

University of Alabama in Huntsville

LOUIS

Honors Capstone Projects and Theses

Honors College

4-28-2024

A Comprehensive Review and Practical Applications of Pathfinding Algorithms

Emily Rose Schoener

University of Alabama in Huntsville

Follow this and additional works at: <https://louis.uah.edu/honors-capstones>

Recommended Citation

Schoener, Emily Rose, "A Comprehensive Review and Practical Applications of Pathfinding Algorithms" (2024). *Honors Capstone Projects and Theses*. 914.

<https://louis.uah.edu/honors-capstones/914>

This Thesis is brought to you for free and open access by the Honors College at LOUIS. It has been accepted for inclusion in Honors Capstone Projects and Theses by an authorized administrator of LOUIS.

A Comprehensive Review and Practical Applications of Pathfinding Algorithms

by

Emily Rose Schoener

An Honors Capstone
submitted in partial fulfillment of the requirements
for the Honors Diploma

to

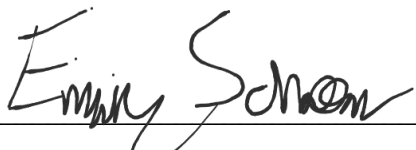
The Honors College

of

The University of Alabama in Huntsville

April 28th, 2024

Honors Capstone Project Director: Mr. Jason Sebastian, Adjunct Lecturer



4/9/24

Student (signature)

Date



4/10/2024

Project Director (signature)

Date

Department Chair (signature)

Date

Honors College Dean (signature)

Date



Honors College Frank Franz
Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)

Honors Thesis Copyright Permission

This form must be signed by the student and submitted with the final manuscript.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Emily Schoener

Student Name (printed)

Emily Schoener

Student Signature

4/9/24

Date

Table of Contents

Abstract	1
Introduction	2
Pathfinding Algorithms	2
Applications	5
Terrain Navigation	6
Robot Navigation	9
Three-Dimensional Navigation	11
Video Games	12
Human Patterns	14
Transportation	15
Electrical Network Pathing	17
Security	18
Loss Location	19
Conclusion	20
Reference List	21

Abstract

Pathfinding algorithms have many uses in modern society. Identifying and learning more about them may give us insight into their uses. Dijkstra's and A* are standard algorithms frequently used in pathfinding. Algorithms like these can be applied in many different ways, from finding the shortest path on a two-dimensional (2D) or three-dimensional (3D) map to helping robots navigate unknown environments. They can also be used in video games, predicting human patterns, transportation, electrical network pathing, security, and loss location. Learning more about these algorithms and their applications will help us identify more uses for them in other fields.

Introduction

Pathfinding is a concept many people have probably heard of when using a Global Positioning System (GPS). Pathfinding algorithms search a graph to find the most efficient path to the destination. Once you understand the basics of these algorithms, you can find many different ways to apply them. Pathfinding algorithms can be found in various applications across many career fields, from GPS to network traffic routing. Identifying multiple implementations will also assist us in understanding new uses or enhancements of these algorithms. Pathfinding algorithms are crucial in various fields, such as robotics, video games, transportation, and terrain navigation, as they efficiently determine the optimal routes within complex environments. Understanding these uses will help us learn new ways to apply these impressive algorithms.

Pathfinding Algorithms

Many pathfinding algorithms exist, but Dijkstra's and A Star (A*) are the most well-known. Pathfinding algorithms are commonly used to find the shortest path, but many variations of these algorithms have different uses.

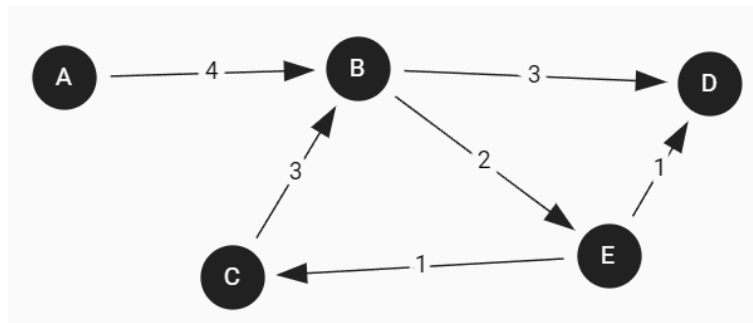


Figure 1. Representation of a directed weighted graph.

To use these algorithms, a graph must be created that represents the problem space within which you are trying to find the shortest path. These graphs consist of nodes (destinations) and edges connecting the nodes. Weights and directions are then placed on the edges. The weights represent how “costly” it will be to travel from one node to the next. The directions represent which way you can move from one node to another. Figure 1 is an example of a directed non-negative weighted graph. So, if you were to travel from node A to B to D, the cost of that path would be 7. Graphs used in pathfinding algorithms are also given a start node and a target node to identify where the path starts and ends.

Dijkstra’s algorithm is one of the more simple pathfinding algorithms that use a directed weighted graph. It is used to find a path with the lowest cost from the start node with every other node, and then when it reaches the target node, it has the shortest path.

```
Dijkstra(Graph graph, Node start, Node target)
  for all nodes in the graph
    node.cost = num

  start.cost = 0
  start.visit = true
  visitCount = 1

  while visitCount < graph.size
    Node min = minUnvisited(Graph graph)
    for each neighbor of current node
      if !neighbor.visit
        neighbor.cost = min(neighbor.cost, min.cost + edge)
    min.visit = true
    visitCount++;
  return target.cost
```

Table 1. Pseudocode for Dijkstra’s Algorithm

As the pseudocode above shows, the algorithm visits every node on the graph before giving the lowest cost for the target node. This can be costly, so many have tried to find new ways to make pathfinding more efficient.

A* is another standard pathfinding algorithm. A* starts out the same way as Dijkstra's with a graph, a start node, and a target node, but it differs in the algorithm. In A*, a heuristic function estimates the cost to the target node. Heuristics are meant to be close to the actual cost of reaching the target. Heuristics differ, but in this example, we use Euclidean distance, which is the measurement of straight line distances, and in A*, it represents the distance from the current node to the target node.

```

A*(Graph graph, Node start, Node target)
  PriorityQueue pq // f(n) = cost(n) + heuristic(n)
  for each node in graph
    pq.push(node)

  set start.cost = 0
  set start.f = start.h

  while !pq.empty()
    Node current = pq.front()
    current.visit = true
    pq.pop()
    if current == target
      return

    for each neighbor of current node
      if !neighbor.visit()
        g = weight of edge + current.g
        if g < neighbor.g
          neighbor.g = g
          neighbor.f = g + neighbor.h
  
```

Table 2. Pseudocode for A* algorithm

In the example above, a priority queue is used with the heuristic and cost so that the popped node has the lowest cost. A* also has an explicit return statement when reaching the target node, which typically causes faster times than Dijkstra's. In both algorithms, there also needs to be a function to give the shortest path in nodes. This is usually a list of previous nodes reread to make a forward path.

A study done by Noori, Azad, and Farzad Moradi examined the efficiency of specific algorithms in games, specifically in informed and uninformed search algorithms (2015, 3). An informed search would require you to know the destination of the target node before running the algorithm. A* is an example of an informed search because you need to know where the target is for the heuristic to work. Uninformed doesn't need the target's destination because the search is performed in all directions, which can be time-consuming. Dijkstra's algorithm is an example of an uninformed search algorithm. The authors split the results into execution time, the number of nodes traversed, and the total length of the path found. They found that both Dijkstra and A* had found a path of the same length, that is, the shortest path. A* had fewer nodes traversed overall to find the path (2015, 7).

You will likely get the shortest path whether you use Dijkstra's algorithm or A*. We make changes and optimizations as more research is done on these algorithms, and with new pathfinding algorithms introduced over time, the ways we can apply them increase. The more we learn about these algorithms, the easier to apply them to situations we never thought of before.

Applications

Using our knowledge of pathfinding algorithms, we can now apply them in many different situations. Commonly, we would use pathfinding for GPS or terrain navigation to find the shortest and most efficient path. This pathfinding can also apply to robotics and video game development. Taking that further can lead to three-dimensional (3D) navigation for underwater applications. There are other ways to apply these algorithms as well. If you modify the algorithms, you can use them to determine human patterns, electrical network pathing, and even

security. Understanding these applications in depth may give rise to new ideas for applying these pathfinding algorithms.

Terrain Navigation

Pathfinding algorithms are commonly used to navigate two-dimensional (2D) maps. However, they can be applied in many different ways. We use them in our everyday lives with GPS navigation, but if you modify the algorithm even slightly, the application can change.

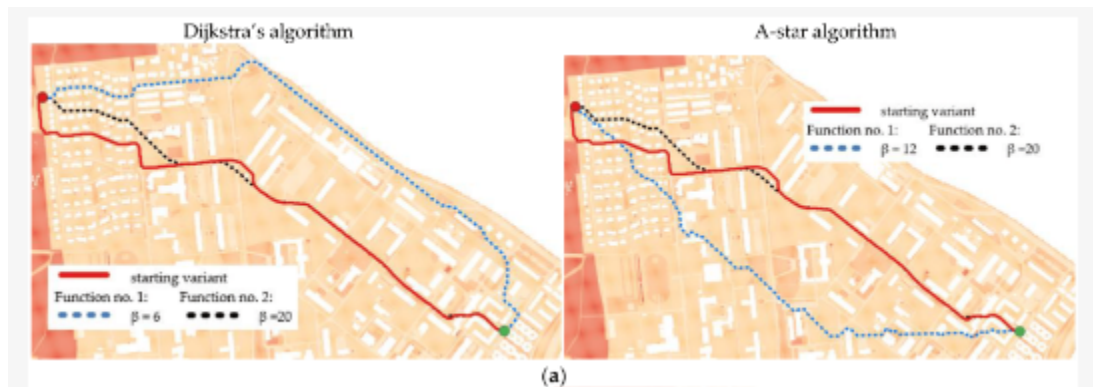


Figure 2. Optimal routes in an urban area (Dawid, Wojciech, and Krzysztof Pokonieczny, 2021).

Dawid, Wojciech, and Krzysztof Pokonieczny sought to use pathfinding algorithms specifically for military operations. Determining a route through dangerous territory is critical in these operations to ensure the safety of our troops. This application uses a map based on the Index of Passability (IOP) for the pathfinding algorithms. Multiple factors are included when generating the passability map, such as land cover types, water features, cultural, industrial, and soil types (2021, 5). After creating a map based on the topography, the authors consider the troop size. Different passability maps were created since larger troops require a larger space to travel (2021, 7). Allowing the grid spaces to be larger to accompany larger troops to create a more reliable passability map for varying sizes. With the grid, nodes are placed with each edge cost

reflecting the IOP maps, and nodes that would be in an impassable area are removed so that the pathfinding algorithms don't consider them (2021,11). Now that a grid of nodes and edges is available, the authors can use pathfinding algorithms to find the optimal route. The authors implemented Dijkstra's algorithm and A* to determine which was better for this application (2021,11). As seen in Figure 2, both algorithms proved helpful in finding the optimal path, with Dijkstra's being more stable than A* (2021,14). Moreover, the authors found that they obtained different paths at different resolutions (larger troop sizes), with some of the impassable areas being too large to maneuver around at the lowest resolution (2021, 15). Overall, this technique can be universally applied in various cases. Though this paper centers on military applications, the authors acknowledge that such pathfinding can be used in non-military and unmanned vehicles.

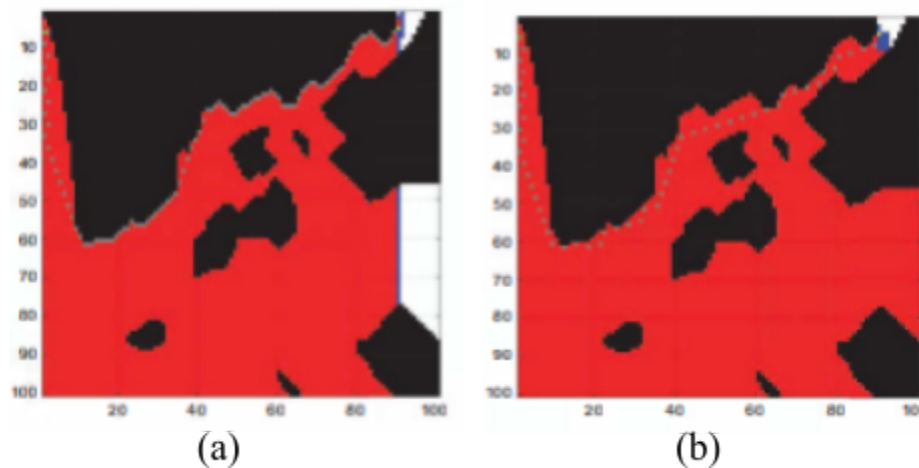


Figure 3. Comparison of (a) Classic Dijkstra's and (b) Improved Dijkstra's in Zhoushan Archipelago (Zhu, Zhenyu, et al., 2021).

Zhu, Zhenyu, et al. found a different application for pathfinding algorithms using a 2D map and an Unmanned Surface Vessel (USV) for maritime use. The authors wanted to create a modified version of Dijkstra's algorithm to limit the unnecessary inflection points of the original algorithm. To do this, they proposed a discriminant factor of pheromone, inspired by ants' path

searching in nature, where they often choose a route with a higher pheromone or higher traversal rate (2021, 3). Using this idea with naval ships, the improved Dijkstra algorithm cuts out the sharp turns and makes the path smoother, as seen in Figure 3 (2021, 5). Creating a more practical path is imperative for this application, and the pheromone factor can also be used in an array of other applications where sharp turns are not applicable.

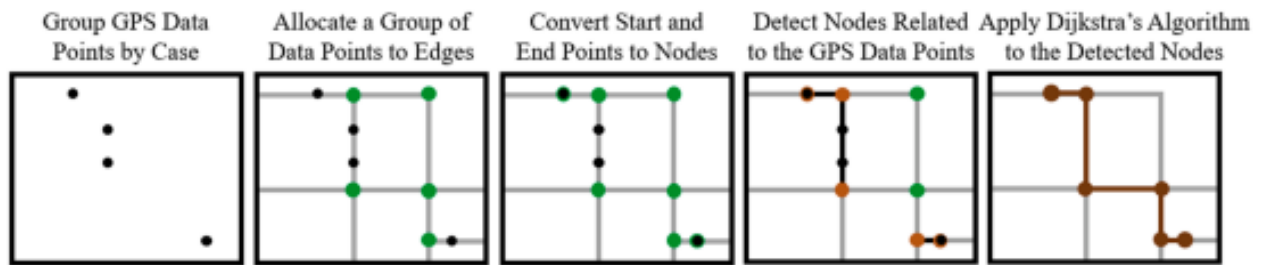


Figure 4. Vehicle path restoration steps (Oh, Heung Jin, and Baabak Ashuri, 2023).

Another way to use pathfinding algorithms is by restoring vehicle paths. Oh, Heung Jin and Baabak Ashuri researched how to enrich GPS data based on emergency vehicles using data with paths restored by Dijkstra's algorithm (2023, 3). Emergency vehicles are essential in society, and if there is a way to decrease response time, it may help save lives. To learn more about emergency vehicle response time, the authors proposed a framework for emergency travel time with vehicle path restoration and data harvesting. As seen in Figure 4, they created a node and edge list to restore the vehicle paths by taking GPS data from vehicles and detecting nodes related to the existing data. Dijkstra's algorithm is then applied to the detected nodes to find the shortest path (2023, 3). Now, they can use data harvesting on the restored paths to get additional information and create the enriched dataset. This enriched dataset can then estimate and identify impacts on driving time with the number of turns, intersections, average road speed limit, and times of day(2023, 5). These methods can potentially create specialized datasets for cities and ultimately create paths for emergency vehicles for the quickest response times.

Robot Navigation

In most instances, applications involving pathfinding algorithms have a known map to traverse, with a graphical representation created for the algorithms. But what about instances that involve navigating an unknown environment? With unmanned vehicles, it is expected to use sensors to compensate for the inability to learn about the environment beforehand. Then, you can apply pathfinding algorithms to the sensor inputs to navigate the area successfully.

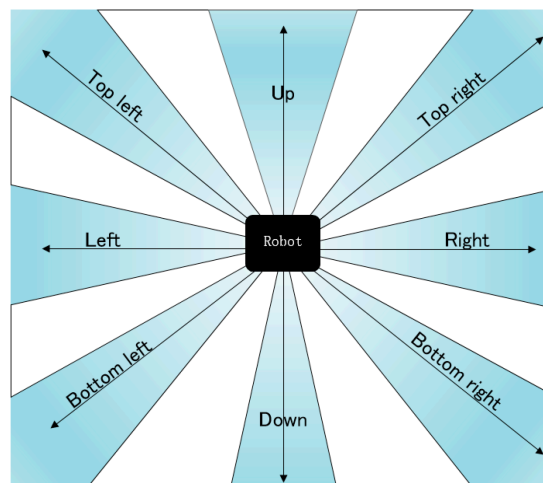


Figure 5. Schematic diagram of the eight-point scan path detection method (Jiang, Le, et al., 2018).

Jiang, Le, et al. aimed to enable a robot to move through an unknown environment by implementing an eight-direction scanning detection (eDSD) algorithm as a pathfinding algorithm (2018, 3). To do this, the robot first scans its surroundings and builds an incremental map based on its localization (2018, 4). This identifies any obstacles and develops a scene to determine where to go. This scan happens in eight directions, with each scan area composed of $\pm 15^\circ$ around the center line of the area, as shown in Figure 5 (2018, 5). This is then used with the weight of rays, grey area exploration, automated obstacle avoidance, and weight of feature pixels as a local pathfinding option to continuously calculate the local optimal direction to move and explore small environments (2018, 11). Since many environments are more complex than small rooms,

the authors proposed a global pathfinding option to scan areas until it has no unknown areas left simultaneously; then, after the scan, the robot can use A* pathfinding to find the shortest path to its starting point and avoid having to traverse the entire environment again (2018, 12). Using this, the authors were able to have a robot that can efficiently traverse an unknown space and help improve automating indoor mapping.

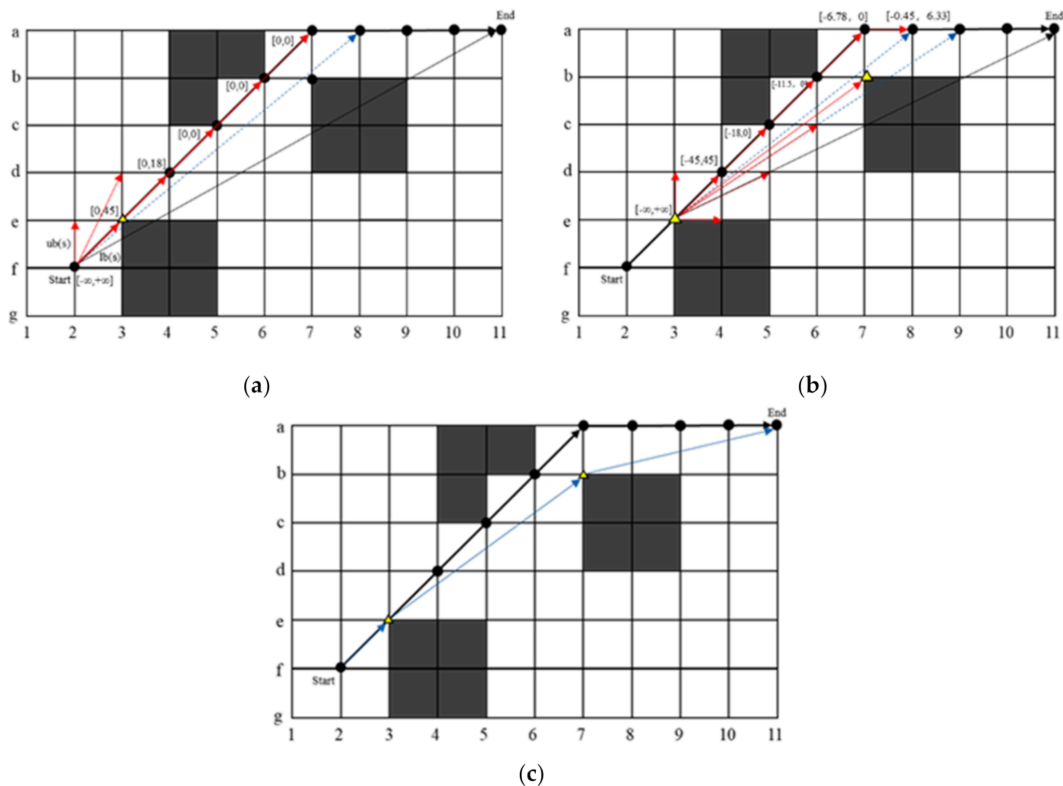


Figure 6. An example of the pathing process is the black line, which is JPS, while the blue line is the optimized version (Luo, Yuan, et al., 2022).

Luo, Yuan, et al. identified that robots are typically limited in movement because they only search at certain angles and plan on a grid map where they can only go in eight directions (2022, 3). This causes the search directions to be extremely limited since they are bound to a grid. The Jump Point Search (JPS) algorithm, based on A*, is unique to A* because it selects successor nodes by searching in a direction and jumping to a point that can skip some path points

and miss out on the shortest path (2022, 5). That is why the authors wanted to optimize the JPS algorithm using angles. To do this, the authors modified the line of sight to view all the reachable nodes at a set angle, as shown in Figure 6 (2022, 9). After identifying nodes, the authors also optimized the path's trajectory to smooth it so that it is more applicable for a robot to traverse (2022, 10). After applying the optimized algorithm, the authors found that although it ran slightly slower than JPS, the path length shortened, and the total turning angle was significantly smaller (2022, 16). Overall, this could be an effective method for robots to effectively find shorter paths without the cost of constantly turning to stay on a grid.

Three-Dimensional Navigation

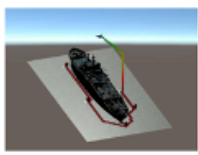
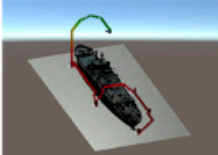
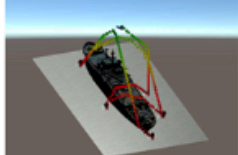
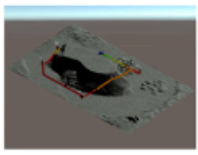
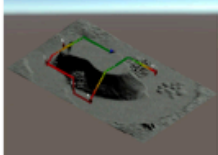
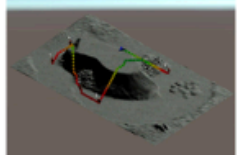
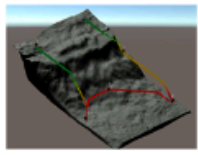
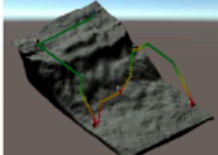
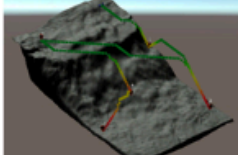
Underwater Site	Strategy		
	Distance	Air Consumption	Decompression
Shipwreck			
Seamount			
Landslide			

Figure 7. The paths generated with three different strategies for three different case studies (Mangeruga, Marino, et al., 2020).

Most pathfinding algorithms require traversing a 2D graph, so what happens when the application requires an algorithm to traverse a 3D graph? Mangeruga, Marino, et al. introduced a novel solution to pathfinding in 3D spaces geared towards pathing out underwater survey dives (2020, 3). This algorithm avoids obstacles in the site's 3D map while considering the divers' decompression limits. To do this, the authors split the algorithm up into different stages: the preprocessing stage to analyze the 3D environment, the creation of links between the points of interest (POI), the optimization of the path, then an external API (application programming interface) takes into account the aspects of the underwater environment to ensure a safe dive (2020, 4). In the preprocessing stage, they identify what regions the diver can pass through and what obstacles need to be avoided; then the space is partitioned into equal-sized voxels, forming a 3D grid (2020, 4). Once the graph is defined, links between the different POIs can be created. The authors used A* to find the paths between the POIs; they mainly had to change the heuristic function for each strategy they wanted to use (2020, 5). The strategies used were distance, air consumption, and decompression cost. Figure 7 shows each strategy's path difference. After creating the links, DecoAPI, developed separately, ensures the paths are safe and feasible for the divers by examining the limits of decompression and ascent pressure, all while going to as many POIs as possible (2020, 8). Depending on which strategy is chosen, the paths created can be more plausible for divers, and with more research, safer paths can be created for underwater survey dives.

Video Games

Pathfinding is a relatively common application in video games. Artificial Intelligence (AI) is commonly used in video games to engage the player and make the experience more

memorable. Game designers commonly use pathfinding algorithms for nonplayer characters (NPCs) to traverse areas in-game to reach a destination. However, as video games become more complex, the demand for more intelligent and humanlike AI rises. This leads to modifying pathfinding algorithms to the requirements of the games.

Several optimizations of A* have proved to be helpful in the gaming industry. Search space, which allows the AI characters to use an underlying data structure to plan a path, and Hierarchical Pathfinding A*(HPA*), which can break up the world into smaller chunks to navigate, leads to faster and improved pathfinding of characters(Cui, Xiao, and Hao Shi 2011, 2). Another optimization is Navigation Mesh (NavMesh). NavMesh takes a 3D environment in the game world and uses polygons to create a “walkable” surface to apply the A* algorithm to find the optimal path (Cui, Xiao, and Hao Shi 2011, 3). This ultimately gives a better description of the map to the pathfinding algorithms instead of trying to create a 2D graph. Memory allocation is also essential in game design. If all of the memory goes to the AI of your game, then it won't run well for the user. That is why game designers try to avoid memory waste by pre-allocating a minimum amount of memory with a buffer or by reducing the space requirements in A* to compute the path in small pieces (Cui, Xiao, and Hao Shi 2011, 4).

Applications of pathfinding algorithms in video games can vary, but they often involve how the enemy moves. Some classic examples involve strategy games such as Age of Empires or Civilization V, which use a tiled map to move units around. A* was used in both games to path enemy movement, but some players found that the pathfinding didn't work perfectly (Cui, Xiao, and Hao Shi 2011, 4). Counter-Strike, a first-person shooter, has better handling with A* since only a few enemies are moving, while in the strategy games, there were hundreds (Cui, Xiao, and

Hao Shi 2011, 4). Video games have continually improved, and with such a high demand for better AI in these games, the algorithms are improving as well.

Human Patterns



Figure 8. An example path produced using the A* and land classification between arbitrary points (Alderton, Simon, et al., 2015).

Understanding human walking patterns in cities or towns can help us know the exposure and risk factors of diseases and how to mitigate a pandemic. Alderton, Simon, et al. wanted to understand these movements to predict disease flow (2015, 2). The authors used a class of computational models called Agent Based Models (ABMs), which helps simulate interactions between individuals with behaviors (2015, 3). To simulate movements in the ABMs, they had to

determine which pathfinding algorithm to use. Instead of using a Euclidean path for the fastest path or Dijkstra's algorithm for the minimal cost, the authors decided to use A* as their algorithm because it is "considered to be the technique which most accurately replicates the process by which a human would devise a route to goal" and it is "computationally economical" (2015, 6). Then, the authors created a map of the area with GPS coordinates, including homes, water sources, and land elevation for traversal. Using the A* algorithm with multiple heuristics, they produced different paths from each household to various water sources, similar to the example path in Figure 8. They then used the ABM to test how long the agents traveled on the paths (2015, 8). This results in a believable set of scenarios that mimic human movements. The authors have also found that the results suggest that trips longer than 1 km or 2 km are the furthest people are willing to travel for water (2015, 13). These simulated human patterns can be used in modeling disease transmission systems, and with further study, it may be able to simulate contacts and infection.

Transportation

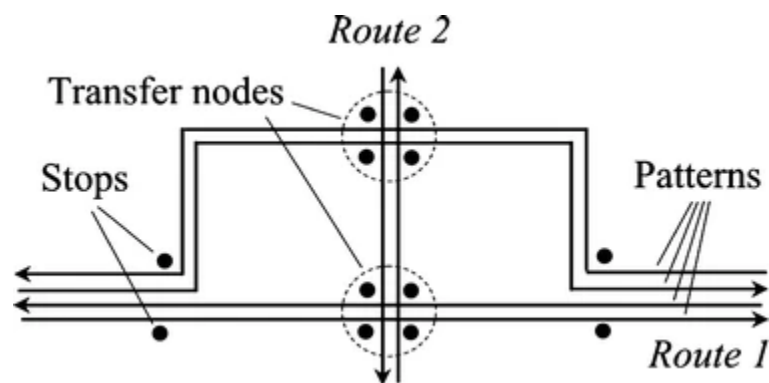


Figure 9. Elements of a transit network (Ruihong Huang, 2007).

Most transportation software is effective at finding the shortest path, but that is not always effective for finding the best path in transit networks. Ruihong Huang proposed a resolution to this problem: use schedules with pathfinding algorithms to ensure a faster path instead of a shorter one (2007, 1). Even though algorithms like Dijkstra's and A* are commonly used for road networks, transit varies in network structure, system operation, and path optimization (2007, 3). The author introduces a schedule-based pattern first search (PFS) algorithm guided by patterns, a transit term for the spatial layouts of routes, and schedules to find the fastest path for a transit network (2007, 4). For a schedule-based algorithm, a network data model must comprise transfer nodes, patterns, and stops in a transit system, as shown in Figure 9. The transfer nodes also identify the arrival time, walking time for transfer, a set of active patterns, a pattern being assessed during this search, the previous node, and the departure time at the previous node (2007, 5). Once the network topology is established and a planned departure or arrival time is ready for the trip, the PFS can determine the path (2007, 6). Real-world transit systems can be highly complicated, so this object-oriented approach to finding the fastest path is applicable. More criteria could even be placed on the path; for example, what if it is more important for you to do fewer transfers than to get the fastest path? As a basis, the PFS algorithm is an interesting approach to using a pathfinding algorithm instead of finding the shortest path.

Electrical Network Pathing

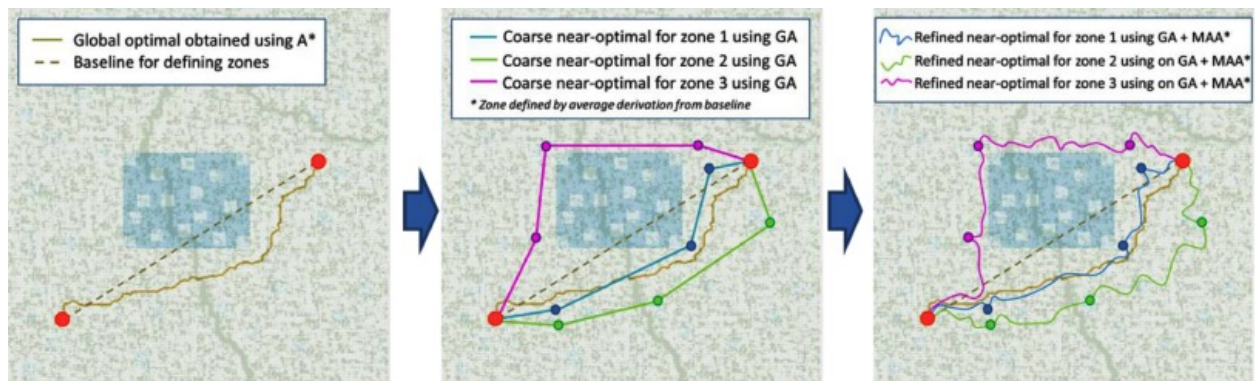


Figure 10. Generating near-optimal paths in different zones using GA and MAA* (Li, Jerry CF, Daniel Zimmerle, and Peter M. Young, 2022).

Energy is an essential commodity in the modern world, but providing energy to everyone is challenging. Since many people often live in dispersed villages in hard-to-reach areas, finding an efficient way to network energy in these communities is imperative. Li, Jerry CF, Daniel Zimmerle, and Peter M. Young wanted to find an approach to changes in demand in villages that considers the topography and accessibility in these regions (2022, 2). To do this, they use a Multiplier-Accelerated A* (MAA*) algorithm, which generally reduces the computation by 90% compared to classic A* (2022, 2). MAA* works by choosing a parameter to selectively attenuate the attractiveness of revisiting prior decisions and reduces the frequency of early-stage restarts, thus reducing the computation time substantially while only reducing the optimality by about 10% (2022, 3). It is also important for other paths to be created. A* and MAA* typically find only one solution for the shortest path. That is why the authors discuss a hybrid pathfinding method that combines MAA* with a modified genetic algorithm (GA), which identifies families of near-optimal solutions, as shown in Figure 10 (2022, 4). This results in a helpful solution with multiple near-optimal paths, reducing the risk of networked rural electrification projects (2022,

13). These methods could also apply to similar infrastructure projects such as gas or oil pipeline routing, communication routing, and transportation system routing. Using this, there is a chance of electrifying rural regions and eventually providing electricity to the world.

Security

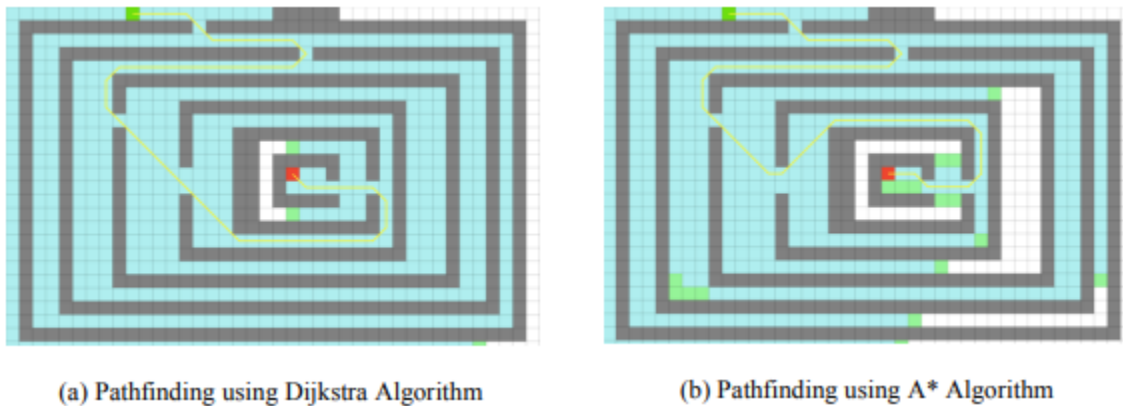


Figure 11. Pathfinding results using Dijkstra's and A* algorithms (Andiwijayakusuma, D. et al., 2019)

Protecting a nuclear facility is extremely important to prevent theft, sabotage, or illegal action. But how would a pathfinding algorithm help with that? Andiwijayakusuma, D. et al. sought to test the security of the facilities with the use of these algorithms (2019). The authors decided to examine both Dijkstra's algorithm and A*. The weights for the nodes used in the algorithms were based on the probability of being detected by security devices like sensors and alarms, which means the path with the smaller weight will be the more vulnerable path (2019, 2). When creating the hypothetical nuclear facility, the authors decided to use a layered system, where there is an outer area, limited area, protected area, and so on, and the main target is the center of the facility (2019, 3). After running the test 10 times, the authors found that the paths Dijkstra's and A* algorithms predicted were slightly different, and they determined that A* took less time and was more efficient than Dijkstra's; the paths are shown in Figure 11 (2019, 5). With

this, we can learn more about high-importance security measures in facilities and determine the weaknesses in such facilities so that we can ensure protection.

Loss Location

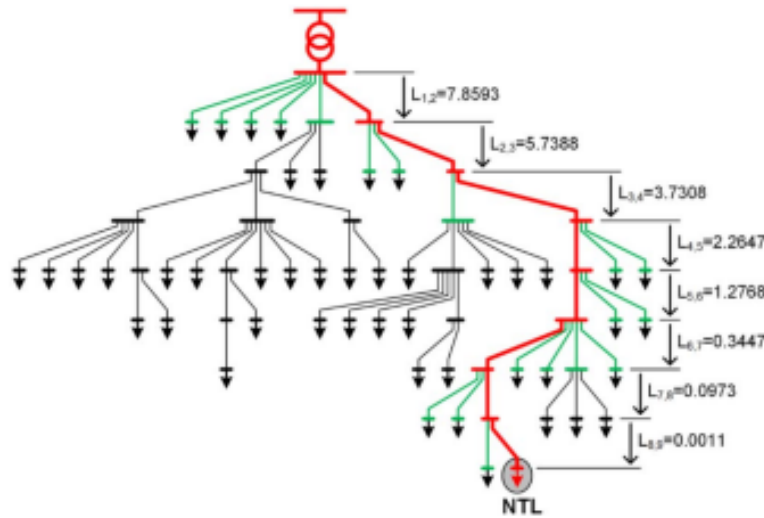


Figure 12. Emphasis of the path built by the search procedure in a low-voltage network. (Leite, Jonatas Boas, and José Roberto Sanches Mantovani, 2016).

Electrical systems can be taken advantage of through fraud, alteration of meter accuracy, and tapping low-voltage lines, and these are typically categorized as non-technical losses. Leite, Jonatas Boas, and José Roberto Sanches Mantovani wanted to discover how to prevent these losses from happening. Smart grids, the next generation of electricity grids, were created to be more efficient, but cyber attacks threatened them (2016, 1). To combat this, the authors wanted to develop a cost-efficient methodology to detect and locate the non-technical losses caused by cyber-attacks (2016, 2). It is simpler for a power system to identify significant voltage or charge changes than small errors. Still, in this case, the authors needed to use a multivariate procedure to monitor the problems in the voltage. First, they sample each field device regularly to find the voltage and current difference (2016, 2). Then, they detect the upper control limit, monitor the

control chart with each sampling, and finally identify the field device with abnormal consumption (2016, 3). Now that they have identified the field device, they can locate the non-technical losses. Using the identified field device as the initial node, they can use A*'s algorithm to minimize the electrical distance, where it iterates through each layer until the value reaches near zero, which will be the target node, similar to Figure 12 (2016, 5). Protecting a power system can come at a high cost, but using a pathfinding algorithm to diagnose and identify cyber-attacks can be more cost-efficient.

Conclusion

Pathfinding algorithms have many uses. After learning the basics of the algorithms with Dijkstra's and A*, their applications widen. The more applications we use them for, the more optimized and modified they become. Since they are pathfinding algorithms, they are often used to find the shortest path. Often using a 2D map, we can use these algorithms for land or sea, military or civilian use, and even robotic use, such as for pathing in an unknown environment. The next step would be 3D pathfinding underwater and in the air. But there are many more applications we can try. Pathfinding algorithms have been used in video games for years, and optimizations are improving with every new game. These algorithms have also been used in transit pathfinding with train schedules and patterns. We can even use pathfinding algorithms to predict human patterns and weaknesses in security. Finally, pathfinding algorithms can be used to create electrical networks in rural areas and identify cyber-attacks that are stealing power. With all these uses, you wonder what the next big step will be for these algorithms.

Reference List

- Alderton, Simon, et al. "Exploiting human resource requirements to infer human movement patterns for use in modelling disease transmission systems: an example from Eastern Province, Zambia." *PLoS One* 10.9 (2015): e0139505.
- Andiwijayakusuma, D., et al. "A Comparative Study of the Algorithms for Path finding to Determine the Adversary Path in Physical Protection System of Nuclear Facilities." *Journal of Physics: Conference Series*. Vol. 1198. No. 9. IOP Publishing, 2019.
- Cui, Xiao, and Hao Shi. "A*-based pathfinding in modern computer games." *International Journal of Computer Science and Network Security* 11.1 (2011): 125-130.
- Dawid, Wojciech, and Krzysztof Pokonieczny. "Methodology of using terrain passability maps for planning the movement of troops and navigation of unmanned ground vehicles." *Sensors* 21.14 (2021): 4682.
- Huang, Ruihong. "A schedule-based pathfinding algorithm for transit networks using pattern first search." *Geoinformatica* 11 (2007): 269-285.
- Jiang, Le, et al. "An eight-direction scanning detection algorithm for the mapping robot pathfinding in unknown indoor environment." *Sensors* 18.12 (2018): 4254.
- Leite, Jonatas Boas, and José Roberto Sanches Mantovani. "Detecting and locating non-technical losses in modern distribution networks." *IEEE Transactions on Smart Grid* 9.2 (2016): 1023-1032.
- Li, Jerry CF, Daniel Zimmerle, and Peter M. Young. "Flexible networked rural electrification using leveled interpolative genetic algorithm." *Energy and AI* 10 (2022): 100186.

- Luo, Yuan, et al. "Improved JPS Path Optimization for Mobile Robots Based on Angle-Propagation Theta* Algorithm." *Algorithms* 15.6 (2022): 198.
- Mangeruga, Marino, et al. "An underwater pathfinding algorithm for optimised planning of survey dives." *Remote Sensing* 12.23 (2020): 3974.
- Noori, Azad, and Farzad Moradi. "Simulation and Comparison of Efficiency in Pathfinding algorithms in Games." *Ciência e Natura* 37 (2015): 230-238.
- Oh, Heung Jin, and Baabak Ashuri. "Enriching GPS data for expanding interpretation of emergency vehicles using a pathfinding algorithm and spatial data harvesting methods." *Sustainable Cities and Society* 95 (2023): 104600.
- Ostrowski, Dustin, et al. "Comparative analysis of the algorithms for pathfinding in GPS systems." *ICN 2015* (2015): 114.
- Zhu, Zhenyu, et al. "Application of improved Dijkstra algorithm in intelligent ship path planning." *2021 33rd Chinese Control and Decision Conference (CCDC)*. IEEE, 2021.