Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 179 (2021) 507–514

Procedia Computer Science

www.elsevier.com/locate/procedia

5th International Conference on Computer Science and Computational Intelligence 2020

A Systematic Literature Review of A* Pathfinding

Daniel Foead[a], Alifio Ghifari[a,*], Marchel Budi Kusuma[a], Novita Hanafiah[b], Eric Gunawan[b]

[a]*Computer Science Department, School of Computer Science, Bina Nusantara University, Jakarta, Indonesia 11480*
[b]*Computer Science Department, BINUS Online Learning, Bina Nusantara University, Jakarta, Indonesia 11480*

Abstract

A* is a search algorithm that has long been used in the pathfinding research community. Its efficiency, simplicity, and modularity are often highlighted as its strengths compared to other tools. Due to its ubiquity and widespread usage, A* has become a common option for researchers attempting to solve pathfinding problems. However, the sheer amount of research done on the topic makes it difficult to know where to start looking. With this paper, we hope to create an accessible, up to date reference on the current state of the A* search algorithm for future pathfinding projects to consider. This paper examines A-Star's current usage in the field of pathfinding, comparing A* to other search algorithms. It also analyzes potential future developments for A-Star's development. A* cannot keep up with the demands of current pathfinding problems. Other algorithms can maintain the same performance while also demanding less overhead and this problem only grows worse as grid size increases. However, use of innovative modifications such as different heuristic types or secondary components to the algorithm allow A* to achieve very fast times with good accuracy when dealing with large maps, while only having slightly increased overhead costs for the modifications. While beginning to show its age, improved algorithms based on the classic A* algorithm are more than capable of keeping up with modern pathfinding demands. These derivative search algorithms such as HPA* is used to overcome the limitations of A*. HPA* can compete with and even surpass its competitors depending on the challenge faced.

Keywords: A-Star, Pathfinding, Heuristic function, HPA-STAR

* Corresponding author. Tel.: +62-811-961-6661.
E-mail address: alifio.ghifari@binus.ac.id

10.1016/j.procs.2021.01.034

1. Introduction

Search algorithms have been a major interest in computer science fields for a long time. Many applications of search algorithms do not have the luxury of time or large amounts of processing power. Searches must be swift, accurate and efficient, any deviation from these qualities is seen as a huge failure. The main two types of searches informed and uninformed both have their places but informed searches are by far the more popular type. They make use of a heuristic function, an addition to the program that measures the distance from the goal, in order to make better decisions. Although there are many types of heuristic functions for many types of search algorithms, this paper will focus on A* search, how well it works in comparison to other pathfinding methods, its advancements and potential future.

1.1. Methodology

The research will be conducted using literature review. Systematic literature review was chosen due to the abundance of research papers related to A*. With the use of systematic literature review we are able to determine the scope of our study and enable us to analyze and synthetize our research through the use of a focused research questions that needs to be answered. The literature will be searched using google scholar as a platform. The keyword that chosen to help refine the search are be using are pathfinding, A* algorithm, and search algorithm.

Before starting the research, some research questions were chosen to be a guide. These research questions allow our paper to be succinct and focused during the research. All the research used should be able to help answer one of these questions, in one way or another. The following are the research questions we chose:

- How effective A* algorithms are compared with other method to solve pathfinding?
- What other methods that can be incorporated with A* algorithm to make it more efficient?
- How will A* be implemented in recent and future pathfinding problems?

1.2. Scope and Limitations

This paper only focuses on A* search. The scope of this review covers 40 papers. The only time this paper discusses other search algorithms will be as comparison or potential future use. There are also a few limitations imposed on the research to ensure quality and relevance. These restrictions are as such:

- Research published within the past 5 years to ensure that information is not outdated.
- Research must be a quantitative research.
- Research must have a transparent and replicable methodology.
- Research must have a focus on A*algorithm and a comparison with other method.
- Research must focus on pathfinding problems.

1.3. Significance

As one of the most popular and widely used search algorithms, there has naturally been a large variety of papers on A* search. Many papers focus on specific, niche applications but few focuses on general applications and characteristics of A* search. This paper seeks to rectify that issue and provide a fairly easy to understand explanation of what makes A* so popular and good. Although impossible to always be accurate to every situation, a baseline idea of what makes an algorithm good will greatly assist fledgling pathfinding projects.

2. Background

*2.1 A*in General*

The A* Algorithm have been a tried-and-true method to use as a basis for pathfinding problems. For Usually A* algorithm use for broad game and strategy-based game, it’s stability and quickness has only been improved over the

last decade and even more improvements could be made to it in the near future. But A* is not a fool proof Algorithm, as in many cases it needs a supplementary algorithm or a tweak to its core functions for it to be able to complete these tricky tasks. As seen in Multi-Agent Pathfinding problems, the A* encounters many hurdles such as conflicting pathways between the multiple agents[1]. The layouts of these test samples also come to mind, as in one of the papers the researcher tested out the Algorithm on similar maps but with a different grid style[2]. A* algorithm may also find a path much quicker than uninformed search however it does not ensure that the result will be the shortest path. On research shows that with a couple of cases of strategy maps and maze, A* algorithm will only show 85% of the time to give a result where it finds the shortest path[3].

2.2 Modifications to Heuristic

With the problem arise within the heuristic function of A* algorithm, many heuristic functions are introduced. One of the common heuristic functions in pathfinding is heuristic distance. There are a couple of heuristic distances such as the Manhattan distance, excluding distance, and diagonal distance[4, 5].

Each of the following heuristic distances can be modified with a change of behaviour such that an efficient and reliable heuristic distance can be found. Other than that direction based heuristic distance is also introduced as there it shows an increase in performance from the generic A* algorithm[6].

Many algorithms introduce heuristic functions such that it is an algorithm that uses an informed search method. An informed search is a method where the algorithm utilizes heuristic functions to reach its target. Unlike uninformed search such as Dijkstra or breadth first search (BFS), it will try to search every possible outcome first before reaching its destination. As such informed search such as A* can be much more efficient than Dijkstra within pathfinding[7,8]. As A* algorithm grows more popular especially within the game industry, problems may arise to which heuristic function could yield worse results than uninformed search such as the BFS algorithm[9].

There is a very clear focus on A* search algorithms when considering problems that have time and memory constraints. It is of no surprise that many have searched for ways to improve the basic algorithm through a variety of methods. Due to A*being so dependent on the heuristic, changing how the heuristic functions is a first stop for many of these papers. Modification of the heuristic could take the form of changing how things are weighted in the heuristic function[10]. Another, more extreme example would be to change entirely how the heuristic function is obtained. This is a potential large benefit, especially when considering different environments in pathfinding[11]. Different maps have different types of obstacles to traverse. This point is especially important to consider when dealing with higher levels of complexity. Another potential way of improvement is to change the cost function itself and add a more thorough priority list to minimize the time spent chasing nonoptimal or dead ends[12]. Other methods were also introduced to incorporate the execution time and the amount of processing to its desired value. Methods such as bidirectional, Iterative Deepening A* (IDA*), and Hierarchical Path-Finding A* is introduced to optimize the overall A*performance[13,14,15].

2.3 Potential Issues

As time passes, new types of problems appear. A* current trend among papers currently, is how to deal with more than one agent efficiently. In most scenarios, agents will experience collisions and are unable to occupy the same space as another one. To combat this, many potential solutions have been proposed. One, newer and more experimental method is to make use of swarm intelligence inspired by animals such as bees[16, 17]. While this method has been proven to decrease computing times, it is not a good solution for everything. This is due to the increased computational load caused by the numerous calculations[18,19.]

Another potential way to increase performance is by cleaning up the dataset itself by removing unnecessary

information. This is not a universal approach but can make a massive difference if done in the proper environment. This has to be done carefully however, the removal of information may make the heuristic less accurate and lead to less optimal paths[20,21]. Finally, there are various potential issues when it comes to the testing phase. In pathfinding, there are very few ways to reasonably judge an algorithm besides testing it in a variety of datasets and situations[22]. Because of this, issues such as biased sampling can occur that result in unfair depictions of an algorithm's performance. This in turn can cause potential algorithms to be discarded when they could have had a good potential niche to fill.

2.4 Future Paths

A* is a common choice due to its accuracy, performance and time complexity but it is not without its shortcomings. It has a relatively high memory storage requirement to hold all of the data it uses. However, if that is not a major issue but other things such as computational time or complexity are, then A* is often close to the number one choice[23]. A* way forward could be to reconsider each problem and think if they really benefit from A-Star. No project is the same and priorities differ, which is why further look into algorithms such as Dijkstra or Deepening Iterative Search could be useful. Dijkstra in particular shines much better on problems where the time spent calculating a heuristic could be better spent actually exploring[24,25] Other contenders to rival A* in Multi-Agent Pathfinding such as Conflict-Based Search. This Algorithm when tested with variety of other A-STAR-based Algorithms proved to surpass them all in a randomly generated dataset[26].

All of these factors in to prove that there are still many improvements to be made for the A* Algorithm in cases such as these, and while some might argue these cases are only a niche, Multi-Agent problems will only grow more and more prevalent as time goes on.

3. Presentation and Analysis of Data

*3.1 A*compared to other methods*

As a whole, there are two general types of searches. Informed searches, such as IDA*, A*, Jump Point Search IDA*, tend to use some sort of third-party data for increased performance and rely on the ability to know and convert goals into data. Heuristic functions tend to work extremely well when the area of search is well known, such as the map of a game, but can suffer if the data is inaccurate or unknown. By contrast, uninformed searches will blindly follow their algorithm until completion, this typically makes them slower but less dependent on external factors. For example, in games, uninformed searches, Dijkstra in this case, performed extremely poorly compared to HPA*, getting the optimal path at almost one third the speed[14]. A* on its lonesome still performed better than Dijkstra, finding the optimal path at two thirds the speed. From this example, the potential increased efficiency of a Heuristic function is very clear.

An important factor of A* search is its efficiency. While most well-made pathfinding algorithms will be able to find the solution, many will take more time, resources or both when compared to A*. There is up to a difference of over 40% when it comes to efficiency in general and close to 30%[28]. On another note, A* is particularly modular and can be adapted and changed to suit a wide variety of needs as necessary. Not all modifications are always better however, and some adaptations that prioritize speed may end up not finding the best path. This was the result of a 29 node, 4 intersection experiment that placed HPA*, A*, Dijkstra and IDA* against each other. HPA* failed to find the fastest path although it still finished incredibly fast, while the other algorithms were slower but managed to achieve the optimal result. Achieving the most optimal path, while desirable is not always necessary depending on application. Some uses, such as a real time GPS management will have users care more about instant, quick

feedback than always having the optimal path. In that situation, HPA* might be the best pathfinding algorithm if adapted properly[29].

Table 1: Selection of Algorithms Comparative to A-Star

Method	Pathfinding Case	Reference Paper	Result
GAMMA(Genetic Algorithm)	Maps againts A-STAR	Leigh R, et al.[38]	85% increase in Efficiency
FAR(Flow Annotation Replanning)	Maps againts WHCA-STAR	Wang K, et al.[39]	86% decrease in solution length
Jump Points	Tile Maps againts A-STAR	Harabor D, et al.[2]	Major(>300% rate) increase in node expansion speed
OA(Optimal Anytime)	Grids with randomized obstacles againts A-STAR	Standley T, et al.[18]	67% increase in efficiency
TASS	Tree Maps against previous A*Tree Searching	Korshid M, et al.[22]	300% increase in speed of node expansion and map completion

With all that in mind, there are still several shortcomings of A* and its variations. As seen in Table 1, there are a multitude of Algorithms made for specific scenarios and niches that can outperform A* or it is variations by a magnitude of even several dozen. One of the main disadvantages of A* is its weaker performance in bidirectional searches. In an experiment on bidirectional graph searches, researchers found that bidirectional A* performed very poorly in comparison to its counterpart of BFS. In particular, BFS was incredibly good at larger scale, '16×16' grids, but A* could sometimes pull ahead when dealing with smaller '8×8' grids[30]. Overall, A* is usually consistent when dealing with a variety of problems but requires specialization to shine. It is still incredibly good at a wide variety of searches as long as it's primary weakness of heuristic function dependency is a minimal or not a downside at all.

*3.2 Use of other methods in A**

When it comes to specialization, A* is incredibly easy to refine as necessary. The different ways to customize and adapt A* is the subject of a great number of research papers. Another major factor when dealing with A* is the type of Heuristic function to use. There are a multitude to choose from and all of them can cause vastly differing effects on the algorithm's result. Factors include the type of situation, priorities when dealing with speed or accuracy, size of space and more. In a 2019 test, researchers found that squared heuristic distance resulted in the fastest run time but suffered in terms of accuracy. The paths generated tended to be longer compared to other heuristic functions such as Manhattan distance or straight Euclidean distance. Straight Euclidean distance in this case generated the most efficient path in a timely manner. Hash tables were generally a detriment or worthless started to become a boon when dealing with maps with a grid size of 300 and up[6].

When dealing with adversarial paths, A* can be modified into Bidirectional A* which allows for faster searches. When faced with a grid of size 26x40, modified A* greatly decreased the time it took to traverse the map. A decrease of 900ms and almost 500 less operations was observed, making it markedly more efficient in both time and resources. In terms of general efficiency, HPA* has taken the lead with its incredibly fast speed in a large majority of cases. It manages to work well with grids, mazes and general pathfinding in game map situations while being able to find the optimal path a vast majority of the time. If fast pathfinding time is the main concern, then HPA* is a great base to build upon in many projects[8].

Table 2: Selection of Improvements made to Pathfinding Algorithms

Method of improvement	Pathfinding Case	Reference Paper	Result
LRA*(Local Repair A-STAR)	Randomly Generated Mazes Againts WHCA-STAR	Silver D.[34]	>100% the number of agents at reduced time interval
DepthD A*(Depth Direction A-STAR)	Predetirmined Maps Againts A*and it's variants	Khantanapoka K, et al.[9]	50% increase in speed and 28.6% decrease in node expansion
Distance Metrics(mainly Euclidean)	Clusters	Singh A, et al.[5]	>100% increase in efficiency
dwA*(Dynamically Weighted A-STAR)	Randomly Generated Grids	Thayer J, et al.[23]	10-20% improvement over past iterations
Improved CBS (Conflict Based Search)	Predetirmined Maps Againts CBS	Felner A, et al.[1]	34% increase in speed and 56% decreas in node expansion

All these improvements made to the base A* Algorithm have shown great promise in the search of a yet-undiscovered algorithm that might one day surpass it. Table 2 shows this clearly as many small additions and changes can lead to monumental improvements from the base Algorithm. A* is of great interest in robotics as pathfinding is a major issue there. Any robot that wants to be mobile and navigate the environment to perform their duties must have some sort of pathfinding algorithm to minimize potential collisions which reduces damage to the robot itself, its environment and people around it. A virtual map is generated through some other algorithm, and once the maze has been created, A* can be used to navigate through it[33]. Through visualization and mapping, the main issue of A* can be solved and the robot is able to navigate through previously unknown environments.

3.3 Refining A and its future*

Finally, a completely new type of Heuristic may result in even faster times than before. Most heuristics deal mainly with distance to and from the agent's current location and the goal. This new type of Heuristic focuses on direction, guiding the agent to explore more based on that. On a 150x150 grid, this type of heuristic resulted in very fast times, only 1-2 ms compared to 6-7 ms[14]. Another advantage of the new type of Heuristic is the reduced overhead costs of the algorithm when dealing with larger spaces. One of the problems with A* is its relatively bad handling of very large maps, it will have a high cost in terms of space and overhead. The algorithm is still quite good at smaller maps but will need changes to efficiently handle larger maps[30]. As challenges continue to come and people continue to push the limits, a new shift in how researches approach pathfinding may be coming.

An example of how a shift in A* algorithms may occur purely from changing the weights and introducing smaller algorithms to speed up the process. Researchers working on optimization of virtual human motion planning found that the improved A* algorithm dropped steps from 200 down to 80 and cut the time from 4.359 seconds down to 2.823 seconds[36]. A massive improvement at the cost of slightly increased overhead from the changes and other algorithms involved. The algorithms involved also mitigate the biggest issue of not having complete information, by use of another heuristic which determines if a path will be a dead end or have no chance of optimal results[37]. Innovations like this is what will lead to faster times, efficiency and overall performance improvements. This type of performance increase is not always viable to have however, as the modifications will need to specialize in order to be good instead of detrimental. What works for one problem will not always be good for another.

4. Conclusion

Overall, the traditional, baseline A* algorithm cannot keep up with the increasing demands of pathfinding. However, if properly modified and improved it can still hold its own against the other algorithms. HPA* is a recent innovation that boasts incredibly fast run time at the cost of not always finding the optimal path. Due to the vast amount of different situations that pathfinding faces, it is not possible to have a solution that works for every problem, but with its incredible speed and relatively low overhead cost, HPA* might become the new default option. It is clear that while classical A* is fading out of usage when dealing with complex problems, its improved variants are easily keeping pace with faster records and increased efficiency. Things like adding hash tables or relying on other algorithms to generate a clearer heuristic, modifications which directly address the weaknesses of the algorithm, will be necessary in the future development of A*.

References

1. Felner A, Li J, Boyarski E, Ma H, Cohen L, Kumar T et al. Adding Heuristics to Conflict-Based Search for Multi-Agent Path Finding. International Conference on Automated Planning and Scheduling. 2018.
2. Harabor D, Grastien A. Online Graph Pruning for Pathfinding on Grid Maps. AAAI Conference on Artificial Intelligence. 2011.
3. Barnouti N, Al-Dabbagh S, Sahib Naser M. Pathfinding in Strategy Games and Maze Solving Using A*Search Algorithm. Journal of Computer and Communications. 2016;04(11):15-25.
4. Yap P. Grid-Based Path-Finding. Advances in Artificial Intelligence. 2002:44-55.
5. Singh A, Yadav A, Rana A. K-means with Three different Distance Metrics. International Journal of Computer Applications. 2013;67(10):13-17.
6. Suryadibrata A, Young J, Luhulima R. Review of Various A*Pathfinding Implementations in Game Autonomous Agent. IJNMT (International Journal of New Media Technology). 2019;6(1):43-9.
7. Mehta P, Shah H, Shukla S, Verma S. A Review on Algorithms for Pathfinding in Computer Games. 2nd International Conference on Innovations in Information Embedded and Communication Systems. 2015.
8. Andiwijayakusuma D, Mardhi A, Savitri I, Asmoro T. A Comparative Study of the Algorithms for Path finding to Determine the Adversary Path in Physical Protection System of Nuclear Facilities. Journal of Physics: Conference Series. 2019;1198(9):092002.
9. Khantanapoka K, Chinnasarn K. Pathfinding of 2D & 3D game real-time strategy with depth direction A*algorithm for multi-layer. 2009 Eighth International Symposium on Natural Language Processing. 2009.
10. Kumar N, Kaur S. Bidirectional Graph Search Techniques for Finding Shortest Path in Image Based Maze Problem. International Research Journal of Engineering and Technology. 2019.
11. Huang H. Intelligent Pathfinding Algorithm in Web Games. Advances in Intelligent Systems and Computing. 2020;:250-257.
12. Zheng T, Xu Y, Zheng D. AGV Path Planning based on Improved A*Algorithm. 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC). 2019;.
13. Smołka J, Miszta K, Skublewska-Paszkowska M, Łukasik E. A*pathfinding algorithm modification for a 3D engine. MATEC Web of Conferences. 2019;252:03007.
14. Noori A, Moradi F. Simulation and Comparison of Efficency in Pathfinding algorithms in Games. Ciência e Natura. 2015;37:230.
15. Aria M. New algorithm for digital way-finding map. IOP Conference Series: Materials Science and Engineering. 2018;407:012074.
16. Sharma S, Pal B. Shortest Path Searching for Road Network using A*Algorithm. International Journal of Computer Science and Mobile Computing. 2020.
17. Booba B, Prema A, Renugadevi R. A Hybrid Optimization Algorithm for Pathfinding in Grid Environment. Data Management, Analytics and Innovation. 2019:713-721.
18. Standley T, Korf R. Complete Algorithms for Cooperative Pathfinding Problems. Twenty-Second International Joint Conference on Artificial Intelligence. 2011.
19. Ivanova M, Surynek P, Nguyen D. Maintaining Ad-Hoc Communication Network inArea Protection Scenarios with Adversarial Agents. Thirty-First International Florida Artificial Intelligence Research Society Conference. 2018.
20. Sharon G, Stern R, Felner A, Sturtevant N. Conflict-based search for optimal multi-agent pathfinding. Artificial Intelligence. 2015;219:40-66.
21. Ko J, Lee D. Path Optimize Research used Ray-Tracing Algorithm in Heuristic-based Genetic Algorithm Pathfinding. Journal of Korea Game Society. 2019.

22. Khorshid M, Holte R, Sturtevant N. A Polynomial-Time Algorithm for Non-Optimal Multi-Agent Pathfinding. The Fourth International Symposium on Combinatorial Search. 2011;.
23. Thayer J, Ruml W. Using Distance Estimates in Heuristic Search. International Conference on Automated Planning and Scheduling. 2009;.
24. Sidhu H. Performance Evaluation of Pathfinding Algorithms (Master's thesis, University of Windsor, Windsor, United Kingdom). 2020.
25. Ranjitha M, Nathan K, Joseph L. Artificial Intelligence Algorithms and Techniques in the computation of Player-Adaptive Games. Journal of Physics: Conference Series. 2020;1427:012006.
26. Sun Y, Ding X, Jiang L. Heuristic Pathfinding Algorithm Based on Dijkstra. Proceedings of the 3rd Annual International Conference on Electronics, Electrical Engineering and Information Science (EEEIS 2017). 2017.
27. Tremblay J, Torres P, Rikovitch N, Verbrugge C. An Exploration Tool for Predicting Stealthy Behaviour. AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. 2013;.
28. Barnouti N, Al-Dabbagh S, Sahib Naser M. Pathfinding in Strategy Games and Maze Solving Using A*Search Algorithm. Journal of Computer and Communications. 2016;04(11):15-25.
29. Aria M. New algorithm for digital way-finding map. IOP Conference Series: Materials Science and Engineering. 2018;407:012074.
30. Kumar N. Bidirectional Graph Search Techniques for Finding Shortest Path in Image Based Maze Problem (Master's thesis, Punjab Technical University). 2019;1411-1414.
31. Cui X, Shi H. A-STAR-based Pathfinding in Modern Computer Games. IJCSNS International Journal of Computer Science and Network Security. 2011;11(1).
32. Hoenig W, Kumar T, Cohen L, Ma H, Xu H, Ayanian N et al. Multi-Agent Path Finding with Kinematic Constraints. International Conference on Automated Planning and Scheduling. 2016;.
33. Muhtadin, Zanuar R, Purnama I, Purnomo M. Autonomous Navigation and Obstacle Avoidance For Service Robot. 2019 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM). 2019;.
34. Silver D. Cooperative Pathfinding. First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. 2005.
35. Hong Jung J, Park S, Kim S. Multi-robot path finding with wireless multihop communications. IEEE Communications Magazine. 2010;48(7):126-132.
36. Mathew G. Direction Based Heuristic for Pathfinding in Video Games. Procedia Computer Science. 2015;47:262-271.
37. Yao J, Lin C, Xie X, Wang A, Hung C. Path Planning for Virtual Human Motion Using Improved A*Star Algorithm. 2010 Seventh International Conference on Information Technology: New Generations. 2010;.
38. Leigh R, Louis S, Miles C. Using a Genetic Algorithm to Explore A-STAR-like Pathfinding Algorithms. 2007 IEEE Symposium on Computational Intelligence and Games. 2007;.
39. Wang K, Botea A. Fast and Memory-Efficient Multi-Agent Pathfinding. Eighteenth International Conference on Automated Planning and Scheduling. 2008;.
40. Sturtevant N. Benchmarks for Grid-Based Pathfinding. IEEE Transactions on Computational Intelligence and AI in Games. 2012;4(2):144-148.