AKADÉMIAI KIADÓ

Pollack Periodica • An International Journal for Engineering and Information Sciences

16 (2021) 1, 14–18

DOI:
10.1556/606.2020.00180

ORIGINAL RESEARCH PAPER

Improving optimization using adaptive algorithms

László Kota[1] and Károly Jármai[2*]

[1] Self-Employed IT Engineer and Software Developer, Budapest, H-1158, Hungary

[2] Institute of Energy and Chemical Machinery, Faculty of Mechanical Engineering and Informatics, University of Miskolc, H-3515 Miskolc, Egyetemváros, Hungary

Received: January 3, 2020 • Revised manuscript received: June 13, 2020 • Accepted: September 7, 2020
Published online: February 26, 2021

ABSTRACT

In the research projects and industrial projects severe optimization problems can be met, where the number of variables is high, there are a lot of constraints, and they are highly nonlinear and mostly discrete issues, where the running time can be calculated sometimes in weeks with the usual optimization methods on an average computer. In most cases in the logistics industry, the most robust constraint is the time. The optimizations are running on a typical office configuration, and the company accepts the suboptimal solution what the optimization method gives within the appropriate time limit. That is, why adaptivity is needed. The adaptivity of the optimization technique includes parameters of fine-tuning. On this way, the most sensitive setting can be found. In this article, some additional adaptive methods for logistic problems have been investigated to increase the effectivity, improve the solution in a strict time condition.

KEYWORDS

heuristic optimization, adaptive optimization, firefly optimization, logistic optimization

1. INTRODUCTION

In research works, logistic problems are solved many times, which are not only nonlinear, but discrete, some are non-continuous, and cannot be represented in diagrams, only with matrix forms. In one of previous research, the problem of adaptive optimization has been employed using evolutionary programming in the optimization of large-scale maintenance systems [1]. If the optimization problem is complex, it is hard to find the global minimum or maximum [2, 3]. In this paper, the firefly algorithm has been applied to a logistic problem with a large variable count. The firefly algorithm is a swarm-based heuristic algorithm presented by Xin-She Yang [4], inspired by the mating behavior of fireflies. The firefly algorithm based on three rules:

- All fireflies are attracted by each other;
- Attractiveness is proportioned by brightness, the less brilliant move towards the brighter one;
- If there is no more luminous firefly than the selected one, that will move randomly in the state space.

The algorithm is shown in *Code 1*.

*Corresponding author.
E-mail: jarmai@uni-miskolc.hu

AKJournals

Code 1. Pseudo-code of the firefly algorithm, (on the basis of [4])

```
Objective function f(x), x=(x1,....., Xd)T
A general initial population of fireflies xi (i=1,2,....., n)
Light intensity Ii at xi is determined by f(xi)
Define light absorption coefficient γ
while (t < MaxGeneration)
    for i = 1 : n all n fireflies
      for j = 1 :n all n fireflies (inner loop)
        if (Ii < Ij), Move firefly I towards j; end if
        Vary attractiveness with distance r via exp[-γ r]
        Evaluate new solutions and update light intensity
      end for j
    end for i
    Rank the fireflies and find the current global best g*
end while
Postprocess results and visualization
```

The firefly algorithm has several control parameters, for example, the absorption coefficient, the randomization control factor, and the firefly population size. The values of these control parameters significantly affect the quality of the achieved solution and the efficiency of the algorithm. It is a problem depends on selecting suitable control parameters for the actual algorithm. Hard to deal with complex issues with many local optima where the most algorithms are trapped. Although it is highly important, there is no consistent methodology for determining the control parameters of the applied firefly algorithm variant. Mostly, the settings are fixed throughout many experiments or set arbitrarily within some predefined ranges [5].

2. THE PROBLEM

The problem to be optimized is a supply chain optimization problem based on the olive oil production by the company Tariş in Turkey described in [6]. The whole problem cannot be described within the framework of this article, so the core function (1) is needed to be optimized here. The main objective function is the profit maximization, as in many other cases:

$$F(X_i) = \sum_{i=1}^{I}\sum_{k=1}^{K}(p_i - c_{ik})Y_{ik} - \sum_{i=1}^{I}(oc\cdot oil_i - pc_i)X_i - t\cdot toil, \quad (1)$$

where p_i is the product price; c_{ik} is the transportation cost; Y_{ik} is the produced quantity; oc is the oil cost (1 L); oil_i is the required oil quantity; pc_i is the packaging cost; X is the total produced quantity of the given packaging unit; $t \times toil$ is the transportation cost of the oil used; and the constraints are:

$$X_i \geq \sum_{k=1}^{K} Y_{ik} \forall i, \quad (2)$$

$$\sum_{i=1}^{I} \mathrm{oil}_i X_i \leq total\ oil, \quad (3)$$

$$Y_{ik} \leq d_{ik}\ \forall i,\ k, \quad (4)$$

$$X_i \in Z \forall\, i, \quad (5)$$

$$Y_{ik} \in Z \forall\, i,\ k. \quad (6)$$

The total produced quantity must less than the entire amount required (1); the oil used must less than the total oil available (2); the amount provided for each region must be less than the required quantity of the given region (3); and the variables must be integers (5), (6).

In the model 10 packaging units (p), and 13 regions (r) have been used, so the matrix (Y) to be optimized, which is the produced quantity matrix, will contain 130 variables.

3. DISTANCE AND MOVEMENT

In the research, the swarm methods like the Particle Swarm Optimization (PSO) [7] and the Firefly Algorithm (FA) [8] are very often used; both of them are common and widely used to solve a wide range of problems. However, until now they were not used for a large number of variables. The firefly algorithm is working well with various test functions, a lot of general problems [9], but how it performs on a large variable count problem.

The first problem was the discrete nature of the problem. The original firefly algorithm was developed to optimize continuous problems and not just continuous problems, but problems where the directions have meaning, so the moving toward have meant so that the fireflies can move toward each other. Thus, the first task was to define the distance of the fireflies and define the movement function. In some articles, they followed that way [10, 11]. The distance of two fireflies (F1, F2) is:

$$dst(F1,\ F2) = \sum_{\substack{i=1..p \\ j=1..r}} \mathrm{Abs}\left(y_{ik}^{F1} - y_{ik}^{F2}\right), \quad (7)$$

where $F1$ and $F2$ are the two fireflies whose distance we want to define; y_{ik} is the decision variable the manufactured quantity of the packaging unit i in the region k.

The movement function is:

$$move\ toward(F1,\ F2):\ y_{ik}^{F1} = y_{ik}^{F1} + \beta\cdot\left(y_{ik}^{F1} - y_{ik}^{F2}\right) + \alpha\cdot rnd - 0.5 \quad (8)$$

executed on every matrix element, where $\beta = e^{-\gamma^* r}$, gamma was set to 1; $\alpha = range$ 0.05 randomization component where the range is the range of interpretation, in this case, it was a set to 30,000 as the upper limit of the decision variables.

When there is no brighter firefly the firefly moves randomly:

$$moverandom(F1)\, y_{ik}^{F1} = y_{ik}^{F1} + rnd(\pm 50). \quad (9)$$

First, the random movement parameter has been fixed, which was selected by guesswork; the firefly count was 5. All runs were done in an average office computer, a 1st generation Intel Core I7-870 processor, 6 GB RAM, the algorithm was programmed in C# .NET and all the test runs were limited 10 min.

The first run, with the usual parameters, as mentioned in the literature [6], gave the following result.

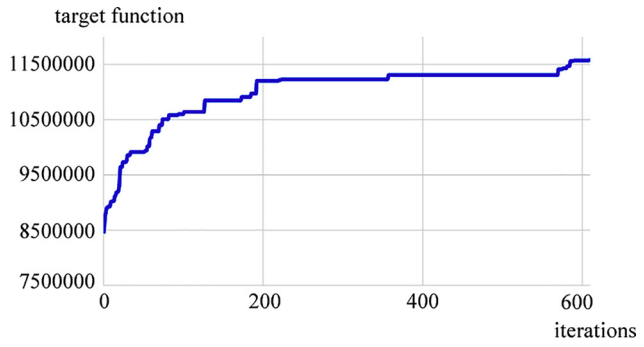

Fig. 1. The target function, firefly algorithm first run without any adjustment

Table 1. Proportional values at random movement

Proportional to the actual value	Target function
$\pm y_{ik}^{F1}$	8862235.24
$\pm y_{ik}^{F1}/2$	10661050.00
$\pm y_{ik}^{F1}/5$	10310898.52
$\pm y_{ik}^{F1}/10$	10167617.44
$\pm y_{ik}^{F1}/50$	11074974.70
$\pm y_{ik}^{F1}/100$	11546460.71
$\pm y_{ik}^{F1}/500$	11618969.21
$\pm y_{ik}^{F1}/1000$	11470180.17
$\pm y_{ik}^{F1}/5000$	11518200.56
$\pm y_{ik}^{F1}/10000$	11600611.80
$\pm y_{ik}^{F1}/25000$	11621109.67
$\pm y_{ik}^{F1}/50000$	11621109.67

Figure 1 shows the convergence of the solution. In the figure, it is visible when the algorithm jumps out in a local optimum, and the function does not start to flatten during the limited 10 min of the run. The best fitness value was: 11579252.39.

4. MODIFIED RANDOM MOVEMENT WITH SELF-ADAPTIVE VALUES

The random movement function (9) moves that firefly, which is not moved in the actual iteration because there was no brightest firefly than him. Several parameters have been tried instead of the first guess ±50 value in the random movement function; the increase was slight. Then a proportional equation has been used where the random movement is proportional to the actual value therefore, the random movement will adapt to the scale of the actual problem.

The proportional value equation is the following:

$$moverandom(F1)y_{ik}^{F1} = y_{ik}^{F1} + rnd\left(y_{ik}^{F1}/n\right) - rnd\left(y_{ik}^{F1}/n\right). \tag{10}$$

The algorithm was run at different proportional values (n) (Table 1). These values show that the target function value is not improved after it reached the 25,000 dividers, because the highest value of the product matrix is about 30,000 so the 25,000 and the 50,000 gives the same random movement value of 1. This modified firefly algorithm with this method of here its brightness improves. So, we generate N random movement variants and choose the best solution; if the solution improves the solution, the firefly will move in that

Table 2. Modified random movement with proportional movement (best values are marked)

N	$\pm y_{ik}^{F1}$	$\pm y_{ik}^{F1}/2$	$\pm y_{ik}^{F1}/5$	$\pm y_{ik}^{F1}/10$
5	11658592.07	11819534.46	11826480.57	11901170.66
10	11708473.55	11850753.79	11922270.75	11933058.08
50	11747862.58	11811321.54	11804486.64	11968312.55
100	11706481.29	11832501.89	11933977.41	11968701.49
500	11898168.58	11777553.58	12069210.91	11971114.62
1,000	11744732.59	11887363.76	11971260.72	11953844.37

N	$\pm y_{ik}^{F1}/50$	$\pm y_{ik}^{F1}/100$	$\pm y_{ik}^{F1}/500$	$\pm y_{ik}^{F1}/1000$
5	11730555.38	11652020.7	11610061.0	11612635.3
10	12028420.92	11831095.4	11512298.0	11433305.7
50	11831781.93	11902323.2	11730092.1	11542878.0
100	11874872.39	11919576.2	11766607.4	11607951.0
500	12060338.92	12120813.7	11683133.6	11571373.9
1,000	12049089.41	12063160.5	11721536.0	11682845.1

N	$\pm y_{ik}^{F1}/5000$	$\pm y_{ik}^{F1}/10000$	$\pm y_{ik}^{F1}/25000$
5	11230433.2	11498258.5	11709050.3
10	11682722.3	11607271.8	11594346.8
50	11639638.0	11524683.7	11567352.8
100	11547163.8	11490801.8	11607922.2
500	11554956.3	11287234.8	11324414.3
1,000	11370669.2	11075469.7	11280026.5

direction, if not, it will stay there. In this case, this firefly program acts as an elitist entity algorithm, because its value does not change; it will go unmodified to the next generation of the fireflies. Several proportional numbers have been tried: the 5, 10, 50, 100, 500 and 1,000 random solutions and selected the best; which has the minimum value of the objective function. Do not forget that the running time is limited, so higher cycle number (N) results in a lower iteration count. Experimental runs have been carried out for all proportional values.

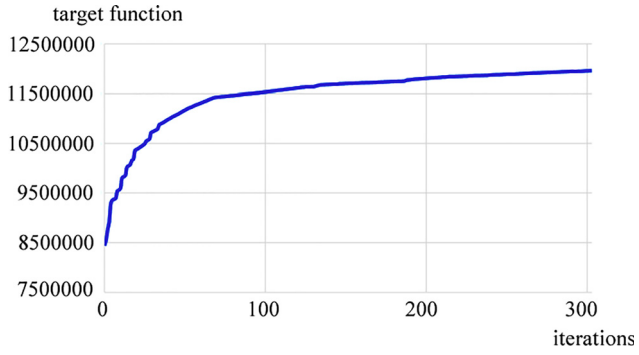

Fig. 2. Modified random movement improved the convergence

The results (Table 2) show that the modified random movement improved the solution in every case. The higher randomization values were better to avoid the local optima or jump out if stuck in a local optimum, but it cannot be too big, because it moves in the state space too far. The best solution was at $\pm y_{ik}^{F1}/100$ and $N = 500$, but a lot of other values are close to this solution. This method also greatly improved the convergence (Fig. 2); the difference can be noticed compared to Fig. 1.

5. ADAPTIVE FIREFLY COUNT

Another critical parameter of the algorithm is the firefly count, which is constant in standard algorithms. The main idea was to improve the search power of the algorithm when the convergence was starting to flatten. So, it was decided to use a simple adaptive method to increase the firefly count. If there is no increase in the global optimum, then the algorithm adds new random initialized fireflies to the state space, *Code 2.*

Table 3. Improvement on the initial solution with adaptive firefly count

N	$\pm y_{ik}^{F1}$	$\pm y_{ik}^{F1}/2$	$\pm y_{ik}^{F1}/5$	$\pm y_{ik}^{F1}/10$
$N = 1$	12070827.22	12122923.38	12125154.01	12144917.07
Imp.	1.43%	1.94%	−0.16%	0.88%
FF count	442	450	451	419
$N = 5$	12051943.05	12098366.66	12087990.38	12108718.04
Imp.	1.28%	1.74%	−0.47%	0.58%
FF count	905.00	895.00	870.00	855.00
$N = 10$	12084262.60	12095241.06	12129499.75	12111717.77
Imp.	1.54%	1.72%	−0.12%	0.60%
FF count	1145.00	1165.00	1135.00	1095.00
N	$\pm y_{ik}^{F1}/50$	$\pm y_{ik}^{F1}/100$	$\pm y_{ik}^{F1}/500$	$\pm y_{ik}^{F1}/1000$
$N = 1$	12134597.06	12142955.4	12010016.1	11978007.0
Imp.	0.94%	1.38%	2.03%	2.46%
FF count	301	274	61	49
$N = 5$	12118359.62	12093069.9	12073598.6	12073523.3
Imp.	0.80%	0.98%	2.54%	3.24%
FF count	780.00	735.00	305.00	925.00
$N = 10$	12124155.78	12083340.9	12101332.1	12106607.0
Imp.	0.85%	0.90%	2.77%	3.50%
FF count	1065.00	1065.00	780.00	565.00
N	$\pm y_{ik}^{F1}/5000$	$\pm y_{ik}^{F1}/10000$	$\pm y_{ik}^{F1}/25000$	
$N = 1$	11300520.0	11830410.1	12082614.0	
Imp.	−3.38%	1.89%	3.09%	
FF count	6	24	457	
$N = 5$	11579538.6	11970996.8	12113865.7	
Imp.	−0.89%	3.04%	3.34%	
FF count	10.00	45.00	905.00	
$N = 10$	11859666.8	12044566.4	12085391.3	
Imp.	1.49%	3.63%	3.11%	
FF count	35.00	375.00	1145.00	

Code 2. A simple adaptive method

```
if iteration-bestiteration > 1 then
      Add N new Firefly
end if
```

This improved method has been tested with $N = 1, 5, 10$ new additional fireflies combined with the self-adaptive random movement method. Table 3 shows the solution. The improvement percentage based on Table 2 improved values with modified random movement and the firefly count at the end of the limited run. According to the results, there is a slight improvement in the results using this method, but at two values/5 and/5,000, there are some minor negative values where the improved method failed. The additional firefly count is a good indicator of the continuous convergence. If it is small it shows that the result gets better and better in every iteration, like at the /5,000 case where the result is worse than the other values, but the firefly count is smaller, so the function has a slow but continuous convergence in the calculated time window.

6. CONCLUSIONS

In this article, some improvement methods have been shown, which have been extensively tested on a large-scale logistic problem, and the combination of these methods has been checked. In discrete logistic problems, where lots of decision variables exist, it is essential to determine the distance metric and the movement function or functions. However, these functions are not precisely specified when to use them and the use of which one from the available variety is appropriate, usually the most straightforward and fastest functions have been used. Several distance metrics can be used, and the movement in most cases in huge state space has to be defined by one or occasionally more matrices, which makes the task even more complicated. These choices can significantly affect the quality of the solution. Because heuristics are used, it is not sure, if the global optimum reached or not and whether the selected movement function for the actual problem is useful or not. The applied improvement methods can help, whether the firefly algorithm or other heuristic methods have been used, and their combination also can be used. Still, it needs serious testing, to determine which helps a lot, and which improves a little and which method did not work in this case.

ACKNOWLEDGEMENTS

The research was partially supported by the Hungarian National Research, Development and Innovation Office – NKFIH – under the project number K 134358.

REFERENCES

[1] L. Kota and K. Jármai, "Adaptive methods in the optimization of large scale technical inspection and maintenance systems," in *XXVII. MicroCAD International Scientific Conference, Material Flow Systems. Logistical Information Technology and Technical Language*, Miskolc, Hungary, Mar. 21–22, 2013, 2013, Paper no. J17.

[2] M. Petrik, G. Szepesi, and K. Jármai, "CFD analysis and heat transfer characteristics of finned tube heat exchangers," *Pollack Period.*, vol. 14, no. 3, pp. 165–176, 2019.

[3] H. N. Ghafil and K. Jármai, "Kinematic-based structural optimization of robots," *Pollack Period.*, vol. 14, no. 3, pp. 213–222, 2019.

[4] X. S. Yang, *Nature-inspired Metaheuristic Algorithms*. Luniver Press, 2008.

[5] N. J. Cheung, X. M. Ding, and H. B. Shen, "Adaptive firefly algorithm: Parameter analysis and its application," *PloS One*, vol. 9, no. 11, Paper no. e112634, 2014.

[6] O. Yurt, L. Kota, K. Jármai, and E. Aglamaz, "Analysis and optimization of an olive oil supply chain: A case from Turkey," *Int. J. Sustain. Agric. Manage. Inform.*, vol. 5, no. 1, pp. 59–79, 2019.

[7] H. N. Ghafil and K. Jármai, "Comparative study of particle swarm optimization and artificial bee colony algorithms," in *XXXII. MicroCAD International Multidisciplinary Scientific Conference*, Miskolc, Hungary, Sep. 5–6, 2018, 2018, Paper no. D1.

[8] L. Kota, "Optimization of the supplier selection problem using discrete firefly algorithm," *Adv. Logistic Syst.*, vol. 6, no. 1, pp. 117–126, 2012.

[9] L. Kota and K. Jármai, "Discretization of the firefly algorithm for the travelling salesman problem," in *28th MicroCAD International Multidisciplinary Scientific Conference*, University of Miskolc, Apr. 10–11, 2014, 2014, Paper no. D30.

[10] S. L. Tilahun and H. C. Ong, "Modified firefly algorithm," *J. Appl. Math.*, vol. 2012, Paper no. 467631, pp. 1–12, 2012.

[11] S. Yu, S. Yang, and S. Su, "Self-adaptive step firefly algorithm," *J. Appl. Math.*, vol. 2013, Article no. ID 832718, pp. 1–8, 2013.