Jurnal Mantik, 7 (2) (2023), ISSN 2685-4236 (Online)

Published by:Institute of Computer Science (IOCS)

Jurnal Mantik

Journal homepage: www.iocscience.org/ejournal/index.php/mantik

Solving the maximum flow problem with the lift-to-front algorithm

Hotler Manurung[1], Tinus Ginting[2], Adnan Surbakti[3], Astro Julida Harahap[4], Resep Sembiring[5]

[1,2,3,4]Informatics Engineering, Academy of Industrial Technology, Indonesia

[5]Mechanical Engineering, Immanuel College of Technology, Indonesia

A R T I C L E I N F O

Article history:

Received Jul 22, 2023
Revised Aug 8, 2023
Accepted Aug 20, 2023

Keywords:

Lift-To-Front
Preflow-Push
Problema Maximum Flow

ABSTRACT

Maximum flow network is an optimization problem related to the effort to optimize the system in maximizing the scale of materials that can be shipped from source to destination based on the system's capabilities. This problem can be solved using the lift-to-front algorithm. The lift-to-front algorithm is based on the application of lists by maintaining a list of vertices in the network. When a vertex is selected it is moved to the front of the list (hence the method is called 'lift-to-front') and the method starts its review once again. The results show that the designed software can demonstrate the application of the Lift-to-Front Algorithm in solving the Maximum Flow problem.

This is an open-access article under the CC BY-NC license.

Corresponding Author:

Hotler Manurung,
Informatics Engineering,
Academy of Industrial Technology,
Jl. Gatot Subroto No 325, Medan, Sumatera Utara, 20119, Indonesia.
Email: hotlermanurungskommkom@yahoo.com

1. INTRODUCTION

In the task of finding the longest path, we are given an n-vertex graph called G as well as an integer called k. (There is a possibility that Graph G will be directed.) The mission is to determine whether or not G has any paths that are at least k steps long(Huang & Li, 2023). Longest Path is a classic algorithmic problem that was fundamental in the development of parameterized complexity and is considered one of the most important(Fomin et al., 2023). The Hamiltonian path and cycle issues in directed graphs (or digraphs), which are respectively termed Hamiltonian Path Problem (HPP)(Wu et al., 2023) and Hamiltonian Cycle Problem (HCP), have been widely investigated(Murua et al., 2022). Many of these studies focus on the boundary between NP-complete and polynomial cases, either in terms of minimal (or just small) classes of digraphs for which the NP-completeness holds, or in terms of maximal (or just large) classes of digraphs for which the problems are polynomial, or even better, a Hamiltonian path or cycle always exist. One way to think about this boundary is in terms of minimal (or just small) classes of digraphs.

A version of the issue of maximum flow on a given directed graph in which some arc pairs are incompatible with one another or conflict with one another; to put it another way, it is forbidden for them to both carry positive flow at the same time. The maximum flow issue with conflicts is a well-known example of a problem that is extremely difficult to solve using NP-hard techniques(Şuvak et al., 2020). Approximation algorithms, heuristic algorithms, and randomized algorithms are some of the successful algorithmic frameworks

that have been presented in order to deal with NP-hard issues. These algorithmic frameworks have been developed in order to solve NP-hard problems. One of the most prominent drawbacks of these methods is that it is impossible to constantly ensure that they will produce ideal answers(W. Li et al., 2020).

The directed graph is modeled as a "flow network" and is intended to answer questions about material flow (Kumudakshi et al., 2022).Flow networks can be used to model liquids flowing through pipes, parts of an object through installation lines, electric current through electrical networks, information through communication networks, and so on (Duan et al., 2022). In graph theory, a flow network is a directed graph that includes a source (S) and a sink (T) in addition to numerous nodes that are connected by edges. Each edge has its own unique capacity, which may be thought of as the greatest amount of flow the edge can accommodate at one time(Monis et al., 2019).

One of the algorithms that can be used to solve the maximum flow problem on this flow network is the Lift-to-Front algorithm (Şuvak et al., 2020). The Ford-Fulkerson method, also known as the Lift-to-Front algorithm, is an algorithm that is commonly used to tackle the problem of achieving maximum flow in a flow network. The maximum flow issue includes determining the maximum amount of flow that can be delivered from a source vertex to a sink vertex in a directed weighted graph, subject to capacity limitations on the edges. This is done by calculating the greatest amount of flow that can be transferred from a source vertex to a sink vertex(Kyi & Naing, 2018).

The method finds an augmenting path, which is a path from the source to the sink in the residual graph(Q. Li et al., 2022). The residual graph is the graph that is created by subtracting the current flow from the capacity of each edge. The iterative process of finding an augmenting path is what makes the algorithm function. The algorithm then raises the flow along this path by the greatest feasible amount, which is the minimum capacity of the edges along the path. This brings the total quantity of flow along this path up to its maximum potential(Hu et al., 2020).

Problem with maximum flow due to Non-deterministic polynomial hard problems, often known as NP-hard issues, are those problems for which fundamental algorithms are unable to arrive at optimal solutions within a computation time that is constrained by a polynomial(Alridha et al., 2021). Complete search algorithms and approximate search algorithms are the two primary types of algorithms that are utilized while attempting to solve or optimize a solution for NP-hard problems(Mor et al., 2021). These two types of algorithms are referred to as "search algorithms."In the case of the comprehensive search algorithms, the suggested algorithms are required to try out all of the potential solutions for the NP-hard issue that has been provided. Therefore, an exponential amount of computer time is required in order to locate the best possible answer(Almufti et al., 2021).

Problems such as the traveling salesman problem(Cárdenas-Montes, 2018), the knapsack problem(Gurski et al., 2019), the maximum flow issue(Orkun Baycik, 2022), and the satisfiability problem are examples of problems that are considered to be NP-hard. These are tough issues to answer due to the fact that the number of viable solutions increases at an exponential rate in proportion to the magnitude of the problem. Computer science and mathematics are important fields because they give a framework for understanding the intrinsic complexity of computing tasks. NP-hard issues and NP-soft problems are examples of problems that fall into these categories. On the other hand, NP-soft issues can be solved in a polynomial amount of time, whereas NP-hard problems are regarded to have an intrinsic difficulty in effectively solving them(Au-Yeung et al., 2023).

The lift-to-front method is a variation or development of the preflow-push method which has an execution time of $O(V^3)$ and asymptotically has an execution time as good as $O(V^2E)$ ((Cheriyan & Mehlhorn, 1999). The lift-to-front method maintains a vertex list on the network(Zhou et al., 2023). Starting from the front, the method reviews the list, iteratively selects a vertex u that overflows and then removes it from the list, i.e., performs 'push and lift' operations until u has no positive excess (Jensen et al., 2023). When a vertex

is lifted, it is moved to the front of the list (hence the method is called 'lift-to-front') and the method starts its review once again (Akram et al., 2022).

2. RESEARCH METHOD

The preflow-push method allows us to apply the basic operations in any order arbitrarily. The lift-to-front method is a variation or development of the preflow-push method that has an execution time of $O(V^3)$ where V is the number of vertices and asymptotically has an execution time that is as good as $O(V^2E)$. The research stages can be seen in Figure 1.

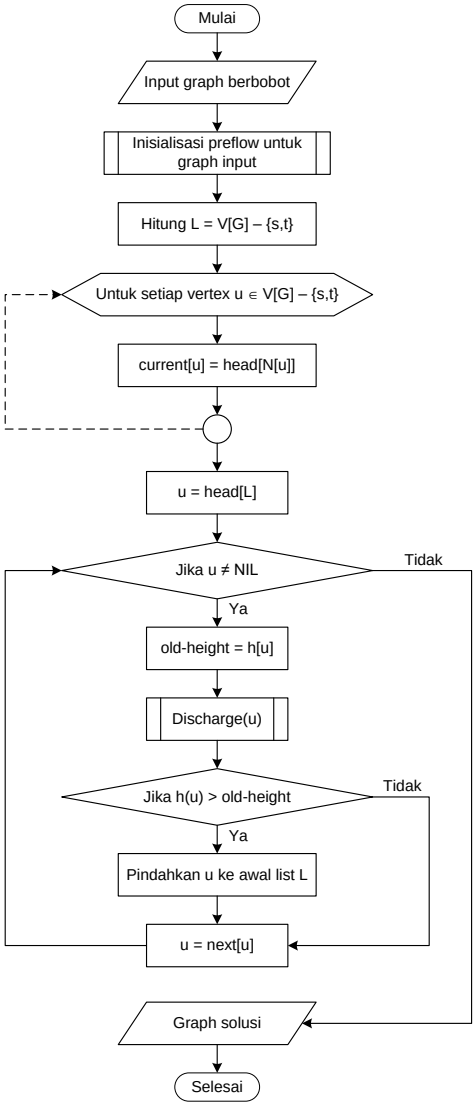

Figure 1: Research Stages

The Discharge (u) algorithm is used to push all vertices that have admissible edges and lift vertices when necessary to make them have admissible edges. The pseudocode of the Discharge (u) algorithm is as follows:

Discharge (u)

```
1. while e[u] > 0
2.         do v ← current [u]
3.             if v = nil
4.               then LIFT (u)
5.                       current[u] ← head [N[u]]
6.             elseif cf (u, v) > 0 and h[u] = h[v] + 1
7.                  then PUSH(u, v)
8.             else current[u] ← next-neighbor [v]
```

The recommended hardware specifications for running this comprehension software are as follows:

1. Pentium IV 1.6 GHz processor.
2. 2 Gb of memory (RAM).
3. Hard disk with at least 500 MB free space.
4. SVGA monitor and 64 MB VGA Card with a minimum resolution of 1024 x 768.
5. Keyboard and Mouse.

The software used to run this application is the Microsoft Windows 7, Windows 8, Windows 10, or Microsoft Windows NT/2000/XP operating system environment..

3. RESULTS AND DISCUSSIONS

Display of the 'Input Graph' form.

This form is where the input graph is designed. The display of the 'Input Graph' form can be seen in Figure 2.

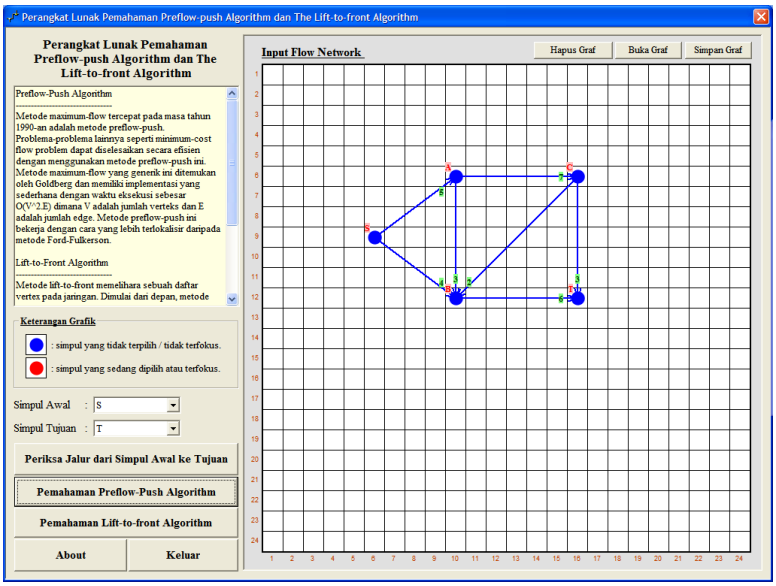

Figure 2. Display of 'Input Graph' Form

Display of 'Understanding Lift-to-Front Algorithm' form

This form serves to display the process of determining the maximum flow network using the Lift-to-Front algorithm. The display of the 'Understanding the Lift-to-Front Algorithm' form can be seen in Figure 3.

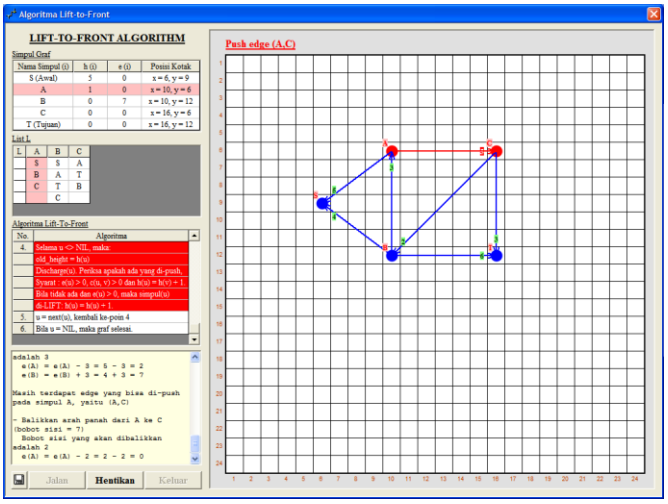

Figure 3. Display of 'Understanding the Lift-to-Front Algorithm' Form

4. CONCLUSION

The results of this study conclude that the Lift-to-Front Algorithm can be used to solve the maximum flow network problem. This algorithm uses a more structured List data structure, resulting in a faster solution graph. In addition, there are several suggestions that can be given based on this research. First, it is important to vary and compare algorithms in solving the Maximum Flow problem to find the most optimal approach. Second, it is also necessary to apply optimization methods in the selection of the List so that the algorithm can run efficiently and more optimally.

REFERENCES

Akram, M., Habib, A., & Allahviranloo, T. (2022). A new maximal flow algorithm for solving optimization problems with linguistic capacities and flows. Information Sciences, 612, *201–230. https://doi.org/10.1016/j.ins.2022.08.068*

Almufti, S. M., Marqas, R. B., Othman, P. S., & Sallow, A. B. (2021). Single-based and Population-based Metaheuristics for Solving NP-hard Problems. Iraqi Journal of Science. *https://doi.org/10.24996/10.24996/ijs.2021.62.5.34*

Alridha, A., Salman, A. M., & Al-Jilawi, A. S. (2021). The Applications of NP-hardness optimizations problem. Journal of Physics: Conference Series, 1818*(1), 012179. https://doi.org/10.1088/1742-6596/1818/1/012179*

Au-Yeung, R., Chancellor, N., & Halffmann, P. (2023). NP-hard but no longer hard to solve? Using quantum computing to tackle optimization problems. Frontiers in Quantum Science and Technology, 2. *https://www.frontiersin.org/articles/10.3389/frqst.2023.1128576*

Cárdenas-Montes, M. (2018). Creating hard-to-solve instances of travelling salesman problem. Applied Soft Computing, 71, *268–276. https://doi.org/10.1016/j.asoc.2018.07.010*

Cheriyan, J., & Mehlhorn, K. (1999). An analysis of the highest-level selection rule in the preflow-push max-flow algorithm. Information Processing Letters, 69*(5), 239–242. https://doi.org/10.1016/S0020-0190(99)00019-8*

Duan, Z., Sun, H., Wu, C., & Hu, H. (2022). Flow-network based dynamic modelling and simulation of the temperature control system for commercial aircraft with multiple temperature zones. Energy, 238, *121874. https://doi.org/10.1016/j.energy.2021.121874*

Fomin, F. V., Golovach, P. A., Lochet, W., Sagunov, D., Saurabh, S., & Simonov, K. (2023). Detours in directed graphs. Journal of Computer and System Sciences, 137, *66–86. https://doi.org/10.1016/j.jcss.2023.05.001*

Gurski, F., Rehs, C., & Rethmann, J. (2019). Knapsack problems: A parameterized point of view. Theoretical Computer Science, 775, *93–108. https://doi.org/10.1016/j.tcs.2018.12.019*

Hu, Y., Zhao, X., Liu, J., Liang, B., & Ma, C. (2020). An Efficient Algorithm for Solving Minimum Cost Flow Problem with Complementarity Slack Conditions. Mathematical Problems in Engineering, 2020, *e2439265. https://doi.org/10.1155/2020/2439265*

Huang, S., & Li, Z. (2023). Vertex-critical (P5,chair)-free graphs. Discrete Applied Mathematics, 341, *9–15. https://doi.org/10.1016/j.dam.2023.07.014*

Jensen, P. M., Jeppesen, N., Dahl, A. B., & Dahl, V. A. (2023). Review of Serial and Parallel Min-Cut/Max-Flow Algorithms for Computer Vision. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45*(02), 2310–2329. https://doi.org/10.1109/TPAMI.2022.3170096*

Kumudakshi, Hegde, S. M., & Anusha. (2022). On sequential directed graphs. Materials Today: Proceedings, 54, *738–742. https://doi.org/10.1016/j.matpr.2021.11.028*

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 (IJSRP), 8*(12). https://doi.org/10.29322/IJSRP.8.12.2018.p8441*

Li, Q., Zhang, T., Chen, C. L. P., Yi, K., & Chen, L. (2022). Residual GCB-Net: Residual Graph Convolutional Broad Network on Emotion Recognition. IEEE Transactions on Cognitive and Developmental Systems, *1–1. https://doi.org/10.1109/TCDS.2022.3147839*

Li, W., Ding, Y., Yang, Y., Sherratt, R. S., Park, J. H., & Wang, J. (2020). Parameterized algorithms of fundamental NP-hard problems: A survey. Human-Centric Computing and Information Sciences, 10*(1), 29. https://doi.org/10.1186/s13673-020-00226-w*

Monis, L., Kunjumon, B., & Guruprasad, N. (2019). Implementation of Maximum Flow Algorithm in an Undirected Network. In V. Sridhar, M. C. Padma, & K. A. R. Rao (Eds.), Emerging Research in Electronics, Computer Science and Technology *(pp. 195–202). Springer. https://doi.org/10.1007/978-981-13-5802-9_18*

Mor, B., Shabtay, D., & Yedidsion, L. (2021). Heuristic algorithms for solving a set of NP-hard single-machine scheduling problems with resource-dependent processing times. Computers & Industrial Engineering, 153, *107024. https://doi.org/10.1016/j.cie.2020.107024*

Murua, M., Galar, D., & Santana, R. (2022). Solving the multi-objective Hamiltonian cycle problem using a Branch-and-Fix based algorithm. Journal of Computational Science, 60, *101578. https://doi.org/10.1016/j.jocs.2022.101578*

Orkun Baycik, N. (2022). Machine learning based approaches to solve the maximum flow network interdiction problem. Computers & Industrial Engineering, 167, *107873. https://doi.org/10.1016/j.cie.2021.107873*

Şuvak, Z., Altınel, İ. K., & Aras, N. (2020). Exact solution algorithms for the maximum flow problem with additional conflict constraints. European Journal of Operational Research, 287*(2), 410–437. https://doi.org/10.1016/j.ejor.2020.04.001*

Wu, J., Cheng, Y., Yang, Z., & Chu, F. (2023). A 3/2-approximation algorithm for the multiple Hamiltonian path problem with no prefixed endpoints. Operations Research Letters, 51*(5), 473–476. https://doi.org/10.1016/j.orl.2023.07.003*

Zhou, J., Yuan, C., Qian, Z.-Y., Wang, B.-H., & Nie, S. (2023). Analysis of cut vertex in the control of complex networks. Chinese Physics B, 32*(2), 028902. https://doi.org/10.1088/1674-1056/aca208*