, Volume 11, Issue 4, April 2012, pp. 111-120

[8] Bacanin N., Tuba M., Artificial Bee Colony (ABC) Algorithm for Constrained Optimization Improved with Genetic Operators, Studies in Informatics and Control, Vol. 21, Issue 2, June 2012, pp. 137-146

[9] Brajevic I., Tuba M., An upgraded artificial bee colony algorithm (ABC) for constrained optimization problems, Journal of Intelligent Manufacturing, published Online First, Jan. 2012, DOI: 10.1007/s10845-011-0621-6

[10] X.S. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press (2008)

[11] Tuba M., Swarm Intelligence Algorithms Parameter Tuning, Proceedings of the American Conference on Applied Mathematics (AMERICAN-MATH'12), pp. 389-394, Harvard, Cambridge, USA, January 2012

[12] X.-S. Yang, Firefly algorithms for multimodal optimization, in: Stochastic Algorithms: Foundations and Applications, SAGA 2009, Lecture Notes in Computer Sciences, Vol. 5792, 2009, pp. 169-178

[13] Subotic M., Misic I., Tuba M., An Object-Oriented Implementation of the Firefly Algorithm for Continuous Unconstrained Optimization Problems, Proceedings of the American Conference on Applied Mathematics (AMERICAN-MATH'12), Harvard, Cambridge, USA, January 2012, pp. 411-416

[14] Subotic M., Tuba M., Stanarevic N., Parallelization of the Firefly Algorithm for Unconstrained Optimization Problems, Proceedings of the 1st Internatonal Conference on Computing, Information Systems and Communications (CISCO '12), 2012, Singapore,, pp. 264-269

[15] A. H. Gandomi, X. S. Yang, A. H. Alavi, Mixed variable structural optimization using Firefly Algorithm, Computers & Structures, Volume 89, Issues 23–24, December 2011, Pages 2325–2336

[16] M.H. Horng, R.J Liou, Multilevel minimum cross entropy threshold selection based on the firefly algorithm, Expert Systems with Applications, Vol. 38, Issue 12, 2011, pp. 14805-14811

[17] X. S. Yang, S. S. Hosseini, A. H. Gandomi, Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect, Applied Soft Computing, Vol. 12, No. 3, 2012, pp. 1180-1186

[18] Efrén Mezura-Montes Carlos A. Coello Coello, Constraint-handling in nature-inspired numerical optimization: Past, present and future, Swarm and Evolutionary Computation, Vol. 1, No. 4, pp. 173-194, 2011

[19] Deb K, An Efficient Constraint-handling Method for Genetic Algorithms, Computer Methods in Applied Mechanics and Engineering, Volume 186, No. 0045-7825, 2000, pp. 311-338