United International Journal for Research & Technology
Volume 04, Issue 08, 2023 | Open Access | ISSN: 2582-6832

Hybridization of A* Pathfinding and Hierarchical Pathfinding A* Algorithm for Pathfinding in Grid Based Games

Clarence Jacob Agcaoili[1], Xenaiah Yzabella Bernabe[2], and Vivien A. Agustin[3]
[1,2]Student, College of Engineering - Pamantasan ng Lungsod ng Maynila
[3]Professor, College of Engineering - Pamantasan ng Lungsod ng Maynila

***Abstract*—** A* algorithm is one of the most used Pathfinding Algorithm to date as well as one of its variants, Hierarchical Pathfinding A*. The primary focus of this paper is A* issue with its scalability and its memory efficiency and HPA* inaccuracy when returning a path. By hybridizing the algorithms, it aims to solve the aforementioned issues of both algorithms and is also aimed to produce an alternative algorithm for game developers working with grid-based games like in open-world games where Pathfinding is a necessary process. A simulator was created wherein it returns specific values which are runtime in milliseconds, final path, length of the final path, the visited nodes, and the total number of nodes visited which is the comparison basis of this study. Using the simulator, it returned the mentioned data in the form of Test Seeds. By analyzing the results reflected in the Test Seeds conducted, it can be seen that by applying the hybrid algorithm, mentioned issues have been resolved, with A*+HPA* producing a faster run time than A* but slower than HPA*, A*+HPA* visiting fewer nodes than A*, and A*+HPA* returning a shorter path compared to HPA*. It can be concluded that the chosen enhancements significantly affected the performance of the algorithms in this study.

***Keywords*—** A Star, Grid-Based Pathfinding Algorithm, Hybrid Algorithm, Hierarchical Pathfinding A Star.

I. INTRODUCTION

Since the start of human civilization, exploration has been a crucial part of our society. People will always attempt to find the shortest path to their destination, and that has not changed, more so with the advent of technology has this ever been more prevalent, and with widespread access to Global Position Systems (GPS), this has allowed people to plan their routes while taking into account a multitude of factors, such as traffic and total distance needed to be traveled in the physical realm. This is known as "Pathfinding". Pathfinding is a method wherein the fastest and most optimal route between a starting point and its final destination is produced.

However, even in the digital space, the need to find the optimal path between two points exists, especially in entertainment software such as video games [1], pathfinding algorithms are commonly used to simulate an entity moving from point to point, and the most common pathfinding algorithm used in this context is the A* algorithm.

A* also isn't perfect for every use case. According to Adi Botea, Martin Muller, and Jonathan Schaeffer, certain programs such as computer games require problems to be solved in real-time, often under the constraints of limited computational resources. This is due to A*'s runtime and memory consumption increasing with the size of the space it needs to search through [2], and this is why different variations of A* have emerged since its creation. These variations offer certain improvements from their predecessor, may it be memory usage, the overall process, or the total runtime of the algorithm. Examples of A* variants are Iterative Deepening A* (IDA*), Theta*, etc [3].

For large spaces, such as open-world games where pathfinding is used, A*'s memory consumption issue comes to light. In addition, in a study conducted by Wang et. al, open-world video games have created a good reputation in the gaming industry, as players are more engaged in the exploration of a certain location within the game [4]. Open-world games require a large scale of pathfinding, therefore, the algorithm to be produced from this study is aimed to be used in game development projects and the like.

This study aims to combine A* and HPA* to yield an algorithm that utilizes the ability of A* to provide the optimal path, but only for a set limited distance, which then switches to HPA* to sacrifice optimality in favor of significantly faster runtime to provide a near-optimal path while maintaining the previous work of A*.

The A* algorithm expands through the search space as it attempts to find a path towards the target, while the algorithms expansion is weighted towards the direction

***All rights are reserved by* UIJRT.COM.**

of the target node with the usage of a heuristic, this does not fully prevent it from expanding to unnecessary nodes, this results to the A* algorithm becoming more computationally intensive over time [5][6].

The HPA* Algorithm divides the abstract graph into clusters, and each cluster has a selection of border nodes that are used as entry points to other neighboring clusters, these entry points can potentially not result in a valid path towards the entry node, or may be suboptimal which results to a suboptimal path [7].

II. REVIEW OF RELATED LITERATURE

Algorithms are tools that are used almost for everything human society needs for their respective industries. Pathfinding is one of such algorithms that provide ease in situations that require navigation. It is defined as determining the fastest possible route between two given points [8]. Examples of applications that use Pathfinding include Google Maps and Waze. Pathfinding in games is no uncommon feat. Games that have made their name in the community such as the Assassin's Creed series, the Far Cry series, and the Left for Dead series all use some variation of Pathfinding. Adi Botea, Bruno Bouzy, Michael Buro, Christian Bauckhage, and Dana Nau b wrote a comprehensive report of the pathfinding in modern computer games. In their paper, they have stated that the future of pathfinding algorithms in games is not lost, and therefore has a lot of potential in being developed into better algorithms. They also mentioned that their survey can be used to identify potential issues and struggles pathfinding may face in the game development industry [9]. This was further supported by Algfoor, Sunar, and Kolivand, that the future of pathfinding in not only in games but also in robotics, are prosperous when it comes to opportunities to flourish [7].

In the years, different kinds of Pathfinding Algorithms have emerged for the industry. They are divided into two different categories: Uninformed and Informed Pathfinding Algorithms. Uninformed Pathfinding Algorithms perform a "blind search" where the only information they use to return a path is the start node and the destination node. On the other hand, Informed Pathfinding Algorithms are those that use knowledge or heuristic to navigate between different nodes to produce an optimal path. These include Dijkstra's Algorithm, A* Algorithm and variants, and many more. These algorithms have evolved and changed over the course of the years since its development [10]. Therefore, this study focuses on two pathfinding algorithms, A* and its variant, HPA*.

In 1968, Peter Hart, Nils Nilsson, and Bertram Raphael invented the A* (A star) algorithm for an existing project called "The Shakey Project" which had the aim to build a mobile robot that could plan its own path. Originally Nils Nilsson proposed the usage of the Graph Traverser algorithm, which was guided by a heuristic function h(n), which is the estimated distance from node n and the goal node, completely ignoring g(n), which was the estimated distance from node n and the starting node, and this is where Bertram Raphael suggested the using the sum of g(n)+h(n), Peter Hart then invented the concept of admissibility and consistency of heuristic functions [11].

Since its creation, many different variants of A* have emerged, where one is significantly different from the other. These include Iterative Deepening A* (IDA*) where, when compared to A*, uses lesser memory but still functions the same way, Theta* where information spreads around the boundaries of the grid without restricting the paths toward grid edges [12], and Near-Optimal Hierarchical Pathfinding or more commonly known as Hierarchical Pathfinding A* (HPA*).

Daniel Foead, et al. wrote a literature review of A* Pathfinding. This paper included a discussion of the algorithm itself, the usage of the algorithm in the industry when compared to other pathfinding algorithms and A* variants, the known issues of the algorithm, and potential development enhancements to the algorithm. The authors of this paper have stated that although A* has been existing for quite a while and has seen better years, it is still a reliable foundation for new and upcoming pathfinding algorithms that are highly usable in the engines used today. One of these usable variations of A is HPA* [6].

In 2004, Adi Botea, Martin Muller, and Jonathan Schaeffer presented a solution to A*'s runtime issue when pathfinding in large graphs or open spaces, they called it Hierarchical Pathfinding A* (HPA*) a derivation of the A* algorithm which reduced the graph into linked local clusters, wherein at the local level, the optimal distances for crossing from cluster to cluster is pre-computed and cached, while at the global level, each cluster is traversed in a single big step [2]. HPA* was developed to enhance the strength of A* while addressing its weakness.

All rights are reserved by UIJRT.COM.

This study aims to combine the strengths of A* and HPA* to produce an algorithm that is flexible when it comes to runtime and size. As said in a study conducted by Anguelov, modern game environments are complex. By putting the said game environment in a straightforward graph format, locating the optimal path is easier [13]. A* algorithm, although produces the fastest runtime, is unreliable when it comes to larger grid or graph sizes, which is typically the case for video games - large scale graphs are needed. HPA* is slower compared to A*, but is more dependable when scaling larger graphs, highly optimal for video game development. The hybrid algorithm A*+HPA* is created to solve these discrepancies between these two algorithms.

A* Algorithm's first problem stems from its scalability, in which it expands towards the target, so if the target is far away or the graph is large, it results in longer run times. A* Algorithm scans the entire graph to reach the target node using a heuristic function, which in turn causes the process to run slower. This is supported by Brand and Bidarra's study wherein they have stated that the more the space grows, the longer the algorithm takes to reach the target node [5].

As presented by another study by David Foead, Alifio Ghifari, Marchel Budi Kusuma, and Novita Hanafiah, a possible solution to A* scalability is by assessing the dataset to be used and removing irrelevant data. However, by doing so, the heuristic of the algorithm may become less accurate and may lead to less optimal paths [6].

Another issue with A* is that the more the graph it searches increases, the more memory it will consume. Anguelov mentioned in his study that "the primary memory cost of the A* algorithm is in the fact that the algorithm allocates memory for each node encountered in the graph (for the algorithm-specific per-node data)" [13].

As for the second algorithm to be used in this study, HPA*, its issue lie on it returning a sub-optimal path, as it does not scan the entire graph in one pass, rather it groups the nodes of the graph in clusters, and then performs pathfinding using A*.

As further supported by Anguelov, HPA* has shown only 10% optimality of paths it returns, counting from "the placement of the abstract nodes within the cluster entrances." [13].

In conclusion, the above used literatures and studies have helped the researchers determine strengths and weaknesses of A* and HPA* algorithm. These studies have also provided enlightenment to the researchers when it comes to potential issues and errors A* and HPA* can produce when testing is implemented. By understanding how A* and HPA* works using the studies and literatures the researchers have read and used in this study, executing the desired solution has become easier and faster.

III. OBJECTIVES

This study aims to combine the A* algorithm and HPA* algorithm, to create a hybrid algorithm that can utilize A* to provide the optimal path within a limited distance, and maintain the work that A* has provided, before switching to HPA* to favor time efficiency while still providing a near-optimal path above the limited distance. This study is aimed to be used by game developers who will be utilizing grid-based pathfinding.

Specifically, this study seeks to address the following objectives:

To utilize HPA*'s ability to create a near-optimal path in favor of finishing a path in a faster time frame for longer distances.

To utilize HPA*'s ability to cluster nodes together and only use the nodes within the selected clusters to reduce the nodes A* will visit to find the near-optimal path.

To utilize A*s ability to find the optimal path, and modify it to stop at a node that is past the set maximum distance if it has not reached the target node, and switch seamlessly to HPA*, continuing on from the path that A* has generated so far in order to increase the paths total optimality.

IV. METHODOLOGY

A. Analysis

The dataset used by the researchers in performing tests is produced by the simulator created by the researchers themselves. It is comprised of averages of multiple test runs of A*, HPA*, and the proposed hybrid A*+HPA* algorithm done on three different cluster sizes, namely 5x5, 10x10, and 15x15.

It also holds the A*+HPA* results for the static distance limit of 50 and dynamic distance limit based on the Euclidean Distance of the starting position to the target position multiplied to 3 different incrementing percentages.

All rights are reserved by **UIJRT.COM.**

Table IV.I

5x5						
Algorithm	A*	HPA*	A*+HPA* (50)	A*+HPA* (25%)	A*+HPA* (50%)	A*+HPA* (75%)
Ave. Runtime	13.01	0.38	6.04	0.57	2.11	6.56
Path Length	99.00	102.00	105.00	101.00	101.00	105.00
Nodes Visited	1805.33	331.33	1282.33	431.67	771.00	1334.00

B. Design

In the year 1968, Peter Hart, Nils Nilsson, and Bertram Raphael invented the A* (A star) algorithm, which is an algorithm that aims to find the shortest optimal path to a target node, it achieves this by maintaining a tree of paths that start from the starting node and expand one edge at a time until its goal has been achieved or a different criterion has been achieved for the algorithm to terminate. In each iteration of its main loop, it determines the cost of a path from a starting node to a node it has extended to by using a heuristic function (1).

$$f(n) = g(n) + h(n) \qquad (1)$$

Where in n is the last node on the path, g(n) is the distance from node n and the starting node, and h(n) is the distance from node n and the goal node, and selects the node with the lowest heuristic, terminating only when the goal node is reached, or if there are no eligible nodes to expand to.[11]

Adi Botea, Martin Muller, and Jonathan Schaeffer presented a hierarchical approach to solve A*'s problem, where the computational resources required to find a path using A*scales with the size of the space it needs to search through, which resulted in major performance bottlenecks. [2]

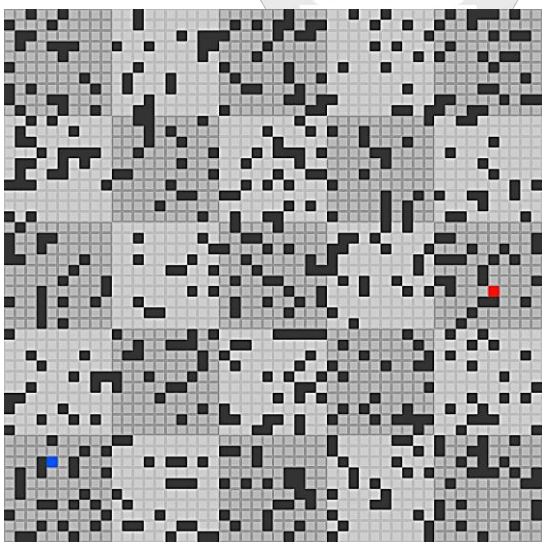

Fig. 3.1 *Division of nodes into clusters*

The solution they presented was to reduce the graph into linked local clusters (see Figure 3.1) which illustrates a graphical representation of a 1-level hierarchy for HPA* pathfinding, with the checkered pattern visually demonstrating the division of nodes into their respective clusters. where in each cluster will be traversed in a single big step. The hierarchy can also be extended to two or more levels, where small clusters are grouped into larger clusters.

Proposed A*+HPA* Hybrid Algorithm Design:

1. The proposed algorithm will first take the abstract graph, start node, and target node as input.
2. From the starting node, the algorithm will attempt to find a path through the graph till it reaches the target node.
3. If the target node was reached within the distance limit, it will return the generated path, however, if the target node has not been reached, it will switch to HPA* algorithm and continue on from where A* left off and find a path to the target node.
4. The HPA* algorithm will then path find through the cluster layer, starting from the continuation nodes cluster to the target node, generating an abstract path.
5. The HPA* algorithm will then refine the abstract path by choosing entry nodes for each cluster.
6. The HPA* algorithm will then generate a detailed path by using the A* algorithm to pathfind from each entry node, only using the nodes within the currently selected cluster.
7. The HPA* algorithm will then return the complete path once the target node has been reached.

C. Development

In retrospect, any programming language is capable of executing a pathfinding algorithm. So, in choosing a programming language to be used for this study, the researchers have chosen languages that are object-oriented, commonly used in game development, and familiar/have been used by the researchers in the past.

Considered languages were Python and Javascript, but what is finalized and used in the study is C#. C# is an object-oriented programming language that was a descendant of C++, another object-oriented language, where it is a natural environment for crafting software components. C# is a language commonly used in game development. A specific game engine called Unity Engine utilizes C#. Since the study is primarily concerned with providing an alternative algorithm for

All rights are reserved by **UIJRT.COM.**

game development, for the simulation of the algorithm, the Unity Engine version used is 2021.3.24f1. An Integrated Development Environment (IDE) called Visual Studio Code for writing code is utilized for its ability to utilize IntelliSense with Unity, which helps reduce the time needed to write code.

The system is created and tested in a computer with a 64bit Windows 11 Pro operating system:

- Ryzen 7 5700x 3.4GHz
- RTX 2060 Super 8GB
- 24GB DDR4 3200MHz

D. Testing

Each test consisted of one of the algorithms mentioned in this study, a quality check of the simulation and output, and a log of the results returned by the Benchmark.

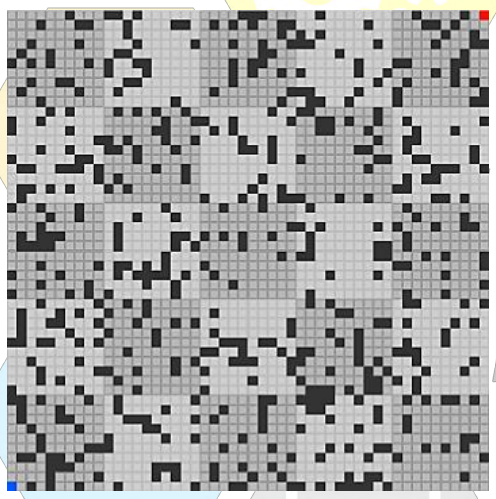

Fig 3.2 *Invalid Seed (1467330106)*

Utilizing the Master Script, the appropriate test configuration for the graph is inserted, once configured, a random seed is generated which will be curated by first generating the graph using the same seed, but in 3 different cluster sizes, 5x5, 10x10, and 15x15, this is to test how each algorithm will perform in small, medium, and large search spaces, cluster density will be kept at a constant 10 on all tests for consistency, this is all done by pressing the Execute button while the selected algorithm is set to None, this will generate the abstract graph and physical grid once executed, for the curation process of a random seed, the start node and target node must not be obstructed on all sides by blocked nodes and a valid path must exist between the Start node and Target Node in all 3 different cluster sizes, this is to ensure that both nodes are reachable during the testing of the algorithms, (see Fig. 3.2 for an example of an invalid seed).

Once the random seeds are chosen for testing, 3 algorithms, A*, HPA*, and A*+HPA*, these algorithms will then be executed 20 times to generate a reasonably accurate average runtime for each algorithm, the same will be done with A*+HPA*, however, A*+HPA* will be tested in 4 different configurations, 1 using a static distance limit which would be set to 50, and dynamic distance limit based on the Euclidian Distance from the start and the target using 3 different multipliers 25%, 50%, and 75%, this is to test which configuration would yield the most optimal outcome in different scenarios.

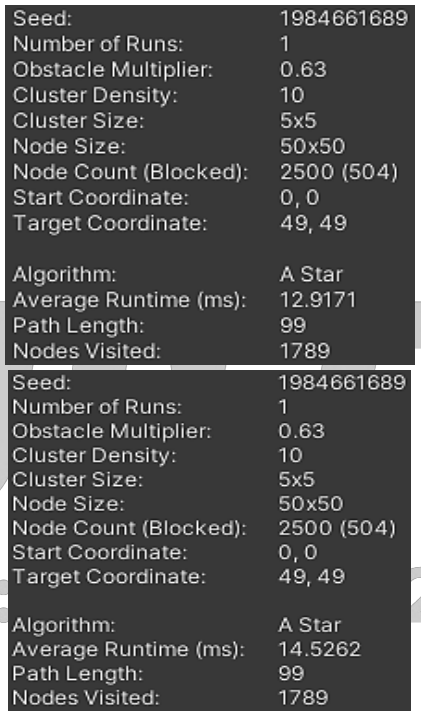

Fig 3.3 *(a) Expected result (b) Unexpected Result*

All results will then be curated and checked for any abnormalities, such as errors and unexpected results (see Fig.3.3), which shows unusually high run times sometimes caused by caching and initialization of the resources required by the program. If no errors or abnormalities are observed, the physical grid showing the path and visited nodes will be saved as a PNG file and test results will be documented in a separate Txt file.

The overall process of the algorithm goes as follows:

1. 5 random seeds are generated and curated for reachability between the start node and target node in all 3 cluster sizes (5x5, 10x10, and 15x15), with a consistent cluster density of 10.

All rights are reserved by **UIJRT.COM.**

2. The appropriate seed and cluster size are applied for the graph and the appropriate configuration for A*+HPA* pathfinding algorithm is applied if required.
3. A*, HPA*, and A*+HPA* pathfinding algorithms are then executed 20 times to generate a path for each algorithm and configuration, to generate a reasonable average runtime for each test run.
4. The results are then curated for any errors and unexpected results.
5. If no errors and unexpected results are observed, the results for each algorithm are saved, including the result on the physical grid which is saved as a PNG, and the performance metrics which is saved as a Txt file.
6. Repeat step 2 until A*, HPA*, and A*+HPA* (4 configurations) are tested on all the 3 curated seeds are tested in the 3 different cluster sizes.

RESULTS AND DISCUSSION

A. Cluster Size 5x5

Table V.I

Algorithm	A*	HPA*	A*+HPA* (50)	A*+HPA* (25%)	A*+HPA* (50%)	A*+HPA* (75%)
Runtime	13.29	0.42	6.07	0.59	2.21	6.64
Path Length	99.00	101.50	104.00	101.50	101.00	103.50
Nodes Visited	1817.75	354.00	1279.25	448.00	793.25	1333.75

The A* algorithm exhibited the longest average runtime and largest on average number of nodes visited of every test in a 5x5 cluster size, with an average runtime of 13.29 ms and an average of 1817.75 nodes visited respectively. The hybrid algorithm with a static distance limit of 50, generated a path length with the most nodes on average among the other algorithms tested with an average path length of 104.00, it is however notable that the hybrid algorithm with a dynamic distance limit of 50% managed to return a shorter path to the target with an average path length of 101.00 within a significantly shorter runtime of 2.21 ms when compared to A*.

B. Cluster Size 10x10

Table V.II

Algorithm	A*	HPA*	A*+HPA* (50)	A*+HPA* (25%)	A*+HPA* (50%)	A*+HPA* (75%)
Runtime	235.85	0.88	8.54	2.76	24.95	96.15
Path Length	199.00	207.50	205.00	205.50	203.00	203.00
Nodes Visited	7385.50	732.50	1843.25	1202.00	2845.50	5090.75

The A* algorithm exhibited the longest average runtime and largest on average amount of nodes visited of every test in a 10x10 cluster size, with an average runtime of 235.85 ms and an average of 7385.50 nodes visited respectively, HPA* produced the highest average path length 207.50, it is, however, notable that the proposed algorithm with a dynamic distance limit of 25%, managed to return a path shorter on average path length of 205.50 within a runtime of only 2.76 ms, while only visiting 1202.00 nodes on average, the proposed algorithm with a dynamic distance limit further reducing the average path length to 203.00, while only visiting 2845.50 nodes on average, however, in a slightly longer runtime of 24.95ms, which remains significantly shorter than A*'s average runtime.

C. Cluster Size 15x15

Table V.III

Algorithm	A*	HPA*	A*+HPA* (50)	A*+HPA* (25%)	A*+HPA* (50%)	A*+HPA* (75%)
Runtime	1332.23	1.93	8.72	10.26	126.51	563.87
Path Length	299.00	312.00	309.50	310.00	306.50	306.00
Nodes Visited	16787.25	1210.00	2269.75	2380.50	6157.00	11491.25

The A* algorithm exhibited the longest average runtime and largest on average amount of nodes visited of every test in a 15x15 cluster size, with an average runtime of 1332.23 ms and an average of 16787.25 nodes visited respectively, HPA* produced the highest average path length 312.00, it is, however, notable that the proposed algorithm with a static distance limit of 50 managed to produce a shorter path on average 309.50 while only visiting 2269.75 nodes, within a runtime of 8.72 ms, and the proposed algorithm with a dynamic distance limit of 50% producing a shorter path on average of 306.50 while visiting 6157.00 nodes on average, within a runtime of 126.51ms which while quicker than A* may not be suitable for an algorithm that is meant to run in near to real-time.

D. Overview of Discussion

With the test results from the 3 different cluster sizes and 4 curated seeds, it is observed that the A* algorithm produced results with a longer average runtime and visited the most amount of nodes when compared to the other algorithms tested in this study. However, it also provided the shortest path in all 3 cluster sizes.

The HPA* algorithm exhibited the shortest run time in all 3 cluster sizes while producing the longest path on average in cluster sizes 10x10 and 15x15.

Moreover, the proposed algorithm managed to perform significantly better than A* in average runtime and average number of nodes in all 3 cluster sizes, all while staying below HPA* average path length in cluster sizes 10x10 and 15x15.

All rights are reserved by **UIJRT.COM.**

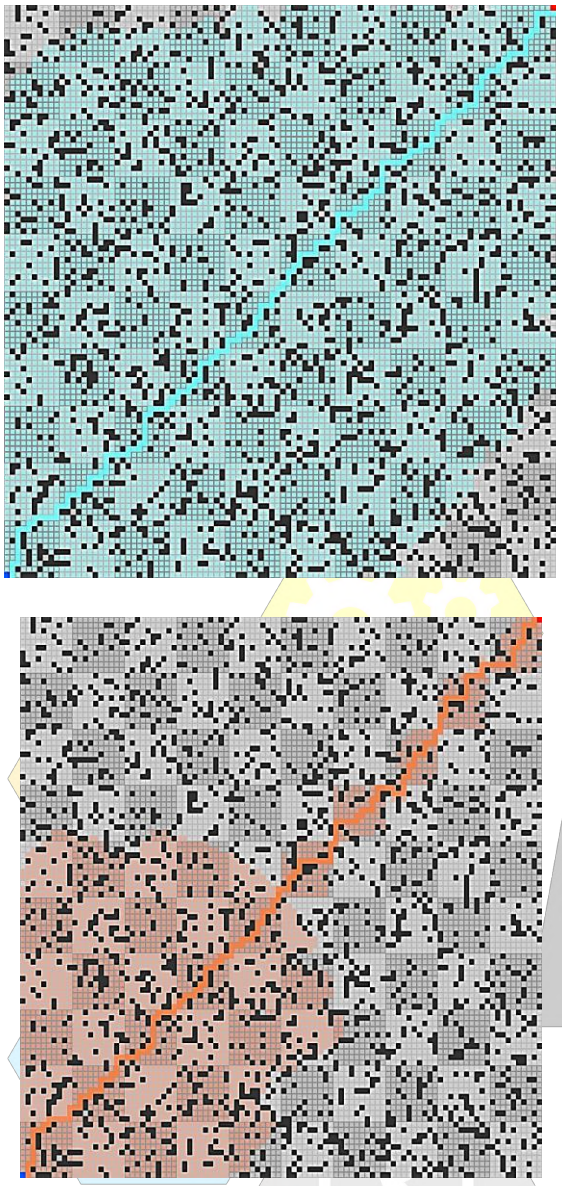

Fig. 5.1 *(a) A* Result (b) A*+HPA* 50% Result*

Both are results of the seed 1160403470 in a 10x10 Cluster

Objective 1 has been achieved by utilizing HPA*'s ability to produce a path in a fast timeframe than A*, and Objective 2 has been achieved by utilizing HPA*'s ability to only path through clusters, and the nodes within the selected clusters, therefore reducing the amount of visited nodes.

Table V.IV

A*		A*+HPA* (50%)	
Ave. Runtime	Nodes Visited	Ave. Runtime	Nodes Visited
225.44	7253	23.92	2797

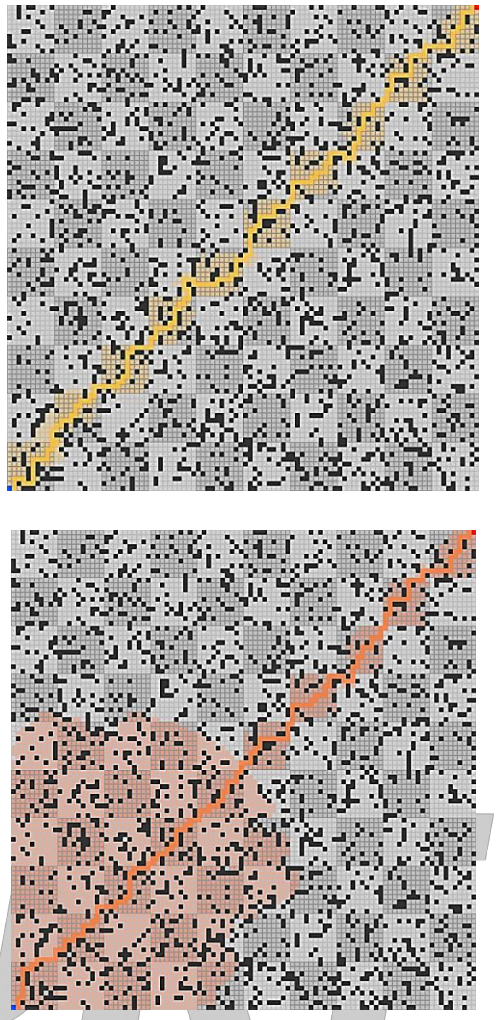

Fig. 5.2 *(a) HPA* Result (b) A*+HPA* 50% Result*

Both are results of the seed 1160403470 in a 10x10 Cluster

Objective 3 has been achieved by utilizing A*'s ability to find an optimal path and modifying it in a way that stops at a maximum distance, and since the target node has not be reached, it seamlessly switched to HPA*, continuing on from the path A* has generated, resulting to improved optimality compared to utilizing HPA* independently.

Table V.VI

HPA*	A*+HPA* (ED. 50%)
Path Length	Path Length
208	201

V. CONCLUSIONS AND RECOMMENDATIONS

A. Conclusion

The A* algorithms main issues when it comes to its scalability lies with its runtime and memory usage when attempting to path through large search spaces, which is

All rights are reserved by **UIJRT.COM.**

a problem as for games with large grid maps, as games typically run at real time or near real time. The HPA* algorithm attempts to solve this issue by grouping nodes into clusters and using only nodes within the selected clusters to path towards the target, therefore significantly decreasing the time required to find a path to the target. This, however, comes at the cost of creating a path that is suboptimal.

The enhancement of combining A* with the HPA* and introducing a distance limit to A* attempts to solve this issue by giving developers a flexible algorithm where A* is still primarily used till it reaches the target or the distance limit. Thus by applying this hybrid algorithm, the benefits of A* and HPA* could be exploited while also mitigating their weaknesses. There are nuances however, specifically in small search spaces where if the distance limit is not set appropriately the result has the potential of being even worse results when compared to A* and HPA*.

That being said, in larger search spaces, this is less of a problem, resulting in an algorithm that is able to produce a path:

When compared to A*, is able to produce a path shorter by modifying A* to only path at a set limited distance, in combination of utilizing HPA*s ability to reduce the path in a shorter runtime.

While visiting fewer nodes when compared to independently using A* by using HPA*s ability to path through clusters and its ability to only use the nodes in the selected clusters, significantly reducing the size of the search space, therefore reducing the number of nodes that could be visited.

The use of A*s ability to produce an optimal path due to its thorough processing of nodes which helps to increase the total optimality of the generated path when compared to utilizing HPA* on its own.

With these algorithms hybridized and modified, a path could be generated in a faster timeframe while visiting fewer nodes when compared to A*, and a shorter path could be generated when compared to HPA*, providing a hybrid algorithm that exploits both algorithm's strengths, and mitigating their weaknesses.

B. Recommendations

The researchers recommend the produced algorithm of this study to game developers who wish to have options when it comes to choosing an algorithm to use for their future projects, as the results reflected by the algorithm are not far off with the results reflected by A* and HPA* alone. The algorithm can also be used in other future studies that may require a hybrid algorithm.

For future researchers who wish to take up this study, the researchers recommend to utilize other pathfinding algorithms and/or other variants of A* while using the same hybrid approach. This is to make sure flexibility and variation is present. Another recommendation would be to use a different programming language when simulating the algorithms in this study. Examples are Python, which is a commonly used language nowadays, and Javascript. The researchers also recommend testing the hybrid algorithm in other development engines that are not restricted to game development (e.g. in Robotics, logistics, etc.).

ACKNOWLEDGMENT

The researchers most especially want to thank God for the enlightenment He has given to them during the course of making this study. The researchers would also like to express their immense gratitude to their thesis adviser, Prof. Vivien A. Agustin, for the guidance and encouragement she has given them throughout the process of creating this study. A heartfelt gratitude is also in order to the Computer Science Department and the researchers' friends and family for their unwavering support regarding this study.

REFERENCES

[1] L. Krayzman, N. Kumar, and S. Scott Lawrence, "Applications of Pathfinding," Pathfinding InfoPages. https://mbhs.edu/~lpiper/pathfinding/applications.php (accessed Jun. 04, 2023).

[2] Botea, M. Müller, and J. Schaeffer, "Near Optimal Hierarchical Path-Finding.," vol. 1, pp. 1–30, Jan. 2004.

[3] Patel, "Variants of A*," Stanford.edu, 2016. http://theory.stanford.edu/~amitp/GameProgramming/Variations.html

[4] Wang, Z. Gao, and M. Shidujaman, "Meaningful Place: A Phenomenological Approach to the Design of Spatial Experience in Open-world Games," May 2023, doi: https://doi.org/10.1177/15554120231171290.

[5] S. Brand and R. Bidarra, "Parallel Ripple Search – Scalable and Efficient Pathfinding for Multi-core Architectures," pp. 290–303, Nov. 2011, doi: https://doi.org/10.1007/978-3-642-25090-3_25.

All rights are reserved by **UIJRT.COM.**

[6] Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, and E. Gunawan, "A Systematic Literature Review of A* Pathfinding," Procedia Computer Science, vol. 179, pp. 507–514, 2021, doi: https://doi.org/10.1016/j.procs.2021.01.034.

[7] Z. Abd Algfoor, M. S. Sunar, and H. Kolivand, "A Comprehensive Study on Pathfinding Techniques for Robotics and Video Games," International Journal of Computer Games Technology, vol. 2015, pp. 1–11, 2015, doi: https://doi.org/10.1155/2015/736138.

[8] X. Cui and H. Shi, "A*-based Pathfinding in Modern Computer Games," IJCSNS International Journal of Computer Science and Network Security, vol. 111, 2011, Available: http://paper.ijcsns.org/07_book/201101/20110119.pdf

[9] Botea, B. Bouzy, M. Buro, C. Bauckhage, and D. Nau, "Pathfinding in Games," doi: https://doi.org/10.4230/DFU.Vol6.12191.21.

[10] H. K. Sidhu, "Performance Evaluation of Pathfinding Algorithms," PDF, University of Windsor, 2020. Accessed: Jun. 01, 2023. [Online]. Available: https://scholar.uwindsor.ca/cgi/viewcontent.cgi?article=9230&context=etd

[11] P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968, doi: https://doi.org/10.1109/tssc.1968.300136.

[12] K. Daniel, A. Nash, S. Koenig, and A. Felner, "Theta*: Any-Angle Path Planning on Grids," Journal of Artificial Intelligence Research, vol. 39, pp. 533–579, Oct. 2010, doi: https://doi.org/10.1613/jair.2994.

[13] B. Anguelov, "Video game pathfinding and improvements to discrete search on grid-based maps," Jan. 2011.

UIJRT
ISSN: 2582-6832

All rights are reserved by **UIJRT.COM.**