Sri Tarun Gulumuru
AI 44800
Yunhua Zhao
11/11/2023

AI MIDTERM

A Comprehensive Exploration of Pathfinding Algorithms and Recent Advances

Abstract
Efficient navigation in artificial intelligence applications, such as robotics and gaming, remains a fundamental challenge. Pathfinding algorithms, notably Dijkstra's, Greedy Best-First Search, and A*, represent crucial components in addressing this challenge. These algorithms, categorized into heuristic and non-heuristic approaches, offer unique perspectives on optimality, efficiency, and heuristic intuition. A comprehensive examination of the A* algorithm reveals its strength in balancing efficiency and optimality through the integration of actual costs and heuristics. Despite computational expenses and reliance on appropriate heuristics, A* is widely implemented in real-life applications, including robotic path planning and logistics. Dijkstra's algorithm, known for its optimality and reliability, is extensively used in geographical mapping and telecommunications. Greedy Best-First Search prioritizes speed over optimality, making it suitable for scenarios like emergency response planning and real-time robotic exploration. The comparative analysis showcases scenarios where each algorithm excels, emphasizing A*'s dominance in real-time, dynamic environments.

Background
The ability to navigate efficiently from one point to another is a fundamental challenge, in the realm of artificial intelligence, within a multitude of applications. Whether it is a robot maneuvering through a dynamic environment or a gaming character exploring a virtual world, the ability to find optimal paths is significant. Pathfinding algorithms are the guiding principles that help in finding optimal routes through complex environments. Three key algorithms- Dijkstra Algorithm, Greedy Best-First search and A*- stand out as cornerstones in the journey of pathfinding. Each of these algorithms offer a unique perspective on optimality, efficiency and heuristic intuition [5].

The two broad categories of these pathfinding algorithms are heuristic and non-heuristic approaches that offer diverse methodologies in pathfinding. Non heuristic approaches such as Dijkstra’s algorithm and Breadth First Search provide systematic exploration to the forefront that allows optimality in determining shortest paths. On the other hand, heuristic approaches like Greedy Best-First Search and the A* algorithms offer a level of intuition. This allows for faster decision making while balancing efficiency and optimality

As we delve into a comprehensive examination of A*, Dijkstra's, and Greedy Best-First Search, the distinctive attributes of each algorithm will be revealed, providing insights into their applications, strengths, and complexities within the intricate domain of pathfinding in artificial intelligence.

A* Algorithm
The "A" stands for "Admissible" as it uses an admissible heuristic to estimate costs. The "*" signifies that it combines actual and estimated costs to make informed decisions during the search process.

A* algorithm is a pathfinding algorithm that is designed to find the shortest path between two points, initial and final state. To illustrate, consider a roadtrip from A to B, A* assists in

navigating the optimal path possible that ensures the shortest amount of time to reach B from A [7].

A* algorithm considers both the actual cost incurred to reach a point and a heuristic estimate of the remaining distance to the destination. A* algorithm consists of three parameters: “f= g + h”. The g represents the cost incurred or time taken to reach the point. The h represents the heuristic estimate of the remaining distance to destination; heuristic estimate is an educated guess on time it takes to reach destination. In conclusion, f gives the total time, shortest in the case of A* algorithm to reach destination[7].

Strengths of A algorithm*
The use of appropriate heuristics guarantees an optimal path. Some of the most commonly employed heuristics are Manhattan distance and Euclidean distance. Euclidean distance finds the straight line distance between two points of a geometric space. The best use of this heuristic is pathfinding in a grid where diagonal movement is allowed, such as finding the shortest route on a map with roads connecting locations(pedestrians have leeway to move diagonally through open space & intersections). On the other hand, Manhattan distance provides the sum of horizontal and vertical distances between two points, following right angle movements. The best use of this heuristic is pathfinding where movement is limited to up, down, left and right.

Furthermore, the algorithm is efficient with the capability of handling large search spaces due to its effective pruning of unpromising paths. The heuristic parameter of the algorithm helps get rid of bad paths, and the cost parameter that holds cost up until the current point/node makes the algorithm more efficient in finding shortest path.

Weakness of A algorithm*
The algorithm is computationally expensive in scenarios where the search space is big and the number of possible paths is large. This is because of its application not just one parameter but two parameters: cost and heuristic in finding optimal paths. This is also the reason for its significant memory and resource consumption.

Additionally, the algorithm is highly reliant on the appropriate heuristic. The application of a poorly designed heuristic or a heuristic that does not accurately estimate the distance to the goal will compromise optimality and performance of the algorithm. Also, A* algorithm struggles with certain types of search spaces that exhibit irregular or unpredictable structures.

Overall, the advantages of A* algorithm overshadows the disadvantages of its usage. Therefore, the algorithm is widely implemented and has a vast amount of resources and support available. Moreover, the algorithm can be tailored to target different problem domains and heuristics. Some of the real life applications of this algorithm are in robotic path planning and in logistics & transportation. In robotic path planning, A* helps navigate through dynamic environments while avoiding obstacles. Additionally, A* algorithm assists

in transportation by providing the optimal routes for delivery vehicles, minimizing the time and cost.

Dijkstra Algorithm

Dijkstra Algorithm is a search algorithm that generates every single route through the path and then selects the path with the lowest overall cost. It is one of the single source shortest path algorithms. This algorithm is named after computer scientist Edsger Dijkstra, who developed it in the 1950s[15].

The algorithm works with the help of a priority queue of nodes. This priority queue continually selects the node with the shortest cumulative distance and continues to expand the neighbors visited. This process continues until all the nodes are visited and provided with the shortest distance and path from the starting node[3].

Strengths of Dijkstra algorithm
The key advantage of this algorithm is the low complexity that is almost linear. The complexity is O(V+E.log(V)), where V is the number of nodes and E is the number of edges in the graph. As a result, it is widely used in google maps. Google Maps uses Dijkstra's Algorithm for finding the shortest paths between two nodes in a graph. For example, it helps find shortest paths between two major cities taking into consideration barriers and other ground realities like rivers or mountain ranges. This gives an accessible shortest route between the two cities. Therefore, this algorithm is widely used in geographical mapping systems[6].

The algorithm's optimality in reliably finding shortest paths to all nodes in a graph makes it an optimal choice for IP routing. Open Shortest Path First (OSPF) is an IP routing protocol that utilizes the Dijkstra algorithm. The OSPF calculates with AS as a starting point the shortest path for all routers in the area; this helps efficiently use network bandwidth, allowing scalability. Similarly for the same reason, it is used in telecommunications to establish connections[2].

Limitations of Dijkstra algorithm
The algorithm cannot handle negative edge weights in the pathfinding. This is the most notable limitation of this algorithm. A real world example of this is in financial systems; negative weights are used to represent losses and vice versa. This is used to maximize profits. However, the Dijkstra algorithm fails to shine in this scenario. Additionally, the algorithm's inefficiency for dense graphs, where the number of edges approaches the square of the number of nodes, poses a drawback[8].

Overall, the Dijkstra algorithm is the most widely used pathfinding algorithm. The algorithm owes this to its ease of implementation and understanding for a broader audience. Despite the various limitations, the algorithm shines with its optimality and reliability in producing

shortest routes to all nodes in a path. This makes it suitable for various real life scenarios like telephone networks and social networking applications[9].

Greedy Best First Search

Greedy Best First Search is an informed search algorithm employed in pathfinding and problem solving. The algorithm strictly relies on a heuristic function and disregards edge weights unlike the A* algorithm and Dijkstra algorithm. This approach rests on the assumption that it is likely to lead to a solution quickly[1].

The function of this algorithm is "f = h". The evaluation function is equal to the heuristic function. The h or heuristic finds the successive node based on how close it is to the destination. In simple words, it finds the immediate low cost option[1].

Limitations of Greedy Best First Search
The Greedy Best First search in a maze with dead ends exhibits weakness. In a scenario where the heuristic is straight line distance to the goal, misleading paths that are short according to heuristic lead to dead ends. Therefore, a short dead end path in a long distance maze might look like a promising path; unnecessary dead ends are often visited before the optimal path is reached.

The solution from this algorithm may not be optimal since a shorter path may exist. This is because of its greedy approach of choosing immediate low cost options based on the heuristic employed.

Overall, Greedy Best-First Search, by prioritizing immediate gains based on heuristic estimates, can be misled by misleading paths, especially in scenarios where the heuristic does not accurately represent the actual cost to reach the goal.

Strengths of Greedy Best First Search
On the other hand, there are many effective scenarios for Greedy Best-First search. We delve into the intricacies of Greedy Best-First Search, examining scenarios where its efficiency outweighs its limitations.

Emergency Response planning is when immediate action is required during emergency situations like disaster response. Greedy Best First search is the best in this scenario as it prioritizes a heuristic that reflects the urgency of reaching certain locations and as a result leads to quickest assistance.

Furthermore, in robotic real time exploration, the goal is to quickly survey an unknown environment. This is a best scenario for the Greedy Best-First search due to its myopic nature; the algorithm moves swiftly through promising paths based on the heuristic values.

This allows for quick decision making and quick exploration. However, in the case for A* algorithm which makes both costs and heuristics into account, it might take more time evaluating paths comprehensively.[4]

Overall, Greedy Best-First Search is like a fast decision-maker. It's really quick in exploring and picking paths that seem good right away. This makes it great for scenarios where speed is more important than finding the absolute best path. It might not always find the overall best solution, but it's excellent for quickly checking out different options.

Comparative Analysis

The Djikstra algorithm without any heuristic excels in scenarios with non-negative edge weights, providing an optimal solution for determining the shortest path for all nodes in the domain. A good example is IP catalog routing like OSPF. The other two algorithms do not accurately align with assistance in this scenario. The Greedy Best First search does not necessarily provide an optimal path. On the other hand, A* algorithm is targeted more towards the most optimal path to destination rather than finding the optimal path to every node.

Greedy Best First Search takes a heuristic centric approach. It is the best example of intuition over exhaustive exploration. This algorithm is most suitable for speed over optimality. This is because it usually leads to suboptimal paths due to its penchant for local optimization. A good real life application of this algorithm is in Emergency Response planning where immediate action is required rather than the most optimal action. The other two algorithms however are time consuming to be of assistance in this scenario.

A* algorithm evaluates based on sum of actual and estimated costs, so it relies both on cost and heuristics. Therefore, it is a balance of efficiency and optimality, and it is a backbone in AI applications. A real life situation that the other two algorithms fail but A* shines is in a scenario, where the road network is presented as a graph with different weights assigned to edges based on factors like traffic congestion, road conditions, and travel time. The goal is to find the shortest path while considering both distance and real-time factors affecting the journey. The intuition and actual costs of this algorithm make it a balance of efficiency and optimality and the perfect choice. Djikstra is shadowed by A* in this scenario due to its blind search which can be time consuming and exhaustive as it explores all the nodes. On the other hand, Greedy Best First search might result in suboptimal paths.

Recent advances

Pathfinding algorithms have recently witnessed a convergence of artificial intelligence and machine learning, especially in the realm of reinforcement learning for pathfinding[10]. Pathfinding strategies that implement machine learning models, including neural networks, into pathfinding methodologies allows dynamic adaptation and optimized navigation based on real time information and historic data.

Additionally, Swarm intelligence algorithms, inspired by natural swarms, are gaining prominence for their decentralized and adaptive decision making capabilities. Ant colony optimization and particle swarm optimization are two of the swarm intelligence algorithms[11].

Furthermore, researchers are exploring hybrid approaches that combine the strengths of traditional pathfinding algorithms like A* and Dijkstra[12].

Overall, the focus has shifted towards handling dynamic and high-dimensional environments that are optimal for real time information and scalability. Furthermore, the advances in multi agent pathfinding and integrated edge computing, for faster decision making, represent the evolution of pathfinding algorithms[13][14].

Conclusion

Overall, in the complex domain of pathfinding in artificial intelligence, A*, Greedy Best First Search and Dijkstra emerge as adaptable tools, each offering a different balance of efficiency and optimality. A* stands out for its adaptability to dynamic scenarios, relying on both cost and heuristics. Dijkstra excels in non-negative edge weight scenarios, finding widespread use in geographical mapping and telecommunications. Greedy Best-First Search, though myopic, proves invaluable in scenarios requiring swift decision-making. Understanding the strengths and limitations of each algorithm allows for informed choices, tailoring solutions to specific AI applications and problem domains. Furthermore, recent advancements signal a convergence of artificial intelligence and machine learning in pathfinding, introducing adaptive strategies and swarm intelligence algorithms. The evolving landscape emphasizes the importance of addressing dynamic and high-dimensional environments, optimizing for real-time responses and scalability. The ongoing research in multi-agent pathfinding and edge computing integration further underscores the continuous development and applicability of pathfinding algorithms in diverse AI scenarios.

References

1) *AI: Search algorithms: Greedy best-first search*. Codecademy. (n.d.). https://www.codecademy.com/resources/docs/ai/search-algorithms/greedy-best-first-search

2) Awati, R., & Burke, J. (2023, October 30). *What is Open shortest path first (OSPF)?: Definition from TechTarget*. Networking. https://www.techtarget.com/searchnetworking/definition/OSPF-Open-Shortest-Path-First

3) Cox, W. by: G. (2022, November 15). *Overview of dijkstra's algorithm*. Baeldung on Computer Science. https://www.baeldung.com/cs/dijkstra#:~:text=Dijkstra's%20Algorithm%20is%20a%20pathfinding,we%20reach%20the%20end%20node.

4) GeeksforGeeks. (2023, April 4). *Greedy best first search algorithm*. GeeksforGeeks. https://www.geeksforgeeks.org/greedy-best-first-search-algorithm/

5) H.urna, H.-. (2020, January 20). *Pathfinding algorithms : The four pillars.* Medium. https://medium.com/@urna.hybesis/pathfinding-algorithms-the-four-pillars-1ebad85d4c6b

6) Lanning, D., Harrell, G.K., & Wang, J. (2014). Dijkstra's algorithm and Google maps. Proceedings of the 2014 ACM Southeast Regional Conference.

7) S, R. A. (2023, October 17). A* algorithm in artificial intelligence you must know in 2023: Simplilearn. Simplilearn.com. https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/a-star-algorithm#:~:text=It%20is%20a%20searching%20algorithm,can%20find%20its%20own%20course.

8) Sryheni, W. by: S. (2022, November 17). *Dijkstra's vs bellman-ford algorithm*. Baeldung on Computer Science. https://www.baeldung.com/cs/dijkstra-vs-bellman-ford

9) Understanding dijkstra algorithm: History, working, advantages, disadvantages, Applications & Complexity. Testbook. (n.d.). https://testbook.com/gate/dijkstra-algorithm-notes

10) Silver, D., et al. (2016). "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm." Nature. [DOI: 10.1038/nature24270]

11) Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). "Swarm Intelligence: From Natural to Artificial Systems." Oxford University Press.

12) Harabor, D., & Grastien, A. (2011). "The Online Any-angle Path Planning Problem." In Proceedings of the AAAI Conference on Artificial Intelligence. [DOI: 10.5555/2909428.2909584]
Multi-Agent Pathfinding:

13) Silver, D., & Hubert, T. (2018). "A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-Play." Science. [DOI: 10.1126/science.aar6404]
Edge Computing for Pathfinding:

14) Wu, Y., Ding, S., & Liang, Y. (2020). "Edge Computing Empowered Internet of Things: A Review." IEEE Access. [DOI: 10.1109/ACCESS.2020.3019972]

15) Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs." Numerische Mathematik, 1(1), 269–271.