ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Improving the firefly algorithm through the Barnes-Hut tree code

Kiril Ralinovski

Berlin Institute of Technology - Machine Learning Group
Marchstr. 23, Berlin, Germany

Abstract. The firefly algorithm is a nature-inspired meta-heuristic algorithm that has a variety of applications such as multimodal optimization, clustering and finding good solutions for NP-hard problems. The original algorithm and modifications thereof have so far always calculated interactions between all fireflies individually which leads to a complexity of $O(n^2)$. In this paper we present a novel approach to reduce the complexity to $O(n \cdot log(n))$ in lower dimensions by using the Barnes-Hut tree code, which is used for n-body simulations in physics. This is possible due to the similar nature of both problems and requires only small modifications.

1 Introduction

Metaheuristic algorithms are often used in a variety of complex problems such as non-convex optimization and NP-hard problems like the traveling salesman problem. Parametric optimization can be used in almost every field - such as economics, image compression and many more. An important class of meta-heuristic algorithms are the so-called bioinspired algorithms, which take examples for solving these problems from nature. One such prominent algorithm is the firefly algorithm proposed by Yang [12]. It imitates the flashing behaviour of fireflies in order to attract mates. Its original purpose was to solve complex multimodal optimization problems, but it has since then been further developed and used for image thresholding [9], clustering [11], scheduling [7] [8] and many others.
Research has been done to parallelize the algorithm and reduce its running time [6], thus enabling it to use a higher number of fireflies. Some versions used problem-specific knowledge in order to optimize it [5]. Papers have been written on making parallel version of it that can run on graphical processors such as CUDA [6] or improving the way how the fireflies move in order for them to find a better solution [3] [10].
In this paper a novel approach is proposed to use the Barnes-hut tree code to optimize the complexity of the algorithm from $O(n^2)$ to $O(n \cdot log(n))$ in lower dimensions.

2 Algorithms

2.1 The Firefly algortihm

The original firefly algorithm [12] relies on an idealized version of fireflies, which are insects that use flashing patterns to communicate with each other. For a

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

complete introduction to the algorithm including the optimization of hyperparameters please refer to [12]. The algorithm makes an idealized imitation of fireflies. In order to achieve this a certain degree of abstraction is required. It assumes that the fireflies in this environment are unisexual. That is to say that all fireflies are attracted to each other.The stronger the glow of a firefly, the more attractive it is. The brightness of the firefly is the function's value of its position.

The formula for the light intensity from one firefly to another is $I = I_0 e^{-\gamma r^2}$, where r is the distance, γ is the absorption coefficient and I_0 was the original light intensity. A more common notion to measure attractiveness is to use β instead of I with $\beta = \beta_0 e^{\gamma r^2}$. The absorption coefficient is due to the fact that the air absorbs light and thus has greater influence over a greater distance. The new position of the firefly is determined by $x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha\epsilon_i$. ϵ is a randomized vector filled with random values from a Gaussian distribution. This adds an extra random factor to the movement of the fireflies. Alpha is a constant factor that determines how much of an influence this randomness has. The pseudo code is presented in Algorithm 1

```
Data: Objective function f(x), x = (x_1, ..., x_d)^T
Initialize a population of fireflies x_i (i = 1, 2, ..., n);
Define light absorption coefficient;
while t < MaxGeneration do
    for i = 1 : n all n fireflies do
        for j = 1 : i all n fireflies do
            Light intensity I_i at x_i is determined by f(xi) ;
            if I_j > I_i then
                Move firefly i towards j in all d dimensions
            end
            Attractiveness varies with distance r via exp[-r] ;
            Evaluate new solutions and update light intensity;
        end
    end
    Rank the fireflies and find the current best
end
```

Algorithm 1: Firefly algorithm

Calculating the distance between two fireflies can be done in several ways, but the most appropriate way should be chosen in accordance to the problem. For most problems the Euclidean distance should suffice.
We can see the similarities between the firefly algorithm and the n-body simulation, where we iterate through all the bodies and calculate the forces between them. There is one major difference between the two however - in the case of the gravitational forces the same force acts on both bodies, but with opposite directions. Simulating the n-body problem is also more complex because it requires algorithms for numerical integration, but this is not required in our variation of the algorithm.

An example of using this algorithm is optimizing a function f(x). Given a firefly x_i, we set its light intensity $I_i = f(x_i)$. After running for the set number of steps, the algorithm returns the maxima that it found. It can easily be modified to find minimas. Other modifications make it possible to use this algorithm for clustering [11], image thresholding [9], clustering [11], scheduling [7] and much more.

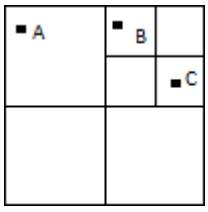
ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

2.2 Barnes-Hut Firefly

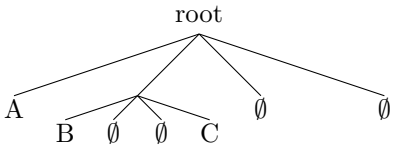
The Barnes-Hut simulation was invented by Josh Barnes and Piet Hut [1]. The main idea of this algorithm is that large clusters of bodies that are far away can be observed as one body and it will lead to almost the same result. As a contribution of this paper we will apply the same notion to the firefly algorithm. Thus there is no need to individually observe every firefly that is part of a large cluster in the distance. We can also imagine that if we had a large amount of light sources in the distance we would see them as a single source of light. It has to be noted that there are several differences between the simulation of the n-body problem and the firefly algorithm. In the former we calculate the accelaration and afterwards need to integrate to obtain the current velocity of the body. This means that we also have to concern ourselves with selecting an appropriate integration method. This is something that we are spared in the firefly algorithm as the individual fireflies simply move and we are not concerned with obeying any laws of physics accurately.

When dealing with 2-D simulation the space is divided up into squares, whereas in a 3D simulation the space is divided up into cubes. Within a 2D simulation every square can then be subdivided into 4 smaller squares of the same size. In a 3D simulation every cube can be subdivided into 8 smaller cubes of the same size. This can easily be represented within a tree structure and can be extended to an arbitrary dimensionality d with 2^d cells. This leads to problems in higher dimensions.Each cell contains an estimated center of gravity of all the bodies contained within it. The mass of this point is the sum of the masses of all the bodies contained within it. An example of this structure can be seen in Figure 1a and Figure 1b.

(a) A Barnes-Hut tree



(b) A Barnes-Hut tree

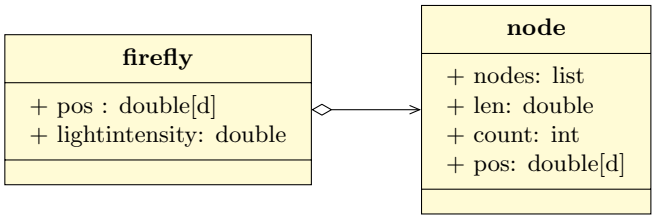


As we can see - the length of the sides of the cells decreases as the regions become denser. Since the cells are either cells, cubes (or other shapes in higher dimensions) they can be described by just two variables - an origin point and a side length. It would be easier for our insertion method if we adoped the corner with the smallest value. That would be the bottom left corner in 2D space. And

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

if we are using a left handed orientation the front bottom left corner in 3D space. This will allow us to automatically be able to determine in which of the child cells the current body belongs, regardless of the dimensionality of the problem. This is applied in Algorithm 2. The firefly structure as shown in Figure 2 represents a single firefly, which has a position pos_i in all d dimensions and a light intensity, which is dependant on its position. Our cells will be represented by the node class/structure. *nodes* represents all of its children nodes, whereas *pos* represents the left corner of the d-dimensional hypercube and *len* is the length of its sides. The innerFirefly object is either a firefly if the cell is a leaf, or the combination of all of the fireflies in its child cells. The count parameter represents the total number of nodes that the subtree starting from this node has. When creating new cells, we have to keep in mind that their side lengths have to be twice as small as those of their parents

Fig. 2: UML



Algorithm 3 is the modified Barnes-Hut algorithm and calculates the changes in the firefly's position. It differs from the original algorithm in one key aspect. The firefly only moves towards fireflies that are more attractive than itself or towards groups whose mean light intensity is larger than its own. This is because we don't want the fireflies to fly towards the nearest largest clusters, but towards clusters who have larger values in general.

Input: firefly f, node *root*
if *root is empty* **then**
  root.innerFirefly := body
end
else
  if *root has children* **then**
    $pos_i := f.pos_i \geq root.pos_i + root.len$;
    insert(root.$nodes_{pos}$) ;
  end
  if *root has no children* **then**
    $pos1_i :=$ body.$pos_i \geq$ root.pos_i + root.len ;
    $pos2_i :=$ root.body.$pos_i \geq$ root.pos_i + root.len ;
    insert(root.$nodes_{pos1}$, f) ;
    insert(root.$nodes_{pos2}$, root.innerFirefly) ;
  end
end

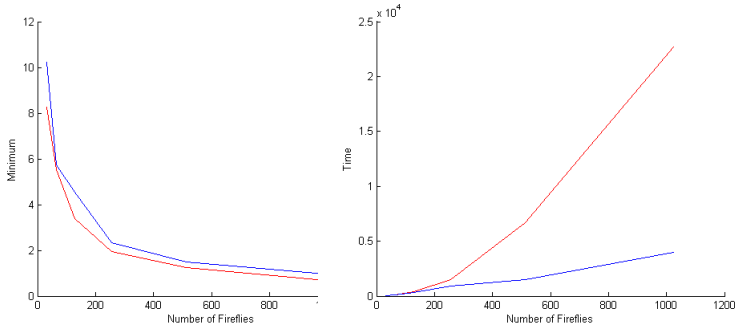
Algorithm 2: insert Function

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Input: node $root$, firefly f
if *root has no children AND root.innerFirefly.lightIntensity≯f.lightIntensity* **then**
  acc $= \beta_0 e^{-\gamma r_{ij}^2} \cdot (root.innerFirefly.pos - f.pos) + \alpha\epsilon_i$
end
if *root has children* **then**
  if $\frac{root.len}{root.innerFirefly.pos - f.pos} < 0.5$ *AND*
  $\frac{root.innerFirefly.lightIntensity}{root.count} > f.lightIntensity$ **then**
    acc $= \beta \cdot root.innerFirefly.lightIntensity \cdot e^{-\gamma r_{ij}^2} \cdot (root.innerFirefly.pos - f.pos) + \alpha\epsilon_i$
  end
  else
    acc $= \sum_i traverse(root.nodes_i, firefly)$
  end
end
return acc ;

Algorithm 3: Get Position Changes

3 Results



(a) Minimum Found (b) Computation time

Fig. 3: Test results. Standard Firefly algorithm(red), Barnes-Hut Firefly(blue)

In order to test the efficiency of the new version of the algorithm we will try to find the minimum of a function and then compare its running time and found solutions to the original algorithm. In this case the Griewank function was chosen. It is defined by the function $f(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod cos(\frac{x_i}{\sqrt{i}}) + 1$, where d is the number of dimensions and the search domain is defined by $-600 < x_i < 600$. In our case we chose to test it in 3 dimensions. The goal of this paper was to compare how the new version of the algorithm fares against the original one. This is why no optimization of the hyper-parameters was attempted and the values used were $\alpha = 1$ and $\gamma = 0.01$. The test programs were written in C++ and all tests were run on a 2.00 Ghz Intel Core Duo. Firstly we investigate for a different count of fireflies - n=32,64,128,256,512 and 1024. The tests for each individual count were repeated 100 times and the mean time and mean minimum found was then averaged. As shown in Figure 3a the Barnes-Hut Firefly underperforms slightly with a smaller number of fireflies, but it becomes almost equal with the original version with $n > 256$. But the real pay-off can be seen in the time it takes the algorithm to finish as shown in Figure 3b. It is only slightly slower for n=32, but then becomes drastically quicker as the number

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

of fireflies increases. The algorithm becomes inefficient as the dimensionality grows, due to the exponential number of children nodes and the fact that most nodes are in a relatively low level of the tree. Another practical issue is the high memory use due to the large number of children nodes. For practical purposes it should be used for problems with a dimensionality less than 8.

4 Conclusion

The proposed Barnes-hut Firefly algorithm has a much quicker running time without suffering a loss in quality. This jump in performance can allow us to handle big data much more efficiently and will allow us to tackle larger problems. Our nearly proposed version can be parallelized just as simply as the original algorithm. It is also possible to extend it by utilizing different tree structures such as k-d trees or metric trees.

References

[1] Josh Barnes & Piet Hut A hierarchical O(N log N) force-calculation algorithm, Nature 324, 446 - 449 (04 December 1986); doi:10.1038/324446a0

[2] Martin Burtscher, Keshav Pingali An Efficient CUDA Implementation of the Tree-Based Barnes Hut n-Body Algorithm, GPU Computing Gems Emerald Edition, pp. 75-92

[3] L. dos Santos Coelho, D.L de Andrade Bernert, V.C. Mariani, *A chaotic firefly algorithm applied to reliability-redundancy optimization*, 2011 IEEE Congress on Evolutionary Computation (CEC)

[4] Ananth Y. Grama, Vipin Kumar and Ahmed Sameh,Scalable Parallel Formulations of the Barnes-hut Method for n-Body simulations, Deparment of Computer Science, University of Minnesota, Minneapolis, MN 55455

[5] R. Falcon, , M. Almeida , A. Nayak, Fault identification with binary adaptive fireflies in parallel and distributed systems, 2011 IEEE Congress on Evolutionary Computation (CEC), Pages 1359 − 1366

[6] A.V. Husselmann ,K.A. Hawick, Parallel Parametric Optimisation with Firefly Algorithms on Graphical Processing Units, Proc. Int. Conf. on Genetic and Evolutionary Methods

[7] U. Hnig , A Firefly Algorithm-based Approach for Scheduling Task Graphs in Homogeneous Systems, DOI: 10.2316/P.2010.724-033

[8] M.K. Marichelvam, T. Prabaharan , X.S. Yang, A Discrete Firefly Algorithm for the Multi-Objective Hybrid Flowshop Scheduling Problems, IEEE Transactions on Evolutionary Computation, Issue 99

[9] Ming-Huwi Horng, Ren-Jean Liou Multilevel minimum cross entropy threshold selection based on the firefly algorithm, Expert Systems with Applications, Volume 38, Issue 12, Pages 1480514811

[10] B. Rampriya; K. Mahadevan ;S. Kannan,Unit commitment in deregulated power system using Lagrangian firefly algorithm, 2010 IEEE International Conference on Communication Control and Computing Technologies (ICCCCT)

[11] J. Senthilnath, S.N. Omkar, V. Mani Clustering using firefly algorithm: Performance study, Swarm and Evolutionary Computation, Volume 1, Issue 3, Pages 164 171

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