Comparison of A* and Iterative Deepening A* Algorithms for Non-Player Character in Role Playing Game

Anggina Primanita
Dept. of Informatics Engineering
Sriwijaya University
Palembang, Indonesia
anggina.primanita@gmail.com

Rusdi Effendi
Dept. of Informatics Engineering
Sriwijaya University
Palembang, Indonesia
rusdiefendi@yahoo.com

Wahyu Hidayat
Dept. of Informatics Engineering
Sriwijaya University
Palembang, Indonesia
yudafyuce@gmail.com

Abstract— **Role Playing Game (RPG) needs realistic Artificial Intelligence, pathfinding is one of the requirements to achieve it. One of the popular algorithm for pathfinding is A*, but A* still has problem about its memory usage. Iterative Deepening A* (IDA*) is an algorithm like A* that uses Depth First Search to prevent the large memory usage. This research develops a game that implements pathfinding method to enemy character using A* and IDA* algorithms to compare their memory and time usages for pathfinding. Heuristic function that used is Manhattan Distance. This research uses 3 different types of map (without obstacle, simple obstacle, and complex obstacle) with 3 different samples in each type of map as tool for comparing the memory and time usage by A* and IDA*. The conclusion of this research are memory and time usage for A* and IDA* is affected by the size of map (node quantity), position of the obstacles on map, and the obstacle quantity. Then, IDA* Algorithm is generally better than A* in case of memory and time usage especially if the map doesn't have any obstacle, but IDA* can be worse if the enemy character and player are at the parallel position that covered by obstacle.**

Keywords— Artificial Intelligence, Pathfinding, Non-Player Character, A* Algorithm, Iterative Deepening A* Algorithm, Role Playing Game

I. Introduction

Nowadays, the business world in the video game industry is growing rapidly. Things that led to the success of the games themselves is not simply because the graphics are well designed, but must be accompanied by a realistic Game Artificial Intelligence. Role Playing Game (RPG) is a game that contains structured storytelling and rules that restrict player to keep follow the plot that created by Game Master [1]. RPG is one of game genre that requires a realistic Artificial Intelligence. The biggest challenge for creating a realistic Artificial Intelligence of RPG game is in the movement agent. Pathfinding is the main thing that must be done by movement agent.

Pathfinding is a way to find routes that can be explored between the starting and ending points. Offline-based RPG games more often apply pathfinding to movement agent of Non-Player Character (NPC). In the NPC context, pathfinding is used to patrol between two specific points and pursue the approaching player characters [2]. There are many algorithms that have been used so far in pathfinding for RPG games. One such algorithm is the A* Algorithm.

The A* algorithm not only finds the route between starting and destination points, but A* also finds the closest route that can be explored from both points quickly [3]. It has been implemented on several genre of games, such as real-time strategy game [4], life simulation game, and turn based strategy games [5] The A* algorithm that used for pathfinding in RPG games has made a lot of success, but there are still shortcomings in that algorithm, namely in terms of memory usage. Iterative Deepening A* (IDA*) is a pathfinding algorithm such as A * that works by using Depth First Search (DFS) on each iteration to avoid excessive memory usage [6]. IDA* is also a searching algorithm that can find the shortest route from a starting point to the destination point.

II. A* Algorithm

A* algorithm is the first and best graph search algorithm, A* finds the most optimal path between starting and given end points. A* uses a *distance-plus-cost* heuristic function ($f(x)$) to determine the order of nodes to be visited in the search area [7]. For every node, the value of the $f(x)$ function is:

$$f(x) = g(x) + h(x) \tag{1}$$

with:
$g(x)$ is the cost for moving from the initial node to the currently visited node.
$h(x)$ is a heuristic value derived from the distance of the currently visited node to the destination node.

The A* algorithm will select the node that has smallest $f(x)$ value to be visited next. The initial node as the beginning of movement is called the *parent node* and the successor of its node is called the *child node*. A* will stop searching if the destination node is reached. A* will repeat the trace that the parent node has taken to create a route that has been found from the starting point to the destination point.

A. *Iterative Deepening A**

The IDA* algorithm is a new search algorithm that develops the graph into a decision tree. In each iteration, IDA* starts its search from the starting point and implements DFS with the specified limit [8]. The initial value of the limit is set by $h(x)$ which is a heuristic value derived from the distance of the initial node to the destination node. The next boundary value is taken from the smallest $f(x)$ value of all iteration that higher and closer than previous boundary. The $f(x)$ value is derived from the same function as the A* $f(x)$ function.

B. *Manhattan Distance*

Manhattan Distance is a heuristic function that commonly used for rectangular gaming environments with four-way movements [9]. Manhattan Distance finds the minimum cost that will be used to move from the starting point to the destination point with four-way movements by using the formula:

$$h = |dx + dy| \qquad (2)$$

with:
dx is the absolute value of the difference between the x coordinates of starting and destination points.
dy is the absolute value of the difference between the y coordinates of starting and destination points.

The closer to the destination point, the smaller value of $h(x)$ of Manhattan Distance is obtained.

In this game, the NPC will perform a pathfinding for the player character when the player character moves, even if the player character hits the obstacle, the NPC character will still search the path and catch the player character. In performing of the pathfinding, the steps that performed by the NPC character are as Figure 1:

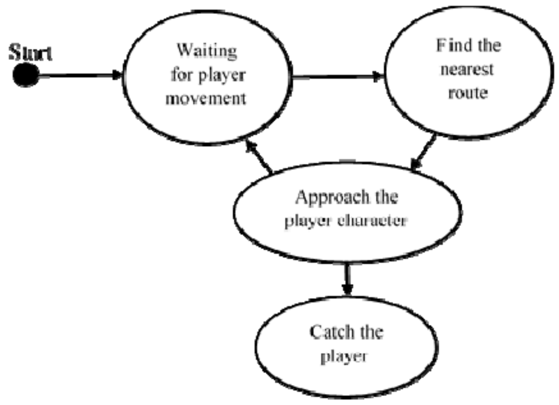


Fig 1. Pathfinding Steps for NPC

The player character and NPC will perform the movement as turn-based game. If the player character has not made any movement, the NPC will not move, too (the NPC will wait until the player character moves as the Figure 1).

To find the nearest route, A* and IDA* algorithms will perform the pathfinding. The pathfinding steps for the A* algorithm are:

1. Pathfinding begins with the initial node that occupied by the NPC, and the destination node that occupied by the player character. The initial node is declared as the first node to be processed (the node to be processed is declared as *current node*), so this node is the only one to be inserted into the *open list* (list of all explored node). Once on the *open list*, since only this node is in this list, this node will be inserted into the *closed list* (list of best node to be member of nearest path).
2. The next step is to add all the neighbors that can be visited from the *current node* into the *open list* and give information that the *parent node* is the *current node*, if the node is not already in the *open list*, if it exists it will be ignored and continued with the node next.
3. Then, the nodes on the *open list* will be compared to find the lowest cost value. The node that has the lowest cost value will be inserted into the *closed list* and will be declared as *current node.*
4. If the *current node* is the destination node, pathfinding will be stopped and the path has been found, otherwise repeat steps 2 and 3 until the *destination node* is found. If the *open list* is empty and also the destination node has not been found yet, then there is no path to go to that node.
5. Once the path is found, enemy characters will save the path by exploring each *parent node* that belongs from the final node to the initial node.

Unlike A*, IDA* algorithm does not require a list to store previously visited nodes. However, IDA* uses the bound value as its search boundary. These are the pathfinding steps for IDA* algorithm:

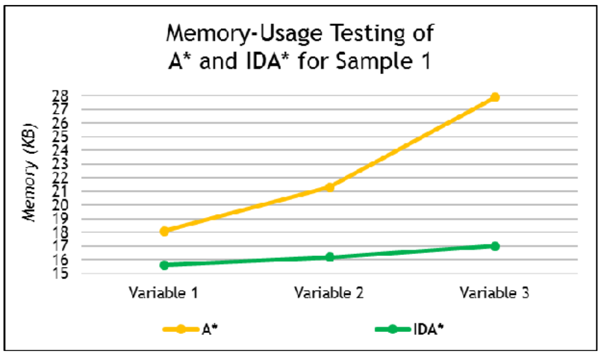
1. Pathfinding also begins from the initial node of NPC. At the beginning of the search, the bound value that used is the $h(x)$ value of the initial node. After obtaining the bound value, the value will call the recursive function of IDA* to perform a pathfinding.
2. In the recursive function, each neighbor of node will be compared to its cost. The neighbor node will also call the same function to compare the next neighbor node until the cost exceeds the specified bound limit.
3. If the value of a node has passed the bound value, then the value will be returned for the cost comparison that lies above the bound value but closest. The closest cost value will be returned as the new bound value. Then, the search will be retrieved from the initial node by using a new bound value (it same as repeating steps 2 and 3).

4. If the node that being examined is the destination node, then the destination node will be saved into the *goal node* (as a sign if the destination node has been found) and also the search will be stopped by returning the infinite value as a sign if the iteration has been complete. If the goal node is the destination node, then the path is found, otherwise there is no path to the node.
5. Once the path is found, enemy characters will save the path by exploring each *parent node* that belongs from the final node to the initial node.

III. Result and Analysis

Pathfinding testing is performed by comparing memory usage and time on A* and IDA* algorithms while performing pathfinding. The testing was performed on 3 different map samples with 3 different variables. The purpose of this test is to see how much difference in memory and time usages of A* and IDA* algorithms for performing pathfinding in RPG games.

In the first map sample, Sample 1, enemy is added to a map without any obstacle, the map size is expanded on each variable, with sample 1-variable 1 as the smallest and sample-1-variable 3 as the largest sample 1 map. The result can be seen in Figure 2.



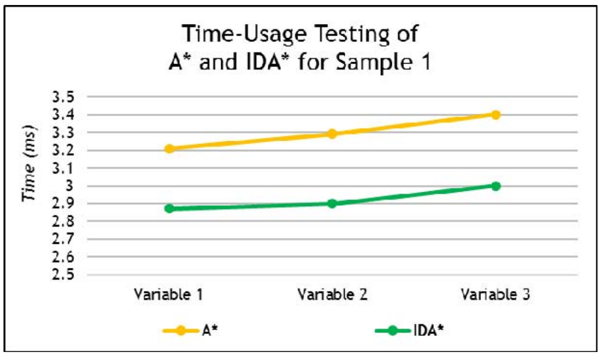
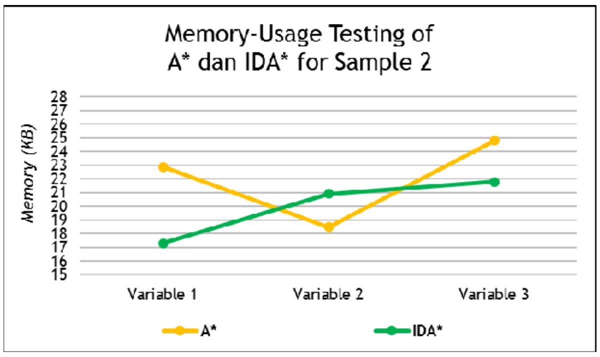


Fig 2. Memory and Time Usages for Sample 1

Based on results displayed in Figure 2, the map size (number of nodes) affects the time and memory usages of the algorithms. The graph shows that in all 3 map sizes, the memory and time usages of IDA* algorithm is lower than the A* algorithm, since f-cost on the map without obstacle is not very varied, it does not make IDA* repeat the iterations. Both IDA* and A* algorithms can find the shortest path from the map variables tested. The A* algorithm stores more nodes than IDA*, since the IDA* algorithm only stores the nodes considered as the shortest path, unlike A* which also stores the explored nodes. It can also be seen from the graph that the difference of memory size used by each algorithm increases significantly as the map is getting bigger, while the difference of time-usage is almost the same on every map sample.

The second map sample, Sample 2, has higher complexity than Sample 1. It has one small region that is considered as obstacle. The map sample size on the variable 1-3 is the same, but the position of player, obstacle, and the enemy were changed for each variable.



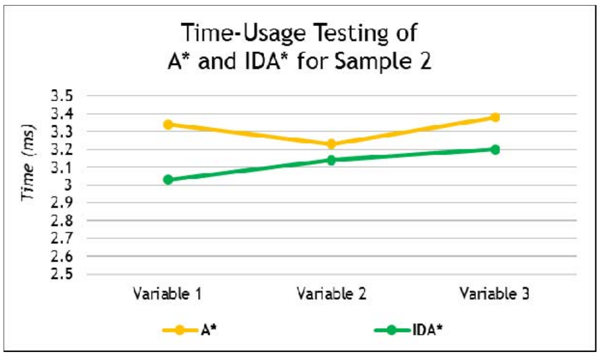


Fig 3. Memory and Time Usages for Sample 2

Based on results displayed in Figure 3, unlike the previous sample, there is one variable (variable 2) where the memory usage of IDA* is higher than A*. In this variable, the player and the enemy are in parallel position, with an obstacle in between them (Figure 4).

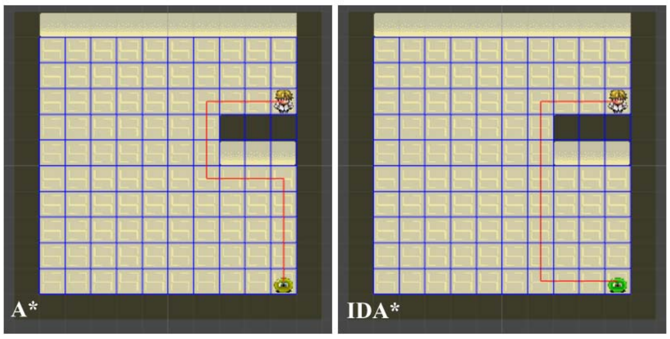
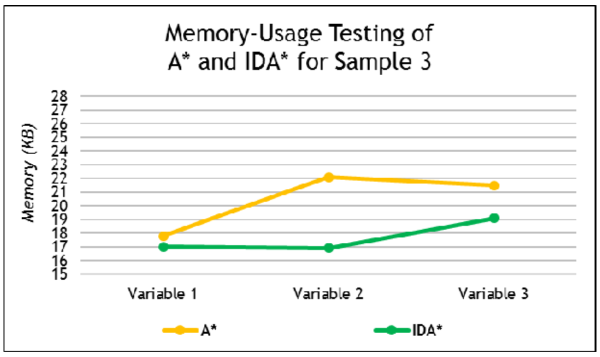


Fig 4. Path result of Sample Map 2, Variable 2.

In this case, IDA* does too many iterations in the pathfinding process because the f-cost value between player and enemy positions that closer is always smaller than the previous iteration but eventually blocked by obstacle (needs to turn right/left) and has to repeat at new iteration to update new higher bound value. This results in the increase of memory allocation of IDA* during pathfinding. It is interesting to see that although the resulting path is different, both path has the same length.

In variable 3, IDA* also almost surpasses the memory usage of A*, this also occurs because during the pathfinding process there is a momentary parallel position between the player and the enemy characters that blocked by an obstacle. The larger memory usage does not make the time usage of IDA* higher than A*.



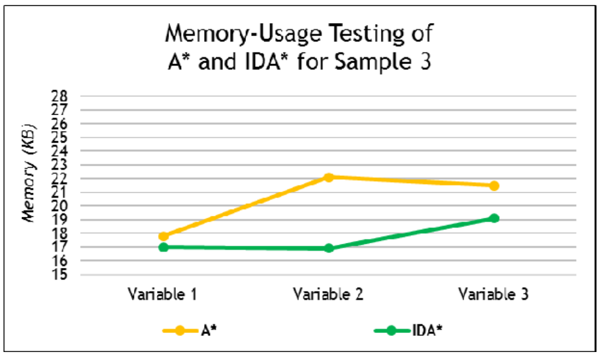


Fig 4. Memory and Time Usages for Sample 3

Based on the Figure 4, memory and time usages of IDA* remains lower than A*. The complexity of obstacle does not make memory and time usages of the IDA* algorithm is worse than A*. The path generated by the two algorithms are different, but still the optimal/nearest path.

IV. Conclusion

The conclusions that obtained from this study are:

1. The memory and time usages of the A* and IDA* algorithms are affected by size of game map (quantity of nodes), obstacle position on the map, and the amount of the obstacle.
2. The increment of memory and time usages in A* and IDA* algorithms is more affected by the obstacle position than the amount of the obstacle in the game.
3. IDA* algorithm is generally better than A* in terms of memory and time usages especially if the map does not have any obstacle, but IDA* may display worse performance if enemy and player characters are in parallel position, but blocked by an obstacle.

further development of this research are as follows:

1. Further research is expected to further optimize A* algorithm for choosing nodes with the same f-cost (adding check for the h-cost of both nodes, the lowest h-cost is preferred).
2. Further research is expected to implement A* or IDA* pathfinding on three-dimensional game map where high variable is also considered to calculate the cost.
3. Further research is expected to add IDA* algorithm capabilities for performing pathfinding of dynamic obstacles.

Acknowledgment

Authors would like to give a big thank you to Department of Informatics Engineering, Sriwijaya University for their support in writing this paper.

References

[1] D. Mackay 2001. The Fantasy Role-Playing Game. McFarland & Company, Inc, Jefferson.

[2] I. Millington dan J. Funge. 2009. Artificial Intelligence for Games 2nd Edition. Elsevier, Oxford, UK.

[3] R. Graham, H. McCabe, and S. Sheridan. 2003. Pathfinding in Computer Games. The ITB Journal: Vol. 4: Iss. 2, Article 6.

[4] J. Hu, W. Wan, X. Yu. 2012. A pathfinding algorithm in real-time strategy game based on Unity3D. 2012 International Conference on Audio, Language and Image Processing. Pp

[5] P. Yap. 2002. Grid-Based Path-Finding. AI '02 Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence. Pp. 44-55.

[6] R.E. Korf. 1985. Depth-First Iterative-Deepening: An Optimal Admissible Tree Search.

[7] Z. Xu and M. Van Doren. 2011. A Museum Visitors Guide with the A* Pathfinding Algorithm.

[8] A. Mahanti, A. K. Pal, S. Ghosh, L. N. Kanal, and D. S. Nau. 1991. Performance of A* and IDA* - A Worst Case Analysis.

[9] G. E. Mathew and G. Malathy. 2015. Direction Based Heuristic for Pathfinding in Video Games.