Journal of Theoretical and Applied Information Technology
31st October 2021. Vol.99. No 20

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DYNAMIC MAP PATHFINDING USING HIERARCHICAL PATHFINDING THETA STAR ALGORITHM

[1]IRFAN DARWIN, [2]SURYADIPUTRA LIAWATIMENA

[1]Bina Nusantara University, Computer Science Department, Jakarta, Indonesia

[2]Bina Nusantara University, Computer Engineering Department, Jakarta, Indonesia

E-mail: [1]irfan@binus.ac.id

ABSTRACT

Theta Star is an efficient algorithm that can be used to find an optimal path in a map with better performance compared to the A-Star algorithm. Combining the Theta Star with Hierarchical Pathfinding further enhances its performance by abstracting a large map into several clusters. What this combination lacks are the capability to handle a dynamic element in the map. Without that capability, the agent could potentially collide with elements in the map that is undesirable in certain conditions, while adding that capability might reduce the pathfinding algorithm's performance. The proposed algorithm aims to provide the capability to handle dynamic elements without severe negative impact on the performance of the algorithm. The effectiveness of the proposed algorithm is verified in terms of execution time, number of nodes explored, final path length, and the number of collisions that occurred.

Keywords: *Artificial Intelligence, Dynamic Map, Grid-Based, Hierarchical Pathfinding, Theta Star*

1. INTRODUCTION

Pathfinding is one of the basic yet essential tasks for Artificial Intelligence. Pathfinding algorithms have many uses, such as autonomous vehicle [1], unit movement in video games [2], path planner application [3], and movement for robots [4]. There are several algorithms for finding the optimum path to travel from one position to another, each with its advantages and disadvantages. Most of the current algorithm, however, primarily deals with a static map. Meanwhile, a specific condition requires a pathfinding algorithm to consider dynamic elements. The navigation application may require viewing a different route due to changing traffic. AI may be required to consider a potentially dangerous area, and an autonomous car will need to consider other cars' locations. Therefore, adding the capability to navigate a dynamic map to a pathfinding algorithm is important to allow the algorithm to be applied in more challenging conditions.

Many researchers have continuously made improvements to create a fast and accurate pathfinding algorithm. They were starting from the simplest one, which is Dijkstra's algorithm, also known as the Shortest Path First (SPF) algorithm. A few improvements were made into what becomes the A-Star (A*) search algorithm, where a heuristic function is added to determine the optimal path. From these algorithms, another improvement was made to make the algorithm suitable for many nodes, which is then called the Hierarchical Pathfinding (HP) algorithm.

Based on the comparison between several pathfinding algorithms, the algorithm which has a good overall performance is the Hierarchical Pathfinding Theta Star algorithm, which can calculate the optimum path to get from one point to another with better memory usage while maintaining the efficiency of the resulting path [5]. The algorithm works by combining the characteristics of the Hierarchical Pathfinding algorithm and the Theta Star algorithm. The Hierarchical Pathfinding algorithm provides the capabilities to process a large number of nodes by separating them into several smaller grids. Some of the grids partitioned from the Hierarchical Pathfinding algorithm already offer the optimal path, which the Theta Star algorithm will disregard. Then, the grids that do not yet have an optimal path will be further processed by the Theta Star algorithm. This method allows the Hierarchical Pathfinding Theta Star algorithm to provide an optimum path for a large grid with the minimum number of processed nodes.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

While the Hierarchical Pathfinding Theta Star algorithm provides a good result for a static map, improvements can still be made to enable the Hierarchical Pathfinding Theta Star algorithm to be used on a dynamically changing map. A dynamically changing map is defined as a map containing elements such as moving obstacles or, more generally, a map in which a path could become invalid at one time and valid at another. The main contribution of this paper is expected to be an algorithm that allows the Hierarchical Theta Star algorithm to be used on a dynamic environment. Meanwhile, the performance of the algorithm should not be significantly worse than the original algorithm. Additionally, the paper also aims to provide additional insight by implementing the algorithm on a hexagon-grid instead of the usual square-grid.

2. LITERATURE REVIEW

2.1 Related Works

Many different methods can be used to find a path from one point to another. It started from the simplest algorithm, which is Dijkstra's algorithm [6]. This algorithm is also known as the Shortest Path First due to its function to find the shortest path to traverse to get from the starting point to the destination. The Dijkstra's algorithm serves as the basis for many other pathfinding algorithms, which improves the performance of the pathfinding by adding one or more criteria to determine the most efficient path. Some of the improvements made for the Dijkstra's algorithm are visualizing the path generated by the algorithm [7], adding the ability to handle parameters given in neutrosophic numbers [8], applying the algorithm on a curved surface [9], combining it with other algorithms such as the Floyd-Warshall algorithm [10], and one of the well-known extensions of the Dijkstra's algorithm is the A-Star algorithm [11]. Adding a heuristic function to the pathfinding algorithm improves the performance of the A-Star algorithm compared to the Dijkstra's algorithm.

The A-Star algorithm is widely used in many applications such as a map, games, or in robotics. The improvement is based on how effective the heuristic function is in approximating the value it represents. The more accurate the heuristic function used, the more efficient the A-Star algorithm will perform. The A-Star algorithm is also provably optimal in regard to the accuracy of the return path. However, improvements can still be made in other areas of the algorithm such as improving the memory usage, adding other constraints, or adding dynamic path planning capability based on the A-Star algorithm [12].

After the A-Star algorithm, various variations were developed, improving the performance of the pathfinding algorithm in other aspects. One of the variations that deals with a dynamic environment are the Dynamic A-Star algorithm, also known as the D Star Algorithm [13]. The D Star algorithm extended the functionality of the A-Star algorithm by allowing the agent to update the heuristic function used in the algorithm as it continually evaluates the changing environment. The dynamic environment can be divided into three categories. A known dynamic environment means that all of the information regarding the obstacles faced by the agent is completely understood. A partially known dynamic environment means that only some of the information is provided to the agent at the pathfinding stage, which will require the agent to determine a path with incomplete information. While a totally unknown dynamic environment means the agent has absolutely no information about the obstacles in the environment.

Another variation of the A-Star algorithm is called the Theta Star algorithm [14]. This variation aims to improve the performance of the A-Star algorithm by reducing the number of nodes that need to be visited to determine the optimum path. The way the Theta Star algorithm works is that the algorithm checks for line-of-sight between the starting point and the destination instead of checking each neighboring node. The Theta Star algorithm can provide a near-optimal path while providing a runtime comparable to A-Star algorithm. Further variations of the Theta Star algorithm, which further improves the runtime of the algorithm is called the Lazy Theta Star algorithm [15]. Other variation includes the Cluster Theta Star which aims to improve performance by dividing the map into several clusters [16], combining the algorithm with a hybrid A-Star algorithm [17], using a visibility graph as a pathfinding method [18], and another variation which aims to improve the line-of-sight efficiency called the Batch Theta Star [19].

There is another method used to optimize a pathfinding algorithm performance when used on large maps. The method is called a Hierarchical Pathfinding [20]. This method works by separating a large map into several smaller clusters, which is

then evaluated separately, leading to a decrease in search space required. This method, however, requires a second step in the planning stage in which the agent is required also to calculate the most efficient cluster to visit, making the implementation of the Hierarchical Pathfinding more complicated. The Hierarchical Pathfinding is used in combination with another pathfinding algorithm to calculate the optimum path in each of the cluster. Several of the implementation such as the Hierarchical Pathfinding A-Star (HPA*) Algorithm or the Hierarchical Pathfinding Theta Star (HPT*) Algorithm, can be used to scale each of the algorithms on a larger map. Hierarchical Pathfinding can also be implemented under multiple conditions such as using a navigation mesh instead of a grid [21] or in a multi-agent condition [22].

Hierarchical Pathfinding Theta Star works by limiting the number of nodes to consider during the pathfinding. Since the visibility check of the Theta Star is a relatively time-consuming process, this allows the algorithm to be executed quicker compared to a non-hierarchical Theta Star algorithm. A comparison was made, determining that the Hierarchical Pathfinding combined with the Theta Star algorithm provides a similar improvement in reducing the number of nodes visited compared with the Hierarchical Pathfinding combined with the A-Star algorithm. As the researchers suggest, further improvement on the Hierarchical Pathfinding Theta Star algorithm can be made by providing the capabilities to traverse a dynamic environment.

2.2 Static Map Pathfinding

Most algorithms primarily deal with a static map. A map is considered static if it contains only stationary obstacles and does not change while the agent deliberates which path is optimum [23]. This condition allows pathfinding algorithm to plan the optimal path from the starting point to the destination only once and ensures that once found, the path will remain valid.

Algorithms such as the A-Star [11] provides a simple and efficient way to calculate the optimum path. While other algorithms, such as the Theta Star [14] improves the execution time and memory usage of the algorithm by optimizing the number of nodes that needed to be visited to determine the optimum path. Further improvement to the execution time was also provided by the Lazy Theta Star [15]. The accuracy of the path produced by both the A-Star and Theta Star algorithm is proven to optimal given an accurate heuristic function.

To determine whether the heuristic function used in the pathfinding algorithm is accurate, one of the criteria required is for the heuristic function to be admissible. For a heuristic function to be admissible, its estimated cost of reaching the destination must never exceed the actual cost [24]. Admissibility is easier to achieve on a static map since the initial cost estimate will not change over time. Therefore, one of the improvements that can be made to both algorithms is by adding the capability for the algorithm to provide an accurate path in a dynamic map.

2.3 Dynamic Map Pathfinding

Some pathfinding algorithms can also be implemented in a dynamic map. A map is considered dynamic if it contains a moving obstacle or changes in such a way, that over time some path will become invalid and other will become valid [25]. This condition requires the initial path that was considered optimum to be refined according to constantly changing environment to ensure it is still valid and optimum.

In a dynamic map the agent should also take into account the movement of an obstacle to ensure no collision will occur while keeping the path as short as possible. Certain path in a dynamic map might be shorter than another but has a high chance of causing a collision between the agent and an obstacle. While a longer path might provide a much safer route to the destination cell. A dynamic map pathfinding algorithm should be able to choose which path is optimum.

One of the pathfinding algorithms that has the capability to provide an accurate result in a dynamic environment is the D Star [13]. The D Star also classifies the dynamic environment into several categories, such as a known dynamic environment, a partially known dynamic environment, and a totally unknown dynamic environment. These three categories are based on the observability of the environment. A known dynamic environment means that information such as the path of the moving obstacle is known. A partially known dynamic environment means that

only some information is available to the agent. While a totally unknown dynamic environment means that the agent does not possess any initial information about the environment, requiring the agent to both collect the required information and refine its path on the go.

The D Star algorithm provides the basis which could be used to allow a pathfinding algorithm to be used on a dynamic map. Based on the observability of the environment, the behavior of the algorithm itself should change to be able to perform optimally. This algorithm could be further improved by increasing the performance of the algorithm when used in a larger known dynamic environment. This improvement can also be applied to partially known dynamic environment, as long as one of the known information is the size of the map. While for totally unknown dynamic environment this improvement might be difficult to achieve.

There are also several other dynamic pathfinding algorithms that can be used as a reference. Algorithms such as the Hierarchical Pathfinding Lifelong Planning A Star (HPLPA*) are algorithms that combine several algorithms to enable an A-Star algorithm to be implemented in a dynamic map [26]. Other dynamic pathfinding algorithms includes the Dynamic Hierarchical Pathfinding A Star (DHPA*) [27] and there is also research on waypoint graph-based pathfinding algorithm that can be used on a dynamic environment [28].

2.4 Hierarchical Pathfinding

Hierarchical Pathfinding provides a method to allow a pathfinding algorithm to deal with a large environment. The general idea of the Hierarchical Pathfinding is to create abstractions that can help simplify a large environment into a more manageable part. One of the abstraction methods used is by dividing a large environment into several clusters. Each cluster can then be treated as a single node by the highest-level entity when performing the pathfinding. The way this method works is similar to a command structure, where the highest-level entity will decide the general strategy and delegate its implementation to a lower-level entity. The delegation will continue until it reaches the lowest level entity which will perform the actual action, in this case determining the optimal path.

Hierarchical Pathfinding has been applied to both the A-Star and Theta Star algorithm. Based on the test performed, the Hierarchical Pathfinding method combined with the Theta Star algorithm is proven to be more efficient than Hierarchical Pathfinding combined with the A-Star algorithm [20]. The variables compared between to determine the performance of both algorithms are path lengths, number of node visits, and number of nodes in memory.

Since both of the algorithms used are static pathfinding algorithms, improvements can be made to allow a combination of the Hierarchical Pathfinding method with dynamic map pathfinding. Exactly which algorithm will be chosen and combined will impact the performance and its ability to handle certain factor such as the observability of the environment.

There is a constraint when combining the Hierarchical Pathfinding method with a dynamic map pathfinding algorithm. The constraint is that the agent's information should include the structure of the map that the agent will traverse. Using that information, the agent will be able to split the map into several smaller clusters. However, a dynamic map with a totally unknown dynamic environment classification cannot implement the Hierarchical Pathfinding method since the agent has little or no information about the map to split it into smaller clusters effectively.

2.5 Algorithm Comparison

Each pathfinding algorithm has different advantages and disadvantages. Some are easier to implement while others provide additional capabilities in handling certain conditions. The following table summarizes the comparison of all the algorithms that have been previously discussed.

TABLE 1: ALGORITHM COMPARISON

No	Algorithm Name	Accurate	Dynamic	Map Size
1	A*	✓	×	Small
2	Theta*	✓	×	Small
3	Lazy Theta*	✓	×	Small
4	D*	✓	✓	Small
5	HPA*	✓	×	Large
6	HPT*	✓	×	Large
7	Mod HPT*	✓	✓	Large

As shown in Table 1, most of the basic form of the pathfinding algorithms can only be applied on a static map. Each algorithms have their own advantage and disadvantage in terms of execution time, memory usage, return path length, and dynamic capability. For example, D Star algorithm provides dynamic capability but requires an undetermined amount of time since it lacks preprocessing ability. Meanwhile, hierarchical pathfinding algorithms provide a longer return path in exchange for a faster execution time when applied to a large map. While Theta Star provides a more efficient memory usage in exchange for a longer execution time.

Prior research regarding the Theta Star algorithm aims to combine the Theta Star with the Hierarchical Pathfinding, creating the Hierarchical Pathfinding Theta Star algorithm. This allows the algorithm to be applied in a large map while also providing a more efficient memory usage advantage that Theta Star provides when compared to the A Star algorithm. For this research, the Hierarchical Pathfinding Theta Star algorithm is further extended by combining it with some aspect of a dynamic pathfinding algorithm. Allowing the modified algorithm to consider dynamic elements when creating a path for the agent.

The proposed algorithm (highlighted in yellow) is a modified version of the Hierarchical Theta Star algorithm. Different from the original algorithm, which is considered a static pathfinding algorithm, the proposed algorithm will also take dynamic elements into consideration, making it a dynamic pathfinding algorithm. The algorithm is expected to add dynamic capability to the algorithm while maintaining all of the advantages of the basic pathfinding algorithm. Which is providing an accurate path as expected from any pathfinding algorithm, efficient memory usage as the algorithm is using the Theta Star as its basis, and can be applied in a large map as the algorithm is also implemented using Hierarchical Pathfinding.

3. RESEARCH METHODOLOGY

3.1 Benchmark

The experiments are done using the benchmark grids from the Dragon Age computer game [29]. The grid is provided in a text file format, which will be interpreted by the simulator to form the environment. Each character represents a different type of grid.

```
type octile
height 49
width 49
map
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
TTT.............TTTT.TTT...TTTT.TTTT.............TT
TT..............TTT........TTT..TTT..............TT
T.................................................T
T.................................................T
T.................................................T
T.................................................T
T.........................TT......................T
T.......................TTT......................T
T.......................TTT......................T
T.................................................T
T.................................................T
T.................................................T
T.................................................T
T.................................................T
TTT.............TTTT.............TTTT.............TT
TTT.............TTTT.............TTTT.............TT
TTT.............TTTT.............TTTT.............TT
TT..............TTT..............TTT..............TT
TT.................................................T
```

Figure 1: Text Representation of the Benchmark Grid Used in the Experiment (Partial)

As shown in Figure 1, the grids provided have several characters used to identify each cell's type in the grid. The experiments are conducted while keeping the original definition for the "." character, which represents a walkable cell. The changes made are on the definition for the "T" character. Initially, the "T" represents trees which is impassable. In this experiment, the "T" will represent a trap cell, the grid's dynamic part. A trap cell will continuously switch between the "off" and "on" states. While the trap is in the "on" state, the agent moving to that cell will count as a collision.

Apart from the text representation of the map, a total of 130 scenarios are also provided benchmarking purpose. These scenarios place a different starting and destination point for the agent. By executing each of the pathfinding algorithm using the same map and scenario, a comparison can be made to determine the efficiency of each algorithm.

TABLE 2: TEXT REPRESENTATION OF THE SCENARIOS (PARTIAL)

Bucket	X Start	Y Start	X End	Y End	Optimal Length
0	19	26	19	29	3.00
0	44	30	43	28	2.41
0	31	23	33	23	2.00
0	30	22	31	21	1.41
0	40	14	43	16	3.82
…	…	…	…	…	…
5	24	35	43	27	22.31

5	26	41	32	20	23.48
5	28	24	26	45	21.82
…	…	…	…	…	…
13	39	7	3	41	50.08
13	15	42	47	6	49.25
13	5	39	39	3	50.08
13	3	33	46	14	50.87
13	4	32	47	19	48.38

Table I displays the text representation of several scenarios from the 130 total scenarios for the map. The text is formatted in a certain pattern, the leftmost value indicates which bucket the scenario belongs to. Each bucket contains 10 scenarios which means there is a total of 13 buckets for this set of scenarios. The bucket value serves to categorize the scenario based on its complexity, with more complex scenario having a higher bucket value. The second and third value indicates the coordinate of the starting point, while the fourth and fifth value indicates the coordinate of the destination point. Finally, the sixth value indicates the optimal length of the scenario, calculated by the square root of the diagonal cost of the distance between the starting and destination point given in the scenario.

3.2 Map Representation

In order to visualize the path formed by each of the pathfinding algorithm, the application developed for the simulation will convert the text representation into an image representation. While originally represented by a square grid, the application for this experiment will use a hexagon grid. A different color will be used to differentiate one type of cell from the other, with a blue-colored hexagon representing the starting position of the agent and a dark green-colored hexagon representing the destination that the agent needs to reach.

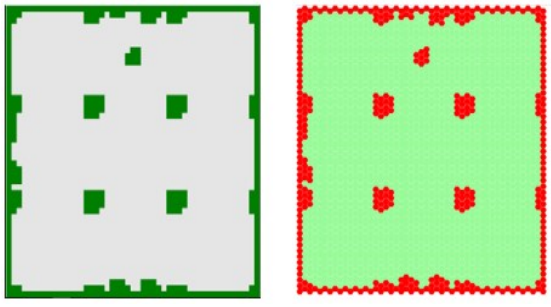

Figure 2: Original Square-Grid Representation (Left) and Hexagon-Grid Representation Used in the Experiment (Right)

Figure 2 illustrates the visual representation of the map. The specific map used in the experiment is the arena map. There are two types of terrains in this map, the "." and "T" cell. In the application used, the "." cell will be represented by a green-colored hexagon which is walkable, while the "T" cell will be represented by a red-colored hexagon which could cause a collision if stepped on by the agent at a particular time.

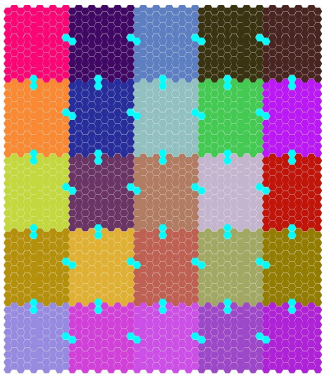

Figure 3: Hierarchical Cluster Representation

Figure 3 shows the clusters formed by the Hierarchical Pathfinding algorithm. For a map with a dimension of 49 x 49, the algorithm creates 25 clusters, represented with 25 different colors, with a dimension of 10 x 10. The various color indicates which grid belongs in which cluster, where a grid with similar color belongs to the same cluster. The clusters act as an abstraction which simplifies the whole map into a few clusters, which minimizes the number of nodes a pathfinding algorithm need to consider when creating a path to a destination. The blue-colored grid represents the connection node, an abstraction that simplifies multiple neighboring grids of two clusters into a single node. The Hierarchical Pathfinding algorithm will only consider the connection node when trying to find a path for the agent to move from one cluster to the other.

3.3 Proposed Algorithms

The proposed algorithm, called the Modified Theta Star, is at its core, a Theta Star algorithm with additional logic during its visibility checking step. The additional logic will allow the algorithm to not only consider the static obstacle but also dynamic obstacle. Meanwhile, other step such as the cluster separation during the hierarchical pathfinding are followed according to a

regular Theta Star algorithm. This setup is expected to allow the modified algorithm to have a comparable performance to the original algorithm, while at the same time adds the capability for the algorithm to consider dynamic obstacle during the pathfinding.

The following pseudocode describe the main point of the Theta Star algorithm. Different from the A-Star algorithm, a parent node in Theta Star does not have to be a direct neighbor of the child node. As long as the parent node has a line of sight to the child node, then it is possible for the two to be connected, bypassing other nodes between the two. Otherwise, the algorithm will work similar to the A-Star algorithm. This allows the Theta Star algorithm to create a more direct path to the destination while minimizing the number of nodes that has to be processed. This results in the Theta Star algorithm to be more efficient in memory usage compared to the A-Star algorithm.

TABLE 3: PSEUDOCODE FOR THE ORIGINAL THETA STAR ALGORITHM

```
if (visibilityCheck(s, n))
    if (gScore(parent(s)) + c(parent(s), n) < gScore(n))
        gScore(n) = gScore(parent(s)) + c(parent(s), n)
        parent(n) = parent(s)
        if neighbor in open
            open.remove(n)
        open.insert(n, gScore(n) + heuristic(n))
else
    if gScore(s) + c(s, n) < gScore(n)
        gScore(n) = gScore(s) + c(s, n)
        parent(n) = s
        if n in open
            open.remove(n)
    open.insert(n, gScore(n) + heuristic(n))
```

As shown in Table 3, the pseudocode for the Theta Star will serve as a basis for the Modified Theta Star algorithm. Most of the variables used are similar to the A-Star algorithm. Variables such as the open, which denotes a list of node that the algorithm can consider to explore, gScore, which denotes the current shortest distance from the start to this node and the c(node1, node2) which denotes the Euclidean distance from node1 to node2. The visibility check is the main function which differentiates the Theta Star from other pathfinding algorithms, as such, the changes made to extend the algorithm will be made in that function, shown by the following pseudocode.

TABLE 4: PSEUDOCODE FOR THE MODIFIED THETA STAR VISIBILITY CHECK ALGORITHM

```
foreach (node in nodeInBetween(node1, node2))
    if (node.isStaticObstacle || node.isActiveDynamicObstacle)
        return false;
return true;
```

As shown in Table 4, the pseudocode for the Modified Theta Star differs from the original Theta Star during the visibility check process. By adding additional condition (highlighted in yellow), the Modified Theta Star will also consider the dynamic element in the environment. The dynamic element will only be considered as an obstacle if the algorithm determine that it will be in an active state when the agent is considering moving there. This will allow the Modified Theta Star to be less likely to pick a path that will cause a collision while at the same time retains the ability to accept collision instead of picking an overly long or failing to find a return path. How the algorithm determines whether a trap will be active or not during a particular time is shown in the following pseudocode.

TABLE 5: PSEUDOCODE FOR THE MODIFIED THETA STAR DYNAMIC OBSTACLE DETECTION ALGORITHM

```
if (node.isDynamicObstacle)
    return numberOfStep % trapFrequency == 0;
return false;
```

As shown in Table 5, the algorithm takes advantage of the information of how often the trap will become active. By keeping track of how many steps the agent needs to reach that cell, the calculation is simply to perform modulo on both numbers. If the number of steps required is a multiple of the trap frequency, then the trap will be active during that time. If that condition is met, the agent will try to avoid moving there during that time.

3.4 Algorithm Limitations

As a result of combining several pathfinding algorithms, the proposed algorithm carries several limitations that is inherent to the basic algorithm used. The first is that by using the Hierarchical Pathfinding algorithm, the algorithm needs information regarding the size of the environment that will be traversed, which is required in order to form the hierarchical clusters.

The second limitation is that in order to form the optimal path, the algorithm also requires information about the dynamic elements in the environment. More detailed information will allow the algorithm to create a more accurate path by simulating what the environment will be like at a certain time.

4. RESULT AND DISCUSSIONS

Six pathfinding algorithms are considered during the experiment. A-Star algorithm (A*), Theta Star algorithm (Theta*), Modified Theta Star algorithm (Mod Theta*), and the Hierarchical counterpart of the three algorithms (HPA*, HPT*, and Mod HPT*). Each of the pathfinding algorithm is executed to provide the return path for each one of the 130 scenarios provided with the map. The result of each execution is then stored and then further processed to determine the general performance of the algorithm on this particular map under varying scenarios. The following figure illustrates the different path produced by Theta Star Algorithm compared to the Modified Theta Star Algorithm when executed on one of the 130 scenarios.

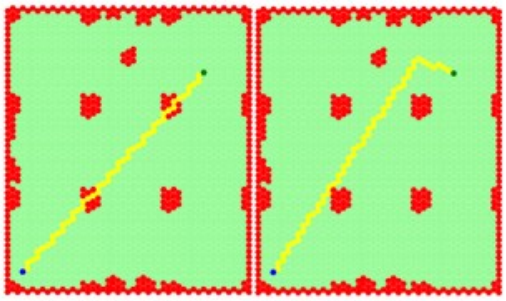

Figure 4: Return Path Theta Star (Left) and Modified Theta Star (Right)

Figure 4 shows the return path comparison between the Theta Star algorithm and the Modified Theta Star algorithm. The yellow-colored cell indicates the path which the algorithm produces. The Theta Star Algorithm creates a path that passes through the red-colored grid, which results in some collision. Meanwhile the Modified Theta Star creates a longer path which evades the red-colored grid, choosing to minimize the number of collisions.

The data used for the final comparison are calculated from after each pathfinding algorithms have finished executing all 130 scenarios. The total result for the execution time, path length, and number of visited nodes are averaged, while number of collisions is summed for the comparison between each of the pathfinding algorithm. Execution time indicates how long the algorithm requires to find the return path and is counted in milliseconds (ms), meanwhile explored node indicates how many cells the pathfinding algorithm need to consider before finding the final return path. Return path length indicates how many steps the agent needs to take to reach the destination, meanwhile collision count indicates how many times the agent steps on an active trap cell. The following figures compares the execution result of each of the six algorithms.

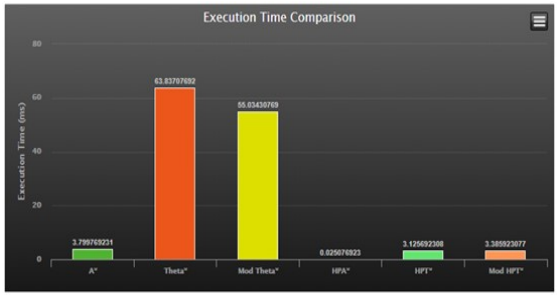

Figure 5: Execution Time Comparison Chart

Figure 5 shows the execution time comparison between the six pathfinding algorithms. For the non-hierarchical pathfinding algorithm, the average execution time for the A-Star algorithm is lower compared to the Theta Star algorithm and the Modified Theta Star algorithm. Implementing the Hierarchical Pathfinding algorithm improves the execution time of all three of the pathfinding algorithms. In comparison between the original Hierarchical Theta Star algorithm and the Modified Hierarchical Theta Star algorithm, adding the dynamic pathfinding capability to the Theta algorithm does not negatively impact its execution time significantly.

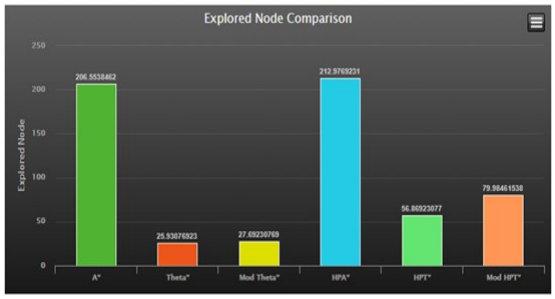

Figure 6: Explored Node Comparison Chart

Figure 6 shows the number of explored graph nodes comparison between the six pathfinding algorithms. For the non-hierarchical pathfinding algorithm, the average number of explored nodes for the A-Star algorithm is higher compared to the Theta Star algorithm and the Modified Theta Star algorithm. Implementing the Hierarchical Pathfinding algorithm increases the number of nodes explored for all three of the pathfinding algorithms. This is due to the preprocessing needed for the Hierarchical Pathfinding algorithm, which aims to split the map into clusters and create a hierarchical abstraction. In comparison between the original Hierarchical Theta Star algorithm and the Modified Hierarchical Theta Star algorithm, adding the dynamic pathfinding capability to the Theta Star algorithm further increases the number of nodes explored due to the need to find alternative path that will not cause a collision.

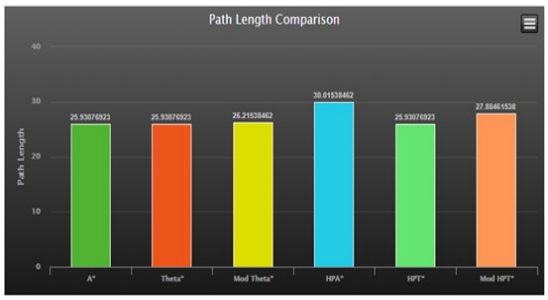

Figure 7: Return Path Length Comparison Chart

Figure 7 shows the return path length comparison between the six pathfinding algorithms. For the non-hierarchical pathfinding algorithm, the average path length for the three algorithms is relatively similar. Implementing the Hierarchical Pathfinding algorithm increases the length of the return path for the A-Star algorithm due to the lack of path smoothing on the final hierarchical result. In comparison between the original Hierarchical Theta Star algorithm and the Modified Hierarchical Theta Star algorithm, adding the dynamic pathfinding capability to the Theta Star algorithm also increases the length of the return path due to the need to find alternative path that will not cause a collision.

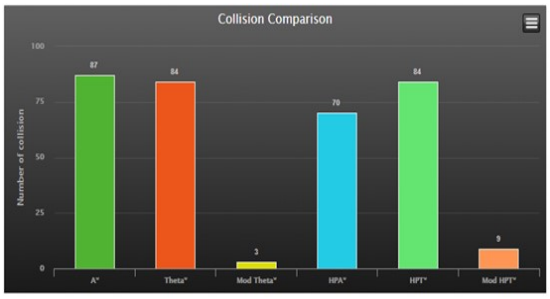

Figure 8: Collision Count Comparison Chart

Figure 8 shows the collision count comparison between the six pathfinding algorithms. For the non-hierarchical pathfinding algorithm, both the A-Star and Theta Star algorithm has a similar number of collisions. Implementing the Hierarchical Pathfinding algorithm does not significantly change the number of collisions. In comparison between the original Hierarchical Theta Star algorithm and the Modified Hierarchical Theta Star algorithm, adding the dynamic pathfinding capability to the Theta Star algorithm successfully reduces the number of collisions that would otherwise occur.

Summarizing the result shown from Figure 5 – 8, a Hierarchical Theta Star algorithm could be extended to be able to handle a hexagon grid and dynamic elements without significant penalty on its performance. The Modified Hierarchical Theta Star algorithm manages to reduce the total number of collisions by up to 90%, while keeping the average execution time within 5% range compared to the original algorithm and keeping the average return path length within 8% range compared to the original algorithm.

Based on the results of the experiment, the Modified Hierarchical Theta Star is capable of performing at a comparable level with the original algorithm. However, this result is achieved with several limitations, where the algorithm has access to complete information regarding the size of the environment and behavior of the dynamic elements. Further improvements can be made to ensure that the algorithm can also perform with incomplete

information. Such as incomplete environment dimension, where the algorithm does not know the exact size of the environment, or unknown dynamic element behavior, where the algorithm cannot perfectly predict what will happen to the dynamic element at a certain time.

5. CONCLUSION

This primary focus of this research is in extending the capability of the Hierarchical Theta Star algorithm. Based on the total number of collisions shown in Figure 8, the Modified Hierarchical Theta Star algorithm can reduce the number of the collision of the final path in most scenarios. This result is achieved while maintaining the efficiency that hierarchical pathfinding provides by abstracting the map into several clusters, reducing the number of explored nodes, and reducing execution time as shown in Figure 5 – 6. Using the Theta Star algorithm as the main pathfinding algorithm ensures that the abstraction will not cause the final path to become significantly longer as shown in Figure 7. These results shows that the Hierarchical Theta Star algorithm can be extended, adding additional capabilities while maintaining the performance of the original algorithm.

Further modifications can still be made to the algorithm. Including adding a more complex dynamic element, multi agent scenario, or implementing the algorithm on a three-dimensional environment. Adding a more complex dynamic element or using multiple agents would require modifying the dynamic obstacle detection algorithm according to the type of obstacle that the agent might encounter. While a three-dimensional environment will require modifying the neighboring node detection algorithm to consider additional directions.

Improvements can also be made to improve the efficiency of the algorithm, such as implementing a Lazy Theta algorithm instead of the original Theta Star algorithm. A Lazy Theta algorithm could further improve the execution time of the algorithm by reducing the number of visibility check required. Another possibility is using a non-uniform cluster separation instead of the basic square cluster. A more efficient cluster separation could potentially reveal a more efficient return path compared to the path that would be generated otherwise.

REFRENCES:

[1] Kulbiej E. Autonomous Vessels' Pathfinding Using Visibility Graph. 2018 Baltic Geodetic Congress (BGC Geomatics). 2018.

[2] Hagelback J. Hybrid Pathfinding in StarCraft. IEEE Transactions on Computational Intelligence and AI in Games. 2016, pp. 319–324.

[3] Imran M, Kunwar F. A hybrid path planning technique developed by integrating global and local path planner. 2016 International Conference on Intelligent Systems Engineering (ICISE). 2016.

[4] Chudý J, Popov N, Surynek P. Deployment of Multi-agent Pathfinding on a Swarm of Physical Robots Centralized Control via Reflex-based Behavior. Proceedings of the International Conference on Robotics, Computer Vision and Intelligent Systems. 2020.

[5] Linus van Elswijk, Hierarchical Path-Finding Theta*, 2013, pp. 11–13.

[6] Javaid MA. Understanding Dijkstra Algorithm. SSRN Electronic Journal. 2013.

[7] Mavrevski R, Traykov M, Trenchev I. Finding the shortest path in a graph and its visualization using C# and WPF. International Journal of Electrical and Computer Engineering (IJECE). 2020.

[8] Broumi S, Bakal A, Talea M, Smarandache F, Vladareanu L. Applying Dijkstra algorithm for solving neutrosophic shortest path problem. 2016 International Conference on Advanced Mechatronic Systems (ICAMechS). 2016.

[9] Luo M, Hou X, Yang J. Surface Optimal Path Planning Using an Extended Dijkstra Algorithm. IEEE Access. 2020, pp. 147827–147838.

[10] Risald, Mirino AE, Suyoto. Best routes selection using Dijkstra and Floyd-Warshall algorithm. 2017 11th International Conference on Information & Communication Technology and System (ICTS). 2017.

[11] Cui X., Shi H. A*-based pathfinding in modern computer games. International Journal of Computer Science and Network Security, 2011, pp 125-130.

[12] Wang C, Wang L, Qin J, Wu Z, Duan L, Li Z, et al. Path planning of automated guided vehicles based on improved A-Star algorithm. 2015 IEEE International Conference on Information and Automation. 2015.

[13] Raheem FA, Hameed UI. Heuristic D* Algorithm Based on Particle Swarm Optimization for Path Planning of Two-Link Robot Arm in Dynamic Environment. Al-Khwarizmi Engineering Journal. 2019, pp. 108–123.

[14] Daniel K, Nash A, Koenig S, Felner A. Theta*: Any-Angle Path Planning on Grids. Journal of Artificial Intelligence Research. 2007, pp. 533–579.

[15] Nash A, Koenig S, Tovey C. Lazy Theta*: Any-angle path planning and path length analysis in 3D. In Proceedings of the AAAI Conference on Artificial Intelligence. 2010.

[16] Mendonca P, Goodwin S. C-Theta*: Cluster Based Path-Planning on Grids. 2015 International Conference on Computational Science and Computational Intelligence (CSCI). 2015.

[17] Михалько ВГ, Круш ІВ. Effective pathfinding for four-wheeled robot based on combining Theta* and hybrid A* algorithms. ScienceRise. 2016.

[18] Abdul Latip NB, Omar R, Debnath SK. Optimal Path Planning using Equilateral Spaces Oriented Visibility Graph Method. International Journal of Electrical and Computer Engineering (IJECE). 2017.

[19] Dang V-H, Thang ND, Viet HH, Tuan LA. Batch-Theta* for path planning to the best goal in a goal set. Advanced Robotics. 2015, pp. 1537–1550.

[20] Botea A, Müller M, Schaeffer J. Near optimal hierarchical path-finding. J. Game Dev. 2004, pp. 1-30.

[21] Pelechano N, Fuentes C. Hierarchical path-finding for Navigation Meshes (HNA∗). Computers & Graphics. 2016, pp. 68–78.

[22] Rahmani V, Pelechano N. Multi-agent parallel hierarchical path finding in navigation meshes (MA-HNA*). Computers & Graphics. 2020, pp. 1–14.

[23] Nilsson NJ. Artificial intelligence: A modern approach. Artificial Intelligence. 2009, pp. 369–380.

[24] Korf RE. Recent Progress in the Design and Analysis of Admissible Heuristic Functions. Lecture Notes in Computer Science. 2000, pp. 45–55.

[25] Charniak E, McDermott D. Introduction to artificial intelligence. Reading, MA: Addison-Wesley; 1991.

[26] Li Y, Zhao W, Zhou Z, Chen C. Hierarchical and Dynamic Pathfinding Algorithms in Game Maps. International Journal of Advancements in Computing Technology. 2013, pp. 87–98.

[27] Kring AW, Champandard AJ, Samarin N. Dhpa* and shpa*: Efficient hierarchical pathfinding in dynamic and static game worlds. In Sixth Artificial Intelligence and Interactive Digital Entertainment Conference. 2010.

[28] Zhu W, Jia D, Wan H, Yang T, Hu C, Qin K, et al. Waypoint Graph Based Fast Pathfinding in Dynamic Environment. International Journal of Distributed Sensor Networks. 2015.

[29] Sturtevant NR. Benchmarks for Grid-Based Pathfinding. IEEE Transactions on Computational Intelligence and AI in Games. 2012, pp. 144–148.