Firefly Programming For Symbolic Regression Problems

Mohamed Aliwi
Department of Computer Engineering
Ondokuz Mayıs University
Samsun, Turkey
mohamed.aliwi@bil.omu.edu.tr

Selcuk Aslan
Department of Computer Engineering
Ondokuz Mayıs University
Samsun, Turkey
selcuk.aslan@omu.edu.tr

Sercan Demirci
Department of Computer Engineering
Ondokuz Mayıs University
Samsun, Turkey
sercan.demirci@omu.edu.tr

***Abstract*—Symbolic regression is the process of finding a mathematical formula that fits a specific set of data by searching in different mathematical expressions. This process requires great accuracy in order to reach the correct formula. In this paper, we will present a new method for solving symbolic regression problems based on the firefly algorithm. This method is called Firefly Programming (FP). The results of applying firefly programming algorithm to some symbolic regression benchmark problems will be compared to the results of Genetic Programming (GP) and Artificial Bee Colony Programming (ABCP) methods.**

Keywords—firefly algorithm, symbolic regression, automatic programming.

I. Introduction

Assuming we have a number of different input values $x_1, x_2, x_3, ..., x_i$, and we also have a number of values $y_1, y_2, y_3, ..., y_i$ that result from applying an unknown mathematical formula to the previous input values. The process of finding a mathematical formula $f(x)$ where $f(x_k) = y_k$ for all $k \in 1, 2, 3, ..., i$ is called "Symbolic Regression". Symbolic regression process will try to find the fittest mathematical formula by searching in mathematical expressions. This process is different from linear and non-linear regressions because it uses specific parameters in order to form an optimal formula [1]. Solving symbolic regression problems started when John Koza introduced a new genetic algorithm (GA) based method which can solve such these problems. This method was called as genetic programming (GP) [2] [3]. Genetic programming relies mainly on the principle of evolution, where it initially creates a set of different mathematical formulas and then performs some operations in order to reach the fittest formula. Another method to solve symbolic regression problems is artificial bee colony programming (ABCP) which was introduced in 2012 [4]. APCB is mainly based on artificial bee colony algorithm, which has been shown to be one of the most popular swarm-based optimization algorithms.
In the remainder of this paper, we will give an overview of firefly algorithm in Section II. We will describe Firefly programming in Section III. In Section IV, we will discuss the practical results of Firefly programming on some common benchmark symbolic regression problems. Last Section will give the conclusion.

978-1-7281-7206-4/20/$31.00

II. Firefly Algorithm

Firefly Algorithm is a meta-heuristic optimization algorithm introduced by Xin-She Yang in 2008 [5] [6]. Firefly algorithm has been inspired from the behavior of the light flashing fireflies in nature. Before going into the details of the firefly algorithm, let us first give an overview of the firefly's behavior in nature.

A. Behavior of Fireflies

Firefly or lightning bug is an insect that is distinguished by its ability to emit light depending on a number of chemicals. These fireflies are found in most hot tropical regions and forests, and there are about 2000 different species. Since these fireflies are located in a harsh environments, they prefer to be active at night. Using light helps them see each other in the vast forests and tall interlocking trees that hide the moonlight and stars on clear nights.

B. Firefly Algorithm

In firefly algorithm (FA), there are some important points to bear in mind [7]:

- All fireflies are unisex, thats means any firefly will be attracted towards all other fitter or brighter ones.
- The attractiveness of any firefly is directly proportional to its brightness.
- The attractiveness of fireflies decreases with increasing distance because the light that these fireflies emit is absorbed by the surrounding medium the greater the distance.
- Fitness or brightness of firefly is determined by an objective function.

FA mainly relies on two concepts: light intensity and attractiveness. While light intensity determines the source of the light, attractiveness determines the change in the position of the firefly attracted to light. Light intensity is defined as:

$$I = I_0 e^{-\gamma r_{ij}^2} \tag{1}$$

where I_0 is the original light intensity which is determined by objective function, γ is the light absorption coefficient, r_{ij} is the distance between firefly i and j. As we can see in (1), light intensity is inversely proportional to the square of the distance

between the fireflies. Also relative attractiveness value can be calculated by:

$$\beta = \beta_0 e^{-\gamma r_{ij}^2} \tag{2}$$

where β_0 is the initial value of attractiveness. The attraction property in FA helps this algorithm not falling into the trap of local minimums because the fireflies is distributed randomly, this means that the best firefly will continue to attract the other ones. After determining the value of attraction, firefly will change its location according to the following formula:

$$x_i = x_i + \beta(x_j - x_i) + \alpha(\phi - 1/2) \tag{3}$$

where x_i, x_j are the locations of firefly i and j respectively. α is constant and ϕ is a random number between 0 and 1.
The distance between two fireflies i and j is defined as the Euclidean distance:

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{d=1}^{D}(x_{id} - x_{jd})^2} \tag{4}$$

where D is dimension. The pseudo code of firefly algorithm is shown below:

Algorithm 1 Firefly Algorithm

```
Initialize MaxGeneration, β0, α, γ parameters
Randomly generate the initial population of n fireflies
Define the objective function f(x)
Calculate light intensity I_i = f(x_i) for all n fireflies
while Generation < MaxGeneration do
  for i = 1 : n all n fireflies do
    for j = 1 : n all n fireflies do
      if I_i < I_j then
        move firefly i towards j
      end if
      Vary attractiveness with distance i via e^{-γ r^2}
    end for
  end for
  Rank the fireflies and find the current global best firefly
  Generation = Generation + 1
end while
Post-process the results
```

III. Firefly Programming

Firefly programming (FP) is an extended version of firefly optimization algorithm to solve symbolic regression problems. Initially, FP creates a set of different mathematical formulas that consist of a set of mathematical expressions and terminals. These initially created formulas represent the first generation of the population of FP. Mathematical formulas are represented by tree structure called Koza Tree that is presented by John Koza [2]. Mathematical expressions used in FP can be simple mathematical operations such as addition, subtraction, division, and multiplication, or they can be mathematical functions like trigonometric functions, logarithm functions or exponential functions. Terminals may be variables like x and y, or some constant integers like 0, 1, 2, ... etc. As in GP, in order to form a mathematical formula, both "ramped and ramped" and "full and grow" methods [2] are used in FP. This combination allows FP to create various mathematical formulas and reduces the similarity between them.

As in FA algorithm, FP is based on both brightness and attractiveness. In FP, the brightness of a firefly is defined by:

$$fit_i = \frac{1}{1 + \sigma_i} \tag{5}$$

where σ_i is the standard error of firefly i which can be calculated by:

$$\sigma_i = \sum_{i=1}^{n} |f(x_i) - g(x)| \tag{6}$$

where $f(x_i)$ is the i^{th} generated formula, $g(x)$ is targeted formula, n is the number points in dataset.
As in standard FA, light intensity and attractiveness are inversely proportional to the square of the distance between the fireflies which can be calculated by:

$$r_{ij} = |fit_i - fit_j| \tag{7}$$

Location-updating operation can be replaced by an operation similar to the crossover operation in GA. This operation is called "Sharing" operation. The probability of sharing operation is directly proportional to both light intensity and attractiveness values. **Fig. (1)** shows an example of sharing operation. We must also show that in the case of division, when the denominator is equal to the value of zero, we adjust the tree structure so that the denominator is randomly chosen to be one of the values of the terminals.

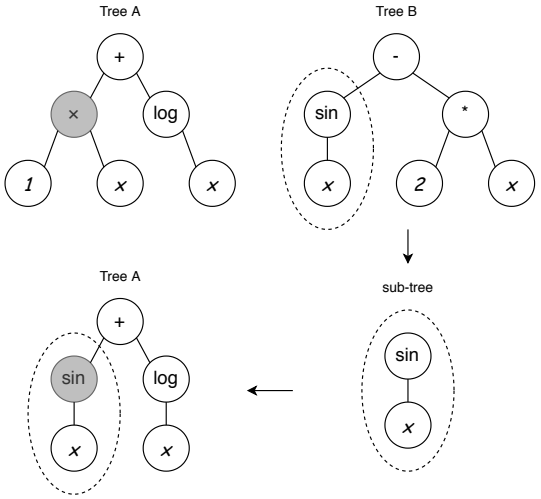

Figure 1: Sharing operation example

IV. Experimental Results

In this section we will compare firefly programming (FP), genetic programming (GP) [9] and artificial bee colony programming (ABCP) [8] using eight symbolic regression benchmark problems [10] [11] [12] which are shown in **Table I**. These equations can be classified into three classes, trigonometric equations, logarithmic equations and linear equations. FP was coded in Python3 and tested using Intel® Core™ i7-7500U 2.7 GHz × 4 processor with 8.0 GB of RAM. All experimental tests was performed using Ubuntu 19.10.

TABLE I: BENCHMARK EQUATIONS

Functions	Fit cases
$F_1 = x^4 + x^3 + x^2 + x$	20 random points $\subseteq$ [-1, 1]
$F_2 = x^5 + x^4 + x^3 + x^2 + x$	20 random points $\subseteq$ [-1, 1]
$F_3 = sin(x) + sin(x + x^2)$	20 random points $\subseteq$ [0, 1]
$F_4 = sin(x^2)cos(x) - 1$	20 random points $\subseteq$ [0, $\pi/2$]
$F_5 = log(x + 1) + log(x^2 + 1)$	20 random points $\subseteq$ [0, 2]
$F_6 = \sqrt{x}$	20 random points $\subseteq$ [0, 4]
$F_7 = sin(x) + sin(y^2)$	100 random points $\subseteq$ [0, 1] $\times$ [0, 1]
$F_8 = 2sin(x)cos(y)$	100 random points $\subseteq$ [0, 1] $\times$ [0, 1]

TABLE II: PARAMETER VALUES USED IN FP, GP AND ABCP

Parameter	Value
Initial max. depth	6
Max. depth	15
Non-terminals	$+, -, \times, \div$, sin, cos, log, exp
Terminals	for 1 parameter, x and 1. for 2 parameters, x, y and 1
Number of runs	100

In order for the comparison to be fair between the three mentioned methods, some of the parameters used have been standardized. These parameters' values are given in **Table II**. Other special parameters used in comparison of FP, GP and ABCP are displayed in **Table III**. Objective function used to determine fitness values in all three methods is:

$$f_i = \sum_{j=1}^{n} \|g_j - t_j\| \tag{8}$$

where g_j stands for generated function's output, t_i stands for targeted function's output for j^{th} point. n is the total number of points. In such these problems, optimization algorithms will try to find the global minimum solution.

TABLE III: OTHER CONTROL PARAMETER VALUES OF FP, GP AND ABCP

FP		GP		ABCP	
Population Size	50	Population Size	500	Colony Size	500
Evaluations	25000	Evaluations	25000	Evaluations	25000
Alpha	1.0	$P_{crossover}$	0.9	Limit	500
$Beta_0$	0.5	$P_{mutation}$	0.05		
Gamma	1.0	Tournament size	3		

After running the FP 100 times independently, the results are compared with the results of both GP [9] and ABCP [8] methods. **Table IV** displays mean error value of each problem.

TABLE IV: MEAN ERROR VALUES OF FP, GP AND ABCP

Method	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
FP	0.24	0.43	0.14	**0.13**	0.26	0.30	**0.12**	**0.09**
GP [9]	0.30	0.40	**0.11**	0.27	**0.15**	0.25	1.67	1.19
ABCP [8]	**0.11**	**0.19**	0.21	0.55	0.24	**0.17**	0.62	0.69

As we can see, FP showed best performance on F_4, F_7 and F_8. Although it gave mean results for both F_1 and F_3, it showed the worst results compared to GP and ABCP in finding solutions for F_2, F_5 and F_6. Bad results of FP were not far from the results shown by other methods. For example, in F_2, the difference in mean error values produced by FP and GP was not significantly different. In FP, the worst mean error value was 0.43 for F_2, while it was 0.69 for F_8 in ABCP and 1.67 for F_7 in GP.

In general, FP showed poor performance in finding logarithmic and square-root equations. On the other hand, it showed excellent results in finding equations containing more than one variable. **Fig. (2)** shows the generated and targeted functions' graphs of F_8 which has two different variables x and y.

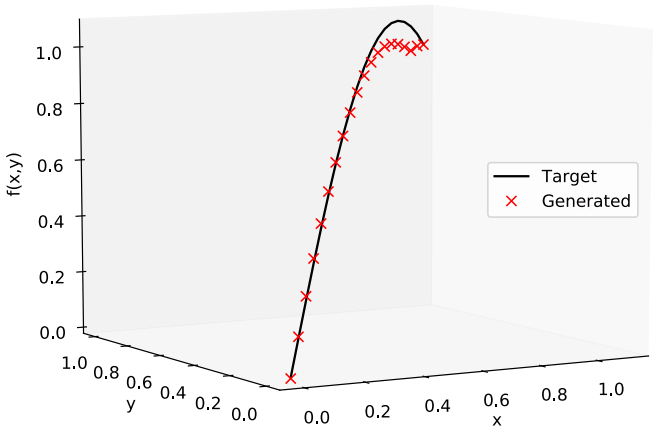

Figure 2: Targeted and Generated functions' graphs of F_8

Although FP is poorly performing in finding some equations, it sometimes gives results very close to the targeted ones. If we take F_2 as an example, **Fig. (3)** shows that the generated equation by FP is very similar to the targeted one.

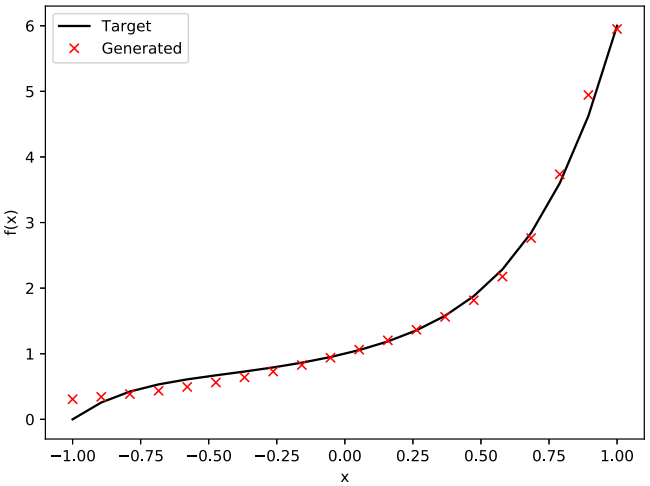

Figure 3: Targeted and Generated functions' graphs of F_2

V. CONCLUSION

In this paper, firefly programming (FP) was presented as a new method for solving symbolic regression problems.

FP is mainly based on firefly algorithm which shows good performance in optimization problems. After performing the necessary experiments, firefly programming performed well in analyzing data and finding suitable mathematical formulas for given symbolic regression problem's datasets.
In the next works we will try as much as possible to minimize the mean error value while solving symbolic regression problems. This can be done by finding optimal values for the parameters used in FP, or by modifying the basic firefly algorithm to give better results than the current ones.

REFERENCES

[1] M.D. Schmidt, H. Lipson, "Co-evolving fitness predictors for accelerating and reducing evaluations", Genetic and Evolutionary Computation, vol. 5, Springer, 2006.

[2] J.R. Koza, "Genetic Programming: On the Programming of Computers by Means of Natural Selection", USA: MIT Press, 1992.

[3] J.R. Koza, F.H. Bennet, D. Andre, M. Keane, "Genetic Programming III", Morgan Kaufnamm pub., 1999.

[4] D. Karaboga, C. Ozturk, N. Karaboga, B. Gorkemli, "Artificial bee colony programming for symbolic regression", Information Sciences, 209, pp.1-15, 2012.

[5] X.S. Yang, "Nature-inspired Metaheuristic Algorithms", United Kingdom:Luniver Press, 2008.

[6] X.S. Yang, "Firefly algorithms for multimodal optimization", Mathematics, vol. 5792, pp. 169-178, 2009.

[7] X.S. Yang, "Nature-inspired algorithms and applied optimization", United Kingdom:Springer, pp. 245-255, 2018

[8] B. Gorkemli, D. Karaboga "Developing new artificial bee colony programming (ABCP) methods and symbolic regression applications", Erciyes University, Kayseri, Turkey, 2015.

[9] N. Uy, N. Hoai, M. O'Neill, R. McKay, D. Phong, "On the roles of semantic locality of crossover in genetic programming", Information Sciences, 235, pp. 195-213, 2013.

[10] N.X. Hoai, R.I. McKay, D. Essam, R. Chau, "Solving the symbolic regression problem with tree-adjunct grammar guided genetic programming: thecomparative results", Proceedings of the IEEE Congress on Evolutionary Computation, CEC'02, pp. 1326–1331, 2002.

[11] C. Johnson, "Genetic programming crossover: does it cross over?", Proceedings of the 12th European Conference on Genetic Programming, Springer, pp 97–108, 2009.

[12] M. Keijzer, "Improving symbolic regression with interval arithmetic and linear scaling", Proceedings of the 6th European Conference on Genetic Programming, Springer, pp. 70–82, 2003.