WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Memory Enhanced Dijkstra

Nachiketa Nalin, Smitha G R

Department of Information Science and Engineering, RV College of Engineering Bangalore, India

ABSTRACT —

The Dijkstra algorithm is a graph-based method that compares node distances, selects the shortest subsequent node, and generates an ideal path. However, The memory utilisation of the algorithm is poor. This problem aggravates when the graph becomes dense that is when it has a lot of nodes and edges. The square space complexity becomes too tough to handle for computer and cloud systems and causes delays in execution. This can have a negative impact on the system and this paper aims to solve that. The time complexity depends on the implementation time like adjacency list or adjacency matrix but the space complexity does not. It always take the same complexity irrespective of the implementation hence, the problem is solved using a different approach.

Keywords—**Graph theory, Dijkstra algorithm**

I. Introduction

Dijkstra algorithm is a path finding algorithm created by research scientists for shortest path finding. It is used to figure out the shortest possible path from a source node to all the other nodes in a graph. The path can be from any source node in the graph and can have only 1 path to any other node. The shortest path is calculated in exponential as well as logarithmic time depending on the implementation of the graph but the space taken is always quadratic irrespective of implementation. The paper aims to solve that.

II. Literature Review

For example, the authors in [1] used a heuristic method for computing the shortest path from one point to another point within traffic networks. They proposed a ―new dynamic direction restricted algorithm obtained by extending the Dijkstra's algorithm [1]. ‖ In another paper [2], a heuristic GA was used for solving the single source shortest path (SSSP) problem. Its main goal was to investigate the SSSP problem within the Internet routing setting, particularly when considering the cost of transmitting messages/packets is significantly high, and the search space is normally very large. In a paper by Li, Qi, and Ruan [3], an efficient algorithm named Li-Qi (LQ) was proposed for the SSSP problem with the objective of finding a simple path of the smallest total weights from a specific initial or source vertex to every other vertex within the graph. The ideas of the queue and the relaxation form the basis of this newly introduced algorithm; the vertices may be queued several times, and furthermore, only the source vertex and relaxed vertices are being queued [3].

III. Existing djikstra's algorithm

Dijkstra algorithm uses memory in terms of square. It uses space complexity of n to the power of 2 where n is the number of vertices in the graph. This causes a problem in execution when the graph becomes very dense and number of nodes increases. This is because the square value will keep increasing the memory requirement. This can eat up the execution time and space complexity. The algorithm picks one node as source node and calculates distances to every other node in the graph. It picks the vertex and sees all the vertices that are unvisited and adjacent to it. If the distance to the current vertex summed with the edge weight to the next vertex comes out to be less than the current distance then the new distance is updated. The same process is repeated for each vertex and the resultant weight of each vertex is the shortest path to that vertex from the selected source vertex.

IV. Proposed Methodology

The algorithm does not handle memory very well as explained. One of the main memory-related challenges with Dijkstra's algorithm is the need to maintain a priority queue or heap to keep track of the nodes and their distances from the source node. This priority queue is used to select the node with the minimum distance in each iteration. The memory usage of this data structure can become a problem when dealing with large graphs or when the algorithm needs to run on memory-constrained devices. When applied on extremely dense graphs, it can cause huge problem of resources. Now to a graph can have two different nodes at separate distances but they may be close to each other. Let two nodes A and B be very near to each other. Now, if node 'source' to node 'A' is x units, the path might be going from A to C then to B. This can yield the shortest path but may yield larger execution time

in real time scenarios. To understand the same lets say A and B are in same locality but C is in different locality so it won't make enough sense to move from A to C then back to B. Instead, one can cover A and B together and then move to C. Hence, the idea of clubbing nodes is used. In such an implementation, two nodes that are nearby are clubbed and seen as one node whereas the others are seen as different. In clubbed nodes' group, all the nodes are visited before moving out to any other node. The entire graph is divided into clusters and base on the threshold distance, two or more nodes are clubbed. Maximum number of nodes in a group can be 4 only otherwise the size can become very large if network consists of small edges.

V. CONCLUSION

This study rolls out a practical approach to solve a real time problem. It can save effort and time when applied to real time graph systems and can help in easing out memory load on the system. Millions of computer based instructions are executed every second so even small optimisations can propagate through the entire graph and save a big amount of time for every execution of task. In the age of cloud computing it becomes even more critical to use space wisely as it costs money as well as system power. Clubbing nodes is an attempt to solve the problem of space and let Dijkstra be used in dense graphs as well for real time applications that are not just theoretically right but practical as well.

VI. References

[1] C. Xi, F. Qi, and L. Wei, —A New Shortest Path Algorithm based on Heuristic Strategy, ‖ Proc. of the 6th World Congress on Intelligent Control and Automation, Vol. 1, pp. 2531 – 2536, 2006.

[2] B.S. Hasan, M.A. Khamees, and A.S.H. Mahmoud, —A Heuristic Genetic Algorithm for the Single Source Shortest Path Problem, ‖ Proc. of International Conference on Computer Systems and Applications, Fig. 5. Results of Testing for Dijkstra's Algorithm Fig. 6. Results of Testing for Floyd-Warshall Algorithm Fig. 7. Results of Testing for Bellman-Ford Algorithm pp. 187-194, 2007.

[3] T. Li, L. Qi, and D. Ruan, —An Efficient Algorithm for the Single-Source Shortest Path Problem in Graph Theory ‖, Proc. of 3rd International Conference on Intelligent System and Knowledge Engineering, Vol. 1 pp. 152-157, 2008

[4] L. N. Hyseni and A. Ibrahimi, "Comparison of the cloud computing platforms provided by Amazon and Google," 2017 Computing Conference, London, UK, 2017, pp. 236-243, doi: 10.1109/SAI.2017.8252109.

[5] Denny, I. Putu Medagia Atmaja, Ari Saptawijaya and Siti Aminah, "Implementation of change data capture in ETL process for data warehouse using HDFS and apache spark," 2017 International Workshop on Big Data and Information Security (IWBIS), Jakarta, Indonesia, 2017, pp. 49-55, doi: 10.1109/IWBIS.2017.8275102.

[6] M. Vijayalakshmi and R. I. Minu, "Incremental Load Processing on ETL System through Cloud," 2022 International Conference for Advancement in Technology (ICONAT), Goa, India, 2022, pp. 1-4, doi: 10.1109/ICONAT53423.2022.9726039.

[7] Z. Bin, S. Shuai, G. Zhi-chun and H. Jian-feng, "Design and Implementation of Incremental Data Capturing in Wireless Network Planning based on Log Mining," 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 2021, pp. 2757-2761, doi: 10.1109/IAEAC50856.2021.9390978.

[8] H. Chandra, "Analysis of Change Data Capture Method in Heterogeneous Data Sources to Support RTDW," 2018 4th International Conference on Computer and Information Sciences (ICCOINS), Kuala Lumpur, Malaysia, 2018, pp. 1-6, doi: 10.1109/ICCOINS.2018.8510574.

[9] F. de Assis Vilela and R. Rodrigues Ciferri, "A novel solution to perform real-time ETL process based on non-intrusive and reactive concepts," 2021 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2021, pp. 556-561, doi: 10.1109/CSCI54926.2021.00158.

[10] K. Ma and B. Yang, "Log-based change data capture from schema-free document stores using MapReduce," 2015 International Conference on Cloud Technologies and Applications (CloudTech), Marrakech, Morocco, 2015, pp. 1-6, doi: 10.1109/CloudTech.2015.7336969.

[11] Dan Sullivan, "Deploying Storage in Google Cloud Platform," in Official Google Cloud Certified Associate Cloud Engineer Study Guide , Wiley, 2019, pp.275-308, doi: 10.1002/9781119564409.ch12.

[12] Dan Sullivan, "Loading Data into Storage," in Official Google Cloud Certified Associate Cloud Engineer Study Guide , Wiley, 2019, pp.309-336, doi: 10.1002/9781119564409.ch13.

[13] J. Shah and D. Dubaria, "Building Modern Clouds: Using Docker, Kubernetes & Google Cloud Platform," 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2019, pp. 0184-0189, doi: 10.1109/CCWC.2019.8666479.

[14] A. Gupta, P. Goswami, N. Chaudhary and R. Bansal, "Deploying an Application using Google Cloud Platform," 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Bangalore, India, 2020, pp. 236-239, doi: 10.1109/ICIMIA48430.2020.9074911.

[15] C. R. Valencio, M. H. Marioto, G. F. Donega Zafalon, J. M. Machado and J. C. Momente, "Real Time Delta Extraction Based on Triggers to Support Data Warehousing," 2013 International Conference on Parallel and Distributed Computing, Applications and Technologies, Taipei, Taiwan, 2013, pp. 293-297, doi: 10.1109/PDCAT.2013.52.

[16] V. Bucur, C. Dehelean and L. Miclea, "Object storage in the cloud and multi-cloud: State of the art and the research challenges," 2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), Cluj-Napoca, Romania, 2018, pp. 1-6, doi: 10.1109/AQTR.2018.8402762.

[17] P. Matesanz, D. Buchner, A. Forderer, A. Koschel, T. Lange and I. Astrova, "ScaLib: Scalable Library System Based on Google App Engine," 2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA), Zakynthos, Greece, 2018, pp. 1-6, doi: 10.1109/IISA.2018.8633629.

[18] B. Posey et al., "On-Demand Urgent High Performance Computing Utilizing the Google Cloud Platform," 2019 IEEE/ACM HPC for Urgent Decision Making (UrgentHPC), Denver, CO, USA, 2019, pp. 13-23, doi: 10.1109/UrgentHPC49580.2019.00008.

[19] S. Challita, F. Zalila, C. Gourdin and P. Merle, "A Precise Model for Google Cloud Platform," 2018 IEEE International Conference on Cloud Engineering (IC2E), Orlando, FL, USA, 2018, pp. 177-183, doi: 10.1109/IC2E.2018.00041.

[20] A. S. Muhammed and D. Ucuz, "Comparison of the IoT Platform Vendors, Microsoft Azure, Amazon Web Services, and Google Cloud, from Users' Perspectives," 2020 8th International Symposium on Digital Forensics and Security (ISDFS), Beirut, Lebanon, 2020, pp. 1-4, doi: 10.1109/ISDFS49300.2020.9116254.