# Software Modularization Using the Modified Firefly Algorithm

**Conference Paper** · December 2014

DOI: 10.1109/MySec.2014.6986037

Ali Safari Mamaghani
Islamic Azad University, Iran.

# Software Modularization Using the Modified Firefly Algorithm

Ali Safari Mamaghani, Meysam Hajizadeh
Department of Computer Engineering
Bonab Branch, Islamic Azad University, Bonab, Iran
Ali.safari.m@gmail.com, Meysam1321@gmail.com

***Abstract*— Software modularization is a significant way to increase the maintenance of software systems. Designers depict the structure of software systems as a directed graph referred to as Module Dependency Graph in which nodes correspond to the software modules (such as files and classes) and edges relate to their relations (like function calls and inheritance relationship); These graphs can be partitioned into the meaningful clusters to be more accessible. Since Graph Partitioning is considered as a NP problem, here the application of exhaustive search algorithms are not so cost-effective that even for very small graphs they are not applicable. Therefore, many random approaches have been proposed for solving the problem. In this paper, a Firefly based method is presented to partition the software systems. However, some adjustments are needed to be made for using the FA for the problem. These modifications are as follows: change of the definition of fireflies, how to move fireflies toward better positions, and the alteration of fireflies' distance definition. Experimental results reveal that in most of the cases the proposed approach has a remarkable superiority over both the Genetic and the Learning Automaton algorithms in solving the problem.Keywords— e-learning, higher education, ICT, search for information.**

***Keywords- Software Clustering; the Firefly Algorithm; the MDG Graph; NP-Complete Problem.***

## I. INTRODUCTION

Today, software supports business, education, and social organizations of countries. As long as the procedures of these institutions modify, the software must back up them, but the alteration of software systems with elaborate functions can be rather challenging, particularly for large and dynamic systems with millions of lines of code. Nevertheless, software modularization is a useful way to simplify software maintenance and understanding. Indeed, modularization is grouping closely related nodes into clusters. To this aim, at first designers use the MDGs to represent the software system's structures. Then, the graph is inputted to clustering algorithm to create a partitioned MDG [2].

However, meaningful clustering of the MDG is not easy due to the substantially large number of feasible clustering. Besides, partial differences between two partitions can lead to considerably distinct results [3].

As Graph partitioning is a NP-Complete problem, using deterministic methods is impractical to deal with the problem; therefore, various approximate approaches have been developed. An illustration of this is the search policy based on classical Hill¬ climbing used by Brian Mitchell [3] that instantly finds fairly satisfactory near-optimal result; nevertheless, getting stuck in local optimal solutions is the potential pitfall of the algorithm. Consequently, evolutionary algorithms were used to tackle this problem. Doval [2, and 3] in the Bunch clustering tool [4] was the first person who used the Genetic algorithm for software clustering, but huge search space was the drawback of the algorithm. However, to counteract the limitations of the Bunch, another GA algorithm called the DAGC presented by Parsa [5]. furthermore, the method based on Learning Automaton offered by Safari Mamaghani could support suitable results [8].

In this paper, an approximated method based on Firefly technique is proposed to solve the software modularization problem. The algorithm has been compared with the Genetic and Learning Automaton algorithms, and the results indicate the algorithms' superiority over the other methods.

The rest of paper includes the following: Modularization Quality (MQ) measurement is introduced in section 2. Section 3 is devoted to a brief explanation about Firefly algorithm. The proposed algorithm is described in section 4, and finally sections 4 and 5 illustrate experimental results and paper summary respectively.

## II. MODULARIZATION QUALITY MEASURMENT

Now, we discuss the Modularization Quality (MQ) measurement which is the fitness function of the software clustering problem [3]. It evaluates the relations between the modules of two separate clusters which is referred to as coupling, and then the connections between the modules of the same cluster which is called cohesion. The MQ is a measurement to increase the value of cohesion, and on the contrary, to reduce coupling.

**Intra-Connectivity (cohesion):**
Cohesion [1] determines the rate of connectivity between the modules that are together in the same partition; In fact, a high value of cohesion implies decent subsystem partitioning since indicates that modules within a subsystem shares many software resources. On the contrary, the low value of cohesion states poor subsystem partitioning. Now, let us define the cohesion measurement $coh_i$ of the partition i comprises $N_i$ nodes and $E_i$ inner edges as

$$Coh_i = \frac{E_i}{N_i^2} \qquad (1)$$

In other words, this measurement is a fraction of the maximum number of inner relations which can be available for the partition i, which is $N_i^{\ 2}$. The rate of $Coh_i$ is limited between 0 and +1. Indeed, $Coh_i$ is +1 when every module in a cluster uses all modules of that cluster, and also it is 0 when none of the modules in the cluster share any software resource.

**Inter-Connectivity (coupling):**

Coupling [1] determines the rate of connectivity between two different partitions; indeed, a high value of coupling expresses a poor clustering; that is, a large number of relations between the modules of different partitions make the system more intricate; therefore, in case of any changes to a cluster, other subsystems would be affected due to the clusters relations.

Let's define the coupling measurement $Coup_{i,j}$ between two clusters of i and j comprises $N_i$ and $N_j$ nodes respectively, and $\varepsilon_{ij}$ inter-edge relations as

$$Coup_{i,j} = \begin{cases} 0 & if \quad i = j \\ \dfrac{\varepsilon_{ij}}{2N_iN_j} & i \neq j \end{cases} \qquad (2)$$

Coupling measurement is a fraction of the maximum number of inter-edge relations between two clusters of i and j ($2N_iN_j$ ). However, the boundary of $Coup_{i,j}$ is the same as cohesion's in that both are limited between 0 and +1. This value is +1 when all of the components of the cluster i are connected to the modules of the cluster j and vice versa, but the rate is 0 when there is not any relationship between two modules.

Now, we can define MQ measurement for a MDG graph clustered into m partitions as

$$MQ = \begin{cases} \dfrac{1}{m}\sum_{i=1}^{m} Coh_i - \dfrac{1}{\frac{m(m-1)}{2}}\sum_{i,j=1}^{m} Coup_{i,j} & if \quad m > 1 \\ Coh_1 \quad if \quad m = 1 \end{cases} \qquad (3)$$

According to the relation (3), the MQ measurement illustrates the tradeoff between cohesion and coupling and this value ranges from -1 (the worst case) and +1 (the best case).

## III. AN INTRODUCTION TO FIREFLY ALGORITHM

The Firefly is a newly developed algorithm which has been influenced by the lifestyle of fireflies, and is presented by Xin-She-Yang in the year of 2008 [6, and 7]. The algorithm is a great example of swarm intelligence in which the cooperation (and sometimes competition) among simple and unintelligent members leads to high level of intelligence. The FA algorithm also takes advantage of the features like quick convergence, not depending on initial values, flexibility, and high fault tolerance [9].

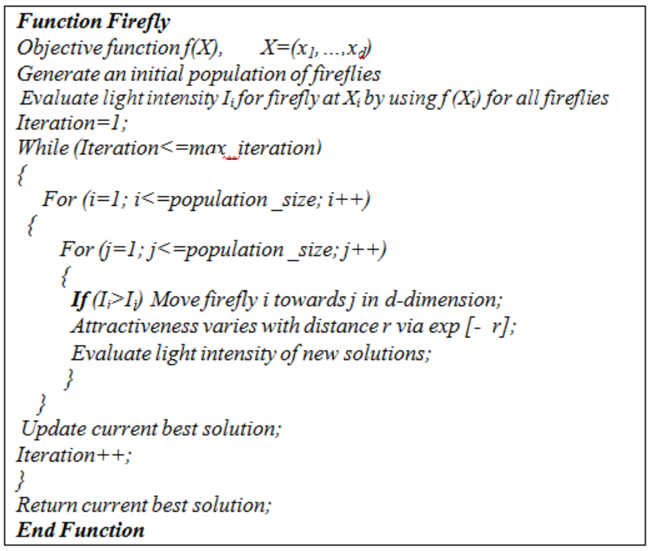
In fact, in the algorithm, each firefly represents a potential solution which has own intensity. This intensity equals the corresponding value of the fitness function. Fireflies try to move toward better (brighter) peers and change their positions in order to acquire sub-optimal and at times even optimal solutions. Three assumptions have been considered to simplify the process of the algorithm. These assumptions are as follows.

1) All fireflies have the same gender, so the pairs are attracted to each other regardless of their genders.

2) A firefly's attractiveness is directly proportional to its brightness which reduces when its distance from another firefly rises, and also if the attractiveness of both fireflies are the same, the movement's direction will be random.

3) The brightness of a particular firefly is evaluated based on the value of fitness function.

Now, the pseudo code of the FA algorithm is shown in Fig. 1.



```
Function Firefly
Objective function f(X),      X=(x1, ...,xd)
Generate an initial population of fireflies
 Evaluate light intensity Ii for firefly at Xi by using f (Xi) for all fireflies
Iteration=1;
While (Iteration<=max_iteration)
{
   For (i=1; i<=population_size; i++)
  {
      For (j=1; j<=population_size; j++)
      {
        If (Ij>Ii) Move firefly i towards j in d-dimension;
        Attractiveness varies with distance r via exp [-  r];
        Evaluate light intensity of new solutions;
      }
   }
 Update current best solution;
Iteration++;
}
Return current best solution;
End Function
```

Fig. 1. The pseudo code of the standard Firefly algorithm.

In the mentioned algorithm, the distance between two fireflies i and j at $x_i$ ,and $x_j$ respectively is defined as

$$r_{ij} = \left\| x_i - x_j \right\| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \qquad (4)$$

Where $x_{i,k}$ is the kth component of the spatial coordinate $x^i$ of ith firefly and d states the number of dimensions. However, for 2-d dimension, $r_{ij}$ is defined as

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \qquad (5)$$

The movement of less bright firefly at $x_i$ toward the brighter one at $x_j$ is characterized as

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha(Rand - \frac{1}{2}) \qquad (6)$$

Where the second and third terms indicate attractiveness and the random movement of firefly i respectively, and The function Rand generates random number ranges from 0 to 1, and mostly $\alpha \in (0,1)$. Also, $\gamma$ is the light absorption coefficient, which is assumed 1 in our implementation.

## IV. THE APPLICATION OF FA ALGORITHM TO SOLVE THE SOFTWARE CLUSTERING PROBLEM

The FA algorithm mainly has been designed for solving continuous optimization problems. However, it also can be used for discrete problems like the software clustering.

Therefore, to adapt the FA to solve the problem, it is necessary to define fundamental concepts such as particle (firefly), the distance between the particles and the movement operator associated with the problem.

**Representation of the particles:** To represent a firefly, a permeation based encoding is used. In fact, each particle is a permutation of the MDG nodes in which i<sup>th</sup> cell presents i<sup>th</sup> node of the graph, but in spite of the Bunch Genetic method [2], its content does not involve the number of cluster, but consists of the number of a particular node of a graph. This representation is decoded to its associated clustering by using the algorithm shown in Fig. 2.

```
Function Decoding
  A permutation of n different nodes of the graph is held in array F.
  For (i=1;i<=n;i++)
   {
     If (F[i]>=i )
       Form a new partition and assign node i to it.
     Else
       Assign i to the same partition as node F[i]'s partition.
   }
End Decoding
```

Fig. 2. The pseudo code of decoding algorithm

Fig. 3 depicts an example of this encoding. It is clearly seen that index 2 of the array is greater than the content of that cell, so node 2 is placed in new partition (cluster 2); however, node 1 has already been placed in partition 1.

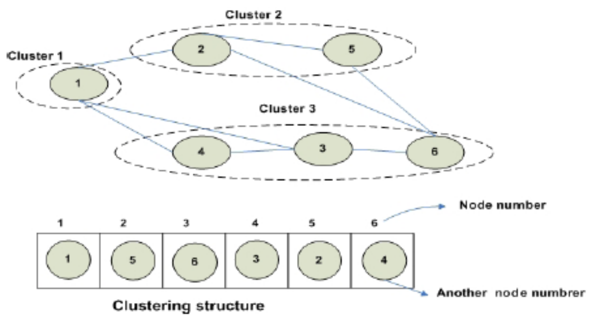


Fig. 3 . an example of permutation based encoding

**Distance:** it can be defined as the hamming distance between two fireflies; in other words, the number of different cells with the same index. Let's examine this example. Consider permutations F1 and F2 as

F1= [1 5 6 3 2 4] , and F2= [1 2 5 6 3 4].

The Hamming Distance (F1, F2) is 4 (the first and last cells are the only cells with the same value).

**Movment:** To move firefly i to j, at first a random number , $R_i$, is generated as $R_i$= Random(1,$d_{i,j}$) where $d_{i,j}$ is the distance between two fireflies. Then, $R_i$ times swap operation is performed between different pairs.

## V. Experimental Results

This section indicates the difference between the proposed algorithm and other approaches in terms of the obtained value of fitness function MQ. Some random MDGs were employed to compare the strength of the algorithms in order to find solutions for clustering the MDGs. Also due to the random nature of the used approximated algorithms, all tests were done five times, and for each sample the average value of the MQ was recorded.

The used parameters include the following: population size= 20 and maximum iteration =100. The results are shown in Table I. It's clearly seen that in most of the cases the results of the new algorithm are more useful than others; in other words, this algorithm acquires greater MQ values in comparison with other approaches; therefore, its clustering will be more meaningful. The following additional tests done on the algorithm are shown in Fig. 4. The test has been performed on a random sample with 15 nodes and number of iterations that increased from 15 to 500. The Fig. 4 demonstrates the changes.

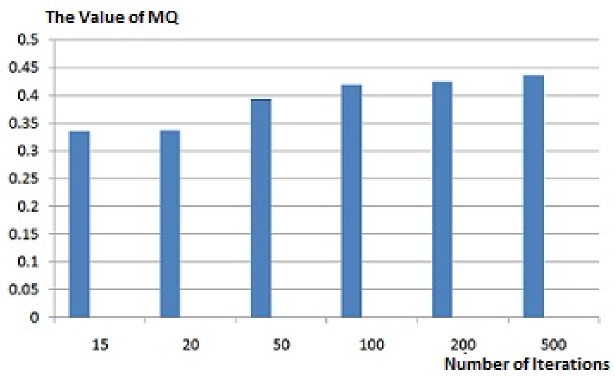


Fig. 4 . the effect of the algoritm's number of iterations on the quality of solutions

Two important points are seen in Figure 4. Firstly, while number of iterations goes up, the quality of MQ will grow; in other words, the algorithm results improve gradually with the rise of iteration. Secondly, the results make changes considerably from the iteration of 100, and they somewhat converge; therefore, to balance the quality of results and time spent by the algorithm, the iteration number is assumed 100.

## VI. Conclusion

Graphs are beneficial abstract model which have varied applications in computer science from computer networking and database design to software engineering and etc. One major application of graphs in software engineering is to depict software system as a MDG graph. However, to improve the maintenance of software systems, a crucial way is clustering them into meaningful partitions in order to prevent propagating the modifications overall system, but the software clustering is a NP-complete problem and we have to use meta heuristics to find solutions. In this paper, we have used a newly developed FA Meta heuristic, and the empirical results states that the modified proposed FA has more capability to find near-optimal and even optimal solutions compared with two other well-known methods: Genetic and learning automaton.

## REFERENCES

[1] B. S. Mitchell, A Heuristic Search Approach to Solving the Software Clustering Problem, Ph.D Thesis, Drexel University, Philadelphia, 2002.

[2] D. Doval and S. Mancoridis, "Automatic Clustering of Software Systems using a Genetic Algorithm", In Proc. Of the IEEE Int. Conf. on Software Tools and Engineering Practice(STEP'99), 1999.

[3] S. Mancoridis, B. S. Mitchell, C. Rorres, Y. Chen and E. R. Gansner, "Using Automatic Clustering to Produce High-Level System Organizations of Source Code", In Proc. Of the Int. Workshop On Programm Understanding, 1998.

[4] S. Mancordis, B. S. Mitchell, Y. Chen and E. R. Gansner, "Bunch: A clustering tool for the recovery and maintenance of software system Structures", in Proc. Of Int. Conf. of Software Maintenance, pp. 50-59, 1999.

[5] S. Parsa, and O. Bushehrian, "A New Encoding Scheme and a Framwork to investigate Genetic Clustering Algorithms", Journal of Research and Practice in Information Technology, Vol. 37, No. 1, pp. 127-143, 2005.

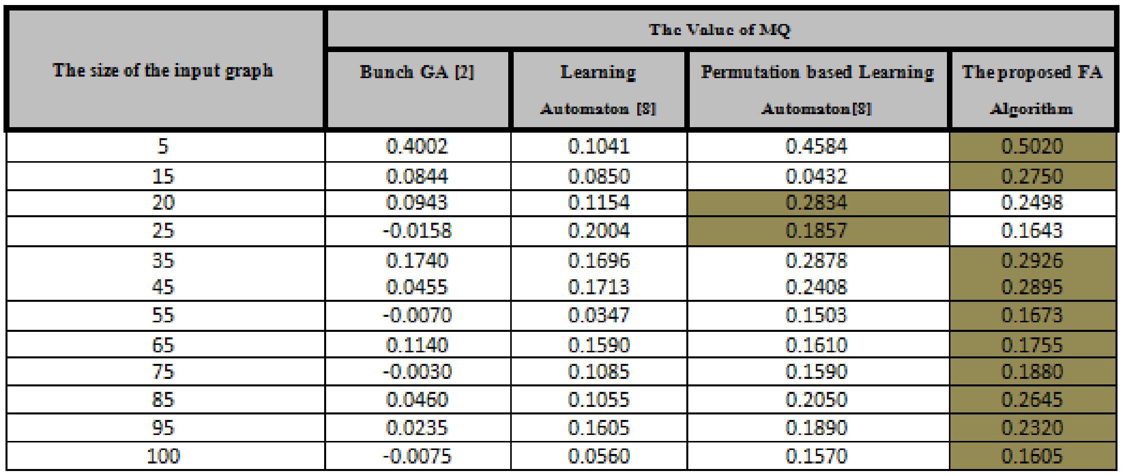
[6] Xin-She Yang. Nature-Inspired Metaheuristic Algorithms. Luniver Press, 2008.

[7] Xin-She Yang. Firefly algorithm for multimodal optimization. Stochastic Algorithms: Foundation and Application, 5th, 2009.

[8] A. Safari Mamaghani, and M. R. Meybodi, , "Clustering of Software Systems using New Hybrid Algorithms", Proceedings of the IEEE Ninth International Conference on Computer and Information Technology (CIT 2009), Xiamen, China, pp. 20-25, October 11-14, 2009.

[9] T. Hassanzadeh, , M. R. Meybodi, and F. Mahmoudi, , "An Improved Algorithm for Optimization in Static Environment", Proceedings of the 5th Iran Data Mining Conference (IDMC'11), Amirkabir University of

TABLE I. THE RESULTS OBTAINED BY THE ALGORITHMS ON THE RANDOM GARPHS



| The size of the input graph | The Value of MQ | | | |
|---|---|---|---|---|
| | Bunch GA [2] | Learning Automaton [8] | Permutation based Learning Automaton[8] | The proposed FA Algorithm |
| 5 | 0.4002 | 0.1041 | 0.4584 | 0.5020 |
| 15 | 0.0844 | 0.0850 | 0.0432 | 0.2750 |
| 20 | 0.0943 | 0.1154 | 0.2834 | 0.2498 |
| 25 | -0.0158 | 0.2004 | 0.1857 | 0.1643 |
| 35 | 0.1740 | 0.1696 | 0.2878 | 0.2926 |
| 45 | 0.0455 | 0.1713 | 0.2408 | 0.2895 |
| 55 | -0.0070 | 0.0347 | 0.1503 | 0.1673 |
| 65 | 0.1140 | 0.1590 | 0.1610 | 0.1755 |
| 75 | -0.0030 | 0.1085 | 0.1590 | 0.1880 |
| 85 | 0.0460 | 0.1055 | 0.2050 | 0.2645 |
| 95 | 0.0235 | 0.1605 | 0.1890 | 0.2320 |
| 100 | -0.0075 | 0.0560 | 0.1570 | 0.1605 |