Firefly Algorithm Applied to Integer Programming Problems

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

Abstract: - Firefly algorithm is a recently added member of the swarm intelligence heuristics family. In this paper the firefly algorithm is adjusted and applied to integer programming problems. In order to deal with integer programming problems, firefly algorithm rounds the parameter values to the closest integer after producing new solutions. The performance of firefly algorithm is tested on seven problems widely used in the literature. Artificial bee colony algorithm is also implemented for comparison with the results of the firefly algorithm. Experimental results show that the firefly algorithm proved to be superior in almost all tested problems.

Key-Words: - Firefly algorithm, Optimization metaheuristics, Integer programming, Swarm intelligence

1 Introduction

Linear programming optimization can find optimal solution to problems with hundreds of thousands of constraints and variables. Modeling strength and solvability makes linear programming technique applicable in real life problems. Integer programming is a form of mathematical optimization program where some or all employed variables are required to have integer values. This method adds additional constraints to linear programming. This change increases the number of problems that can be modeled, but also makes the models more difficult to solve [1]. One interesting aspect of integer programming is that two similar formulations of the same problem can produce different computational results – one formulation can quickly converge to the optimal solution, while the other may take long time to solve the problem in hand [1].

Integer programming models arise in almost every area of applied mathematical programming [2]. Warehouse location problem deals with the tradeoffs between transportation costs and operational costs of distribution centers. Capital budgeting problem refers to the assessment of long term investments, such as new machinery, new plants, products, etc. VLSI (very large scale integration) circuits design problems and robot path planning problems with variables taken from Delaunay triangulation of the free configuration space [3] are also solvable by integer programming method. The entire class of problems referred to as sequencing, scheduling and routing is inherently integer programs. This class of problems might for example deal with the scheduling of students and classrooms in such a way that the number of students who cannot take their first choice of classes is minimized [2].

There are many variants of integer programming methods. In pure integer programming problems, all variables are integers. These problems can be formulated as follows:

$$minimize\ f(x),\ x \in F \subseteq Z^n \qquad (1)$$

where F is the feasible region of the search space, and Z is the set of integer values. One instance of integer programming programs, where the integer variables can be only 0 or 1 is called binary integer programming [3]. There are also mixed binary integer programming problems where some decision variables are binary, and other variables are either general integer or continuous valued.

Exact integer programming techniques such as Branch and Bound or dynamic programming have high computational cost, since they explore a search tree containing hundreds or more nodes on large-scale real-life problems [4]. On the other side heuristic methods can be used for solving integer programming problems. Swarm intelligence metaheuristics [5], among which ant colony optimization [6], [7], artificial bee colony

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

optimization [8], [9], [10] and cuckoo search [11] are prominent and were used successfully for similar problems. Heuristics typically have polynomial computational complexity, but they do not guarantee that the optimal solution will be captured. In order to solve integer programming problems, the most of heuristics truncate or round the real valued solutions to the nearest integer values. 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 were successfully applied in solving integer programming problems [5], [6], [14].

Firefly Algorithm (FA) is a very promising recent developed metaheuristic algorithm inspired by the flashing behavior of fireflies [15]. The basic firefly algorithm is proposed to solve unconstrained optimization problems. There are object-oriented software implementations [16], as well as parallelized versions for unconstrained optimization problems of the FA [17]. Since its invention different variants of FA have been developed and used in many practical fields [18], [19], [20]. In this paper, firefly algorithm is applied to integer programming problems. The performance of the proposed firefly algorithm was compared with those of ABC algorithm.

This paper is organized as follows. Section 2 presents firefly algorithm. A brief description of ABC is provided in Section 3. Section 4 describes seven benchmark problem formulations. Section 5 presents the experimental setup adopted and provides an analysis of the results obtained from our empirical study. Our conclusions and some possible plans for future research are provided in Section 5.

2 Firefly algorithm

The Firefly algorithm proposed by Xin-She Yang is based on the idealized behavior of the flashing characteristics of fireflies [15]. These flashing characteristics can be summarized by the following three rules:

1. All fireflies are unisex so that one firefly will be attracted to 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 affected or determined by the landscape of the objective function to be optimized.

The attractiveness of the firefly and the movement towards the attractive firefly are two significant points in the firefly algorithm. 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 I of a firefly is chosen to reveal its recent position of its fitness value or objective function $f(x)$:

$$I_i = f(x_i) \tag{2}$$

The less bright (attractive) firefly is attracted and moved to the brighter one. 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. Therefore the attractiveness of firefly β is relative. Therefore, it will vary with the distance r_{ij} between firefly *i* and firefly *j*. In addition it should be allowed the attractiveness to vary with the degree of absorption because light intensity decreases with the distance from its source, and light is also absorbed in the media. It is known that the light intensity $I(r)$ varies following the inverse square law:

$$I(r) = \frac{I_0}{r^2} \tag{3}$$

where I_0 represents the light intensity at the source. The combined effect the inverse square law and absorption can be approximated using the following Gaussian form:

$$I(r) = I_0 e^{-\gamma \cdot r^2} \tag{4}$$

As a firefly's attractiveness is proportional to the light intensity seen by adjacent fireflies, the attractiveness function of the firefly is established by:

$$\beta(r) = \beta_0 \cdot e^{-\gamma \cdot r^2} \tag{5}$$

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

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:

$$x_{ik} = x_{ik} + \beta_0 \cdot e^{-\gamma \cdot r_{ij}^2} \cdot (x_{ik} - x_{jk}) + \alpha \cdot S_k \cdot \left(rand_{ik} - \frac{1}{2} \right) \quad (6)$$

where $k = 1,2,...,D$ (D is dimension of problem). In Eq. (6) third term is randomization term where $\alpha \in [0,1]$, S_k are the scaling parameters and $rand_{ik}$ is random number between 0 and 1. In our firefly algorithm employed for integer programming problems, as in the version of FA proposed to solve structural optimization problems [19], 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:

$$\alpha = \alpha_0 \cdot \theta^t \quad (7)$$

where $0 < \theta < 1$ is the reduction factor of randomization. In addition, the scaling parameters S_k in all D dimensions are determined by:

$$S_k = |u_k - l_k| \quad (8)$$

where l_k and u_k are the lower and upper bound of the parameter x_{ik}.

Distance r_{ij} between fireflies i and j is obtained by Cartesian distance form by:

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

The pseudo code of the firefly algorithm adjusted for integer programming problems is given below:

```
Input
Objective function f(x), $x = (x_1,...,x_D)^T$ {cost function}
S = [$l_k$, $u_k$]; k = 1, 2,..., D {given constraints}
$\alpha_0$, $\beta_0$, $\gamma$ {algorithm's parameters}

Output
$x_{\min}$ {obtained minimum location}

Begin
Initialize a population of fireflies $x_i$ (i = 1, 2, ..., n) randomly
Initialize algorithm's parameters $\alpha$, $\beta_0$, $\gamma$

while (t <MaxGeneration)
  for i = 1 : n
    for j = 1 : i
      if (f(x_j) < f(x_i)){move firefly i towards j}
      Obtain attractiveness which varies with distance r via exp[-$\gamma$ r] by Eq.(5)
      Move firefly i towards j in all D dimensions by Eq.(6)
      Evaluate and update the new solution
      end if
    end for j
  end for i

Reduce the randomization parameter $\alpha$ by Eq.(7)
Rank the fireflies and find the current best
end while
```

3 Artificial bee colony algorithm

In ABC algorithm the colony of artificial bees consists of three groups of bees: employed bees, onlookers and scouts **Error! Reference source not found.**. The number of the employed bees is equal to the number of food sources and an employed bee is assigned to one of the sources. The employed bees search the food around the food source in their memory and also they share their food information with onlookers. The onlookers tend to select good food sources from those founded by the employed bees and then they calculate a new solution from the selected food source. The scout bees randomly search the environment surrounding the hive for new food sources.

Short pseudo–code of the ABC algorithm is given below:

```
Initialize the population of solutions
Evaluate the population
t = 0
repeat
    Employed bee phase
    Calculate probabilities for onlookers
    Onlooker bee phase
    Scout bee phase
    Memorize the best solution achieved so far
t = t+1
until t = Tmax
```

In ABC algorithm adjusted for integer programming problems [14], in employed bee phase an update process is performed for each solution x_i in order to produce a new solution v_i :

$$v_{ij} = \begin{cases} x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}), if & R_j \le MR \\ x_{ij} & , \quad otherwise \end{cases} \quad (10)$$

where R_j is a uniformly random real number within [0, 1), k is randomly chosen index from the whole population and has to be different from *i*, *MR* is the modification rate parameter that controls the possible modifications of optimization parameters , φ_{ij} is a random number between [-1, 1) and $j = 1,2,...,D$. Then, a greedy selection is done between x_i and v_i, which completes the update process. The main distinction between the employed bee phase and the onlooker bee phase is that every solution in the employed bee phase involves the update process, while only the selected solutions have the opportunity to update in the onlooker bee phase. An inactive solution refers to a solution that does not change over a certain number of generations. In scout bee phase one of the most inactive solutions is selected and replaced by a new randomly generated solution.

4 Benchmark Problems

To test the performance of artificial bee colony and firefly algorithms on integer programming problems, we used seven problems widely used in the literature. These problems are listed below:

Test Problem 1. This problem is defined by:

$$F_1(x) = \|x\|_1 = |x_1| + |x_2| + \dots + |x_D|$$

where D is the dimension, and $x \in [-100, 100]^D$. The global minimum is $F_1(x) = 0$.

Test Problem 2. This problem is defined by:

$$F_2(x) = x^T \cdot x = [x_1 \quad \dots \quad x_D] \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix}$$

where D is the dimension, and $x \in [-100, 100]^D$. The global minimum is $F_2(x) = 0$.

Test Problem 3. This problem is defined by:

$$F_3(x) = (9 \cdot x_1^2 + 2 \cdot x_2^2 - 11)^2 + (3 \cdot x_1 + 4 \cdot x_2^2 - 7)^2$$

The global minimum is $F_3(x) = 0$.

Test Problem 4. This problem is defined by:

$$F_4(x) = (x_1 + 10 \cdot x_2)^2 + 5 \cdot (x_3 - x_4)^2 + 5 \cdot (x_2 - 2 \cdot x_3)^4 + 10 \cdot (x_1 - x_4)^4$$

The global minimum is $F_4(x) = 0$.

Test Problem 5. This problem is defined by:

$$F_5(x) = ({x_1}^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

The global minimum is $F_5(x) = 0$.

Test Problem 6. This problem is defined by:

$$F_6(x) = -3803.84 - 138.08 \cdot x_1 - 232.92 \cdot x_2 + 123.08 \cdot {x_1}^2 + 203.64 \cdot {x_2}^2 + 182.25 \cdot x_1 \cdot x_2$$

The global minimum is $F_6(x) = 0$.

Test Problem 7. This problem is defined by:

$$F_7(x) = 100 \cdot (x_2 - {x_1}^2)^2 + (1 - x_1)^2$$

The global minimum is $F_7(x) = 0$.

For each test problem the solutions were constrained in $[-100, 100]^D$, where D is the dimension of the corresponding problem.

5 Parameter settings, results and discussion

In order to solve the integer programming problems, artificial bee colony and firefly algorithms round the parameter values to the closest integer after evolution according to Eq.(6) and Eq.(10). The solutions were also rounded after initialization phase of the algorithms. Therefore, they were considered as integer numbers for all operations.

The artificial bee colony and firefly algorithms have 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 7x64 Professional operating system.

In all experiments for both algorithms the same size of population (SP) of 40 and the same maximum number of iterations (MCN) of 100 is used. Parameters of ABC algorithm are the same as in [14]: modification rate parameter is 0.8 and limit parameter is $2.5 \cdot SP \cdot D$. In FA, the reduction scheme described by Eq.(6) was followed by reducing α from 0.5 to 0.00001. In our implementation of FA we set parameter γ , which characterizes the variation of attractiveness to the value 1. Also, the initial value of attractiveness $\beta_0 = 1$ was used. Each of the experiments was repeated 50 runs and the algorithms are terminated when correct solution is reached.

The results of ABC and FA for the test problems P1-P7 are shown in Table 1. Results in Table1 are the number of success rates to correct solution of 50 independent runs along with the mean and standard deviation of the required number of function iterations generated from all tests included unsuccessful cases.

From Table1 it can be seen that FA produces better results in terms of success rate in allowed maximum number of function iterations for the majority of tested functions. Specially, for higher dimensions of P1 and P2 functions, firefly algorithm performs much better than ABC algorithm. In these cases the success rate of FA is higher than 85%, while ABC algorithm was not able to find the correct solution in allowed maximum number of function iterations in none of the runs.

In terms of mean number and standard deviation of iterations, the FA produces better results than ABC algorithm for lower dimensional functions (P3, P4, P5, P6 and P7) and for higher dimensions of P1 and P2 functions, while ABC has better results for test functions P1 and P2 (for D = 5). Hence, FA can reach the correct solution faster than ABC in the most of tested cases. Although ABC performs better than FA on P1 and P2 when *D*=5, by the increase in dimension, the performance of ABC significantly deteriorates, but FA preserves its robustness.

		ABC			**FA**		
Problem	Dimension	Succ Rate	Mean Iter	St Dev	Succ Rate	Mean Iter	St Dev
P1	5	50/50	25.66	3.17	49/50	37.64	9.47
P1	10	50/50	60.30	4.41	49/50	47.76	7.56
P1	15	29/50	97.18	3.68	48/50	51.08	10.00
P1	20	0/50	100.00	0.00	48/50	51.94	9.82
P1	25	0/50	100.00	0.00	47/50	53.2	11.83
P1	30	0/50	100.00	0.00	43/50	57.4	17.20
P2	5	50/50	31.48	4.59	50/50	36.92	2.39
P2	10	50/50	78.04	4.95	49/50	48.04	7.49
P2	15	0/50	100.00	0.00	49/50	50.00	7.19
P2	20	0/50	100.00	0.00	47/50	52.84	11.92
P2	25	0/50	100.00	0.00	47/50	53.14	11.85
P2	30	0/50	100.00	0.00	45/50	55.46	14.86
P3	2	50/50	8.74	3.25	50/50	4.34	2.54
P4	4	17/50	95.56	8.02	50/50	32.76	2.77
P5	2	50/50	11.4	4.52	50/50	8.28	4.45
P6	2	50/50	9.72	4.20	50/50	8.80	3.82
P7	2	49/50	26.42	21.11	50/50	8.68	3.88

Table1. Dimension, Success Rate, Mean and Standard Deviation of iteration numbers for ABC and FA for test problems P1-P7

6 Conclusion

In this paper, performance of the firefly algorithm was investigated on integer programming problems. The results were compared to the results of artificial bee colony algorithm adjusted for integer programming problems. We have tested artificial bee colony and firefly algorithms on the set of seven integer programming problems widely used in the literature. Compared with ABC, FA can produce better performance in terms of both, efficiency and success rate for the majority of tested functions. This implies that FA is potentially more powerful in solving NP-hard problems, regard to the fact that the most of combinatory problems can be converted to integer programming problems. Future work will include investigation on practicability of FA on some NP-complete combinatory problems. In addition, possible modifications of FA in order to improve its performance will also be considered.

References:

[1] Bosh A. Robert, *Integer Programming and Conway's Game of Life*, SIAM Review, Vol. 41, No. 3,1999., pp. 594–604.

[2] Bradley P. Stephen, Hax C. Arnoldo, Magnati L. Thomas, Applied Mathematical Programming, Addison-Wesley Publishing, 1977. pp.716.

[3] Habibi G., Masehian E., Behetshti M.H.T., Binary Integer Programming Model of Point Robot Path Planning, IECON 2007. 33rd Annual Conference of the IEEE, 2007. pp. 2841 – 2845

[4] Rouillon S., Desaulniers G., Soumis F.: An extended branch-and-bound method for locomotive assignment. Transportation Research Part B: Methodological, 2006, Vol 40, No.5,2006, pp.404–423

[5] Tuba M., Swarm Intelligence Algorithms Parameter Tuning, Proceedings of the American Conference on Applied Mathematics, pp. 389-394, Harvard, Cambridge, USA, January 2012

[6] Jovanovic R., Tuba M.: 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

[7] Jovanovic R, Tuba M: Ant Colony Optimization Algorithm with Pheromone Correction Strategy for Minimum Connected Dominating Set Problem, Computer Science and Information Systems (ComSIS), Vol. 9, Issue 4, Dec 2012, DOI:10.2298/CSIS110927038J

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

[9] 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

[10] 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

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

[12] Parsopoulos K. E., Vrahatis M. N., Recent Approaches to Global Optimization Problems through Particle Swarm Optimization. Natural Computing, Vol. 1, Issues 2-3, 2002., pp. 235-306.

[13] Liu J., Sun J., Xu W., Quantum-Behaved Particle Swarm Optimization for Integer Programming, LCNS, Vol. 4233, 2006, pp. 1042-1050.

[14] Karaboga D., Akay B., Solving Integer Programming Problems by Using Artificial Bee Colony Algorithm, LCNS, Vol. 5883, 2009., pp. 355-364.

[15] 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

[16] 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, Harvard, Cambridge, USA, January 2012, pp.411-416

[17] Subotic M, Tuba M, Stanarevic N: Parallelization of the Firefly Algorithm for Unconstrained Optimization Problems, Proceedings of the 1st International Conference on Computing, Information Systems and Communications, 2012, Singapore, pp. 264-269

[18] 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

[19] 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, pp. 2325–2336

[20] Brajevic I, Tuba M, Bacanin N, Firefly Algorithm with a Feasibility-Based Rules for Constrained Optimization, Proceedings of the 6th WSEAS European Computing Conference (ECC '12), ISBN: 978-1-61804-126-5, Prague, Czech Republic, September 24-26, 2012, pp. 163-168