Bulgarian Chemical Communications, Volume 52, Issue A (pp. 184-188) 2020   DOI: 10.34049/bcc.52.A.255

Maximum-flow problem in networking

F. I. Sapundzhi*, M. S. Popstoilov

South-West University "Neofit Rilski", 2700 Blagoevgrad, Bulgaria

Received August 01, 2019; Accepted October 29, 2019

The communication network is made up of nodes and links. It carries traffic where traffic flows from a start node (source) to an end node (sink). In general, a communication network can be represented as: (1) a directed network - the flow is directional from one node to another and the links are considered as directional links; or (2) an undirected network - there is no differentiation between the directions of flow. The aim of the maximal flow problem is to find the maximum flow that can be sent from a specified start node to a specified end node through the edges of the network. The maximum flow problem asks for the largest amount of flow that can be transported from one vertex to another. Network flow modelling is used for traffic engineering of networks. It can help in determining routing decisions.

Keywords: maximum flow problem, routing, traffic engineering, communication network.

INTRODUCTION

The graphs theory plays a significant role in many fields of applications in real world such as travelling, transportation, various computer applications, traffic control, communications, and so on [1-4]. We can consider the graph as a mathematical representation of a network that describes the relationship between a finite number of points connected by lines. The maximum flow problem and its dual, the minimum cut problem, are classical combinatorial optimization problems with many applications in science and engineering.

The algorithm for finding the maximum flow models several important problems as the traffic in transportation networks and the data packets in a communication network. A widely used algorithm for finding the maximum flow problem based on the augmenting path method was developed by Ford and Fulkerson [5,6]. The Ford–Fulkerson algorithm (FFA) is a greedy algorithm that computes the maximum flow in a flow network. Later Edmonds and Karp introduced the shortest augmenting path method [7]. To define path lengths, their method uses the unit length function that sets the length of each edge to one. Other popular maximum flow algorithms are Dinic's blocking flow-based algorithm [8], and the Goldberg-Tarjan push-relabel algorithm [9,10]. A special case of the maximum flow problem involving all arcs having unit capacities was reported by Even-Tarjan [11] and Karzanov [12].

The main purposes of this study are: (a) to determine and identify the concepts of the maximum flow problem in networking; (b) to determine the representation of graphs in the computer in order to solve this path problem, as well as to explore and understand he different basic terms of graphs; (c) to explain the general concepts and the C# implementations of the algorithms for finding the maximum flow in networking.

METHODS

We consider a directed $G = (V, E)$, where V is a set of vertices and E is a set of connections (edges) between the vertices, $e = (x, y)$ consists of two vertices such that $x, y \epsilon V$. Let $G = (V, E)$ is connected and loopless, G has only one start node, named $source\ (s)$ and only one end node, named $sink(t)$.

The weight $c(e)$ of the arc $e \in E$ is called the $capacity$ and it is a nonnegative real number, i.e. we have a mapping.

A flow f of a network is a weight mapping $E \to \mathbb{R}_0$, which satisfies for each arc e, we have the capacity constraint $f(e) \leq c(e)$, and each vertex $v \neq s, t$. In this notation $val(f_e)$ denotes the value of the flow for the arc $e(u, v)$. With f^* we denote a *maximum flow*, if the value is the largest possible and for every flow is in effect $val(\, f_e{}^*) \geq val(f_e)$.

$$\sum_{y \in V} f(x,y) - \sum_{y \in V} f(y,x) \; = 0, \quad x \neq s,t \qquad (1)$$

$$0 \leq f(x,y) \leq c(x,y), \qquad (x,y) \in E \qquad (2)$$

$$\sum_{y \in V} f(s,y) - \sum_{y \in V} f(y,s) \; = 0, \qquad (3)$$

$$\sum_{y \in V} f(y,t) - \sum_{y \in V} f(t,y) \; = 0, \qquad (4)$$

Here $f(x, y)$, $(x, y) \in E$ is named flow if and only if the equations (1) - (4) are in effect.

* To whom all correspondence should be sent.
E-mail: sapundzhi@swu.bg

The idea of the Ford-Fulkerson algorithm is simple [7]. As long as there is a path from the *source* (s) to the *sink*(t), with available capacity on all edges in the path, we send flow along one of these paths. Then we find another path, and so on. A path with available capacity is called an *augmenting path*. According to the algorithm the flow in a network is a collection of chain flows which has the property that the sum of the number of all chain flows that contain any edge is no greater than the capacity of that edge [13-16]. A flowchart illustrating the maximum flow network is shown in Figure 1 [17]. A pseudocode of Ford-Fulkerson algorithm is given below (Figure 2).

On Figure 3 a simple web architecture is presented, with users, web servers, databases and data where the ends should be considered in an abstract way. We can consider this diagram as an example of flow networking with source (users) and sink (data) (Figure 2).

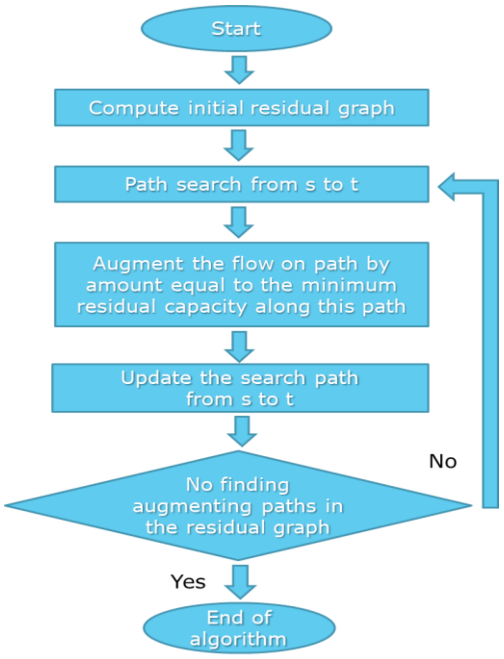

Start
Compute initial residual graph
Path search from s to t
Augment the flow on path by amount equal to the minimum residual capacity along this path
Update the search path from s to t
No finding augmenting paths in the residual graph
No
Yes
End of algorithm

Fig. 1. Flowchart for the maximum flow problem in network.

Algorithm: *Ford-Fulkerson*

```
1: Input: $G = (V,E), s, t$. Any network flow from $s$ to $t$ is selected. Initially, label the source. If there is no
   such flow, a flow of 0 is set for each arc $(x,y), f(x,y) = 0$.
2: By the ordered pair $(-,\infty)$ the source $s$ is coloured.
3: If there is a non-coloured vertex $y$ in coloured $x$:
    • If $(x,y) \in E$ and $f(x,y) < c(x,y)$ then color the arc $(x,y)$ and mark the node $y$ by the pair $(x^+,\Delta_y)$,
      where $\Delta y = \min\{\Delta x, c(x,y) - f(x,y)\}$, forward label.
    • If $(x,y) \in E$ and $f(y,x) > 0$ then color the arc $(y,x)$ and mark the node $y$ by the pair $(x^+,\Delta y)$,
      where $\Delta y = \min\{\Delta x, f(x,y)\}$, backward label.
      After that go to step 4, else return to step 2.
4: If $y = t$, go to step 5. Else, return to step 3.
5: Let the node $y$ is labeled by pair $(d_y, \Delta y)$. Then:
    • If $d_y = x^+$ then $f(x,y) = f(x,y) + \Delta t$.
    • If $d_y = x^+$ then $f(x,y) = f(x,y) - \Delta t$.
6: If $x = s$, remove all labels and go to step 2. Else apply $y = x$ and go back to step 5.
7: Output: The value of the maximum flow $f^*$.
```

Fig. 2. Ford-Fulkerson algorithm.

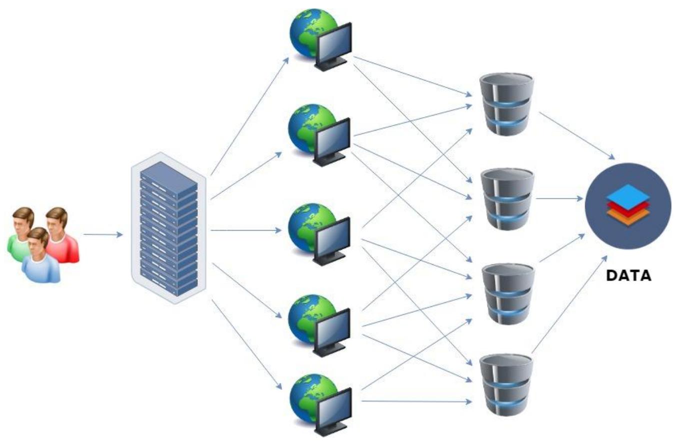


Fig. 3. Diagram for example of flow networking with source (users) and sink (data).

SIMULATION RESULTS

The maximum-flow problem in networking has been investigated by many researchers because of its importance for many areas of applications [3]. Max-flow and min-cut are widely applicable algorithms for solving the following problems: network reliability, network connectivity, distributed computing, data mining, bipartite matching, image segmentation, and more.

Popular algorithms for finding the max-flow in a network are Ford-Fulkerson algorithm, Edmonds-Karp algorithm, Dinic's algorithm and Goldberg-Tarjan algorithm. The values of execution time and different number of vertices for these are presented in Table 1. It depends on the number of n, m and c (number of nodes, edges and capacities) respectively.

The main task of the present work was the design, implementation and analysis of the Ford-Fulkerson algorithm for finding max-flow in network (Figure 4). The C# implementation was developed by using Visual Studio Community 2018. The developed program was tested in the PC with following parameters: Intel® Pentium® Processor N3710, 1.6 GHz (4 CPUs), 4096 MB RAM [18,19]. The experimental results are shown on Figure 5.

The algorithms of Edmonds-Karp and Goldberg-Tarjan use breath-first-searches and are performed from the sink, labelling each vertex with the distance to the sink, while the algorithm of Ford-Fulkerson can be applied to find the maximum flow between a single source and a single sink in a network [20, 21-29].

As can be seen, the time complexity for Ford-Fulkerson algorithm is higher than for other algorithms, it is simple to implement in finding the maximum flow. The algorithm of Edmonds-Karp is a version on the Ford-Fulkerson algorithm. Goldberg-Tarjan algorithm manipulates the preflow rather than an actual flow in a graph.

Table 1. Comparison for time complexity of maximum flow algorithms.

Maximum-flow algorithms	Method	Asymptotic Time	Year
Ford-Fulkerson	Augmenting path	$O(nm\ c)$	1955
Edmonds-Karp	Shortest path	$O(nm^2)$	1970
Edmonds-Karp	Fattest path	$\mathrm{O}(\mathrm{mlog} \cup (\mathrm{mlogn}))$	1970
Dinic	Improved shortest path	$O(mn^2)$	1970
Goldberg-Tarjan	FIFO preflow-push	$O(mn\log(n^2/m))$	1986

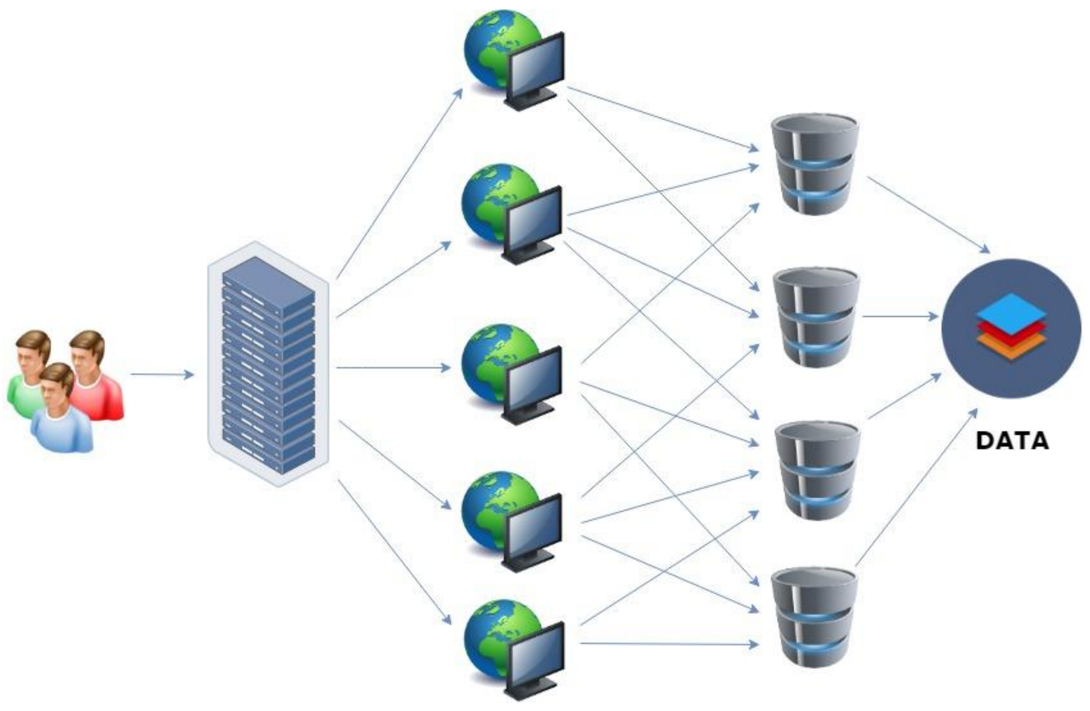
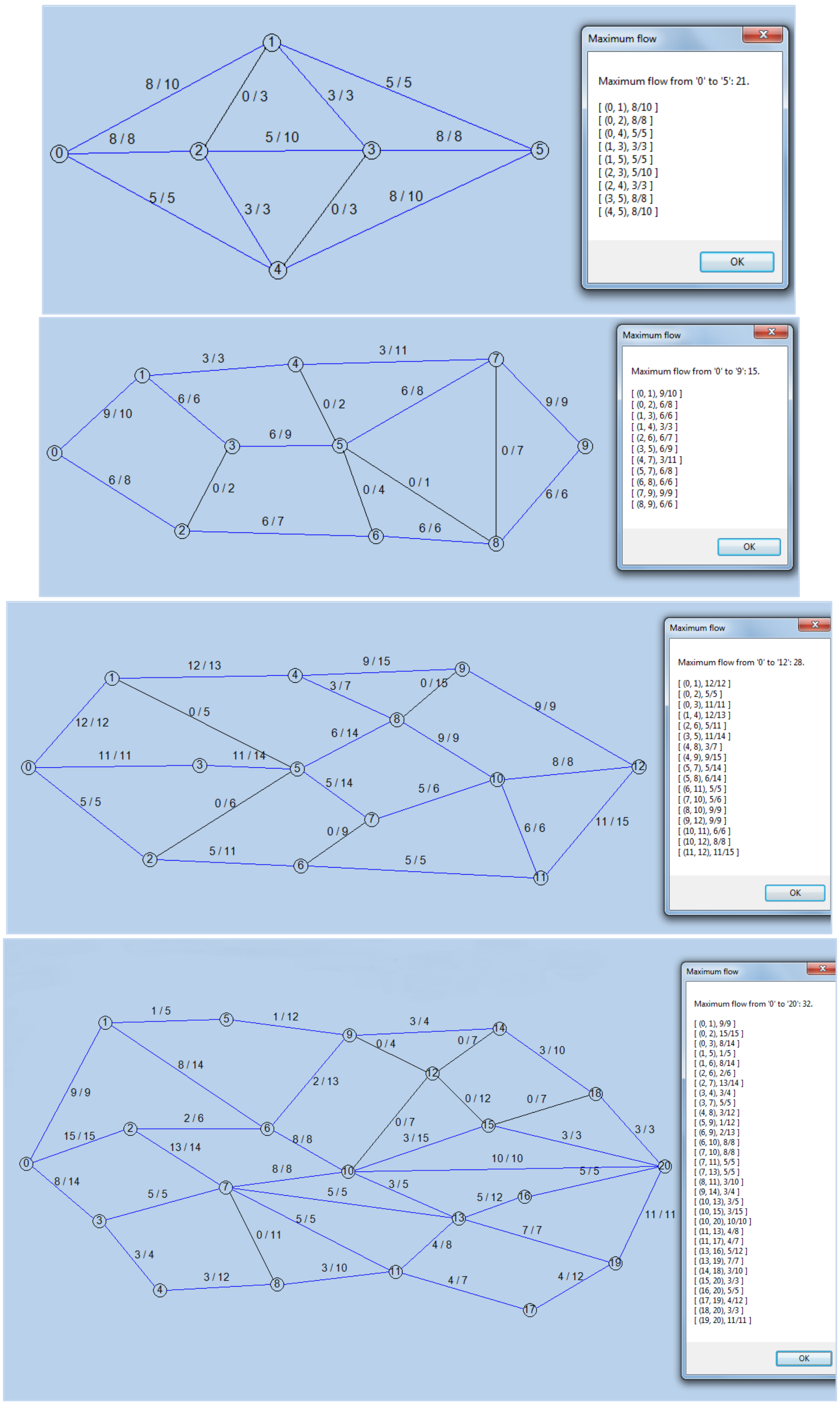


Fig. 4. Diagram for example of flow networking with source (users) and sink (data).



```
Maximum flow from '0' to '5': 21.

[ (0, 1), 8/10 ]
[ (0, 2), 8/8 ]
[ (0, 4), 5/5 ]
[ (1, 3), 3/3 ]
[ (1, 5), 5/5 ]
[ (2, 3), 5/10 ]
[ (2, 4), 3/3 ]
[ (3, 5), 8/8 ]
[ (4, 5), 8/10 ]
```

```
Maximum flow from '0' to '9': 15.

[ (0, 1), 9/10 ]
[ (0, 2), 6/8 ]
[ (1, 3), 6/6 ]
[ (1, 4), 3/3 ]
[ (2, 6), 6/7 ]
[ (3, 5), 6/9 ]
[ (4, 7), 3/11 ]
[ (5, 7), 6/8 ]
[ (6, 8), 6/6 ]
[ (7, 9), 9/9 ]
[ (8, 9), 6/6 ]
```

```
Maximum flow from '0' to '12': 28.

[ (0, 1), 12/12 ]
[ (0, 2), 5/5 ]
[ (0, 3), 11/11 ]
[ (1, 4), 12/13 ]
[ (2, 6), 5/11 ]
[ (3, 5), 11/14 ]
[ (4, 8), 3/7 ]
[ (4, 9), 9/15 ]
[ (5, 7), 5/14 ]
[ (5, 8), 6/14 ]
[ (6, 11), 5/5 ]
[ (7, 10), 5/6 ]
[ (8, 10), 9/9 ]
[ (9, 12), 9/9 ]
[ (10, 11), 6/6 ]
[ (10, 12), 8/8 ]
[ (11, 12), 11/15 ]
```

```
Maximum flow from '0' to '20': 32.

[ (0, 1), 9/9 ]
[ (0, 2), 15/15 ]
[ (0, 3), 8/14 ]
[ (1, 5), 1/5 ]
[ (1, 6), 8/14 ]
[ (2, 6), 2/6 ]
[ (2, 7), 13/14 ]
[ (3, 4), 3/4 ]
[ (3, 7), 5/5 ]
[ (4, 8), 3/12 ]
[ (5, 9), 1/12 ]
[ (6, 9), 2/13 ]
[ (6, 10), 8/8 ]
[ (7, 10), 8/8 ]
[ (7, 11), 5/5 ]
[ (7, 13), 5/5 ]
[ (8, 11), 3/10 ]
[ (9, 14), 3/4 ]
[ (10, 13), 3/5 ]
[ (10, 15), 3/15 ]
[ (10, 20), 10/10 ]
[ (11, 13), 4/8 ]
[ (11, 17), 4/7 ]
[ (13, 16), 5/12 ]
[ (13, 19), 7/7 ]
[ (14, 18), 3/10 ]
[ (15, 20), 3/3 ]
[ (16, 20), 5/5 ]
[ (17, 19), 4/12 ]
[ (18, 20), 3/3 ]
[ (19, 20), 11/11 ]
```

Fig. 5. Results of the C# implementation of the maximum-flow algorithm for different nodes, edges and capacities

When the capacities of the edges are integers, the runtime of Ford-Fulkerson algorithm is bounded by the following notation: $O(|E|f)$, where $|E|$ is the number of edges in the graph and f is the maximum flow in the graph. In this case each augmenting path can be found in $O(|E|)$ time. Other modified algorithms for finding the maximum flow in networking are developed to reduce computation time, but they require high-speed processors such as supercomputers to simulate the problem and calculate the optimal flow. In conclusion, we can say that the algorithm of Ford-Fulkerson guarantees to stop work if the edge capacity has a non-negative real value.

Acknowledgement: *This paper is partially supported by SWU "Neofit Rilski", Bulgaria, Project RPY – B4/19; Project of the National Science Fund of Bulgaria, BNSF H27/36 National Scientific Program "Information and Communication Technologies for a Single Digital Market in Science, Education and Security (ICTinSES)", financed by the Ministry of Education and Science.*

REFERENCES

1. K. Erciyes, Guide to Graph Algorithms: Sequential, Parallel and Distributed, Springer, 2018.
2. M. Kochenderfer, T. Wheeler, Algorithms for Optimization, The MIT Press, 2019.
3. J. Evans, E. Minieka, Optimization Algorithms for Networks and Graphs, Second edn., Inc., New York and Basel, 1992.
4. R. Gould, Graph Theory (Dover Books on Mathematics), US, Cafifornia, 2012.
5. L. Ford, D. Fulkerson, D. *Canadian Journal of Mathematics*, **8,** 399, (1956).
6. T. Cormen, Ch. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, Second edn., MIT Press and McGraw–Hill, 2001, p. 651.
7. J. Edmonds, R. Karp, *Journal of the ACM*, **19**, **2**, 248, (1972).
8. E. Dinic, *Soviet Mathematical Docladi*, **11,** 1277, (1970).
9. A. Goldberg, A New Max-Flow Algorithm. MIT Technical Report MIT/LCS/TM-291, Laboratory for Computer Science, MIT, 1985.
10. A Goldberg, R. Tarjan, *Proceedings of the eighteenth annual ACM symposium on Theory of computing – STOC* **'86**, 136, (1986).
11. S. Even, R. Tarjan, *SIAM Journal on Computing*, **4,** 507, (1975).
12. A. Karzanov, *Matematicheskie Voprosy Upravleniya Proizvodstvom*, **5,** 81, (1973).
13. F. Sapundzhi, M. Popstoilov, Proc. 27th National Conference with International Participation "TELECOM 2019", Sofia, Bulgaria, 2019, p. 62.
14. F. Harary, Graph Theory, Addison-Wesley, 1969.
15. D. West, Introduction to Graph Theory, Prentice Hall, 1996.
16. F. Sapundzhi, M. Popstoilov, *Bulgarian Chemical Communications*, **50**, Special Issue B, 115 (2018).
17. M. Kyi, L. Naing, *International Journal of Scientific and Research Publications*, **8(12)**, 306 (2018).
18. M. Negnevitsky, Artificial Intelligence: A Guide to Intelligent Systems, Addison-Wesley, 2011.
19. B. Johnson, Professional Visual Studio, 2015.
20. F. Sapundzhi, *International Journal of Online and Biomedical Engineering*, **15 (11),** 139 (2019).
21. F. Sapundzhi, T. Dzimbova, P. Milanov, N. Pencheva, *International Journal Bioautomation*, **17 (1)**, 5 (2013).
22. F. Sapundzhi, T. Dzimbova, N. Pencheva, P. Milanov, *Der Pharma Chemica*, **8**, 118 (2016).
23. F. Sapundzhi, T. Dzimbova, N. Pencheva, P. Milanov, *Bulg. Chem. Commun.*, **50**, Special Issue B, 44 (2018).
24. R. Mavrevski, M. Traykov, I. Trenchev, M. Trencheva, *WSEAS Transactions on Systems and Control* ,**13**, 242 (2018).
25. F. Sapundzhi, T. Dzimbova, N. Pencheva, P. Milanov. *Bulg. Chem. Commun.*, **50**, Special Issue B, 15 (2018).
26. F. Sapundzhi, *International Journal of Online and Biomedical Engineering*, **15 (12)**, 88 (2019).
27. F. Sapundzhi, T. Dzimbova, *International Journal of Online and Biomedical Engineering*, **15 (15)**, 39 (2019).
28. F. Sapundzhi, K. Prodanova, M. Lazarova *AIP Conference Proceedings*, **2172**, 100008 1-6 (2019).
29. V. Kralev, R. Kraleva, IJACR, **7** (28), 1 (2017)