Proceedings of the 4th International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/2026.38873

A and Its Variants for Pathfinding in Grid-Based Game Environments: Techniques and Trends*

Siyuan Wang

School of Computing and Software Systems, The University of Melbourne, Melbourne, Australia
siyuan_wang2021@163.com

Abstract. This review examines A* and its major variants for pathfinding in grid-based game environments, this review include Hierarchical Path-Finding A* (HPA*), Jump Point Search (JPS), Theta*, and Constrained Jump Point Search (CJPS). It compares their search mechanisms, performance, and suitability for different game-map conditions. The reviewed evidence indicates that standard A* remains practical for small and simple maps, while HPA* supports a long-distance search through hierarchical abstraction. JPS improves search efficiency on uniform-cost grids, Theta* produces more natural any-angle paths, and CJPS is more useful when environmental changes create repeated search work. However, differences in maps, implementations, hardware, preprocessing requirements, and evaluation measures make direct comparison between studies difficult. The review summaries that no single method is optimal for every game environment. Algorithm selection should instead balance runtime, memory use, preprocessing cost, path optimality, movement quality, and adaptability to map changes. It also identifies standardised benchmarking and learning-assisted guidance as important directions and future research.

Keywords: A* algorithm, game pathfinding, Jump Point Search, hierarchical pathfinding, any-angle pathfinding

1. Introduction

Pathfinding is a significant part of artificial intelligence in digital games. It leads non-player characters, enemies and players to move from one location to another location which avoids obstacles and minimises the total cost. An effective pathfinding influences not only whether a character can reach its destination, but also the responsiveness, visual realism and overall quality of gameplay. However, pathfinding in games is subject to constraints that are not always equally important in other domains. A game may require hundreds of units to calculate their paths within a limited frame-time budget, and doors, buildings, terrain and other characters can continuously change the environment. Therefore, a suitable game pathfinding algorithm not only returns the mathematically shortest route, but also balances computational efficiency, memory consumption, path suitability, scalability and adaptability.

The A* algorithms is one of the most important foundations for navigating in grid-based game environments. Each candidate node will be evaluated by adding expected cost to the destination and actual cost from the starting location. According to this heuristic function, A* algorithms are able to

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.38873

search more selectively than any other algorithms which explore possible routes without considering the direction of the goal [1]. However, the result quality from the heuristic function has a significant impact on the size and complexity of the map. In some large maps, algorithms will search more unnecessary nodes and also waste a large amount of considerable processing time and memory. In some dynamic environments, the changing obstacle or movement cost will lead the path to be calculated again. Due to these limitations, several variants of A* algorithms have been developed with the goal of enhancing search speed, lowering memory usage, supporting huge maps, or better adapting to environmental changes.

In order to address these previous limitations of standard A* algorithms which mentioned above, previous study have explored several significant improvement directions. Hierarchical Path-Finding A* (HPA*) which was raised by Botea et al. [2] to reduce the complexity of pathfinding on large grid-based maps. Not like HPA* Jump Point Search (JPS) which improves efficiency through map abstraction, proposed by Harabor and Grastien [3], focused on reducing unnecessary node expansions in uniform-cost grid-based maps. While JPS improves the search efficiency by reducing unnecessary node expansions, there is another improvement algorithm which solves the direction restrictions in traditional grid-based pathfinding and generates any angle of paths. Nash et al. [4] proposed Theta* to improve these promotions mentioned above. Furthermore, CJPS was proposed to reduce repeated scanning and unnecessary node expansions that may remain in the original JPS process, particularly in changing grid environments [5]. Each method improves a different aspect of A*. Therefore, no single method is the best choice for every situation.

This review will focus on the A* and its major variants including HPA*, JPS, Theta*, CJPS, and explain their applicability on different grid-based. The remainder of the review will introduce the algorithm which selects, compares their application and performance in games, discusses current challenges, future directions, and ends with a summary of the key conclusion.

2. Overview of the techniques

2.1. Standard A* algorithm

A* algorithm is a heuristic search algorithm which aims to find a minimum-cost path between a starting node and a destination node in a graph. Unlike uninformed search methods that explore the search space without guidance, A* uses a heuristic function to estimate which candidate nodes are more likely to lead to the target. It evaluates each node n through the function $f(n) = g(n) + h(n)$, where $g(n)$ represents the actual cost from the starting node to the node n, and $h(n)$ means the estimation of remaining cost from the node n to the destination [1].

During the search process, A* keeps an open list in order to record the candidate nodes which are available for exploration, and also keeps a closed list to record the nodes that have already been examined. The starting node is added to the open list. At each searching process, the node which has the lowest $f(n)$ will be selected and expanded, then calculate and update the cost of the neighbour of the chosen node. If you find a route which has a lower cost than the previously discovered node, then change its cost and parent information. This process will repeat until the target node is selected for expansion. The final path will be created by the recorded parent nodes from the objective back to the starting node [1].

In game pathfinding scenarios, A* provides a balance between path quality and search efficiency. For the suitable choice of the evaluation function f, A* algorithm guarantees to find the optimal path of a preferred target node, which is acceptable [1]. That means When using an appropriate heuristic function, the algorithm will find an optimal path, and also avoid brute exploration which is related to

DOI: 10.54254/2755-2721/2026.38873

uninformed search. Although it has a clear evaluation mechanism which is well implemented in grid-based games, the performance mainly depends on the quality of the heuristic function and the size of searching space. Furthermore, standard A* algorithm may not have a better performance on large maps, Botea et al. [2] also summarised that on large maps, A* algorithm might extend, store and calculate more nodes, that means it will cause an increasing trend in processing time, and memory consumption. Moreover, the moving cost and the changing of obstacles might need repeating computation.

2.2. Hierarchical Path-Finding A* algorithm

Hierarchical Path-Finding A* (HPA*), an algorithm which can reduce the complexity of search space by dividing maps into interconnect local cluster, and precompute optimal cost of traveling across each cluster, and then pathfinding will perform on the higher abstraction level, treat each cluster as single step and then combine them into a larger clusters to build a multi-level hierarchy [2].

For the searching process of HPA* Botea et al. [2] state that insert starting point S and G into the abstract graph and connect the border of their clusters, then respectively search the both actual paths from S and G to the border of the cluster which contains S and G. Then using A* to search the path between S and G in an abstract graph, if a detailed route is required, the abstract path can subsequently be refined into a sequence of moves on the original grid.

For the result, Botea et al. [2] reported that the experimental findings demonstrate that the hierarchical abstraction will lead a low research effort while keeping approximately optimal paths, and when the solution length is small, A* is mildly better than HPA*. That means for short paths, standard A* may perform slightly better because the overhead of inserting the start and goal nodes into the abstract graph can exceed the savings achieved by HPA*, they also present that when the problem map is large, more hierarchy format can reduce the search effort, but it may cost more storage and preprocessing time.

2.3. Jump Point Search (JPS)

In uniform-cost grid maps, the major source of unnecessary computation is path symmetry; several routes connect the starting point and ending point with the same cost, but the difference is the order of movement. JPS improves grid-based pathfinding by pruning the equivalent paths and selecting the significant nodes for extension [3].

For the searching process, Harabor and Grastien [3] explain that according to the current node direction, JPS prunes the neighbours around the current node. c Natural and forced neighbours are kept, but neighbours that can be accessed by comparable or less expensive routes without going via the current node are eliminated. Furthermore, JPS continues choosing direction until it reaches the destination or crosses a node that has a forced neighbour. After being recognised as a jump point, this node will add in searching for extra extension.

From the game pathfinding perspective, JPS provides several advantages. This selective expansion mechanism can reduce unnecessary search work. For regular, large, and some which contain several symmetric path maps, JPS will lead an effective searching process, because such maps provide more opportunities for pruning. Furthermore, it maintains optimality without requiring additional memory, it is fast without requiring preprocessing [3]. However, skipping intermediate node expansions still remains computation, because intermediate grid cells may still need to be examined when searching for jump-point successors [3].

DOI: 10.54254/2755-2721/2026.38873

2.4. Theta*

Standard A* restricts path segments to grid edges, which limits the available movement directions and may produce a path which is longer and less realistic than the actual path of terrain. Theta* directly solve this limitation through allowing path segments to follow any heading when an unobstructed straight-line connection is available. That is the reason why it has been classified as any-angle pathfinding algorithm [4].

Basic Theta* remains the whole searching frame of the standard A*, but modifies the selection method of parent nodes of neighbouring nodes. When current extend point does not consider the an unexpanded neighbour (s'), algorithm will check (s') the line of sight between (s) and (parent(s)), if the direct path through (parent(s)) is shorter than the path which through (s), (parent(s)) will become the parent of (s'). This mechanism allows the path to directly cross the grid and not only go along the border of each grid [4].

From the game pathfinding perspective, this parent-update mechanism can reduce unnecessary changes in direction which are caused by the limitation of grid direction, and moreover, it can provide shorter and more natural paths for characters in game. However, Basic Theta* can not guarantee the shortest path in the game. In addition, the extension cost of each node might increase with the number of grid cells examined, because of frequent line-of-sight check [4].

2.5. Constrained Jump Point Search (CJPS)

Constrained Jump Point Search (CJPS) is the improvement algorithm for JPS, although JPS reduces node expansions by pruning symmetric paths, it can still perform redundant work during successor generation. In particular, overlapping diagonal recursions may repeatedly scan the same areas of a map. JPS may also generate and expand nodes reached with suboptimal costs. To address these problems, Zhao et al. proposed Constrained Jump Point Search (CJPS), a purely online extension of JPS based on geometric constraints [5].

When a cardinal scan discovers a previously generated jump point, CJPS creates a constraint containing its location, direction and cost information. This constraint establishes a dynamic limit for later scans performed in the same direction during diagonal recursion. It can stop a scan from entering an area that can be reached at a lower cost from the previously discovered jump point. Constraints can also be updated or removed as the search progresses. Through this mechanism, CJPS attempts to reduce repeated scanning and prevent the generation of some suboptimal successors while preserving optimality [5].

From a game-pathfinding perspective, CJPS is particularly relevant to dynamic grid environments because it operates online and does not depend on map preprocessing. However, maintaining and applying constraints introduces additional local reasoning. When a map contains little redundant work that can be eliminated, this overhead may cause CJPS to perform less efficiently than the original JPS. Its advantage therefore depends on map topology and the amount of repeated scanning present in the search [5].

3. Results and applications

3.1. Reported results comparison

The chosen pathfinding algorithm enhances the performance in various ways, but only execution time can not be used to compare the given outcomes. Therefore, this section will evaluate them in

DOI: 10.54254/2755-2721/2026.38873

search efficiency, memory consumption, the quality of path, processing requirements, and adaptability to different grid-based environments. This need for a broader comparison was mentioned by Harabor et al. [6], they explained that different pathfinding methods are based on different map representations, development and restrictions. Furthermore, the experiment evaluations are often application specific, use unavailable maps, and only compare compatible algorithms. Therefore, it is difficult to directly compare the performance from different studies.

Search efficiency is one of the most important factors when evaluating pathfinding algorithms for games, especially in need to compute a large amount of routes in a limited processing time. A* and JPS provide a significant comparison, because they are both suitable to find the optimal path on grid maps, despite the fact that their examination of the search space varies greatly. In the experiment Harabor et al. reported that in each map group of Iron Harvest, in table 1, A* require approximately hours to solve 70k queries, but JPS only completed these queries in approximately 5 minutes, and both algorithms returned optimal paths, therefore, under the tested conditions JPS reduces running time without compromising path optimality [6]. This comparison shows that path optimality can not determine real performance because both algorithms might require significantly different amounts of computation. Overall, for games which include frequent pathfinding queries on suitable maps, JPS might offer a more practical balance between solution accuracy and response time than standard A*.

Table 1. Cumulative runtime results for A* and JPS in Iron Harvest map groups

Instance	Queries	A*	JPS
cha	8 k	5,891	53.05
pol	12 k	4,757	47.24
rus	14 k	6,305	62.62
sax	14 k	7,172	56.44
mp	20 k	6,055	50.02
end	2 k	2,948	25.60
all	70 k	33,130	295.0

Note: Values are cumulative runtimes in seconds. The data are selected from Harabor et al. [6].

The comparison above mainly focus on the efficiency of A* and JPS under static map conditions, whereas the adaptability to map changes provide another basis for comparison. Zhao et al. found that compared with the JPS in static environments, CJPS does not have performance advantages. However, as the data shows in table 2, after introducing dynamic map changes, CJPS achieved lower cumulative runtimes and a maximum speedup of 7.34 in the tested domains [5]. These results show that the relative performance of JPS and CJPS changes along with static or dynamic changes.

Table 2. Selected cumulative runtimes of CJPS and JPS in static and dynamic environments

Change rate r (%)	Domain	CJPS (s)	JPS (s)	Speedup
0.0	bgmaps	4.95	3.49	0.71
0.0	iron	167.35	175.17	1.05
0.1	bgmaps	6.49	7.51	1.16

DOI: 10.54254/2755-2721/2026.38873

Table 2. (continued)

0.1	street	28.09	98.57	3.51
0.1	iron	341.78	2508.75	7.34

Note: Values represent cumulative runtime. The value r = 0.0 represents static conditions, and the value r = 0.1 represents dynamic map changes. The data are selected from Zhao et al. [5].

Instead of a single performance ranking, the remaining results show various trade-offs. HPA* exchanges preprocessing time and storage space to reduce the workload of online search [2]. JPS avoids these preparation costs and also maintains the optimality of uniform cost grid [3]. Theta* prioritizes the path quality at any angle, but also sacrifices the guaranteed global optimality [4]. Therefore, the runtime result should be considered with memory consumption, processing requirements and path quality

3.2. Applications in game environments

The actual choice of the pathfinding algorithm depends on the structure of the game map and the movement requirement of the character. For small and simple static maps, standard A* may provide sufficient performance. More complicated methods may introduce unnecessary preprocessing or implementation costs, and a small search space should limit the benefits brought by hierarchical abstraction. For large maps, HPA* is a better choice which involves long-distance queries, because its experimental advantages increase with the length of the solution [2]. In stable game environments, the precomputed hierarchy can also be used repeatedly across multiple searches. For uniform-cost grid maps, JPS performs pathfinding online, preserving optimality and does not require preprocessing or additional memory [3]. However, for static maps which receive frequent pathfinding queries, JPS can be combined with goal-bounding data to improve the online performance, although it will lead to additional preprocessing and storage costs [7]. Therefore, there is a significant question for developers, whether the faster online queries provided by the extension based on preprocessing are worth the additional preparation and storage requirements.

Dynamic environment and movement quality of character impose different requirements on pathfinding algorithms. When obstacles change during pathfinding searching, CJPS can reduce the repeated scanning in the original JPS process. Zhao et al. [5] reported that compared with JPS, CJPS does not show obvious advantages over JPS in a static environment, but it provides a better performance, after introducing CJPS in a dynamic environment. In contrast, Theta* may be more suitable for games that prioritise natural-looking character movement. The experimental results showed that its paths contained fewer heading changes and were more realistic-looking than those produced by Field D*. However, neither version of Theta* was guaranteed to find the shortest path [4]. Overall, the improvement of online speed, adaptability, and path quality should consider the balanced trade-off among preprocessing costs, memory usage, path optimality, and suitability for different map types.

4. Discussion: challenges and future directions

4.1. Current challenges

The practical application of A* variants in grid-based game environments still need to face several challenges. First, the results of different study reports are difficult to compare directly, because their

DOI: 10.54254/2755-2721/2026.38873

maps, hardwares, movement rules, and implementations are all different. Iron Harvest benchmark partially solves this problem by providing 35 game maps and a common set of publicly available problem instances [6]. However, it still needs a broader standardized evaluation to compare the algorithm in different game environments and execution conditions. Second, the adaptability is still limited. In some particular assumptions, HPA* and JPS are most effective, for the changing obstacle they might require repeated searching. Although CJPS reduces the work after map changes, the advantages are still different between static and dynamic environments [5]. Third, balancing search efficiency with storage, preprocessing, and path quality is a further challenge. Precomputed guidance can reduce the amount of online searching, but it may need more preparation and memory. A synthesised heuristic generally becomes less effective at guiding A* when applied to a map other than the one for which it was originally developed [8]. To solve these limitations requires future research to evaluate pathfinding methods in the same experimental conditions and different map structures.

4.2. Future directions

Based on some of the limitations mentioned above, future research should focus on a way of evaluation with more adaptive guidance and more consistent. Bulitko et al. [9] automatically synthesised compact, map-specific heuristic functions for video-game pathfinding. Under their experimental conditions, the synthesised heuristics reduced the search effort of weighted A* while maintaining comparable solution quality, demonstrating that automatic heuristic design can improve performance without the large memory requirements of precomputed tables. Learning-based search provides another direction. Neural A* achieved a better balance between search efficiency and solution optimality than existing state-of-the-art data-driven planners [10]. However, learned guidance must still be tested for generalisation, training cost, and reliability on maps that differ from the training data. Future studies should therefore compare classical and learning-assisted methods using the same maps, movement rules, and hardware. Evaluations should report runtime, memory, preprocessing or training cost, path length, and movement smoothness together. This would help developers determine whether improvements measured in experiments remain useful under practical game constraints.

5. Conclusion

This review examined A* and several of its major variants for pathfinding in grid-based game environments. The findings show that no single algorithm is suitable for every game scenario. Standard A* remains a reliable choice for small and simple maps. HPA* can reduce online search effort for large and stable environments, although it requires preprocessing and additional storage. JPS provides efficient and optimal search on uniform-cost grids, while CJPS is more beneficial when dynamic changes create redundant scanning. Theta* can generate more natural-looking any-angle paths, but does not guarantee global optimality. Therefore, algorithm selection should reflect map size, environmental stability, movement quality, memory availability and preprocessing requirements. Future research should develop more consistent benchmarks and examine whether automatically generated or learned guidance can generalise across different game maps. Overall, practical game pathfinding requires a balance between computational efficiency, path quality and adaptability rather than selecting an algorithm solely according to runtime or path optimality. In addition, the review considers whether recent preprocessing and learning-based approaches can address the limitations of conventional search methods under practical game constraints.

DOI: 10.54254/2755-2721/2026.38873

References

[1] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics, 4*(2), 100–107. https://doi.org/10.1109/TSSC.1968.300136

[2] Botea, A., Müller, M., & Schaeffer, J. (2004). Near optimal hierarchical path-finding. *Journal of Game Development, 1*(1), 7–28.

[3] Harabor, D., & Grastien, A. (2011). Online graph pruning for pathfinding on grid maps. *Proceedings of the AAAI Conference on Artificial Intelligence, 25*(1), 1114–1119. https://doi.org/10.1609/aaai.v25i1.7994

[4] Nash, A., Daniel, K., Koenig, S., & Felner, A. (2007). Theta*: Any-angle path planning on grids. *Proceedings of the AAAI Conference on Artificial Intelligence*, 1177–1183.

[5] Zhao, S., Harabor, D., & Stuckey, P. J. (2023). Reducing redundant work in jump point search. *Proceedings of the International Symposium on Combinatorial Search, 16*(1), 128–136. https://doi.org/10.1609/socs.v16i1.27291

[6] Harabor, D., Hechenberger, R., & Jahn, T. (2022). Benchmarks for pathfinding search: Iron Harvest. *Proceedings of the International Symposium on Combinatorial Search, 15*(1), 218–222. https://doi.org/10.1609/socs.v15i1.21770

[7] Hu, Y., Harabor, D., Qin, L., & Yin, Q. (2021). Regarding goal bounding and Jump Point Search. *Journal of Artificial Intelligence Research, 70*, 631–681. https://doi.org/10.1613/jair.1.12255

[8] Bulitko, V., Wang, S., Stevens, J., & Lelis, L. H. S. (2022). Portability and explainability of synthesized formula-based heuristics. *Proceedings of the International Symposium on Combinatorial Search, 15*(1), 29–37. https://doi.org/10.1609/socs.v15i1.21749

[9] Bulitko, V., Hernandez, S. P., & Lelis, L. H. S. (2021). Fast synthesis of algebraic heuristic functions for video-game pathfinding. *Proceedings of the 2021 IEEE Conference on Games (CoG)*, 1–8. https://doi.org/10.1109/CoG52621.2021.9618995

[10] Yonetani, R., Taniai, T., Barekatain, M., Nishimura, M., & Kanezaki, A. (2021). Path planning using Neural A* Search. *Proceedings of the 38th International Conference on Machine Learning, 139*, 12029–12039. https://proceedings.mlr.press/v139/yonetani21a.html