ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 10(23), DOI: 10.17485/ijst/2017/v10i23/112150, June 2017* ISSN (Online) : 0974-5645

Adaptive Optimized Open Shortest Path First Algorithm using Enhanced Moth Flame Algorithm

Pertik Garg[1] and Ashu Gupta[2]

[1]Department of Computer Science and Engineering, I. K. Gujral Punjab Technical University, Jalandhar – 144603, Punjab, India; waytogarg@gmail.com
[2]Department of Computer Applications, Apeejay Institute of Management Technical Campus, Jalandhar – 144007, Punjab, India; guptaashu1@rediffmail.com

Abstract

Objective: To builds up a versatile algorithm in view of Open Shortest Path First (OSPF) algorithm by using improved moth flame optimization. The designed algorithm gives the upgraded performance even with dynamic topology. **Methods/Statistical Analysis:** The optimized OSPF algorithm utilizes the directing conduct of moth to exchange the information from source to destination. The optimized OSPF algorithm performs well however the execution debases with frequently change in topology. This work upgrades the optimized OSPF by changing the path of moth based on the delay parameter. Additionally, the neighbor determination criteria have been changed to diminish the delay. **Findings:** The analysis of the adaptive optimized OSPF algorithm for the dynamic network has been finished by utilizing delay and energy consumption. The performance of the outlined algorithm is done with existing OSPF, modified OSPF and the optimized OSPF algorithm. The reduced delay and energy consumption as contrasted with optimized OSPF and OSPF based algorithms demonstrates the significance of algorithm. **Application/Improvements:** The adaptive optimized OSPF can be used for routing for the dynamic networks. The algorithm can be utilized where dependable information conveyance is required with least deferral.

Keywords: Adaptive, Delay, Moth-Flame, Optimized OSPF, Routing

1. Introduction

Routing is the way toward choosing paths in a network along which to send arrange movement. Most routing is performed by routers which work at layer 3 of the OSI model. These routers utilize either dynamic directing conventions or static courses to settle on choices on where to send arrange movement to achieve its goal[1]. Static routes are courses arranged by an overseer which advise a router how to deal with a system that is not specifically associated. Dynamic routing conventions permit routers to promote the systems with which they know how to convey. This data is gotten by different routers on the system and these different routers use this information to populate their routing table with data to interface with each system inside the same independent framework. This procedure of engendering topology data all through a system is called convergence. At the point when an arrangement of routers has met, the system topology data must not negate between any of the routers, and it must mirror the genuine condition of the system[1,2].

Dynamic routing conventions[3] can be part into three unmistakable sorts: distance vector algorithms, link-state algorithms, and path vector protocol. Distance vector algorithm relegates a cost to each connection between nodes in a system. They will then pick courses in light of the most minimal cost way by processing the total of the cost of each connection the system activity must navigate to achieve its goal. Link-state algorithms determine that every hub surges the whole system with data about what different nodes it can associate with. This permits every node to freely gather this data into a system guide that can

* *Author for correspondence*

then be utilized to decide slightest cost ways by utilizing a briefest way calculation. Path vector convention is utilized for routing between self-governing frameworks and is fundamentally the same as distance vector routing. Path vector convention utilizes a node in each self-ruling framework, called a speaker hub, to publicize its directing table to other neighboring speaker hubs[4].

Open Shortest Path First (OSPF)[1,5] is a famous interior gateway routing convention. Such conventions give routing functionality inside a domain, which is for the most part, in spite of the fact that not really, contained inside a self-ruling framework (AS)[6]. OSPF has a place with the class of link state routing conventions that for the most part require every router in the system to think about the entire system topology[7]. The working of OSPF begins with Routers to communicate interface state information. Then each router stores the connection state data in memory utilizing a structure named the topology table. The router forms all information in the topology table and makes utilization of the Dijkstra calculation to decide all routes to all networks, and also the minimum cost routes. All this data is put away in the SFP tree, distinguishing favored and optional courses[7,8]. The steering data is proliferated to the routing table.

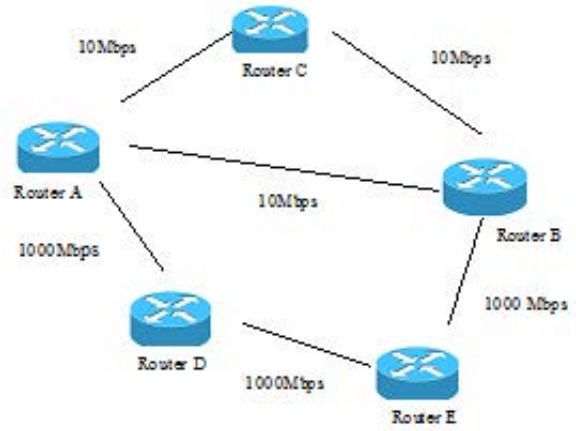

Figure 1. Anonymous network.

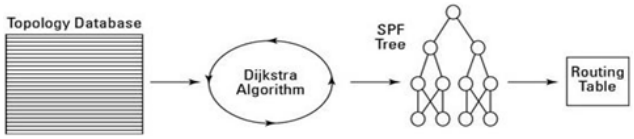

Figure 2. OSPF working.

The OSPF finishes its working in three stages(for any network example network shown in Figure 1. first is to procure the neighbor data by sending the hello bundle as appeared in Figure 2.Then in the second stage, routers trade their database as likewise appeared in Figure 2. At that point in the last stage Dijkstra's Shortest Path algorithm[9] is utilized for the route computation. At that point the information is exchanged utilizing the chosen path[10-12].

Diverse authors have adjusted OSPF convention to upgrade the QoS, few of them are introduced here. The author of[6] proposes a dynamic calculation to deal with the connection failure problem[6]. The author adjusted the OSPF for ongoing application by utilizing multipath data. The accessible substitute way utilized by the creator if there should be an occurrence of connection disappointment, diminished re-merging time. The author demonstrated the better execution of the convention when contrasted with existing convention utilized as a part of routers. The author in 2009 presents the appropriated OSPF with decreased convergence time without anyone else's input adaption with connection failure[13]. In[14] proposes a versatile connection foundation plan to enhance the execution if there should be an occurrence of portability. Be that as it may, the calculation works productively just if there should arise an occurrence of low versatility. In[15] uses the computational insight system to improve the planning of routing table estimation to lessen the convergence time IQBAL. In[16] investigations the OSPF over the voice and the video activity. This paper adjusts the Optimized OSPF calculation by utilizing enhanced MOTH-FLAME improvement calculation examined in next section.

2. Enhanced Moth-Flame Optimization (MFO) Algorithm

The existing Moth-flame optimization algorithm is an optimization in light of the routing system of the moth which is a sort of butterfly towards the moon utilizing the moon light. The moth moves towards the goal in a straight line when the goal is far away as shown in the Figure 3.

While the moth play out the winding movement when goal (flame) is nearer. Be that as it may, the calculation actualize the marginally changes conduct to keep away from the neighborhood minima. The calculation utilizes the explorative and additionally exploitation pursuit to

get an enhanced solution[17]. Here, we design an enhanced moth flame algorithm by updating the position of moth with respect to the moth moving towards moon with minimum cost. The position of the moth is updated by using the eq. (1).

Figure 3. Moth-flame.

$$MO_i' = MO_i + rand * [(FO]_i - MO_i * improvement) \quad (1)$$

Here, MO_i is the position of the ith moth to be updated with respect to the best moth position i.e. FO_i. The rand is random variable ranging from 0-1. The improvement factor is considered in-between 1-2 on experimental basis. The updated value of the moth will lead to the generation of the corresponding fitness function generationFO_i'. The updated value of the MO_i can be obtained using eq. (2).

$$if FO_i' < FO_i \{MO_i = MO_i'\} \quad (2)$$

The equation (2) demonstrates that the change in position of moth is accepted only if the updated leads to better fitness cost. The algorithm of the enhanced moth flame algorithm is as follow:

Algorithm

The MFO is a populace based calculation, here N moths are moving in D measurement whose position grid is given in eq. (3) shown as:

$$MO = \begin{bmatrix} MO_{1,1} & \cdots & MO_{1,D} \\ \vdots & \ddots & \vdots \\ MO_{N,1} & \cdots & MO_{N,D} \end{bmatrix} \quad (3)$$

The fitness value of each moth can be evaluated by using the fitness function and stored in a matrix as shown in eq. (4).

$$O_{MO} = \begin{bmatrix} O_{MO1} \\ \vdots \\ O_{MON} \end{bmatrix} \quad (4)$$

The best position of the moths in each dimension is stored in matrix given as flames and corresponding fitness value is generated shown in eq. (5) and (6) respectively.

$$FO = \begin{bmatrix} FO_{1,1} & \cdots & FO_{1,D} \\ \vdots & \ddots & \vdots \\ FO_{N,1} & \cdots & FO_{N,D} \end{bmatrix} \quad (5)$$

$$O_{FO} = \begin{bmatrix} O_{FO1} \\ \vdots \\ O_{FON} \end{bmatrix} \quad (6)$$

The moth moves around the flame in a spiral motion given by the eq. (7).

$$S_{MO_i,FO_j} = |FO_j - MO_i| * e^{bt} * cos(2\pi t) + FO_j \quad (7)$$

Here, S_{MO_i,FO_j} denotes the spiral motion, b is the constant to shape the spiral motion, t is the random variable to select the behavior of moth with respect to flame i.e. if t=-1 then the moth select the position closest to the flame and if t=1 then the farthest position is selected. The positions of the moth are updated using the eq. (7) and the best moth position updates the flames position. Moreover, the number of flames is decreased in each an iteration using the eq. (8).

$$NOF = ceil\left(N - c_{iter} * \frac{N-1}{T_{iter}}\right) \quad (8)$$

Here, T_{iter}, C_{iter} is the total number of iteration and current iteration number respectively. The overall process of the MFO algorithm can be easily understood by the following algorithm:

- Initialize the Moths:

  For i=1:N

  For j=1:D

  $MO_{i,j}$= random position between given bounds

  End

  O_{MOi} = Fitness value (MO_i)

  End
- C_{iter}=1
- While $C_{iter} \leq T_{iter}$

  $MO_i = |FO_j - MO_i| * e^{bt} * cos(2\pi t) + FO_j$

  O_{MOi} = Fitness value (MO_i)

if $C_{iter} == 1$

FO_i=QuickSort (MO_i)

O_{FOi}= QuickSort (O_{MOi})

else

FO=QuickSort ($MO_{C_{iter-1}}$, $MO_{C_{iter}}$)

O_{FO}= QuickSort ($MO_{C_{iter-1}}$, $MO_{C_{iter}}$)

End if

Update b and t

$$MO_i = |FO_j - MO_i| * e^{bt} * cos(2\pi t) + FO_j$$

$$MO_i' = MO_i + rand * [(FO]_i - MO_i * improvement)$$

FO'=QuickSort(MO')

if $C_{iter} == 1$

$if FO_i' < FO_i \{MO_i = MO_i'\}$

$end\ if$

Update number of flames

$$N = ceil\left(N - c_{iter} * \frac{N-1}{T_{iter}}\right)$$

$$C_{iter} = C_{iter} + 1$$

End while

End

3. Proposed Approach

This section describes the proposed i.e. adaptive optimized Moth Flame optimization algorithm by using the enhanced moth flame algorithm discussed in previous section. This algorithm modifies the optimized Moth flame algorithm[3] by selecting farthest node in range of transmission of the node. This process will reduce the number of hops to transfer the data from source to destination. The reduction in number of hop must improve the performance of algorithm. Additionally, the algorithm depicts the behavior of enhanced moth flame algorithm instead of moth flame algorithm. The corresponding algorithm is described in next subsection.

3.1 Adaptive Optimized OSPF based on Enhanced Moth Flame Algorithm

In this work the OSPF algorithm is optimized by using the enhanced moth flame algorithm. The moth are supplanted by the nodes i.e. there is one moth for every node and the position of the node is considered as the position of the moth. As the nodes are set in real world condition so the quantity of measurement D=3. The position matrix representing the moth based on nodes is given by eq. (9).

$$M_N = \begin{bmatrix} M_{N_{1,1}} & M_{N_{1,2}} & M_{N_{1,3}} \\ M_{N_{2,1}} & M_{N_{2,2}} & M_{N_{2,3}} \\ \vdots & \vdots & \vdots \\ M_{N_{n,1}} & M_{N_{n,2}} & M_{N_{n,3}} \end{bmatrix} \quad (9)$$

Here, n is the number of nodes in the network. The distance of current node from the destination is used to generate the corresponding fitness value. The fitness value is generated by using the eq. (10).

$$M_{N_1} = \sqrt{\left(M_{N_{d,1}} - M_{N_{1,1}}\right)^2 + \left(M_{N_{d,2}} - M_{N_{1,2}}\right)^2 + \left(M_{N_{d,3}} - M_{N_{1,3}}\right)^2} \quad (10)$$

Corresponding fitness value matrix is given by the eq. (11):

$$O_{M_N} = \begin{bmatrix} O_{M_{N_1}} \\ \vdots \\ O_{M_{N_n}} \end{bmatrix} \quad (11)$$

The flame matrix gives the position matrix of the best neighbor node to be selected in the route by any node given by eq. (10) and corresponding fitness value matrix generated using the eq. (10) is given in the eq. (13).

$$F_N = \begin{bmatrix} F_{N_{1,1}} & F_{N_{1,2}} & F_{N_{1,3}} \\ F_{N_{2,1}} & F_{N_{2,2}} & F_{N_{2,3}} \\ \vdots & \vdots & \vdots \\ F_{N_{n,1}} & F_{N_{n,2}} & F_{N_{n,3}} \end{bmatrix} \quad (12)$$

$$O_{F_N} = \begin{bmatrix} O_{F_{N_1}} \\ \vdots \\ O_{F_{N_n}} \end{bmatrix} \quad (13)$$

The position of the node to be selected next in the route is given by the eq. (14).

$$M_{M_i F_j} = |F_j - M_i| * e^{bt} * cos(2\pi t) + F_j \quad (14)$$

The new position obtained using the equation (14) need not be the exact position of the node, so the node nearest to the updated position is to be selected, so the eq. (14) is applied to generate the matrix in eq. (15).

$$D_{M_N} = \begin{bmatrix} D_{M_{N_1}} \\ \vdots \\ D_{M_{N_n}} \end{bmatrix} \quad (15)$$

The position of each node is further updated by using eq. (16).

$$M'_{M_i,F_j} = M_{M_i,F_j} + rand * (\lceil F \rceil_i - M_i * improvement) \quad (16)$$

Where the terms has the same meaning as described in previous section. The updated position is accepted only by use of eq. (17).

$$if M'_{M_i,F_j} < M_{M_i,F_j} \{M_i = M'_i\} \quad (17)$$

Perform the quick sort on matrix D_{M_N} to generate the updated M_N matrix. Now, the algorithm select the M_{N_1} or F_{N_1} based on the corresponding fitness value. It means the nodes with better fitness value (lower value is better) are getting selected. If the route failed due to any attack then the alternative node get selected. The process gets repeated until the destination gets reached. The complete process can also be understood by the following algorithm:

3.2 Adaptive Optimized OSPF Algorithm (Source, Destination)

- Initialize the Network:

  For i=1:N

  For j=1:3

  $M_{Ni,j}$= random position between given network area

  End

  Calculate fitness value

  $$M_{N_i} = \sqrt{\left(M_{N_{d,i_1}} - M_{N_{i,1}}\right)^2 + \left(M_{N_{d,2}} - M_{N_{i,2}}\right)^2 + \left(M_{N_{d,3}} - M_{N_{i,3}}\right)^2}$$

  End

- F_N=QuickSort (M_N)
- O_{FN}= QuickSort (O_{MN})
- $Current_{node}$ =Source
- While $Current_{node} \neq Destination$
  - $M_{M_i,F_j} = |F_j - M_i| * e^{bt} * cos(2\pi t) + F_j$
  - $M'_{M_i,F_j} = M_{M_i,F_j} + rand * (\lceil F \rceil_i - M_i * improvement)$
  - $if M'_{M_i,F_j} < M_{M_i,F_j} \{M_i = M'_i\}$
  - Generate the Distance matrix by using eq. (12)
  - Select Next Node in The route
  - F_N=QuickSort ($M_{Current_{node}}$, $F_{Current_{node}}$)
  - O_{FN}= QuickSort ($M_{Current_{node}}$, $F_{Current_{node}}$)
  - End if
  - Update b and t
  - Update $Current_{node}$

  End while
- Exit

The above algorithm is used to generate an optimized path. The implementation and results of the algorithm are discussed in next section.

4. Performance Evaluation

The performance evaluation section includes two parts, the first part discussed about the simulation parameters and the second one revealed the experimental results.

4.1 Simulation Parameters

Firstly, we assumed network range of 100*100 measurements and which is further partitioned into 9 squares. Each square contains approach measure of nodes. The position of the source and goal are settled in the block 1 and block 9 individually and every node is furnished with some underlying vitality esteem. The trading of data would be in the middle of source and goal and it is mandatory that the route goes into each shut and select a node from each piece. Toward the start there is an era of beginning populace, which gives the arrangement of routes arbitrarily between sources to goal. The vitality of each course is accumulated i.e. the vitality of every node in each piece is summed up and the wellness capacity

is assessed on the premise of most astounding vitality esteem recovered by any course. The best course is along these lines chose. The reenactment parameters are given beneath in Table 1.

Table 1. Simulation parameters

Parameter	Values
Area (user defined)	100*100 m^2
Number of blocks	9
Number of Routers per section	10
Energy	0.5 Joule
Population Size	10
Number of iterations	100

Figure 4 to 6 compares the proposed adaptive optimized OSPF algorithm with traditional OSPF algorithm, Modified and Optimized OSPF algorithm in terms of Delay, Energy Consumption and throughput.

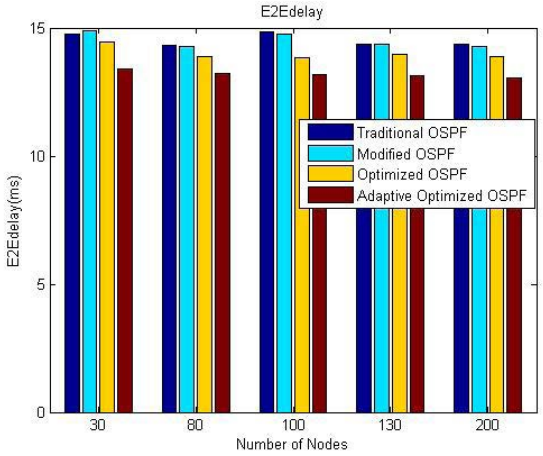

Figure 4. Comparison of delay for traditional, modified, optimized OSPF and adaptive optimized OSPF.

Figure 4 compares the delay of the traditional, modified, optimized OSPF and adaptive optimized OSPF. The performance shows that the delay gets decreased in the adaptive optimized OSPF due to inclusion of the dynamic optimization process.

Figure 5 and 6 compares the energy consumption and throughput of the traditional, modified, optimized OSPF and adaptive optimized OSPF respectively. The performance shows that the energy consumption gets decreased while the throughput is enhanced in the adaptive optimized OSPF as more reliable path is selected for the backup.

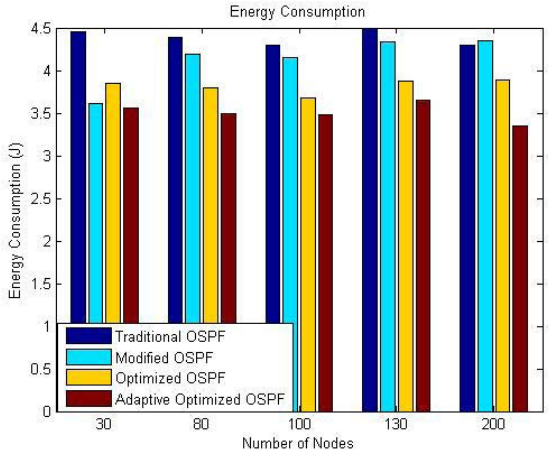

Figure 5. Comparison of energy consumption for traditional, modified, optimized OSPF and adaptive optimized OSPF.

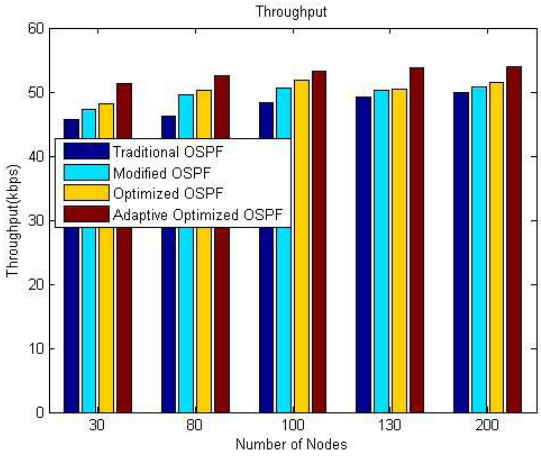

Figure 6. Comparison of throughput for traditional, modified, optimized OSPF and adaptive optimized OSPF.

5. Conclusion

This paper designs an adaptive optimized OSPF algorithm by modifying the optimized OSPF algorithm. The optimized OSPF algorithm is based on the moth-flame algorithm. This paper firstly enhances the moth flame algorithm then uses the same to extend the performance of the optimized OSPF algorithm. The adaptive optimized OSPF algorithm is capable to handle the dynamic topology. This algorithm transfer the data with less number of hops as compared to optimized OSPF and adapts to the topology dynamically. The simulation

results prove the significance of the work. In future this algorithm can be in other area of application.

6. References

1. Mandekar AV, Chandramouli K. Centralization of Network using Open Flow Protocol, Indian Journal of Science and Technology. 2015 Jan; 8(S2):165-70. Crossref.
2. Barreto F, Wille EC, Junior LN. Fast Recovery Paths: Reducing Packet Loss Rates during IP Routing Convergence. In: Telecommunications, Fourth Advanced International Conference; 2008 Jun. p. 101-10.
3. Garg P, Gupta A. Optimized Open Shortest Path First Algorithm Based on Moth Flame Optimization, Indian Journal of Science and Technology. 2017 Jan; 9(48):1-9. Crossref.
4. Ravikumar CV, Srikanth YM, Sairam P, Sundeep M, Bagadi KP, Annepu V. Performance Analysis of HSRP in Provisioning Layer-3 Gateway Redundancy for Corporate Networks, Indian Journal of Science and Technology. 2016 May; 9(20):1-5. Crossref.
5. Goyal M, Soperi M, Hosseini H, Trivedi KS, Shaikh A, Choudhury G. Analyzing the Hold Time Schemes to Limit the Routing Table Calculations in OSPF Protocol. In: International Conference Advanced Information Networking and Applications. 2009 May, p. 74-81. Crossref.
6. Jiang X, Xu M, Li Q, Pan L. Improving IGP Convergence through Distributed OSPF in Scalable Router. In: High Performance Computing and Communications, 11th IEEE International Conference. 2009 Jun, p. 438-43. Crossref.
7. Krishnan YN, Shobha G. Performance Analysis of OSPF and EIGRP Routing Protocols for Greener Internetworking. In: Green High Performance Computing (ICGHPC), IEEE International Conference. 2013 Mar, p. 1-4. Crossref.
8. Bahattab AA. A Comparative Analysis of TCP/IP and ROACM Protocols-A Simulation Study, Indian Journal of Science and Technology. 2016 Jul; 9(28):1-9. Crossref.
9. Choi Y, Kang D, Bahk S. Improvement of AODV Routing Protocol through Dynamic Route Change using Hello Message. In: International Conference on Information and Communication Technology Convergence (ICTC). 2014 Oct, p. 117-21. Crossref.
10. Xiao X, Ni L. Reducing Routing Table Computation Cost in OSPF. In: Internet Workshop. 1999, p. 119-25. PMid:10320045.
11. Lan L, Li L, Jianya C. A Multipath Routing Algorithm based on OSPF Routing Protocol. In: Semantics, Knowledge and Grids (SKG), Eighth International Conference. 2012 Oct, p. 269-72. Crossref.
12. Kari H, Moghadam HY, Monsefi R. Design and Simulation of OSPF Routing Protocol Fuzzy Model by Two Costraint: Bandwidth and Queue Size Of Router. 2010 Dec, p. 249-54.
13. Eramo V, Listanti M, Cianfrani A. Design and Evaluation of a New Multi-Path Incremental Routing Algorithm on Software Routers, IEEE Transactions on Network and Service Management. 2008 Dec; 5(4):188-203. Crossref.
14. Kang J, Mariusz AF, Samtani S. ALE: Adaptive Link Establishment in OSPF Wireless Ad-Hoc Networks. In: Military Communications Conference, 2010-Milcom. 2010 Oct, p. 983-88.
15. Haider M, Soperi MZ, Bakar KA. Comparison of Intelligent Schemes for Scheduling OSPF Routing Table Calculation. In: Hybrid Intelligent Systems (HIS), 11th International Conference. 2011 Dec, p. 145-50. Crossref.
16. Iqbal A, Khan SL. Performance Evaluation of Real Time Applications for RIP, OSPF and EIGRP for Flapping Links using OPNET Modeler, International Journal of Computer Networks and Communications Security (IJCNCS). 2015 Jan; 3(1):16-26.
17. Mirjalili S. Moth-Flame Optimization Algorithm: A Novel Nature-Inspired Heuristic Paradigm, Knowledge-Based Systems. 2015 Nov; 89:228-49. Crossref.