International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 05 | May - 2023 Impact Factor: 8.176 ISSN: 2582-3930

An Improved Shortest Path Campus Navigation Algorithm

Dr. Ankita Gupta
Assistant Professor, *Computer Science Engineering*
Maharaja Agrasen Institute of Technology
Rohini Sector-22, New Delhi, India

Mayank Arya
B.Tech *Computer Science Engineering*
Maharaja Agrasen Institute of Technology
Rohini Sector-22, New Delhi, India

Rishabh Garg
B.Tech *Computer Science Engineering*
Maharaja Agrasen Institute of Technology
Rohini Sector-22, New Delhi, India

Ashish Prakash
B.Tech *Computer Science Engineering*
Maharaja Agrasen Institute of Technology
Rohini Sector-22, New Delhi, India

Abstract: Navigation within college campuses can be a challenging task for students, staff, and visitors. With the increasing reliance on smartphones as a common utility, it is possible to develop a dedicated navigation system to aid in navigating these complex environments. In an effort to improve upon traditional navigation methods, we explored the use of the A* algorithm and identified an opportunity to enhance its performance by implementing bidirectional search. This allows the algorithm to search simultaneously from both the start and end points, reducing the search space and improving efficiency. Additionally, we utilized a caching mechanism to store previously calculated paths, which allows the algorithm to reuse these values and further improve its performance. Our approach offers a unique solution to the problem of navigating on college campuses and has the potential to significantly improve the user experience.

1. INTRODUCTION

With an increasing rate of college-related activities, be it admission, fests, events, campus placement drives, etc organized by universities, it is difficult for new students to navigate the college campus by themselves. Even students who have been attending the college for a couple of years aren't well-versed in the layout of the campus. Students have a hard time seeing the complete picture of the college campus in detail. The usual solution to this is to ask others for directions, taking the help of landmarks made popular by students (canteens, grounds, library, etc) to figure out where they are. Many colleges and universities try to help students by displaying a map of the campus, using sign boards, and directions to blocks in the campus but they often aren't used in general by the students. Students prefer their sense of direction to their most visited locations on the campus but struggle to reach a new location.

At present, the navigation tools students can use are applications like Google Maps, Apple Maps, etc. But in a relatively small place like a college campus, they are not precise enough to show the route which the students prefer to take. The routes taken by students include paths that are not listed even on college maps since students tend to take the shorter route rather than the given route. In addition to this, areas of importance inside the campus aren't listed on applications like Google Maps. Block Numbers, their respective names, canteens, and stationeries aren't listed.

This research aims to develop a more accurate and efficient method of navigation within a college campus. By conducting a small-scale study on navigation techniques, we aim to provide a practical solution that will assist students, teachers, and visitors in quickly reaching their destinations. Our goal is to improve the overall user experience on the college campus and make it easier for individuals to navigate and find their way around.

Dijkstra's Algorithm [1] is a popular method for finding the shortest path in a simple, weighted graph with non-negative edge weights. It begins at the specified source node and expands outward to reachable nodes. While Dijkstra's Algorithm is guaranteed to produce correct results, it can be inefficient due to its blind search approach, which may result

in the traversal of many unnecessary nodes. To address this issue, some researchers have incorporated heuristic information into the algorithm to guide the search and reduce the number of nodes that need to be traversed. This can improve the time and space complexity of the algorithm. An alternative approach is the bidirectional A* algorithm [2] , which searches simultaneously from both the source and destination nodes. This results in the use of half the space required by the unidirectional A* algorithm [3] and a reduction in running time.

2. Literature survey (navigation system)

More recent campus navigation systems have utilized a variety of technologies, including GPS, Bluetooth, and QR codes, to provide real-time location information and turn-by-turn directions to users. These systems have been implemented in a range of institutions, including universities, community colleges, and technical schools.

Research has shown that campus navigation systems can have a number of benefits, including improving the efficiency and effectiveness of wayfinding, reducing the time and effort required to find one's way around campus, and increasing the overall satisfaction of students and faculty with the campus environment. However, there are also potential drawbacks to these systems, such as the cost of implementation and maintenance, and the need for ongoing updates to keep the information accurate.

There has also been research on the use of augmented reality (AR) in campus navigation systems. AR systems use a device's camera to superimpose digital information onto the physical environment, providing users with additional context and guidance as they move around campus. While AR campus navigation systems are still in the early stages of development, they hold promise as a way to enhance the user experience and make it easier for people to find their way around unfamiliar environments.

3. Methodology

i) Fundamentals of A* Algorithm

The basic implementation of the A* Algorithm is the use of informed search instead of blind search used in Dijkstra's Algorithm.

$$f(n) = g(n) + h(n)...(1)$$

f(n) is the evaluation function from the source node to the destination node, g(n) is the actual cost of reaching the target node and h(n) refers to the heuristic value which represents the estimated cost.
In our case, the heuristic value/ estimate can be the euclidean distance between two nodes. This will help generate the optimal solution and won't include nodes that stray further away from the destination in the computation overhead. Use of the heuristic search results in a significant efficiency boost at the backend of the application.

ii) Concept of Bi-directional A*

Bidirectional A* is the modification made to the unidirectional A* algorithm, which initiates a search from the destination node. When the forward search and backward search reach the same node, the search stops. Applying the A* search algorithm from both source and destination improves the time complexity as well as space complexity as compared to a unidirectional A* search.

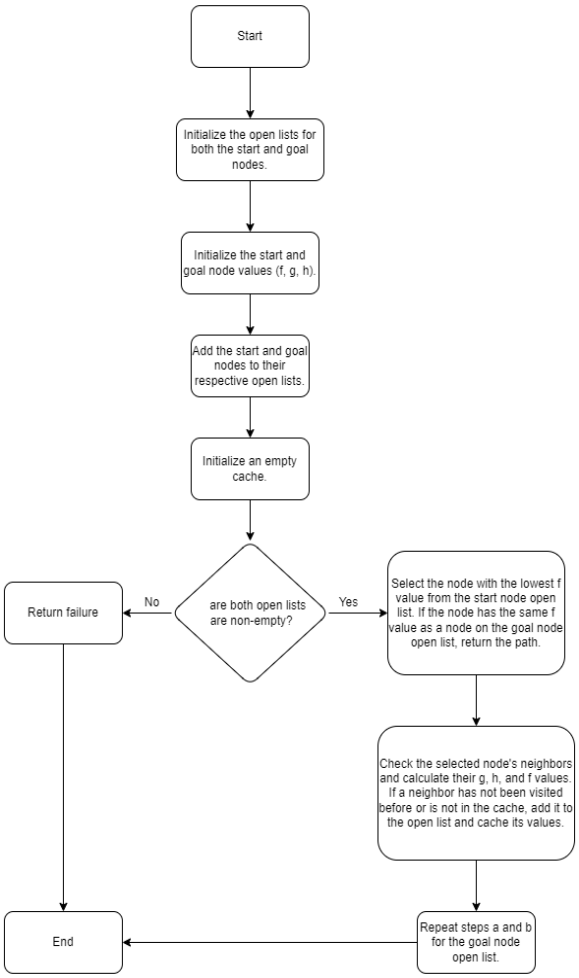

Figure 1. Flow Chart for Improved Algorithm

4. Philosophy and Initialization of the improved algorithm

As we progressed from Dijkstra's Algorithm to unidirectional A* and eventually to the bidirectional A* algorithm, the run time and space complexity of the algorithms improved. In our proposed navigation algorithm, we plan to incorporate the benefits of the bidirectional A* algorithm and add a caching mechanism to further improve performance. The caching mechanism will be used to handle the volume of requests received by the server, helping to reduce the workload and improve the speed of the algorithm. Overall, this approach is expected to provide a more efficient and effective solution for navigation within a college campus.

One way to improve the bidirectional A* algorithm by using a caching mechanism is to store the values of previously visited nodes in a hash table or dictionary. This can help reduce the number of nodes that need to be recalculated by the algorithm, which can significantly improve its performance.

Here is a modified version of the bidirectional A* algorithm that makes use of a caching mechanism:

1. Initialize the open lists for both the start and goal nodes.
2. Initialize the start and goal node values (f, g, h).
3. Add the start and goal nodes to their respective open lists.
4. Initialize an empty cache.
5. While both open lists are non-empty:

a) Select the node with the lowest f value from the start node open list. If the node has the same f value as a node on the goal node open list, return the path.

b) Check the selected node's neighbors and calculate their g, h, and f values. If a neighbor has not been visited before or is not in the cache, add it to the open list and cache its values.

c) Repeat steps a and b for the goal node open list.

6. If the open lists are empty and no path has been found, return failure.

This modification allows the algorithm to reuse the values of previously visited nodes, which can greatly improve its performance in cases where the same paths are accessed multiple times.

The algorithm will take the help of five global tables, namely: initSrc, and initDest which will keep track of nodes that have already been generated without inspection. Table endSrc and endDest save the nodes already visited in the forward and backward search. Table pathsCache is used to save the request of paths.

The idea is to look for the path in the pathsCache first. If the path is present already, there will not be a need to recompute it. The application will return the path and update the path's priority. Since the cache will have limited space, cache replacement techniques will need to be implemented. For

this, the LRU cache [4] replacement method will be implemented. Every time an existing path in the cache is referred, it will have a low chance of being replaced.

If the path is not available in the cache,
1) Determine whether the source node can reach the destination node directly. If possible, a path will be inserted into the pathsCache, replacing a suitable victim selected by the LRU scheme[7] and the path will be returned to the request.

2) Otherwise clear the tables initSrc, initDest, endSrc, and endDest and put the target node into the initDest list.

3) If the search is forward (backward) and the initial source (destination) node is not null, select the node with the smallest f value in the initial source (destination) table. If this node is present in the destination (source) table, the path can be formed by moving both the starting and target nodes in opposite directions along the father node until they meet at the selected node. This path is then stored in the cache using the LRU (Least Recently Used) replacement algorithm. The final path is then returned.

5. Result and Analysis

For the result and analysis, the algorithms and their performance was tested on the prototype map of a college campus.
Figure 1 shows the working screenshots of the application used on a smartphone device. Table 1 compares the execution times of the paths computed using different algorithms including Dijkstra, unidirectional A*, bidirectional A*, improved A*.

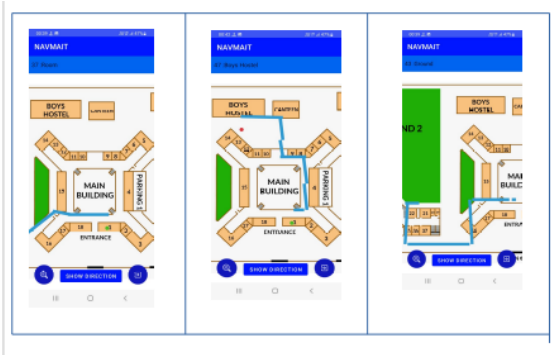

Figure 2. Effect picture of running software

Request Time	Dijkstra Algo	Unidirectional A*	Bidirectional A*	Improved A*
	Time	Time	Time	Time
1	258706	435	303	201
10	71571	953	391	236
100	125538	390	359	299

Table 1. Test of path algorithm and its comparison

6. Conclusion

It is generally accepted that the bidirectional A* algorithm is more efficient than the unidirectional A* algorithm, as it allows the search to start from both the source and destination nodes simultaneously, reducing the search space by half.

Furthermore, implementing a caching mechanism to store the most frequently visited paths can further improve the performance of the bidirectional A* algorithm by allowing it to reuse previously calculated values and reducing the number of nodes that need to be recalculated.

Overall, the improved navigation algorithm that is a bidirectional A* algorithm with a caching mechanism is likely to be more efficient than both the unidirectional A* algorithm and Dijkstra's algorithm, as it combines the benefits of both bidirectional search and caching, resulting in a faster and more efficient pathfinding algorithm.

7. References

1. *He, Baoyi. "Application of Dijkstra algorithm in finding the shortest path." Journal of Physics: Conference Series. Vol. 2181. No. 1. IOP Publishing, 1957.*
2. *Pijls, Wim; Post, Henk. Yet another bidirectional algorithm for shortest paths (PDF) (Technical report). Econometric Institute, Erasmus University Rotterdam. EI 2009-10.*
3. *Hart, P. E.; Nilsson, N.J.; Raphael, B. (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". IEEE*
4. *Transactions on Systems Science and Cybernetics. 4 (2): 100–7. doi:10.1109/TSSC.1968.300136.*

5. *J. Ousterhout, "A Memory-Efficient, High-Performance Cache," ACM Transactions on Computer Systems, vol. 2, no. 4, pp. 307-321, Nov. 1984.*
6. *Basumatary, Dipali, Swapna Rawat, and Ranjan Maity. "Exploring the Campus of a University—An AR-Based Application—"Drishti"." Proceedings of the 3rd International Conference on Communication, Devices and Computing. Springer, Singapore, 2022.*
7. *S. Dutta, M. S. Barik, C. Chowdhury and D. Gupta, "Divya-Dristi: A Smartphone based Campus Navigation System for the Visually Impaired," 2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT), 2018, pp. 1-3, doi: 10.1109/EAIT.2018.8470397.*
8. *Hou, Jiacheng, et al. "A Graph Neural Network Approach for Caching Performance Optimization in NDN Networks." IEEE Access 10 (2022): 112657-112668.*
9. *Pagare, Akshay S., et al. "An Event Driven Campus Navigation System On Android Platform." International Journal of Advances in Scientific Research and Engineering. Retrieved from http://www. ijasre. net/uploads/1/2833_pdf. pdf (2017).*
10. *R. Al-Jumaili, M. Al-Shabi, and N. Al-Shabi, "A Review of Indoor Navigation Systems for Campus Environments," IEEE Access, vol. 9, pp. 83052-83068, 2021.*
11. *Bacchewar, Y., Morwadkar, S., Chandegave, R., Dendage, P., & Dhamgunde, S. (2023). Literature Survey: Indoor Navigation Using Augmented Reality. In International Conference on Data Management, Analytics & Innovation (pp. 387-400). Springer, Singapore.*
12. *R. A. El-Sayed, M. A. El-Sayed, and M. M. Abo-Zahhad, "A Review of GPS Map-Matching Algorithms," IEEE Access, vol. 8, pp. 101248-101263, 2020.*
13. *bin Sabri, Syahier Aqif, et al. "Development of a Mobile Application for Room Booking and Indoor Navigation." International Conference on Information Systems and Intelligent Applications. Springer, Cham, 2023*
14. *Tudpor, Kukiat, et al. "Geographic Information System-Based Mobile Application Design for Health Care in Older Persons in Rural Community by Village Health Volunteers." Stud. Health Technol. Inform 289 (2022): 426-429.*
15. *Gao, Meirong. "Smart campus teaching system based on ZigBee wireless sensor network." Alexandria Engineering Journal 61.4 (2022): 2625-2635.*
16. *Jana, Susovan, and Matangini Chattopadhyay. "An event-driven university campus navigation system on android platform." 2015 Applications and Innovations in Mobile Computing (AIMoC). IEEE, 2015.*
17. *ElAlAMI, M. E., IBRAHIM A. ElSHAFEI, and HANAN E. ABDELKADER. "LOCATION-BASED SERVICES USING WEB-GIS BY AN ANDROID PLATFORM TO IMPROVE STUDENTS'NAVIGATION DURING COVID-19." Journal of Theoretical and Applied Information Technology 100.10 (2022)*
18. *A. C. Raza, A. M. Al-Dubai, and M. A. Imran, "A Review of Indoor Navigation Systems for Campus Environments," IEEE Access, vol. 8, pp. 24159-24179, 2020.*
19. *M. K. Hasan, A. M. Al-Dubai, and M. A. Imran, "A Survey of Campus Navigation Systems," IEEE Access, vol. 9, pp. 24441-24460, 2021.*
20. *R. A. Al-Jumaili, M. Al-Shabi, and N. Al-Shabi, "A Review of Campus Navigation Systems," IEEE Access, vol. 9, pp. 83052-83068, 2021.*
21. *H. Yang, J. Chen, and L. Hanzo, "Survey of Indoor Navigation Systems for Campus Environments," IEEE Access, vol. 9, pp. 90922-90940, 2021.*