INTERNATIONAL JOURNAL OF TECHO-ENGINEERING

ISSN : 0975-4520

Artificial Intelligence Based Shortest Path Finding Using Dynamic Direction Restricted Searching Algorithm

Dr Ravindar Reddy Thokala,Professor,

Department of computer science and Engineering,

Brilliant grammar school educational society's group of institutions- Integrated campus,

Abdullapur Mettu,Ranga Reddy Dist,Hyderabad,Telangana

Abstract: It is the most popular but frustrating game artificial intelligence (AI) difficulty in the game industry. Several search algorithms, such as Dijkstra's algorithm, bread first search algorithm, and depth-first search algorithm, were designed to determine the shortest path issues. Several investigations about shortest path search determine the probability of using graphs for this design. Dijkstra's algorithm is one of the classic shortest path search algorithms. But, this algorithm was not well satisfied for the shortest path search in large graphs. This is why many changes to Dijkstra's algorithm have been suggested by several authors using heuristics to decrease the run time of the shortest path search. One of the most utilized heuristic algorithms is the A* algorithm, and the main goal is to reduce the run time by reducing the search space. This article proposes modifying Dijkstra's shortest path search algorithm in reduced graphs by employing a dynamic direction restricted searching (DDSE) algorithm. Shortest path problems occupy an essential position in Operations Research as well as in Artificial Intelligence. The performance of the computer implementation of the dynamic direction restricted algorithm is compared to the Dijkstra algorithm, A*, and so on.

Keywords: Artificial intelligence, shortest path finding, Graph theory, heuristic strategy, the dynamic direction restricted algorithm.

I. INTRODUCTION

From the development of the 3D gaming world in the past decade, it is not difficult to see that the gaming industry focuses more on 3D video games. In popular video games like Dead Trigger 2, Counter-Strike, and Get In Touch with duty, gamers can visit and explore unique and exciting worlds on their PC, players are drawn to the concept of walking in theatrical environments in 3-D worlds, or the idea of playing in a 3D digital background.

ISSN : 0975-4520

While there are plenty of entertaining AI issues in the industry, leadership can be the ultimate celebrity for individuals — and indeed, the frustration of one of them all. The point of location by address is to learn how to get from one area to another. Most algorithms rely primarily on maps (networks of records) that store the price of each node or factor in the network. These algorithms attempt to discover a path along with the network. Many pathfinding algorithms can usually be considered and used, but the quality they typically apply is a particular issue. While some plot to detect the path at the lowest total cost (or distance), some algorithms no longer rely on any value register of the path in its extended form. Therefore, the goal is to discover finding directions, a seemingly common but challenging problem, in a world of 3D games. You can run the actual implementation of the issue at hand to find out the fundamental difficulties in solving the problem and discover the resulting implications [1].

Choosing a correctly reproduced path detection method is critical to achieving AI in 3D video games and simulation initiatives because path detection is a fundamental building block in science. AI entertainment. There are many search strategies for video games and simulation projects. Technology may also excel in some instances, but it excels in others. The primary purpose of this document is not always to provide solutions; however, to draw attention to the most common factors that influence the performance of pathfinding strategies. The developer intends to integrate more elements into the game, and the more complex the search for paths will be. Finding paths is usually associated with finding the shortest address, but other possibilities may also have to be solved in another way. This includes locating any address, locating the route with the highest coverage of any location, finding the path with minimal exposure, and finding the set of paths with the largest capacity to carry as many devices as possible to the vacation spot [2].

This article presents a path-finding algorithm that allows us to move from one point to another in the 3D world, depending on how your system is configured. As shown in the example, we will use a grid system that contains blocks from 1 to 42. The dark block represents impassable squares in the grid system. This dark block can be described as buildings, mountains, walls, etc. since you cannot walk from this dark space in the game world. This algorithm uses lists and nodes to help the player determine the best path. Each box of this box can be thought of as a node, and each node has a data pair

ISSN : 0975-4520

with it as H value (indicative), G value (traffic cost), F value (G + H), and the origin (a node to reach this node). Lists are divided into two different types in this proposed system: 1. An open list, and 2. A closed list. The open list is the list of nodes to be verified, and the completed list is the nodes that have been verified. In the opened list, we add the nodes that must be checked over time. The selected nodes will be added to the closed list in this proposed algorithm.

3D Games and Gaming Industry

A computer sport or online game is an interactive program with the primary purpose of entertaining a person (player). Initially, computer scientists developed advanced video games to entertain themselves and their colleagues. The history of video games dates back to the early 1950s when educators began designing simple games, simulations, and artificial intelligence programs as part of their computer science studies. In the late 1970s, with the massive increase in computing power, graphic talent, and mass production of private computers, sports improvement became a legitimate business.

Competition between esports builders and the increasing expectations of players have forced developers to make esports as computationally complex and graphically attractive as the use of standard end-person devices allowed [3].

The '90s saw a massive improvement in video game presentation. From just a four-to-eight color palette and pixelated characters in the early '90s, it evolved into 32-bit color masterpieces of 2D graphics to the early '90s. In addition, processing from 2D to 3D images is quickly observed. Today, a commercial game takes 2 to 3 years of development time and 20 to 50 construction workers.

In the late 1990s, with the global adoption of mobile phones, part of the development efforts of game companies focused on making small, low-budget 2D video games for these new devices. However, these games were unsuccessful due to limited distribution opportunities, insufficient consumer awareness of pricing techniques, heavy hardware segmentation, and the overall coffee performance of the Java MIDP platform used by mobile phones and video games[4].

In 2007, a new gadget known as iPhone came out with Apple inc. It has grown into the critical peripheral area of mobile gaming in a precious space for small independent developers and large organizations. Apple introduced a single distribution channel (AppStore) that a person can access directly and without

ISSN : 0975-4520

difficulty from the tool, a stable standard development system, and a rapidly growing target market for gamers equipped to pay for video games.

The iPhone runs an iOS device entirely based on Apple's Macintosh workstation for laptops and computer systems. A new term, "smartphone," has been coined to explain cell phones with a massive, nearly 5-inch touch screen and more powerful computing and graphics capabilities than an ordinary cell phone.

In 2009, a competitive business device, Android, was launched by Google Inc. And in 2011, it overtook iOS in reputation for being open source and less restrictive. Figure 1.1 represents the number of games developed for smartphones that grew at an exceptional rate during 2009-2012. At the time of writing, mobile gaming is the only growing market for video games, while all the different needs (PC games, console games, web browser games, video games for mobile devices) are stagnant or stagnant[5].

II. REVIEW OF LITERATURE

HaitaoWei et al [2021] Fast and accurate address calculation is essential for software that comprises vehicle navigation structures and transportation community roads. However, many shortest path algorithms for restricted search areas have been developed during the past ten years to speed up the performance of the routing query. The overall performance, including practicality, still needs to be improved to solve this problem; this article proposes a new method for computing statistical parameters Based on a larger, real-world, one-way street network model and a set of rules to make address plans dynamically limited search areas. That build digital obstacles with a lower degree of self-confidence. We have conducted a detailed experiment on the proposed rule set with the actual avenue network in Zhengzhou. As the investigation suggests, compared to the current algorithms, the proposed rule set significantly improves the overall hunting performance under the condition of the high-quality track below the rule to ensure the best steering response.

Zhu et al. [2018] A vehicle that uses cycle planning is an essential record holder in ITS. As an important basis for improving road construction plans, the driving time prediction method has attracted much attention. However, traffic drift is characterized by high non-linear capabilities, time change, and uncertainty, making it difficult for a single-function prediction method to meet the exact demand of an intelligent transmission device in a high-traffic environment. In

ISSN : 0975-4520

this article, the vehicle's historical GPS history statistics create a visitor prediction version. First, a clustering algorithm that mainly relies on QUEst (CLIQUE) V-CLIQUE is proposed to analyze the vehicle's historical GPS statistics. Second, a prediction version based on artificial neural networks (ANN) has been proposed. Finally, a weighted shortest path rule set based primarily on ANN, A-Dijkstra is proposed. We use the implicit absolute percentage error (MAPE) to evaluate the predictive version and check it against the mean and the helper regression vector (SRV). Experiments show that the improved version of the proposed ANN route mapping can, as it should, predict real-time traffic popularity in a given area. Moreover, it has fewer relative errors and saves time for users' travel while saving social resources.

Linet al. [2020] Shortest path problem (SPP) is an optimization problem for determining the path between the vertex of the target source and the vertex of the leave point t in a fuzzy community. Fuzzy logic can handle the uncertainties associated with the statistics of any real lifestyle problem, where traditional mathematical fads may not show the correct result. In the classic SPP, the real numbers are used to denote the duration of the network arc. However, uncertainties related to the linguistic description of arc duration in SPP are not adequately represented by true diversity. Therefore, we need to address essential issues in SPP with unclear arc lengths. The first is based on a method for calculating the path duration using the fuzzy addition process. The second to be remembered is how to evaluate the two specific path lengths indicated by the undefined parameter. We use the gradual hints merge technique of fuzzy triangular numbers to solve these problems. The standard heuristic algorithm for treating SPP is the genetic base set. This manuscript presents an algorithmic method that relies entirely on a set of genetic rules to determine the shortest direction between the width peak and the recess apex t in a fuzzy graph with fuzzy arc lengths in the SPP. A new cross and mutation is brought to resolve this SPP. We also described a QoS routing issue on a dedicated wireless network.

Songet al. [2018] Fast and accurate address calculation is essential for applications such as onboard navigation systems and traffic network routing. While some inferential algorithms have been developed in recent years for faster path queries, their accuracy is always less than somewhat satisfactory. This article first extends the block graph splitting technology to produce hyper-balanced

ISSN : 0975-4520

cross-sections, forming a 3D graph version mainly based on the city street network structuring graph splitting scheme. Next, we recommend the new hierarchical course computation algorithm, which takes advantage of the hierarchical scheme version and takes advantage of the area pruning approach to reduce the search area without compromising accuracy seriously. Finally, we provide a detailed empirical evaluation of the actual road community in New York City. The observed effects demonstrate the effectiveness of the proposed technology in creating the most valuable highways and facilitating real-time routing applications.

Zhouet al. [2013] The incredible complexity of Dijkstra's algorithm limits its software to find the shortest address in the road community. Therefore, it is necessary to improve the performance of finding the shortest route in the street network. It combining the dominant traditional algorithms for the finite search area, a set of shortest path rules based on the low-angle direction rectangle is proposed. Based on the thrust rectangle, the location of rules aims to reduce corresponding corner areas to minimize the locus of search. The proposed set of regulations makes full use of the statistical parameter of the city road network to obtain the economic cost of the control situation. Theoretical calculations and experimental results are presented. They all show that the proposed set of rules can by default decorate the efficiency of finding the shortest path, reducing the time complexity when using 10%-40% without affecting its reliability compared to the current traditional algorithms.

III. LEARNING SYSTEMS

Among the challenging areas covered by artificial intelligence, the study of structures requires a special mention. The idea of gaining knowledge is illustrated here in the topic of a natural problem for the study of pronunciation using the child's mother. The child's hearing device gets to pronounce the letter "A," and the sound system tries to imitate it. The difference between the mother's utterance and the child's utterance, hereinafter referred to as the error signal, is received when the child knows the auditory nerve of the device, and a performance signal is generated when knowledge of the device is acquired through a modulating motor nerve, of the child's pronunciation. The model of the child's voice device is maintained until the amplitude of the error signal is significantly low. Each time the vocal system goes through a cycle of variation, the position of the child's tongue for

ISSN : 0975-4520

speaking "A" is memorized by the mastery model. The study inconvenience discussed above is an example of famous parametric knowledge, where the acquisition of adaptive knowledge of a technique independently adjusts the parameters of a child's voice machine to keep his response close enough to the 'presentation education sample.' Artificial neural networks, which are the electrical analog of terrifying biological structures, are gaining importance in their increasing packages in supervised (parametric) study problems. Besides this kind, knowing the other unfamiliar way, which we unintentionally do, is the inductive field and is based on comparisons. In the inductive domain, the student makes generalizations from examples. For example, when referring to "cuckoo flies," "parrot flies," and "sparrow flies," the student generalizes that "birds fly." On the other hand, in knowledge-based mainly on comparisons, the student learns, for example, the motion of electrons in an atom in a similar way from his knowledge of the motion of planets in solar structures.

a) Knowledge Representation and Reasoning

One has to access the previously described pre-defined one or more particular initial states in a reasoning problem. So the fewer turns to get to the intended state, the better the efficiency of the thinking tool. Increasing the efficiency of the thinking apparatus, for this reason, requires reducing intermediate states to a minimum, which does not directly need an organized and comprehensive base of understanding. A complete and equipped knowledge store at least strives to be familiar with the appropriate technical knowledge in a particular troubled country and, as a result, produces the proper subsequent country in the forefront of troubleshooting methods. Therefore, information organization is of paramount importance in information architecture. In artificial intelligence, a technique of clarifying information is used. Production rules, semantic networks, frames, padding, slots, and original logic are just a few of what can be said. The choice of a particular type of representative comprehension scheme depends on the nature of the programs as on the choice of clients.

b) Planning:

Another excellent place for artificial intelligence is planning. Thinking and planning problems offer many unusual topics but have a simple distinction from their definitions. A reasoning problem should relate specifically to testing the satisfaction of a purpose from a given set

ISSN : 0975-4520

of data and information. Again, making plans is presented by devoting the technique by which can carry out a successful purpose from the initial known cases. Automated planning finds oversized packages in robotics and navigation problems

c) Pathfinding in 3D games

Game characters usually want to move around their score. Sometimes this move is set in stone by developers, including a patrol road that a ranger can blindly pursue or a small fenced area that a dog can roam around at random. Static paths are easy to implement but can easily be cheated if an element is pushed into the form. Free-roaming characters can seem aimless and can get stuck without any issues.

The most complex characters never before understood what they would have to move them. The player can order a unit in a real-time strategy game for any factor on the map at any time; The Patrolling guard in a stealth game may wish to move to the nearest alarm point to summon reinforcements; Recreation of the platform may require fighters to chase the participant across an abyss using existing structures.

For each of these characters, the AI must calculate an appropriate direction through the game level to get from where you are now to your goal. We wanted the path to be reasonable and as short or as fast as possible (it doesn't seem smart if our character is going from the kitchen to the living room through the attic.

This is pathfinding, sometimes called path planning, and it is everywhere in game AI.

d) Search algorithm used in 3D Pathfinding

Graphs are used closely in finding video game paths; For this reason, it is not always surprising that researching graphics has become a core topic in game programming. Graphics search is one of the essential algorithms in game programming. Depth-first Search (DFS) and Breadth-first Search (BFS), and Dijkstra are the most common algorithm for implementing Pathfinding and AI in 3D video games. The following sections discuss these algorithms.

IV. DYNAMIC DIRECTION RESTRICTED ALGORITHM

We introduce a new shortest path algorithm in combination with these previous algorithms: the dynamic constrained routing algorithm. This set of rules is mainly based on a narrow-looking path. People constantly assume that the

ISSN : 0975-4520

shortest path is a set of paths that follow or analyze straight-line direction from the single node and the leave position node. But this principle is not appropriate from time to time. As shown in Figure 1, the shortest direction from node A to node B is not the set of paths that include arc 1, 2, 3, 4, 5, 6, 7, 8, but is the arc 9 and arc 10. Therefore, we must allow some of The arcs to deviate from the straight line AB, so we let some of the arcs factors in the opposite direction to the straight line AB while finding the shorter direction. Then we started thinking about a new bound path algorithm. First, we must consider using a line perpendicular to the given straight line AB as the boundary of the adjacent region to choose whether the new filtered arc follows or points to the straight line AB to be extended. This circumstance of the restricted path is not sufficient because the starting arc, which belongs to the shortest path, probably refers to the reverse direction of the straight line AB. Second, we recommend that this constrained trend line passes at a distance PY from the original node A on the path opposite to the straight line AB, including what Figure 2 suggests. Therefore, we can limit the number of search nodes and reduce the computational complexity, but we can also solve the distribution problem. The unusual weight of the Internet is partly on the road. By comparing the cost of the beeline function AB, we can effortlessly determine whether the new candidate arc node is located in the restricted path region. The extension node can be entered into the path-constrained neighborhood in step 2 of Dijkstra's algorithm in section 2; Conflicting nodes enter the third step. This approach can educate calculus functions.

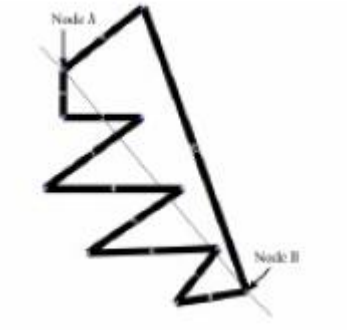

Fig.1 The shortest path from node A to node B

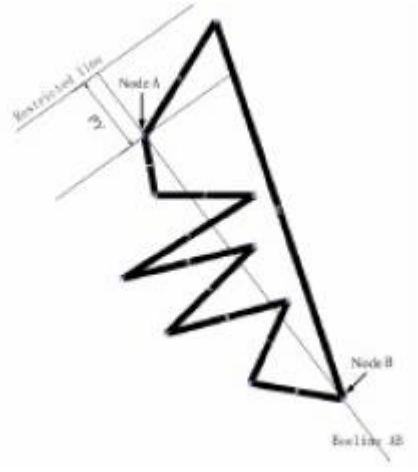

Fig.2 The direction restricted line and the PY

According to the above, the improved algorithm can be described below (in the

ISSN : 0975-4520

same condition as the classical Dijkstra algorithm).

Step 1: Mark the origin node i(N[i]). Initialize the temporary variable Vtempas the origin node. Compute the initial direction restricted searching beeline.

Step 2: Select node j (N[j]), whose label S[j] is 'FALSE', make sure the dist [j] is minimum.

Step 3: According to Vtemp, calculate the new direction restricted searching beeline. Judge whether the node j (N[j]) satisfy the condition of direction restricted. If yes, mark the label S[j] as 'TRUE', set Vtemp as the node j (N[j]), and go to step 4. If no, mark the label S[j] as

'MAYBE', and go to step5.

Step 4: Calculate the movement cost (dist[k]) between the origin node and each node which is connected to the N[j] and whose label S[k] is 'FALSE'. Modify these relevant assistant vector dist[k]. If dist[j]+cost[j, k]<dist[k], then set dist[k]=dist[j]+cost[j, k].

Step 5: go to step 2 until the destination node is marked or the n-1 time loop is ended. If the destination node isn't marked and none of the label S[j] is 'FALSE',set the label S[j] of those nodes whose label S[j] is 'MAYBE' to 'FALSE' , and go to step 2.

The dynamic constrained path algorithm mentioned above can also be combined with many other shorter path experimental algorithms, along with A*, bounded ellipse region ruleset, restricted rectangle area algorithm, etc. An example of such algorithms: The restricted rectangular area algorithm is discussed here. The area of a constrained rectangle can be determined by the four direct lines: the mainline is the constrained line discussed in this previous section; the second is parallel to the primary straight line and must flow along the distance PY from the destination node; Beeline 1/3 and the last are respectively parallel to the beeline AB from two sides. They have the same distance PY from the straight line AB. The remaining three broadcast lines retain their reputation in the way the rules are calculated, while the main direct line does not. It can be modified with the remaining elongated nodules.

V. CONCLUSION

In this paper, a new shortest path finding algorithm based on heuristic strategy for computing shortest path between one origin node and one destination in traffic network is presented. It exploits the knowledge that the shortest path always follows or points toward the direction of a beeline from original node and destination

ISSN : 0975-4520

node. Particularly, the new algorithm discussed in this article can be combined with other shortest path finding algorithms such as the A* algorithm. The new algorithm was implemented and its computational performance was experimentally evaluated and tested. The performance of the computer implementation of the dynamic direction restricted algorithm is compared to Dijkstra algorithm, A*, and so on.

REFERENCES

1. Breadth First Search or DFS for a Graph'(2018). Retrieved from https://www.geeksforgeeks.org/ breadth first-search-or-bfs-for-agraph.
2. Research," ActaGeodaeticaetCartographicaSinica, vol.30, no. 8, pp.269-275, 2008. (in Chinese).
3. Yan Weimin and Wu Weimin, Data Structure. Tsinghua University Press, 2008, pp.186-187.
4. Jung, S.; Pramanik, S. An efficient path computation model for hierarchically structured topographical road maps. IEEE Trans. Knowl. Data Eng. 2015, 14, 1029–1046.
5. Gao, Z.; Yu, Y." Interacting multiple model for improving the precision of vehicle-mounted global position system". Comput. Electr. Eng. 2016, 51, 370–375
6. HaitaoWei, Shusheng Zhang and Xiaohui He, 2021, "Shortest Path Algorithm in Dynamic Restricted Area Based on Unidirectional Road Network Model", pp. 1-16.
7. Zhu, D.; Du, H.; Sun, Y.; Cao, N. Research on path planning model based on short-term traffic flow prediction in intelligent transportation system. Sensors 2018, 18, 4275.
8. Lin, L.; Wu, C.; Ma, L. "A genetic algorithm for the fuzzy shortest path problem in a fuzzy network". Complex. Intell. Syst. 2020, 3, page no.1–10.
9. Prasadu Peddi (2016), "Comparative study on cloud optimized resource and prediction using machine learning algorithm", ISSN: 2455-6300, volume 1, issue 3, pp: 88-94.
10. Song, Q.; Li, M.; Li, X. "Accurate and fast path computation on large urban road networks: A general approach. PLoS ONE" 2018, 13, e0192274
11. Zhou, W.; Qiu, Q.; Luo, P.; Fang, P. "An improved shortest path algorithm based on orientation rectangle for restricted searching area", pp. 692–697.
12. Prasadu peddi (2016), "Experimental Study On Cloud Resource Prediction

ISSN : 0975-4520

And Allocation Using Bat Algorithm",
volume 1, issue 2, pp: 78-82.