Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/150411

A New Direction to The Maximal Flow Problem: A Route Merging Approach

Elias Munapo [1,*] *Santosh Kumar*[2,3] *and Kolentino Mpeta* [1]

[1]School of Economics and Decision Sciences, North West Uiversity, Mafikeng Campus Private Bag X2046, Mafikeng, 2735, South Africa
[2]School of Mathematical and Geospatial Sciences, RMIT University,124 La Trobe Street, Melbourne VIC 3000, Australia
[3]Department of Mathematics and Statistics, University of Melbourne, Grattan Street, Parkville, Victoria, 3010, Australia

Received: 2 Mar. 2021, Revised: 2 May 2021, Accepted: 7 Jun. 2021
Published online: 1 Jul. 2021

Abstract: The paper presents a new direction to the maximal flow problem. A maximal-flow problem involves finding the largest flow rate a network supply can have from the source to sink. The proposed algorithm solves the flow problem by merging flow routes iteratively until there is only one route left. It has the strength that the problem reduces in size at every iteration. The route merging algorithm belongs to a family of algorithms that solves by reducing the complexity of the flow problem. The flow problem has applications in the networking of pipes and road transportation networks.

Keywords: Maximal flow, Route merging, Linear programming, Source and sink

1 Introduction

A maximal-flow problem is the problem of finding the largest flow rate a network can supply from the source to sink. The network can be a road or pipe network system. The first algorithm to solve this problem is the Ford-Fulkerson algorithm that was developed in 1955 [1]. There have been other developments which provide many methods available for solving the maximal-flow problem today. These methods are classified as three main families by Verma and Batra [2]. The families are *augmenting-path, push relabel* and *pseudoflow* algorithms. Augmenting-path algorithms always satisfy the capacity and flow conservation constraints [3,4]. The push-relabel algorithms satisfy the capacity constraints but may violate the conservation constraints. With these algorithms we may have flow excesses at nodes but no flow deficits [5]. The third family which is the Hochbaum's pseudoflow algorithms satisfy the capacity constraints but sometimes violate the conservation constraints such that we have flow excess or deficit at nodes [6,7]. We refer the reader to Verma and Batra [2] for detailed information. The maximal-flow model has direct application in road and pipe network systems. In road network systems, the objective is to find the maximum amount of traffic that can flow from a starting point (source) to some destinantion or sink while in fluid network systems, the objective is to find the maximum amount of liquid/gas that can flow from a source to a sink. Besides the direct application in traffic and pipe transportation, the flow problem has other applications, as follows:

Binary assignment problem: The binary assignment problem can be reduced to a standard maximal-flow problem. For example passenger airlines with thousands of routes are scheduled using maximum-flow models saving huge amounts of money in terms of costs.

Match elimnation problem: Some games such as baseball can be modelled as maximum-flow problems. With this model match analysts can tell which teams are mathematically eliminated even before all the matches are played.

Edge - disjoint paths: Maximum-flow model is used to find the maximum number of edge-disjoint paths between two specified vertices in a directed graph. If we have capacities on both verices and edges then we call these *vertex-disjoint paths*.

Energy function: A large class of energy functions can be efficiently optimized by modelling them as maximum-flow problems [4,8].

* Corresponding author e-mail: Elias.Munapo@nwu.ac.za

Maximum matching in bipartite graphs: Maximum-flow problems are used in bipartite graphs.The problem is to find a matching with the maximum number of edges in a given bipartite graph. This can be modelled as a maximum-flow problem.

Dancing problem: Suppose we have the same number (p) of ladies and gentlemen at party and each man knows exactly a number (e) of women and each woman knows exactly a number (e) of men. The problem of arranging a dance so that each man dances with a different woman that he knows can be formulated as maximum-flow model.

Project selection: Suppose we have n projects such that some projects depend on the completion of others before they can commense. If each project is associated with some profit or loss, then the problem of finding the set of projects including all its depedences that give the maximum profit can be formulated as a maximum-flow model.

Image segimentation: The image processing problem can also be modelled as maximum-flow problem [9].

This paper presents an algorithm which solves the maximum-flow network model by merging routes. The proposed node merging algorithm for the maximum-flow problem belongs to a family of algorithms we can call *complexity reduction* family of algorithms. This family of algorithms solve the flow problem by reducing the size or complexity of the problem in stages or iterations. The worst case complexity of the algorithm together with computational experiments are presented.

2 Maximum flow problem

2.1 Single source single sink

Suppose a maximum - flow network problem is represented as shown in Figure 1.

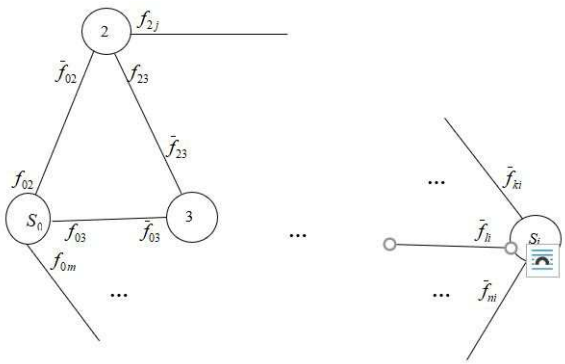

Fig. 1: Maximum-flow network problem.

Where f_{kl} is the flow into node l from node k,
$\bar{f}_{kl}$ is the flow into node k from node l,
$S_o$is the source and
$S_i$is the sink.
The objective is to find the maximum-flow from source $(S_o)$to sink (S_i).

2.2 Multi - source and multi-sink problem

The problem given in Figure 1 is a single source and a single sink. In real life, we usually have more than one source or sink. In such a case, the multi-source to multi-sink problem can be transformed into a single-source to single-sink one. The new source and sink are called supersource and supersink, respectively. Readers are encouraged to see Ahuja et al. [10] for more information and examples on transforming multi-source multi-sink maximum flow problem into a single-source single-sink maximum flow problem. The hunt for more efficient algorithms is an ongoing exercise [8,11].

3 Merging routes

Routes in the maximum-flow network diagram can be merged into one. This simplifies and reduces the complexity of the problem. There are several ways and rules of merging routes. The various rules, ways or theorems for merging routes in a maximum-flow network diagram are presented in this section.

3.1 Theorem 1

Suppose a single route maximum - flow network diagram is represented as shown in Figure 2.

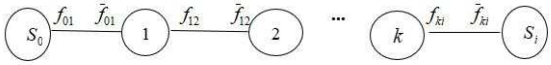

Fig. 2: Single route flow network diagram.

In Figure 2 the maximal - flow P_f,from S_o to S_i is given by (4.1).

$$P_f = \min[f_{01}, f_{12}, ..., f_{ki}] \quad (1)$$

The reverse maximal-flow (i.e. from S_i to S_o) is given by (2).

$$\bar{P}_f = \min[\bar{f}_{01}, \bar{f}_{12}, ..., \bar{f}_{ki}] \quad (2)$$

3.1.1 Proof by linear programming

Suppose P_f is the maximum-flow as shown in Figure 3.

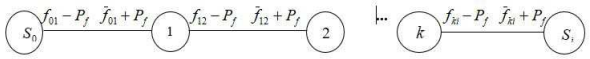

Fig. 3: Maximum-flow for single route network diagram.

Then linear programming (LP) can be used to prove Theorem 1. In other words the same maximum-flow problem given in Figure 3, in linear programming form is (3).

$$Maximize\, P_f, \tag{3}$$

Subject to: $f_{01} - P_f \geq 0, f_{12} - P_f \geq 0, \ldots, f_{ki} - P_f \geq 0.$
Where $P_f \geq 0$and constants $f_{01}, f_{12}, \ldots, f_{ki} \geq 0.$

This is a single variable LP and the optimal solution is given as (4).

$$P_f = \min[f_{01}, f_{12}, \ldots, f_{ki}]. \tag{4}$$

Similarly it can also be shown that the maximum reverse flow is (5).

$$\bar{P}_f = \min[\bar{f}_{01}, \bar{f}_{12}, \ldots, \bar{f}_{ki}]. \tag{5}$$

3.2 Theorem 2 (special case - two routes)

Suppose a flow network diagram is presented as given in Figure 4.

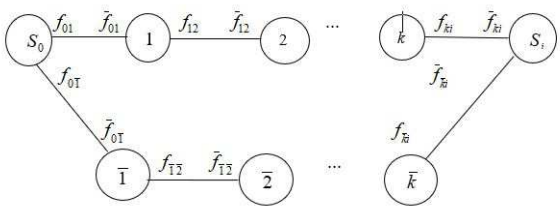

Fig. 4: Two route flow network diagram.

The maximum-flow $(P_f)$from S_o to S_i is given in (6).

$$P_f = \min[f_{01}, f_{12}, \ldots, f_{ki}] + \min[f_{0\bar{1}}, f_{\bar{1}\bar{2}}, \ldots, f_{\bar{k}i}]. \tag{6}$$

3.2.1 Proof

From the diagram given in Figure 4 we have (7).

$$P_f = P_f^1 + P_f^2. \tag{7}$$

Where $P_f^1$is the maximum-flow in the first route and $P_f^2$is the second one. Then,

$$P_f^1 = \min[f_{01}, f_{12}, \ldots, f_{ki}]. \tag{8}$$

$$P_f^2 = \min[f_{0\bar{1}}, f_{\bar{1}\bar{2}}, \ldots, f_{\bar{k}i}]. \tag{9}$$

Since there are two separate or different routes then the maximum-flow$(P_f)$is the sum of (8) and (9) and is given in (10).

$$P_f = \min[f_{01}, f_{12}, \ldots, f_{ki}] + \min[f_{0\bar{1}}, f_{\bar{1}\bar{2}}, \ldots, f_{\bar{k}i}]. \tag{10}$$

3.3 Theorem 3 (general case - more than two routes)

Theorem 3 is the general case of Theorem 2 in which the number of separate or different routes is greater than 2. This is represented as shown in Figure 5 and the maximum flow

P_f is given in (11).

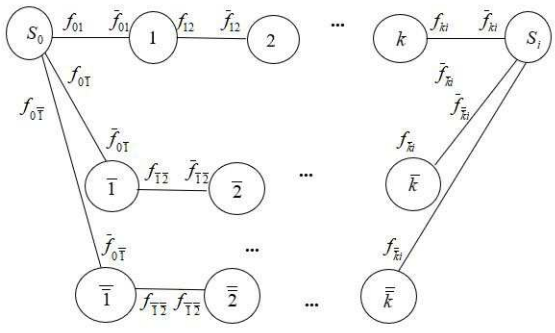

Fig. 5: Multi-route flow network diagram.

$$P_f = \min[f_{01}, f_{12}, \ldots, f_{ki}] + \min[f_{0\bar{1}}, f_{\bar{1}\bar{2}}, \ldots, f_{\bar{k}i}] + \min[f_{0\bar{\bar{1}}}, f_{\bar{\bar{1}}\bar{\bar{2}}}, \ldots, f_{\bar{\bar{k}}i}]. \tag{11}$$

3.3.1 Proof

Each of the multiple routes is treated separately, then the maximum-flow is the sum of all these routes. Suppose there are m routes and the maximum flow of the m^{th} route isP_f^m, then

$$P_f = P_f^1 + P_f^2 + ... + P_f^m. \tag{12}$$

From Figure 5,

$$P_f^3 = \min[f_{0\bar{1}}, f_{\bar{1}\bar{2}}, ..., f_{\bar{k}i}]. \tag{13}$$

Then,

$$P_f = \min[f_{01}, f_{12}, ..., f_{ki}] + \min[f_{0\bar{1}}, f_{\bar{1}\bar{2}}, ..., f_{\bar{k}i}] + \min[f_{0\bar{1}}, f_{\bar{1}\bar{2}}, ..., f_{\bar{k}i}] \tag{14}$$

3.4 Theorem 4 (Factorization rule)

Given (15),

$$P_f = \min[f_{0c} + f_{i1}, f_{0c} + f_{j1}]. \tag{15}$$

Where$i \neq j$andf_{0c} is common then this is simplified as (16).

$$P_f = f_{0c} + \min[f_{i1}, f_{j1}]. \tag{16}$$

3.4.1 Proof by linear programming

Flows are nonnegative, i.e., $f_{0c}, f_{i1}, f_{j1} \geq 0$. Modelling (16) as an LP becomes (17).

$$Maximize P_f, \tag{17}$$

Subject to: $f_{0c} + f_{i1} - P_f \geq 0, f_{0c} + f_{j1} - P_f \geq 0$.
Where $f_{0c}, f_{i1}, f_{j1} \geq 0$.

Rearranging we have (18),

$$Maximize P_f, \tag{18}$$

Subject to: $P_f \leq f_{0c} + f_{i1}, P_f \leq f_{0c} + f_{j1}$.
Where $f_{0c}, f_{i1}, f_{j1} \geq 0$.
The optimal solution of (18) is given in (19).

$$P_f = f_{0c} + \min[f_{i1}, f_{j1}]. \tag{19}$$

3.5 Theorem 5a (Triangle rule-1 for merging routes)

The two routes $i-j-k$and $i-k$presented in Figure refFig6 can be merged into a new route $i*-j*-k*$ as presented in Figure refFig7.

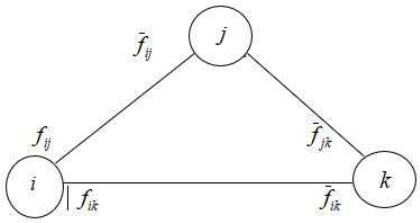

Fig. 6: Triangle rule for merging routes.

Fig. 7: Merged route way 1.

3.5.1 Proof

The theorem is valid if the maximum-flow of the two original routes and the maximum-flow of the merged route are the same.The maximum-flow (P_f) from node i to node k in Figure refFig6 is given in (19). Using Theorem 2,

$$P_f = \min[f_{ij}, f_{jk}] + f_{ik}. \tag{20}$$

Using Theorem 1, the maximum-flow $(P_f)$of merged route from node $i*$ to node $k*$ given in Figure 7 , becomes (21).

$$P_f = \min[f_{ij} + f_{ik}, f_{ik} + f_{jk}]. \tag{21}$$

Using Theorem 1 in this case. Since $f_{ik}$is common and by the Factorization Theorem.

$$P_f = f_{ik} + \min[f_{ij}, f_{jk}]. \tag{22}$$

This is exactly equal to the maximum-flow before merging.

3.6 Theorem 5b (Triangle rule-2 for merging routes)

The two routes $i-j-k$and $i-k$in Figure refFig6 can also be merged into a new route $i*-k*$as given in Figure 8.

Fig. 8: Merged route way.

3.6.1 Proof

As stated in the proof of way 1, the theorem is valid if the maximum-flows of the original routes and the merged route are the same. The flow leaving i^* is given in (23).

$$P_f = \min[f_{ij}, f_{jk}] + f_{ik}. \tag{23}$$

This is the same as the maximum-flow given in (20)

3.7 Theorem 6 (Kite rule for merging routes)

Given a kite as shown in Figure 9, the maximum-flow (P_f)from i to l is given as (24).

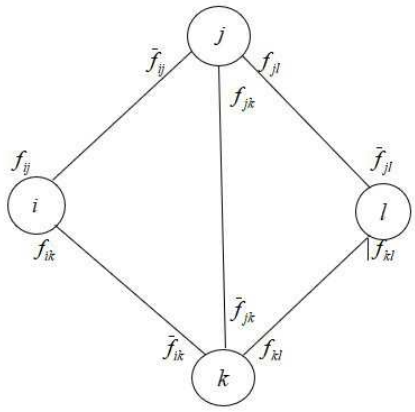

Fig. 9: Kite rule for merging routes.

$$P_f = \min[f_{ij} + f_{ik}, f_{ik} + f_{jk} + f_{jl}, f_{kl} + f_{jl}]. \tag{24}$$

3.7.1 Proof

Using Theorem 5A we can merge i-j-k and i-k as shown in Figure 10. Theorem 5A is used again to merge j-l and j-k-l as shown in Figure 10.

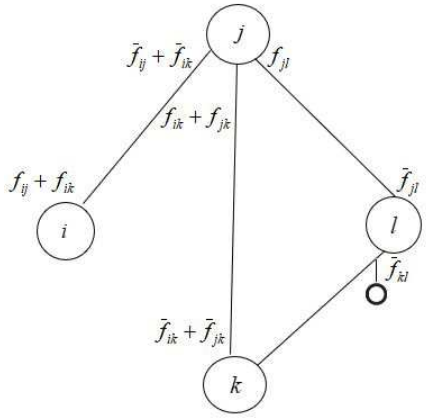

Fig. 10: Merging routes i-j-k and i-k.

Theorem 1 is then used to determine the maximum-flow in Figure 11 as given in (25).

$$P_f = \min[f_{ij} + f_{ik}, f_{ik} + f_{jk} + f_{jl}, f_{kl} + f_{jl}]. \tag{25}$$

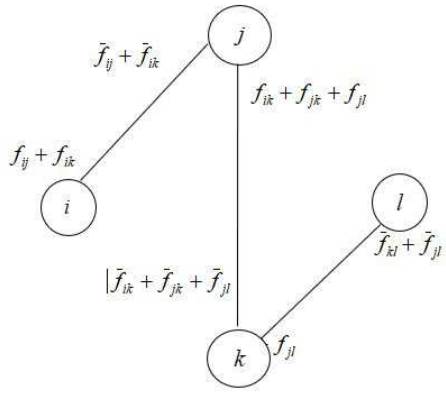

Fig. 11: Maximum-flow for single route network diagram.

The reverse maximum-flow can be shown as (26).

$$P_f = \min[\bar{f}_{ij} + \bar{f}_{ik}, \bar{f}_{ik} + \bar{f}_{jk} + \bar{f}_{jl}, \bar{f}_{kl} + \bar{f}_{jl}]. \tag{26}$$

An alternate way is to merge i-j and i-k-j. This will give the maximum-flow as shown in (27).

$$P_f = \min[f_{ij} + f_{ik}, f_{ij} + \bar{f}_{jk} + f_{kl}, f_{jl} + f_{kl}]. \tag{27}$$

The reverse maximum-flow is (28).

$$P_f = \min[\bar{f}_{ij} + \bar{f}_{ik}, f_{jk} + \bar{f}_{ij} + \bar{f}_{kl}, \bar{f}_{jl} + \bar{f}_{kl}]. \tag{28}$$

4 Applying the route merging approach

4.1 Illustrative example

Question: A road network system is shown in Figure 12. The numbers by the nodes represent the traffic flow in hundreds of cars per hour. What is the maximum-flow of cars from node S_0 to node S_i?

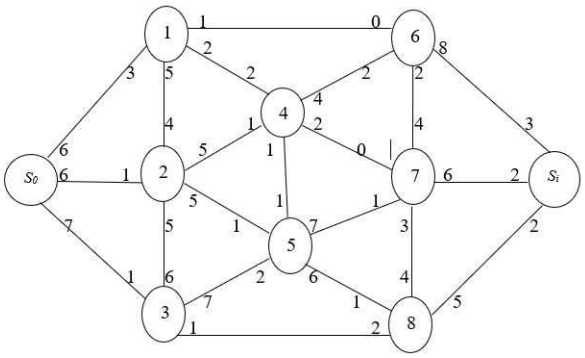

Fig. 12: Road network system.

Solution: Triangle rule-2 is applied to outermost arcs to get Figure 13 which is iteration 1.

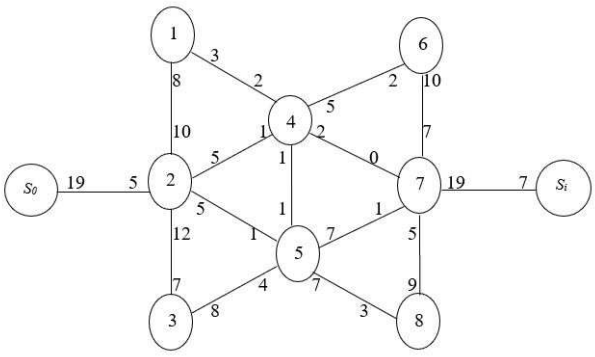

Fig. 13: Triangle rule - 1 applied to outmost arcs: iteration 1.

Then triangle rule-2 is applied to outmost arcs to obtain Figure 14 which is iteration 2.

The kite rule is used to merge routes 2-4-7 and 2-5-7 to get a single route flow problem shown in Figure 15.

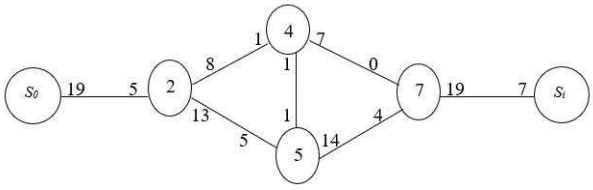

Fig. 14: Triangle rule - 2 applied to outmost arcs: iteration 2.

Fig. 15: Kite rule used to merge routes 2-4-7 and 2-5-7: iteration 3.

Maximum flow for kite rule is given in (29) which is simplified to (30).

$$P_f = \min[8+13, 7+1+13, 7+14], \tag{29}$$

$$P_f = \min[21, 21, 21] = 21. \tag{30}$$

The reverse maximum-flow is given in (31) and then simplified to (32).

$$\bar{P}_f = \min[1+5, 0+1+13, 0+4], \tag{31}$$

$$\bar{P}_f = \min[6, 14, 4] = 4. \tag{32}$$

Using Theorem 1, the maximum-flow in hundreds of cars is given in (33).

$$P_f = \min[19, 21, 19] = 19. \tag{33}$$

The maximum-flow of cars is **1 900** cars per hour. Applying the the available maximum-flow algorithm [12] we have the iterations given in Table .

Table 1: Iterations using the proposed route merging approach.

S_0	1	6				$S_i : P_f = 1.$
S_0	1	4	6			$S_i : P_f = 2.$
S_0	1	2	4	6		$S_i : P_f = 2.$
S_0	1	2	4	6	7	$S_i : P_f = 1.$
S_0	2	4	7	6		$S_i : P_f = 1.$
S_0	2	4	5	7		$S_i : P_f = 1.$
S_0	2	5	7			$S_i : P_f = 4.$
S_0	3	2	5	7		$S_i : P_f = 1.$
S_0	3	5	7	7		$S_i : P_f = 1.$
S_0	3	5	8	7		$S_i : P_f = 5.$
Sum						$P_f = 19.$

5 Complexity of the route merging algorithm

Using worst case complexity analysis the route merging algorithm can be found to be quadratic. A flow network diagram can be viewed as a structure of nodes and polygons. The smallest of these polygons is the triangle which connects any three neigbouring nodes. The other types of polygons are pentagon, hexagon or any other polygon that can connect a higher number of neighbouring nodes. The route merging algorithm works on collapsing a polygon at time until the whole network reduces to a set of arcs in a line. In this paper, we define

1.A small polygon as one that connects a small number of neigbouring nodes.
2.A route merging iteration as the collapsing of a polygon in a flow network diagram.

From (ii), we can conclude that the more the number of polygons in a flow network, the more route merging iterations are required to solve the problem and the more complex is the problem. In other words the flow problem is more complex if it is made up of triangles only. The route merging algorithm works on collapsing a polygon connecting some neighbouring nodes. Thus we can determine the worst case number of route merging iterations as the largest number of triangles(τ)that can be formed in an n-node flow network diagram.

When the number of nodes is $n = 3$.

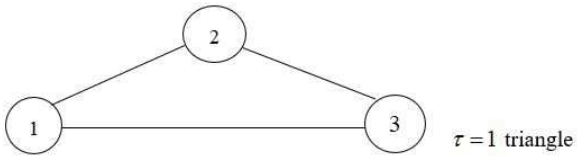

Fig. 16: Largest number of triangles when $n = 3$.

When the number of nodes is $n = 4$.

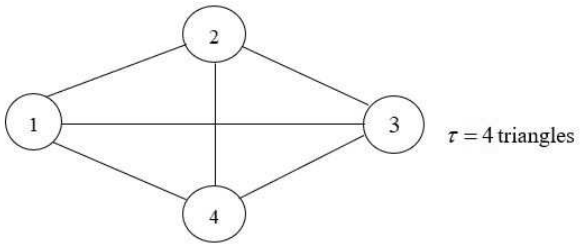

Fig. 17: Largest number of triangles when $n = 4$.

When the number of nodes $n = 5$.

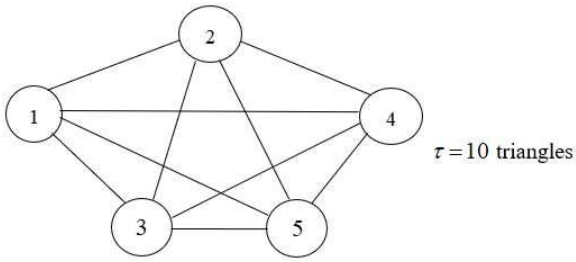

Fig. 18: Largest number of triangles when $n = 5$.

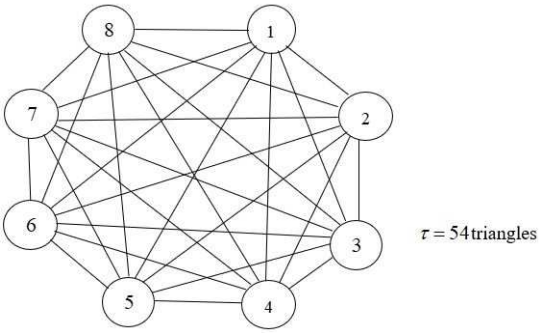

Fig. 19: Largest number of triangles when $n = 8$.

Table 2: Largest number of triangles as number of nodes increases.

Nodes	3	4	5	6	7	8	9	...	N
No. of triangles	1	4	10	20	35	56	84	...	τ

When the number of nodes $n = 8$.

From Table 2 given above, it can be shown that the largest numbers of triangles from $n = 3$ form a sequence whose n^{th} term is given by,

$$\tau = \tfrac{1}{2}(n-1)(n-2) + \tfrac{1}{2}(n-2)(n-3) + ... + \tfrac{1}{2}(n-r+1)(n-r). \quad (34)$$

Where,

$$n - r = 1. \quad (35)$$

Thus, the route merging algorithm has a quadratic (worst) case complexity. It is also noticeable that the number of nodes on its own is enough to determine the complexity of a maximum flow when the choice is to apply the route merging algorithm.

6 Conclusion

The paper presented a route merging algorithm which solves the maximum - flow network problem. The algorithm solves the flow problem by merging routes until there is only one route left. The proposed algorithm has

the strength that the problem reduces in size at every iteration. The flow problem has direct application in road and pipe networks. The route merging algorithm belongs to a family of algorithms solved by reducing the complexity of the flow problem. The flow problem has applications in the networking of pipes and road transportation networks.

6.1 Further research

There is need for computational experiments to define the ways that improve the proposed algorithm. More algorithms in parallel form [13] are also required for the flow problem. Some of the existing approaches, such as [14], [15], [16], [17] and [18].

Acknowledgement

We are grateful to the editors and the anonymous reviewers.

Conflict of Interest

The authors declare that they have no conflict of interest.

References

[1] L. R. Ford and D. R. Fulkerson, Maximal Flow Through a Network, *Canadian , Journal of Mathematics*, 399-404 (1956).
[2] T. Verma and D. Batra, MaxFlow Revisited: An Empirical Comparison of Maxflow Algorithms for Dense Vision Problems, BMVC, 1-12 (2012).
[3] Y. Boykov and V. Kolmogorov, An experimental comparison of mini-cut/max-flow algorithms for energy minimization in vision, *PAMI*, **26**(9), 1124-1137 (2004).
[4] V. Kolmogorov and R. Zabih, What energy functions can be minimized via graph cuts?, *PAMI*, **26**(2), 147-159 (2004).
[5] B. V. Cherkassky and A. V. Goldberg, On implementing the push-relabel method for the maximum flow problem, *Algorithmica*, **19**(40)390-410 (1997).
[6] B. G. Chandran and D. S. Hochbaum, A computational study of the pseudoflow and push-relabel algorithms for the maximum flow problem, *Operations Research,* **57**, 358-376 (2009).
[7] D, Goldfarb and M. D. Grigoriadis, A computational comparison of the Dinic and network simplex methods for maximum flow, *Annals of Operations Research*, 13, 83-123 (1988).
[8] C. Rother, P. Kohli, W. Feng and J. Y. Jia, Minimizing sparse higher order energy functions of discrete variables. *In CVPR*, 1382-1389 (2009).
[9] M. Rubinstein, A. Shamir and S. Avidan, Improved seam carving for video retargeting, *ACM Transactions on Graphics* (SIGGRAPH), 27(3), 1-9 (2008).
[10] R. K. Ahuja, T. L. Magnanti and J. B. Orlin, *Network Flows: Theory, Algorithms and Applications*, Prentice Hall. (1993).
[11] D. S. Johnson and C. C. McGeoch, Network flows and matchings. DIMACS series in discrete mathematics and theoretical computer science, Providence, R1: *American Mathematical Society*, **12** (1993).
[12] B. Render, R. M. Stair and M. E. Hanna, (2012). *Quantitative analysis for management*, 11^{th} edition, Pearson, 453-459. (2012).
[13] O. M. Surakhi, M. Qatawneh and A. Hussein, A Parallel Genetic Algorithm for Maximum Flow Problem, *International Journal of Advanced Computer Science and Applications*, **8**(6), 159-164 (2017).
[14] C. Caliskan, A computational study of the capacity scaling algorithm for maximum flow problem, *Computers & Operations Research,* **39**, 2742-2747(2012).
[15] D. S. Hochbaum, D.S., The pseudoflow algorithm: A new algorithm for the Maximum-Flow problem, *Operations Research*, 56(4), 992-1009 (2008).
[16] Y. Hu, X. Zhao, J. Liu, B. Liang and C. Ma, An Efficient Algorithm for Solving Minimum Cost Flow Problem with Complementarity Slack Conditions, *Mathematical Problems in Engineering*, 1-5 (2020).
[17] A. V. Goldberg, S. Hed, H. Kaplan, R. E. Tarjan, and R. F. Werneck, Maximum flows by incremental breadth-first search. *In Proceedings of the 19^{th} European conference on Algorithms*, ESA'11, 457-468 (2011).
[18] R. Masadeh, A. Alzaqebah, A. Sharieh, Whale Optimization Algorithm for Solving the Maximum Flow Problem, *Journal of Theoretical and Applied Information Technology*, **96** (2018).

Elias Munapo has a PhD from N.U.S.T., Zimbabwe which he obtained in 2010 and is a Professor of Operations Research at the North West University, Mafikeng Campus in South Africa. He is a guest editor of the Applied Sciences journal, has published two books, edited a number of books, a reviewer of a number journals, he has published a significant number of journal articles and book chapters. In addition he has presented at both local and international conferences and has supervised a couple of doctoral students to completion. His research interests are in the broad area of operations research. Professor Munapo is a member of the Operations Research Society of South Africa (ORSSA), South African Council for Natural Scientific Professions (SACNASP) as a Certified Natural Scientist, European Conference on Operational Research (EURO) and the International Federation of Operations Research Societies (IFORS).

Santosh Kumar is author and co-author of 195 papers and 3 books in the field of Operations Research. His contributions to the field of OR have been recognized in the form of ?Ren Pot Award? from the Australian Society for Operations Research in 2009 and a recognition award from the South African OR society as a non-member of the society in 2011. He has served as the President of the Asia Pacific Operations Research societies (1995-97), where ASOR was a member along with China, India, Japan, Korea, Malaysia, New Zealand, and Singapore. He is currently an Adjunct Professor at the RMIT University, Melbourne. He is a Fellow of the Institute of Mathematics and its Applications.

Kolentino Mpeta has a PhD from NWU, South Africa which he obtained in 2019 and is a Senior Lecturer of Statistics at the North West University, Mafikeng Campus in South Africa. He is a budding researcher and a reviewer for a number of journals. In addition he has presented at international conferences and has supervised a couple of masters' students to completion. His research interests are in the broad area of Statistics. He also has research interests in Operations research. Dr Mpeta is a member of the Operations Research Society of South Africa (ORSSA) and Statistics Association of South Africa (SASA).