ZULFAQAR Journal of Defence Science, Engineering & Technology
e-ISSN: 2773-5281
Vol. 6, Issue 1 (2023)
DOI: https://doi.org/10.58247/jdset-2023-0601-02
Journal homepage: https://zulfaqarjdset.upnm.edu.my

ON MAXIMUM FLOW OF NETWORKS

Khairani Abd. Majid[a*], Suzaimah Ramli[b], Zaharin Yusoff[c], Norazman Mohamad Nor[d], Nor Afiza Mat Razali[b]

[a] Department of Defence Science, Faculty of Science and Defence Technology, National Defence University of Malaysia, Sg. Besi Camp, 57000 Kuala Lumpur, Malaysia
[b] Department of Computer Science, Faculty of Defence Science and Technology, National Defence University of Malaysia, Sg. Besi Camp, 57000 Kuala Lumpur, Malaysia
[c] Institute of Research, Development & Innovation, International Medical University (IMU), 126, Jln Jalil Perkasa 19, Bukit Jalil, 57000 Kuala Lumpur, Malaysia
[d] Department of Civil Engineering, Faculty of Engineering, National Defence University of Malaysia, Sg. Besi Camp, 57000 Kuala Lumpur, Malaysia

ARTICLE INFO

ARTICLE HISTORY
Received: 10-01-2023
Revised: 30-03-2023
Accepted: 30-04-2023
Published: 30-06-2023

KEYWORDS
Dinic's algorithm
Flow network
Ford Fulkerson algorithm
Maximum flow
Pseudocode

ABSTRACT

This paper aims to introduce and discuss two existing algorithms, namely Ford-Fulkerson's Algorithm and Dinic's Algorithm. These algorithms are for determining the maximum flow from source (s) to sink (t) in a flow network. A numerical example is solved to illustrate both algorithms, and to demonstrate, study, and compare the procedures at each iteration. The results show that Dinic's Algorithm returns the maximum flow that takes a smaller number of iterations and augmentations than the Ford-Fulkerson Algorithm. In terms of complexity, the running time of Dinic's algorithm is O(n^2m), which should make it perform better on dense graphs. This goes to show that the claim by many researchers that Dinic's Algorithm is very powerful in solving big network flow problems is justified.

1.0 INTRODUCTION

A network is a rather simple mathematical concept but has many uses. A network can be defined as a triple {S | N, A}, where S is a schema made up of elements of N and A, with N being a set of nodes (or vertices) and A is a set of arcs (or edges, or links), and that the following rules apply:

- If an arc a ϵ A is in the schema S, then a necessarily joins two nodes n_1, n_2 ϵ N.
- If a node n ϵ N is in the schema S, n may stand alone or may be joined by 1 or more arcs.
- An arc may be undirected, uni-directional, or bi-directional.

The schema S can thus be viewed diagrammatically and is the one often simply referred to as the network. Figure 1 is an illustration of a schema.

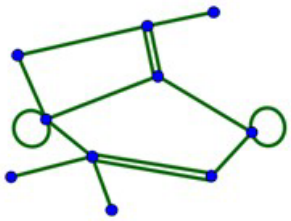

Figure 1. An example of a schema (network)

*Corresponding Author | Abd Majid, K. | khairani@upnm.edu.my

Being a fundamentally simple concept, networks can be readily studied mathematically, and the results would give very good insights into the applications where networks are applied. Networks are mainly used to represent or model various phenomena, typically

- Dynamic – flows, e.g. workflow, traffic flow, communication patterns, etc.
- Static – information, e.g. linguistic structures, ontology, etc.

In such representations, several guidelines apply:

- The nodes would normally represent, people, objects, states, countries, etc.
- In flows, the arcs would be directed, either uni-directional or bi-directional.
- In static, the arcs usually represent relations between nodes.
- Nodes and arcs have labels.
- Nodes and arcs may also be weighted to represent variations in types.

There are many forms of networks with differing properties, which would determine the appropriate applications to be represented or modeled. The following are some examples:

- The most well-known networks are computer networks with their different topologies – ring, mesh, bus, tree, and star.
- In general, networks can be a single node, a line of nodes joined by arcs, a tree where no node can lead back to itself by tracing the arcs and nodes, and a graph being the most general kind of network. As an example, lines and trees are used to represent linguistic structures (morphology, syntax, meaning, etc.)
- Groups of nodes and arcs that form equivalence relations (reflexive, symmetric & transitive) would show clusters in the schema. An example application for clusters is for the detection of possible terrorist cells, where the nodes represent suspects and the arcs the communications amongst them.
- If the set of nodes N is formed from two non-intersecting groups, $N = X \oplus Y$, and that all arcs in the schema join elements of X to elements of Y only – and vice versa, but never within X or within Y, the network is referred to as bipartite. Bipartite networks have been used to trace the movements of rare animals or insects that carry diseases.

Research in the network's domain can be empirical or analytical, or both. Empirical research is the most common and typically more applied, where the topology of the network used is already known and hence its mathematical properties. This would involve some hypothesis on the performance of the network, the collection of data (usually large), a simulation experiment, followed by a statistical analysis, and then the conclusions vis-à-vis the hypothesis. The main contribution here would be the (statistical) proof of the hypothesis. Examples include comparisons of networks of different topologies, performance of new devices over certain networks, etc. One of the demonstrations of the comparisons of algorithms to determine the maximum flow is between the Ford-Fulkerson's algorithm and Dinic's algorithm.

Analytical network research would typically involve the observation of a certain phenomenon and to model it using a network, which would entail formulating the topology of the network. Should the discovered topology be different from the well-known topologies, then the (mathematical) properties should be studied, followed by some proof of concept via an experiment (albeit small). This is the type of research that should be carried out. This paper, which is for the purpose of comparing different techniques for determining maximum flow problems, and for supporting the postulation that both algorithms are often used in solving network flow problems where Dinic's Algorithm is preferred to solve a bigger network.

2.0 LITERATURE REVIEW

The maximum flow problem is an optimization problem. There are many algorithms to solve the problem of sending flow at the greatest rate without violating any capacity constraints. A maximum flow problem aims to send as much flow as possible between two special nodes, from a source/supply node s to a sink/demand node t, without exceeding the capacity of any arc. In this paper two maximum flow algorithms are discussed, namely the Ford-Fulkerson's algorithm and the Dinic's algorithm. These two have been chosen because they are the most discussed and applied to solve the maximum flow problems.

*Corresponding Author | Abd Majid, K. | khairani@upnm.edu.my

Pattipati [1] introduced the concept of maximum flow of a network, as well as discussed, the historical perspective on maximum flow algorithms as shown in Table 1. Ford-Fulkerson & Edmond & Karp first attempts to push flow on one path at a time, called the augmentation path, and if a path cannot be found from source to sink, it would then just stop. Other algorithms would try to push flows through several paths at the same time, for which a series of layered networks would have to be constructed. If this cannot be done, then stop. In more recent algorithms, work is done on the arcs in the form of distributed computation.

Table 1. Historical perspective on maximum flow algorithms

Year	Algorithm	Complexity
1956	Ford & Fulkerson	can be exponential
1969	Edmonds & Karp	(nm^2)
1970	Dinic	(n^2m^2)
1974	Karzanov	$O(n^3)$
1977	Cherkaski	$O\left(n^2m^{1/2}\right)$
1978	Galil	$O\left(n^{5/3}m^{1/2}\right)$
1978	Malhotra et al.	$O(n^3)$
1979	Galil et al.	$O(nm(\log n)^2)$
1980	Sleator & Tarjan	$O(nm \log n)$
1986/1987	Golberg & Tarjan	$O(n^3)$
1987	Bertsekas	$O(n^3)$
1989	Ahuja & Orlin	survey of max. flow algorithms

Kyi & Naing [2], applied the Ford-Fulkerson algorithm to calculate the maximum flow in a water distribution pipeline network. The flow rate of water in a pipeline is very much dependent on the size of the pipes used and the pressure at each node, but they had only considered the capacity and the flow rate of water in the network, and not the size and pressure. The highest possible amount of energy flow for an eleven-node network, with the source node located in the generation zone and the target node located in the target zone, was calculated by Bulut & Ozcan [3] using the Ford-Fulkerson algorithm. They found that it was quite simple to use, and the results would be obtained within a reasonable time.

Kovalev & Novichikhin [4] introduced the inverse values for the railway line class in their analysis. By using the inverse values approach, the maximum flow in the system can be determined and it provides a clear view of relations of transportation capacities between railway lines and stations. Dergachev et al. [5] demonstrated that the methods of graph theory, fuzzy set theory, and mathematical programming can be used to solve the problem of determining the optimal option for work teams to carry out complex restoration works at railway facilities destroyed as a result of an emergency. Filipova-Patrakieva [6] provided examples showing that it is faster to converge to the final solution using Dinic's algorithm when compared with Ford-Fulkerson's algorithm and Edmond-Karp's algorithm. The complexity of Dinic's algorithm is equal to the other two algorithms. Bahadra et al. [7] used Dinic's algorithm on urban road junctions to determine the maximum flow and to reduce traffic congestions. Safadi et al. [8] applied Ford-Fulkerson algorithm, Dinic's algorithm and Edmund-Karp algorithm to a real-life traffic problem with 24 nodes and 60 edges in Kota Kinabalu. A comparative analysis was conducted, and it was discovered that all the methodologies showed the same amount of flow in the traffic network. However, there is a significant different in the CPU times, where Edmund-Karp's was lesser compared to the others, whereas the complexity time is less for Dinic's algorithm.

3.0 METHODS

3.1 Ford-Fulkerson Algorithm

The Ford-Fulkerson algorithm is an algorithm that deals with maximum-flow and minimum-cut problems. It was designed by L. R. Ford, Jr. and D. R. Fulkerson in 1956 [1]. The Ford-Fulkerson algorithm assumes that the input is a graph, G, along with a source node, s, and a sink node, t. The graph is any representation of a weighted graph, where the vertices are connected by edges of specified weights. The source node and the sink node are required to denote the beginning and the end of the flow network. This algorithm would send flow as long as there is a path from the source to the sink that can handle it. This path is called an augmenting path. To find the maximum flow (and min-cut as a product), the Ford-

*Corresponding Author | Abd Majid, K. | khairani@upnm.edu.my

Fulkerson method repeatedly finds augmenting graphs through the residual graphs and augments the flow until no more augmenting paths can be found. Figure 2 shows a version of the pseudo-code that explains the flow augmentation in more depth:

```
1.  flow = 0
2.  for each edge (u, v) in G:
3.  flow(u, v) = 0
4.  while there is a path, p, from s -> t in residual network G_f:
5.  residual_capacity(p) = min(residual_capacity(u, v) : for (u, v) in p)
6.  flow = flow + residual_capacity(p)
7.  for each edge (u, v) in p:
8.  if (u, v) is a forward edge:
9.  flow(u, v) = flow(u, v) + residual_capacity(p)
10. else:
11. flow(u, v) = flow(u, v) - residual_capacity(p)
12. return flow
```

Figure 2. A pseudo-code showing the flow of the Ford-Fulkerson algorithm

3.1.2 Illustrative Example for Ford-Fulkerson Algorithm

Consider the flow network given by Figure 3 below. The source node is denoted by s and the sink note is denoted by t. The capacities are shown on the respective arcs. The objective is to send the maximum amount of flow in this network from source s to sink t using the Ford-Fulkerson's algorithm.

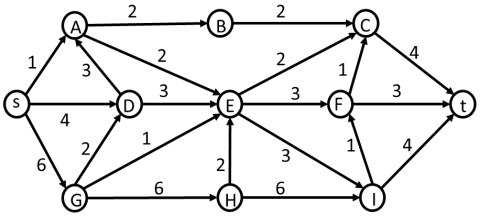

Figure 3. A network flow diagram

Initialize the value of f for each edge to 0 in the flow network G.

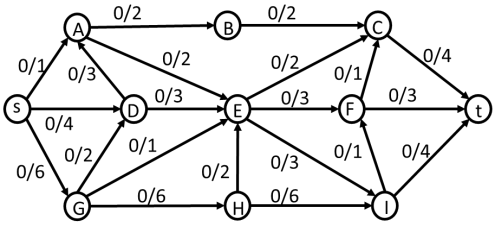

Figure 4. Initial flow network

The value of flow = 0. There does not exist any flow from source to sink.

Iteration 1:
Flow = min $\{1,4,6\} = 1$ and so, flow value = $0 + 1 = 1$

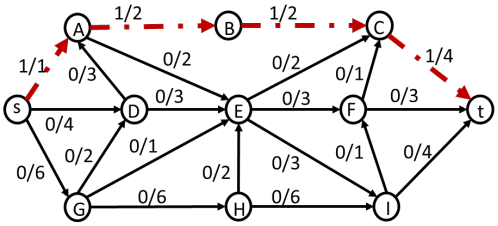

Figure 5. First iteration

Iteration 2:
Flow = min $\{4,3,3,3\} = 3$ and so, flow value = 0 + 1 + 3 = 4

*Corresponding Author | Abd Majid, K. | khairani@upnm.edu.my

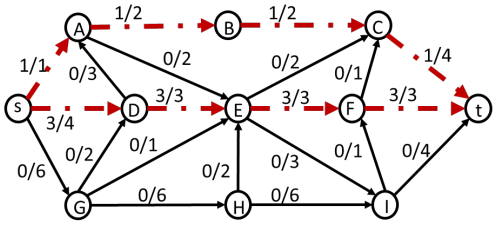

Figure 6. Second iteration

Iteration 3:
Flow = min {1,3,1,1,3} = 1 and so, flow value = 0 +1 + 3 + 1 = 5

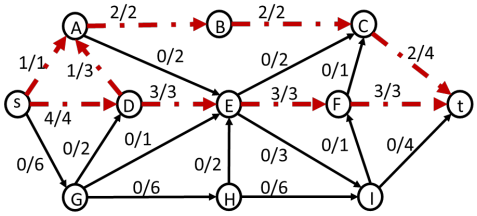

Figure 7. Third iteration

Iteration 4:
Flow = min {6,2,2,2,2,2} = 2 and so, flow value = 0 + 1 + 3 + 1 + 2 = 7

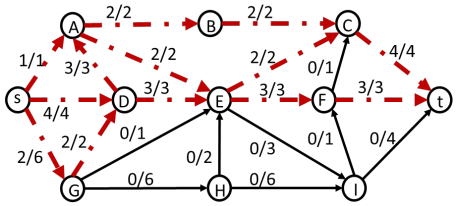

Figure 8. Fourth iteration

Iteration 5:
Flow = min {4,6,6,4} = 4 and so, flow value = 0 + 1 + 3 + 1 + 2 + 4 = 11

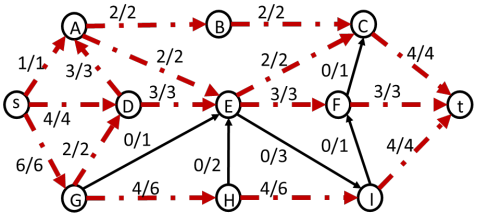

Figure 9. Fifth iteration

3.2 Dinic's Algorithm

In 1969 Yefim Dinitz designed the Dinic's algorithm and published it in 1970 [1]. Dinic's Algorithm is a strong polynomial maximum algorithm. The algorithm is more robust because it is based on novel concepts such as building a level graph, blocking flow, and the use of multiple graph traversal techniques, including breadth-first search (BFS) and depth-first search. A level graph is one where the value of each node is its shortest distance from the source. Blocking flow is used if no more flow can be sent using a level graph, and it includes looking for a new path from the bottleneck node. Dinic's algorithm runtime does not depend on the capacity values of the flow graph, which in some cases could contain very large graphs. In practice, the algorithm works better on bipartite graphs and can handle large sizes of such graphs. Figure 10 gives a pseudocode for Dinic's algorithm.

```
1.  function: DinicMaxFlow(Graph G, Node S, Node T):
2.      Initialize flow in all edges to 0, F = 0
3.      Construct level graph
```

*Corresponding Author | Abd Majid, K. | khairani@upnm.edu.my

```
4.      while (there exists an augmenting path in level graph):
5.          find blocking flow f in level graph
6.          F = F + f
7.          Update level graph
8.      return F
```

Figure 10. A pseudo-code showing the flow of the Dinic's algorithm

3.2.1 Illustrative Example for Dinic's Algorithm

Initial Residual Graph (same as given Graph). Total Flow = 0.

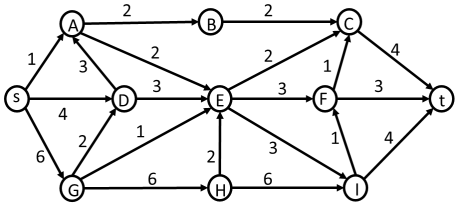

Figure 11. Initial Residual graph

Iteration 1:
We assign levels to all nodes using BFS. We also check if more flows are possible (or there is a $s-t$ path in the residual graph).

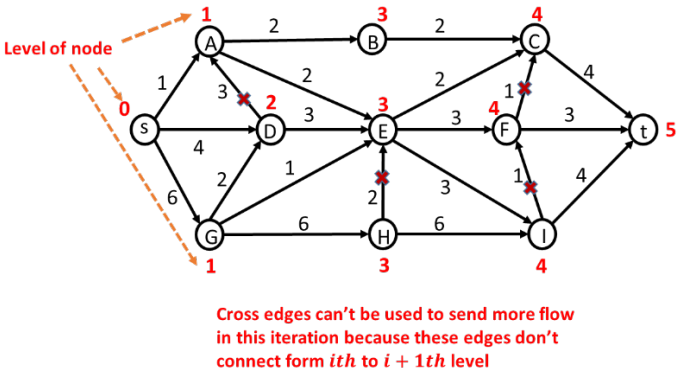

Figure 12. Level Graph

Blocking Flow Path:
s – A – B – C – t Flow = 1
s – G – H – I – t Flow = 4
s – D – E – C – t Flow = 2 Total Flow = 9
s – D – E – F – t Flow = 1
s – G – E – F – t Flow = 1

After finishing the level graph, the residual graph is as given in Figure 13.

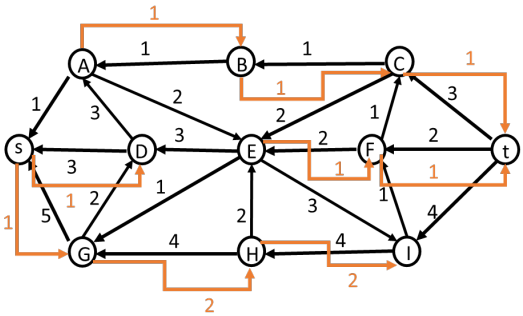

Figure 13. The residual graph

*Corresponding Author | Abd Majid, K. | khairani@upnm.edu.my

Iteration 2:
We then repeat the BFS to create a new level graph.

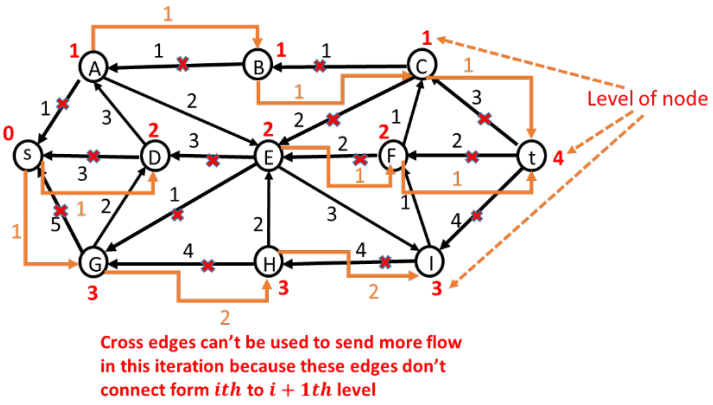

Figure 14: Level graph

Blocking flow Path:
s – D – A – B – C – t Flow = 1
s – G – H – E – F – t Flow = 1 Total Flow = 9 + 2 = 11

After finishing the level graph, the residual graph is as given in Figure 15. After the removal of edges with full capacity, the next level graph is as given in Figure 16. Since there is no more $s-t$ path, the algorithm is terminated. The maximum flow value is 11.

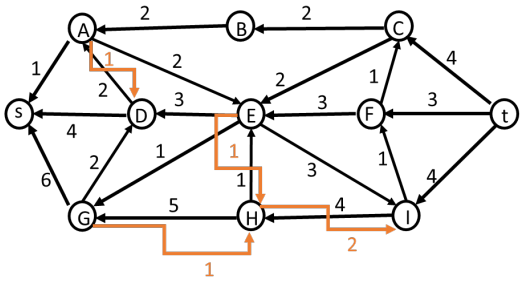

Figure 15. The residual graph

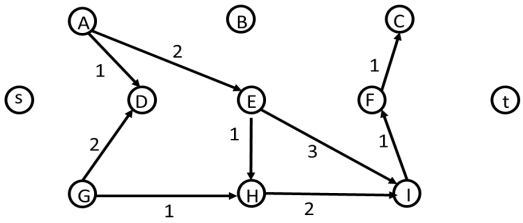

Figure 16. The level graph

4.0 RESULTS AND DISCUSSION

When solving a maximal flow problem, we found that the Ford-Fulkerson algorithm takes five (5) iterations to obtain the maximum value, whereas Dinic's algorithm only takes 2 iterations to get the same value.
Table 2 display the outcomes from the two algorithms. With a simple illustration of a network flow problem, we can see clearly that the path chosen and the flow value on each iteration for the Ford-Fulkerson's Algorithm. However, in Dinic's Algorithm, the paths and the flow values seem to be grouped together in each iteration. The difference between the two algorithms is due to the nature of the design of each algorithm. The Ford-Fulkerson's algorithm repeatedly finds augmenting graphs through the residual graphs, whereas Dinic's algorithm works with level graphs and blocking flows.

*Corresponding Author | Abd Majid, K. | khairani@upnm.edu.my

Table 2. Results from the Ford-Fulkerson's algorithm and Dinic's algorithm

Iteration	Result From the Two Techniques		
	Path	Flow Value	Method
1	s – A – B – C – t	= 1	Ford-Fulkerson
2	s – D – E – F – t	= 1 + 3 = 4	
3	s – D – A – B – C – t	= 1 + 3 + 1 = 5	
4	s – G – D – A – E – C – t	= 1 + 3 + 1 + 2 = 7	
5	s – G – H – I - t	= 1 + 3 + 1 + 2 + 4 = 11	
1	s – A – B – C – t s – G – H – I – t s – D – E – C – t s – D – E – F – t s – G – E – F – t	= 1 + 4 + 2 + 1 + 1 = 9	Dinic's
2	s – D – A – B – C – t s – G – H – E – F - t	= 9 + 2 = 11	

5.0 CONCLUSION

Analysing a directed graph based on the links in its topology is a necessity for the optimal use of resources in a system. Since the type of problem, we discussed was to maximize, we need to determine and assume that the allocation of flows has been maximized. The process was to optimize the problem using the graph's model with capacities on its arcs. This paper compared two existing techniques for solving the maximum flow problem. Both techniques have their strong points and weak points. We solved the same network problem using Ford-Fulkerson's algorithm and Dinic's algorithm, using both algorithms to find the maximum flow from source (s) to sink (t) in a flow network. We found that Dinic's algorithm returns the maximum flow that takes a much lesser number of iterations and augmentations than the Ford-Fulkerson algorithm. Using novel concepts such as blocking flow, a level graph, and unique applications of both BFS and DFS, Dinic's algorithm outperforms Ford-Fulkerson algorithm due to its high performance and strong polynomial time. Since the running time of Dinic's algorithm is $O(n^2m)$ [1], it should perform better on dense graphs. A possible problem with Dinic's algorithm now is that it might process the same part of the layer graph multiple times. It might be possible to speed it up by saving information about the paths in the layer graph that have already been visited. Dinic's algorithm is one of the most useful because it is very fast in practice in competitive programming. Both algorithms can also be used to model traffic in a road system (e.g. [9]), fluids in pipes, currents in an electrical circuit, or anything similar in which something travels through a network of nodes. Moreover, based on the pseudocodes and detail calculation procedure, one can code in C/C++, or JAVA, running these codes by using nonnegative real weighted values from actual information and data to solve general Ford-Fulkerson's and Dinic 's maximum flow problems.

6.0 ACKNOWLEDGEMENT

This work was supported under the National Defence University of Malaysia Short Term Research Grants UPNM/2020/GPJP/ICT/3.

List of Reference

[1] Elias, P., Feinstein, A., & Shannon, C. (1956). A note on the maximum flow through a network. IRE Transactions on Information Theory, 2(4), 117-119.

[2] Kyi, M. T., & Naing, L. L. (2018). Application of Ford-Fulkerson algorithm to maximum flow in water distribution pipeline network. International Journal of Scientific and Research Publications, 8(12), 306-310.

[3] Bulut, M., & Özcan, E. (2021). Optimization of electricity transmission by Ford–Fulkerson algorithm. Sustainable Energy, Grids and Networks, 28, 100544.

[4] Kovalev, K. E., & Novichikhin, A. V. (2021, December). Ford-Fulkerson algorithm refinement for the cooperation effectiveness increase of intensive and low-density lines. In Journal of Physics: Conference Series (Vol. 2131, No. 3, p. 032008). IOP Publishing.

*Corresponding Author | Abd Majid, K. | khairani@upnm.edu.my

[5] Dergachev, A. I., Dergachev, S. A., Chernykh, A. K., & Abu-Khasan, M. S. (2021, November). On the Approach to Optimization of Restoration Works at Railway Facilities. In Journal of Physics: Conference Series (Vol. 2096, No. 1, p. 012006). IOP Publishing.

[6] Filipova-Petrakieva, S. K. (2020). Applications of the heuristic optimization approach for determining a maximum flow problem based on the graphs' theory. Advances in Science, Technology and Engineering Systems, Special Issue on Multidisciplinary Sciences and Engineering, 5(6), 175-184.

[7] Bhadra, S., Kundu, A., & Khatua, S. (2021). Optimization of road traffic congestion in urban traffic network using dinic's algorithm. In Innovations in Bio-Inspired Computing and Applications: Proceedings of the 11th International Conference on Innovations in Bio-Inspired Computing and Applications (IBICA 2020) held during December 16-18, 2020 11 (pp. 372-379). Springer International Publishing.

[8] Safadi, Ş., Durmuşoğlu, A., & Özceylan, E. (2021). A COMPARATIVE STUDY FOR THE MAXIMUM FLOW PROBLEM ARISING AT ROAD NETWORKS IN KOTA KINABALU. Honorary Chair, 33.

[9] Majid, K. A., Ramli, S., & Ali, S. A. S. (2021). PREDICTION OF CONGESTIONS USING BASIC TRAFFIC UNIT. Malaysian Journal of Computer Science, 21-29.

*Corresponding Author | Abd Majid, K. | khairani@upnm.edu.my