On path planning with waypoints as benchmark for the planning abilites of LLMs*

Laura Mandow[1] and Lorenzo Mandow[2][0000−0002−8141−498X]

[1] Independent researcher. Granada, Spain lauramandow@correo.ugr.es
[2] Universidad de Málaga, Andalucía Tech, Dpto. LCC, Spain lmandow@uma.es

Abstract. This paper critically investigates LLM-A*, a recent path planner for 2D grid environments using LLM-generated waypoints. We analyze performance on standard benchmark problems when the LLM is substituted with an oracle that generates waypoints of varying quality.

Keywords: path planning · Grid search · Waypoints · LLMs

1 Introduction

Machine learning and problem solving have traditionally been two major driving goals in artificial intelligence (AI) research.

Learning is a key human cognitive ability. *Foundation models* in machine learning have successfully captured large amounts of general knowledge using *transformer* architectures trained with self-supervision over massive datasets [19]. Large language models (LLMs) in particular, are highly successful in natural language processing (NLP) when trained on text datasets. Most models use an intensive *pre-training* stage to learn representations that allow natural language *text generation*. These are latter adapted (or transferred) to learn particular tasks through a less computationally intensive second training stage of fine-tuning.

Recent research claimed that such generative pre-trained transformers (GPTs) can show improved performance without the fine-tuning stage in NLP tasks (like question answering or completing translations), when size and training are properly upscaled. This is known as *zero-shot task transfer*. Empirical performance improved when the input query (or prompt) to the model included one or a few examples of the task to be solved (typically between 10 and 100 input-output pairs), limited by the model's input size (or context window). These are known as *one-shot* or *few-shot* task transfer respectively [1]. This suggested GPTs could be understood as exhibiting *meta-learning*, a term used in the literature to denote the ability of "learning to learn" a new task from a few examples [3].

Other authors have hypothesized this emerging ability could be extended to sufficiently large LLMs beyond NLP tasks and/or using different techniques of *prompt engineering*, like *chain-of-thought* (CoT) prompting [20], which aims at

* Laura Mandow is student at Universidad de Granada (Spain). Lorenzo Mandow is supported by the Spanish Ministry of Science, Innovation and Universities through the research project PID2024-1620110B-I00.

extending few-shot prompting to reasoning tasks. A chain of thought is a question accompanied by a sequence of reasoning statements leading from question facts to some conclusion. Results showed that eight-shot CoT prompts allowed LLMs to improve empirical performance to produce CoT answers for some arithmetic, commonsense, and symbolic reasoning tasks on multiple benchmarks.

This suggested CoT reasoning is also an emergent property of sufficiently large scale LLMs. For symbolic reasoning tasks (concatenation of last letters of names; determining the heads/tails state of a coin after a number of flip/no flip actions), some generalization ability was observed (three or four words/actions) over the lengths (two) of the provided exemplars in the CoT prompt. However, authors acknowledged their research did not answer whether the neural network was actually "reasoning", a question currently under active debate [8][10][18].

Problem solving is another key human cognitive ability. This involves finding the appropriate means to achieve certain ends or goals, usually under constraints and limited resources [11]. Different strategies have been studied for problem solving in AI, under the paradigm of the *heuristic search hypothesis* [12].

One relevant example is *classical planning*, which advocates logic-based formal representation languages and domain independent inference mechanisms [4]. The above mentioned research on the reasoning capabilities of LLMs has naturally posed the question whether LLMs exhibit general capabilities for plan generation [15][18]. The work of [17] has suggested that, while the ability of LLMs to generate executable plans is poor, "LLM-generated plans can improve the search process for underlying sound planners". This idea is now being extended and put to the test in different problem solving tasks.

Path planning is one of such tasks that is generally efficiently solved using specialized representations and heuristic graph search algorithms [2]. The goal is to find an adequate path from start to goal locations avoiding obstacles in a given space. Square eight-neighbor grids are a frequent space representation formalism [16], and A* search [7] is usually considered a baseline solving algorithm.

The recent work of [9] has proposed the use of an LLM with CoT prompting to generate a sequence of intermediate waypoints in a 2D grid path planning task. Then, A*-like search is run using a waypoint-guided heuristic to try to avoid obstacles in path generation. This system received the name of LLM-A*.

This paper critically investigates the general architecture of LLM-A* in an LLM agnostic way. More precisely, we analyze what could be expected if the LLM were substituted with an oracle that generates obstacle-avoiding waypoints of varying quality. We compare several path planning alternatives with and without waypoints and address the following research questions: which algorithm is best? how do waypoints, their quality, and placement, influence the result?

This paper is organized as follows. Section 2 briefly describes grid-based path planning state spaces and algorithms, and introduces relevant aspects of LLM-A*. Section 3 describes our oracle for waypoint generation. Section 4 describes our experimental setup and results. These are discussed in section 5. Finally, conclusions are summarized and future lines of research are presented.

2 Antecedents

2.1 Grid-based path planning

Grid-based path planning consists of a grid environment where each cell is either an obstacle or passable, possibly with different degrees of cost. An adequate obstacle-free path needs to be found between start and goal cells, moving between neighboring cells according to some definition of neighborhood. Standard benchmark problem sets are available[3] and frequently used for research in graph search algorithms [16]. These are provided in different map sizes, accompanied by sets of randomly generated problem instances (origin and destination pairs).

In this paper we consider 2D grid maps where each square cell in the map is either strictly passable or an obstacle. We define the *standard grid state space* as follows. Agents are allowed to move from their current cell to adjacent passable cells using cardinal or diagonal moves (i.e. eight-neighbors). Diagonal moves that pass between two obstacles or cut a corner are not allowed. This is done to account for the fact that moving agents occupy some volume in space. These are the benchmarks that most closely resemble the problems addressed by [9].

We will consider shortest path search using algorithm A* over the standard grid state space as a baseline. This is the framework proposed by [9]. A* is an efficient general purpose heuristic graph search algorithm. It keeps a tree rooted at the start state that records the shortest known paths to each visited state. States where the tree can grow further are said to be *open*. The algorithm iteratively selects an open state until the goal is selected or no more open alternatives remain. The successors of the selected state are generated and considered for inclusion in the tree and the open set. A* always selects an open state s with minimum value of $f(s) = g(s) + h(s)$, where $g(s)$ is the cost of the path in the tree to s and $h(s)$ an heuristic estimate of the cost from s to the goal. Using an optimistic $h(s)$, A* always finds an optimal solution path [13]. In 2D grid search the *octile* heuristic, that estimates path costs assuming no obstacles and cardinal and diagonal moves, is a usual choice.

We will also find interesting to consider *weighted A** [14] (wA*), which is a simple variant of A* that selects open alternatives that minimize $f(s) = g(s) + wh(s)$. When $w > 1$ the search becomes increasingly greedy, which can be faster in domains like grid search [21] at the cost of sacrificing path optimality.

Finally, we consider JPS (Jump Point Search) [5][6], a state-of-the-art specialized procedure for uniform-cost grid-based search. JPS re-defines the state space to be searched, i.e. the way successor states are generated. For any given state, only *natural* directions to neighbor cells that are not better reached by its parent are considered. These are supplemented with *forced* directions, when grid obstacles prevent direct paths to some neighbor cells from the parent. Additionally, these natural and forced directions are explored until a cell is reached which is a potential turning point or the goal. Only in such cases, that cell becomes a successor state. We will call this the *JPS state space*. A* in the JPS state space using the octile heuristic guarantees a minimum cost solution path.

[3] https://movingai.com/benchmarks/

2.2 LLM-A*: path planning guided by LLM generated waypoints

LLM-A* [9] is a path planner that combines LLM prompting with A* search. Given a pair of start and goal points in a 2D grid with obstacles, the LLM provides a sequence $(p_0 = start, p_1, \ldots, p_n = goal)$ of passable *waypoints* (including both start and goal) meant to avoid the obstacles. Then, A* searches for a solution path heuristically biasing the search towards the waypoints in the sequence and, hopefully, around the obstacles. LLM-A* was tested on 50×30 grid environments scaled from 1 to 10, each with two horizontal and one vertical barriers, and randomly generated start and destination cells. Reported results show 90% of optimal paths, and improved operations and storage performance[9].

Three prompting techniques were tested: few-shot promtpting, CoT prompting, and recursive path evaluation (RePE). This latter technique elaborates the chain of thought, sequentially detailing the purpose of each waypoint and why more waypoints may be needed to avoid further obstacles. More precisely, such prompts included "standard 5-shot examples, chain of thought with 3-shot, and recursive path evaluation with 3-shot"'. The system was evaluated with two LLMs (GPT-3.5-TURBO and LLAMA3-8B-16bit), though the approach should hopefully improve with larger models according to the meta-learning hypothesis.

LLM-A* uses a variant of A* with a modified dynamic heuristic that takes into account the waypoints, and changes as these are sequentially dicovered during the search. Let $h_{A*}(s)$ be the standard estimated distance from each state s to the goal used by A*. LLM-A* adds to this heuristic a new term $c(s, p_i)$, that estimates cost from state s to the current waypoint p_i. When the search starts, the current waypoint is set to p_1. As soon as a path to the current waypoint p_i is added to the search tree, it is updated, i.e it is set to the next waypoint p_{i+1} in the sequence. All heuristic values are then recalculated and updated, and search proceeds with the updated heuristic until the new reference waypoint is again reached. Therefore, LLM-A* uses in fact a sequence of heuristics as the search progresses: $h_i(s) = h_{A*}(s) + c(s, p_i), i = 1, \ldots n$.

We note that, in the final part of the search, when the current waypoint is the goal, LLM-A* searches the grid using as evaluation function $f(s) = g(s) + 2h_{A*}(s)$, i.e. it actually uses weighted A* with $w = 2$. The work of [9] compares LLM-A* to A*, but in light of the above remark, we feel LLM-A* is better understood as a preliminary exploratory search stage that biases the search sequentially towards each waypoint, followed by a final search stage with wA*.

The original LLM-A* proposal runs the proposed search algorithm over the standard grid state space [9]. The authors claim that their proposal "retains the versatility of the original A*, making it suitable for a wide range of pathfinding tasks across various environments", explicitly citing JPS. However, we note here that most grid cells are not states in the JPS state space (see section 2.1). Therefore, it is a really challenging task to accurately predict obstacle avoiding waypoints that are as well jump point states for a given problem instance, unless these are previously enumerated. The proposed updating mechanism for the dynamic heuristic scheme relies on finding paths to the waypoints, i.e. these would *need to be jump points*, or the heuristic would never be updated.

One alternative would be to explicitly include the waypoint cells as jump point states. However, jump point states may typically be quite far away from their parents. This means a jump point may be generated when the search is still active in a far away region of the state space, questioning the rationale of the LLM-A* heuristic scheme. Thus, there does not seem to be a single straightforward way to effectively combine search in the JPS state space with LLM-A*.

3 Oracles for waypoint generation

In the context of this research, planning with waypoints assumes that a useful sequence of waypoints is provided (by an LLM) in the right order. In order to investigate the influence of such sequence in search efficiency, we use an LLM-agnostic oracle to generate candidate sequences of different quality. We also investigate possible differences in waypoint placement. In LLM-A* [9] one waypoint per avoided obstacle is presented in the CoT exemplars and expected in the CoT result. These were intentionally located close to obstacle corners, where turning points of the path should appear. We devise two waypoint generation procedures. The first one is meant to emulate the behavior described by [9], and the second one tries to generate waypoints between the corners of obstacles.

To rigorously evaluate different search algorithms, we require a standardized and reproducible method for generating waypoints. While the work of [9] used LLMs to generate waypoints, we explicitly avoid this approach. LLM capabilities evolve rapidly, and benchmarking algorithms against waypoints generated by a specific model risks tying our results to a transient baseline. Instead, our goal is to test the fundamental efficiency of the path planning systems themselves.

We extract "ideal" waypoints for each problem instance directly from an optimal A*-generated path and systematically degrade them. This methodology ensures that we are evaluating the heuristics' ability to leverage navigational hints of varying quality, independent of the particular external generator's linguistic or spatial reasoning flaws. Our extraction strategy is based on the spatial relationship between the optimal path and the obstacles in the environment. Rather than blindly sampling points along the path, we target critical navigational junctions by identifying obstacle boundaries.

First, an optimal path P is computed from start to goal node using A* over the standard grid state space. Next, we identify cell clusters from obstacle's bounding boxes across the grid map, which effectively outline the bounding geometries of the non-passable regions. Let $C = \{C_1, C_2, \ldots, C_k\}$ represent the set of these obstacle clusters. For each cluster $C_j \in C$, we evaluate its distance to every state $s_i \in P$. We define the distance from a state to a cluster as the minimum Chebyshev distance to any cell within that cluster. A waypoint is generated for each obstacle along the path. If the minimum distance falls within a predefined obstacle radius threshold R_{obs}, the state is considered a candidate. To prevent waypoint crowding (e.g., choosing several waypoints for the same obstacle), we enforce a minimum spacing constraint S_{min} between selected waypoints. The state that minimizes the distance to the cluster, while satisfying the

spacing constraint, is selected as the waypoint for that specific obstacle. Start and goal states are explicitly added to the waypoint list, just like in [9].

To simulate the imperfect nature of externally generated or real-world waypoints, we apply a controlled perturbation strategy to the ideal waypoint set. The process is governed by two parameters: a perturbation probability p and a maximum octile radius r. For every waypoint w (excluding the protected start and goal states), we sample a random variable $q \sim U(0, 1)$. If $q \leq p$, the waypoint is subjected to perturbation. A new location w' is uniformly sampled from the set of all valid grid cells within an octile distance r from w.

If the newly sampled location w' lands on a non-passable (obstacle) cell, the waypoint is discarded. This design choice replicates the one used in [9], allowing us to evaluate how gracefully the heuristics handle missing or geometrically invalid waypoints. Due to the nature of the waypoint generation algorithm, most of this waypoints are located nearby a "corner" of one of the obstacles.

We additionally explore the possibility of using waypoints for each avoided obstacle that are far from the corners. Therefore we calculate for each problem instance an alternative set of *midpoints*. These are calculated as points in an optimal path that lay between each of the waypoints returned by the previous "corner" waypoint generation procedure, except for the last two (so we keep the same ideal number of waypoints per instance). Midpoint and corner waypoints calculated for a sample problem instance are shown in Figure 1.

4 Experimental evaluation

4.1 Benchmarks and algorithms

The experiments of [9] included custom maps with few simple longitudinal obstacles. Our experiments are carried out over three different 512×512 2D grid maps [16] from the "real-world benchmarks" set: "Berlin_1_512", "Boston_0_512" and "Paris_1_512". These are somewhat larger than the largest maps tested by [9], and contain much richer and structured obstacles representing buildings among city streets. Obstacles represent approximately rectangular areas, and are possibly the closest match to the longitudinal obstacles used in context prompting by [9]. However, obstacles are not perfectly rectangular nor orthogonally aligned with map coordinates. We introduced the Paris map, which contains curved obstacles, anticipating it might be a qualitatively more challenging scenario for LLM prompting and reasoning. For all maps we consider the standard problem instance sets. Costs are 1414 for diagonal moves and 1000 for cardinal ones.

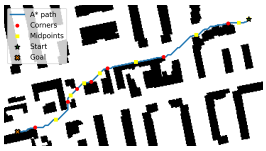


Fig. 1. Corner and midpoint waypoints with no perturbation for a sample path.

We consider A*, weighted A* ($w = 2$), LLM-A*, and a naive waypoint algorithm we call A*Way. This latter algorithm carries out a sequence of A* searches between waypoints, connecting them to build the final solution path. Note that A*Way will provide a solution that forcibly passes through all waypoints. This is in contrast to LLM-A*, which uses the waypoints only to force the exploration of their vicinity, and may provide solutions that do not traverse (all of) them. We test all algorithms in the standard grid and JPS state spaces, except LLM-A* which is tested only on the grid state space as discussed in section 2.2. In the following, we refer by LLM-A* to the combination of their proposed search algorithm with our oracle-generated waypoints (not actually generated by any particular LLM). Reported runtimes do not consider the time taken to generate the waypoints in any case. For the waypoint-based algorithms (LLM-A* and A*Way) we consider the same waypoints generated for each problem instance as described in section 3 using no perturbation and low ($p = 0.1, r = 2$), medium ($p = 0.5, r = 2$) and high ($p = 0.9, r = 2$) perturbation.

We use our own implementation of all the algorithms in Python sharing as much code as possible between them[4]. We store search tree nodes in a 512×512 array for constant-time access and checking of duplicate states. We sort the open set using a *heapdict* which allows efficient key updates. This is particularly important for LLM-A*, which needs to re-sort the open set each time the current waypoint (and, in consequence, the heuristic) is updated during the search.

4.2 Results

We measure the time taken by each algorithm in each setting, as well as the ratio of its solution cost to the optimal cost (i.e. the one found by A*). We label each alternative combining the state space (Grid or JPS) with the base search algorithm (A*, wA*, A*Way)[5]. LLM-A* always uses the grid state space.

Figure 2 plots averaged values for both measures for each alternative and map for corner waypoints without perturbation (top). It also plots the effect of waypoint perturbation on the cost ratio (center) and runtime (bottom). Figure 3 plots the same results for midpoint waypoints. Table 1 shows average runtime and optimal cost-ratio with their standard deviation.

5 Discussion

The results in Figure 2 (top) show a recurrent pattern for all maps. As expected, searching the JPS space is faster for all algorithms, except A*Way which obtains almost identical results for both. This is likely due to the fact that searches in A*Way are never very deep. The fastest search is wA*, but also the one that deviates more from the optimal cost (over $8\%, 6\%$ and 9% respectively). JPS-A* is the second fastest search, attaining also the optimal cost. The Pareto front is formed by JPS-A* and JPS-wA*, leaving all other alternatives dominated.

[4] Code available through https://riuma.uma.es

[5] Grid-A* refers to the baseline A*, and JPS-A* to the standard JPS system.

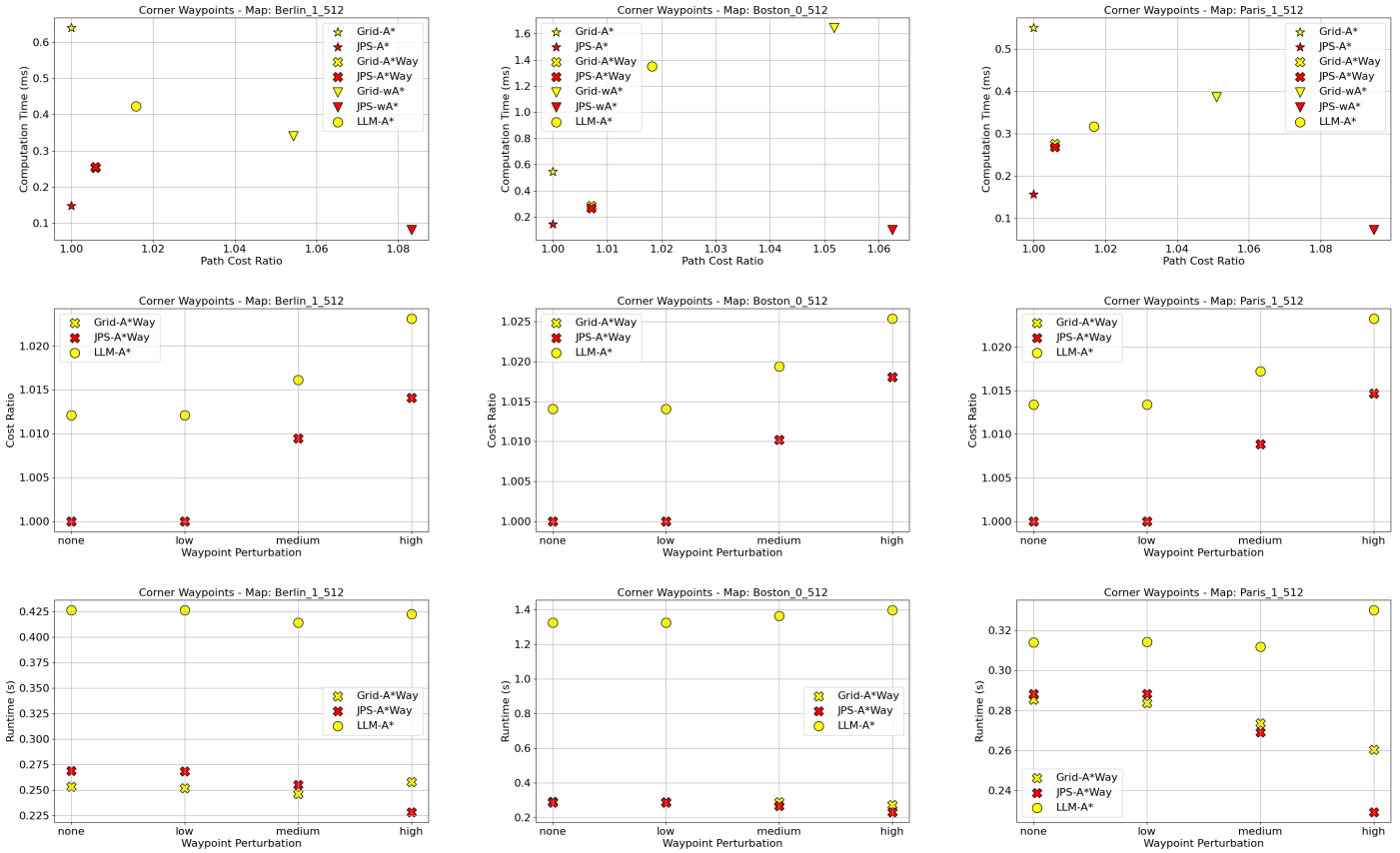


Fig. 2. Performance of each alternative for unperturbed corner waypoints (top) and effect of perturbation on solution path cost-ratio (center) and an runtime (bottom)

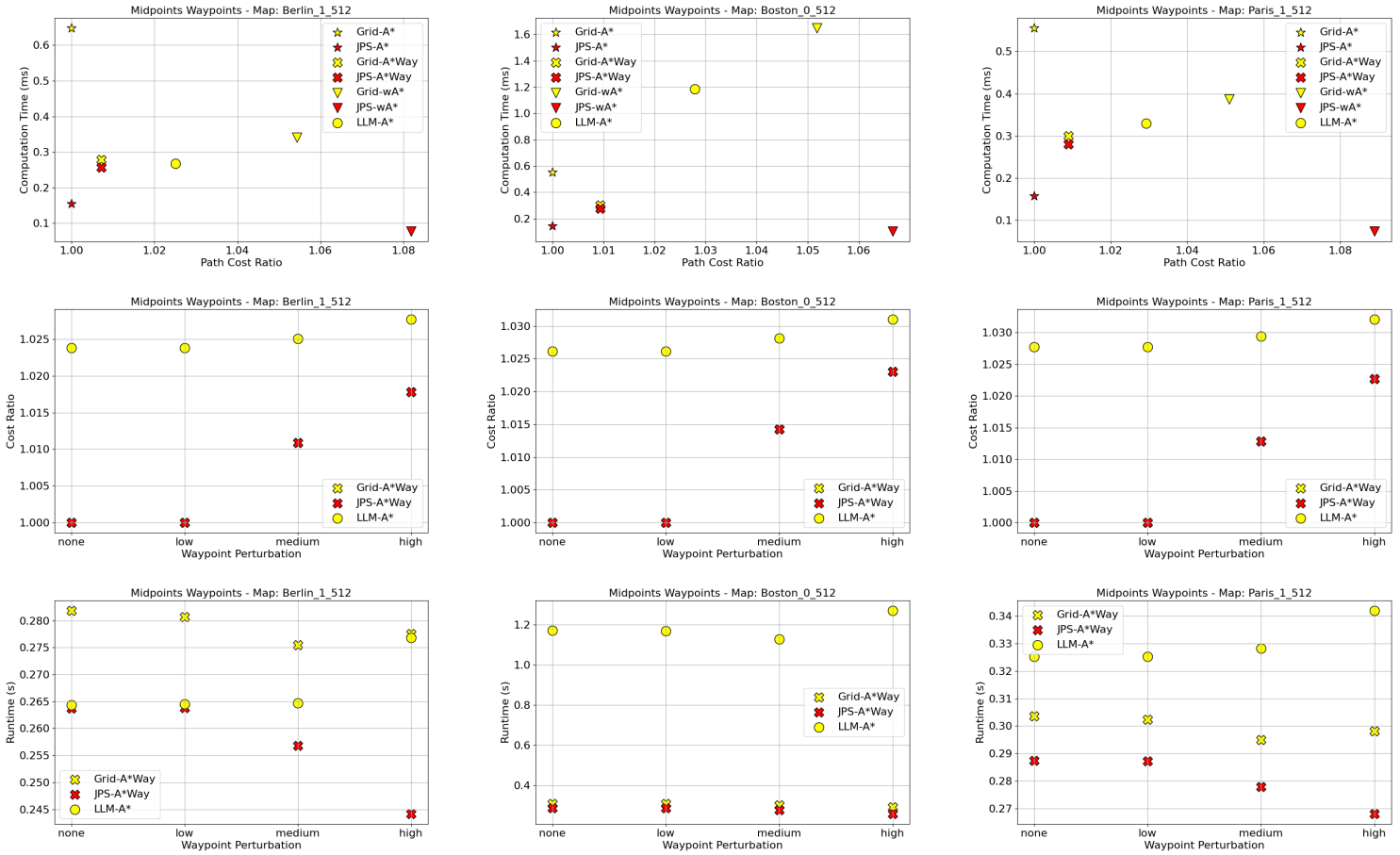


Fig. 3. Performance of each alternative for unperturbed midpoints (top) and effect of perturbation on solution path cost-ratio (center) and an runtime (bottom)

LLM-A*, the algorithm motivating this research, is always dominated by JPS-A*, but also by A*Way. In the Boston map it is also dominated by the baseline Grid-A*. As expected, Figure 2 (center) shows that the cost-ratio performance of A*Way degrades with waypoint perturbation. However, results show that such performance also degrades similarly for LLM-A* in spite of its more relaxed treatment of the waypoints. Figure 2 (bottom) shows the effect of waypoint perturbation on runtime. The more perturbation, the more likely waypoints will be found inside obstacles and discarded. The only significant difference is found for JPS-A*Way. We attribute the reduction in runtime to savings in the overhead of data structure initialization due to the reduced number of searches. In light of results, A*Way algorithms look better alternatives to LLM-A*.

Compared to Grid-wA*, LLM-A* does always improve on path cost-ratio. This can be attributed to the more exploratory nature of the dynamic heuristic in the latter. Also Grid-wA* was faster in the Berlin map, but slower in the Boston and Paris ones, showcasing the variability of greedy search.

Figure 3 shows similar trends for midpoint waypoints. This means waypoint-based searches do not change much with waypoint placement. In the case of LLM-A* some improvements in runtime can be observed at the cost of somewhat worse cost ratios in the Berlin and Boston maps.

Results show that LLM-A* does not provide an attractive performance trade-off between JPS-A* and JPS-wA*, even without taking into account the time needed for the LLM query. Restricting results to the standard grid state space showed A*Way to be a better alternative. In order to be an effective alternative, LLM-A* should be improved to be a competitive trade-off between JPS-A* and JPS-wA* in terms of runtime and optimal cost ratio. This would require a drastic improvement in runtime since the time for the LLM query should also be taken into account. Current JPS-based planners seem hard to beat in this sense.

Table 1(a) shows that standard deviation is quite low for all tests with the exception of Grid-wA* and LLM-A* for the Boston map. We attribute these to their more greedy nature. In Table 1(b) standard deviation is very low. Thus, the reported means are significant.

6 Conclusion

This paper critically investigates LLM-A*, a recent proposal for the integration of LLMs and A* for grid-based path finding. We analyze what could be expected if the LLM were substituted with an oracle that generates obstacle-avoiding waypoints of varying quality. We compare search on standard grid and JPS state spaces using A*, weighted A*, LLM-A* and A*Way (a naive concatenation of sequential A* searches).

Our experiments show that the best algorithms are JPS-A* (optimal path cost) and JPS-wA* (fastest search). They form a Pareto front that seems hard to beat by other alternatives. Regarding waypoint strategies, both JPS-A*Way and Grid-A*Way consistently beat LLM-A* in both cost ratio and runtime (except in the Berlin map with the midpoints strategy for non-high perturbed waypoints).

Improving waypoint quality slightly improves the optimal cost ratio of solutions. Highly perturbed waypoints also made JPS-A*Way slightly faster. Finally, regarding waypoint placement, LLM-A* runs faster with the midpoints but gives a slightly worse path than with the corners waypoints. JPS-A*Way and Grid-A*Way aren't significantly affected by the change of waypoint location.

Our results are hopefully useful for future research on the integration of heuristic search with LLMs, especially in the grid-based path planning domain. In any case, the possible integration of LLM-A* waypoints in the JPS state space for different scenarios deserves attention as future work.

References

1. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: Proc. NIPS'20. Curran Associates Inc., USA (2020)
2. Edelkamp, S., Schrödl, S.: Heuristic Search. Academic Press (2012)
3. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. pp. 1126–1135. ICML'17, JMLR.org (2017)
4. Ghallab, M., Nau, D.S., Traverso, P.: Automated planning. Elsevier (2004)

	Corner waypoints			Midpoint waypoints		
Alg.	Berlin	Boston	Paris	Berlin	Boston	Paris
Grid-A*	0.640(0.59)	0.546(0.53)	0.550(0.52)	0.648(0.60)	0.554(0.54)	0.555(0.52)
JPS-A*	0.147(0.10)	0.145(0.11)	0.157(0.12)	0.155(0.10)	0.146(0.11)	0.157(0.13)
Grid-A*Wy	0.253(0.13)	0.285(0.15)	0.276(0.14)	0.279(0.14)	0.302(0.16)	0.300(0.16)
JPS-A*Wy	0.255(0.13)	0.267(0.13)	0.269(0.14)	0.257(0.13)	0.276(0.14)	0.280(0.15)
Grid-wA*	0.341(0.27)	1.645(3.10)	0.386(0.54)	0.341(0.27)	1.648(3.11)	0.387(0.54)
JPS-wA*	0.081(0.02)	0.101(0.08)	0.073(0.02)	0.078(0.02)	0.104(0.08)	0.074(0.03)
LLM-A*	0.423(0.42)	1.352(2.54)	0.318(0.50)	0.268(0.21)	1.183(2.23)	0.330(0.52)

(a)

	Corner waypoints			Midpoint waypoints		
Alg.	Berlin	Boston	Paris	Berlin	Boston	Paris
Grid-A*	1.000(0.00)	1.000(0.00)	1.000(0.00)	1.000(0.00)	1.000(0.00)	1.000(0.00)
JPS-A*	1.000(0.00)	1.000(0.00)	1.000(0.00)	1.000(0.00)	1.000(0.00)	1.000(0.00)
Grid-A*Wy	1.006(0.02)	1.007(0.01)	1.006(0.01)	1.007(0.01)	1.009(0.02)	1.009(0.01)
JPS-A*Wy	1.006(0.02)	1.007(0.01)	1.006(0.01)	1.007(0.01)	1.009(0.02)	1.009(0.01)
Grid-wA*	1.054(0.03)	1.052(0.02)	1.051(0.02)	1.054(0.03)	1.052(0.02)	1.051(0.02)
JPS-wA*	1.083(0.05)	1.062(0.03)	1.095(0.05)	1.082(0.05)	1.067(0.03)	1.089(0.04)
LLM-A*	1.016(0.01)	1.018(0.01)	1.017(0.01)	1.025(0.01)	1.028(0.01)	1.029(0.01)

(b)

Table 1. (a) Average runtimes (and standard deviation). (b)Average cost ratio (and standard deviation).

5. Harabor, D.D., Grastien, A.: Online graph pruning for pathfinding on grid maps. In: Burgard, W., Roth, D. (eds.) Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011. pp. 1114–1119. AAAI Press (2011)
6. Harabor, D.D., Grastien, A.: The JPS pathfinding system. In: Borrajo, D., Felner, A., Korf, R.E., Likhachev, M., López, C.L., Ruml, W., Sturtevant, N.R. (eds.) Proceedings of the Fifth Annual Symposium on Combinatorial Search, SOCS 2012. pp. 207–208. AAAI Press (2012)
7. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Systems Science and Cybernetics SSC-4 **2**, 100–107 (1968)
8. Lingo, R., Arroyo, M., Chhajer, R.: Enhancing LLM problem solving with REAP: Reflection, explicit problem deconstruction, and advanced prompting (2024), https://arxiv.org/abs/2409.09415
9. Meng, S., Wang, Y., Yang, C.F., Peng, N., Chang, K.W.: LLM-A*: Large language model enhanced incremental heuristic search on path planning. In: Al-Onaizan, Y., Bansal, M., Chen, Y.N. (eds.) Findings of the Association for Computational Linguistics: EMNLP 2024. pp. 1087–1102. Association for Computational Linguistics, Miami, Florida, USA (Nov 2024)
10. Mirzadeh, I., Alizadeh, K., Shahrokhi, H., Tuzel, O., Bengio, S., Farajtabar, M.: GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models. In: The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025. OpenReview.net (2025)
11. Newell, A., Shaw, J.C., Simon, H.A.: Report on a general problem-solving program. In: Proc. 1st Int. Conf. on Inf. Proc., UNESCO. pp. 256–264 (1959)
12. Newell, A., Simon, H.A.: Computer science as empirical inquiry: symbols and search. Commun. ACM **19**(3), 113–126 (Mar 1976)
13. Pearl, J.: Heuristics. Addison-Wesley, Reading, Massachusetts (1984)
14. Pohl, I.: Heuristic search viewed as path finding in a graph. Artif. Intell. **1**(3), 193–204 (1970)
15. Stechly, K., Valmeekam, K., Kambhampati, S.: Chain of thoughtlessness? an analysis of CoT in planning. In: Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J.M., Zhang, C. (eds.) Advances in Neural Information Processing Systems 38: Annual Conf. on Neural Inf. Proc. Syst., NeurIPS 2024 (2024)
16. Sturtevant, N.R.: Benchmarks for grid-based pathfinding. IEEE Trans. Comput. Intell. AI Games **4**(2), 144–148 (2012)
17. Valmeekam, K., Marquez, M., Sreedharan, S., Kambhampati, S.: On the planning abilities of large language models - A critical investigation. In: Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (eds.) Adv. in Neural Inf. Proc. Syst. 36: Annual Conf. on Neural Inf. Proc. Syst, NeurIPS 2023 (2023)
18. Valmeekam, K., Stechly, K., Gundawar, A., Kambhampati, S.: A systematic evaluation of the planning and scheduling abilities of the reasoning model o1. Trans. Mach. Learn. Res. **2025** (2025)
19. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Proc. of the 31st Int. Conf. on Neural Information Processing Systems. pp. 6000–6010. Curran Assoc. Inc. (2017)
20. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E.H., Le, Q.V., Zhou, D.: Chain-of-thought prompting elicits reasoning in large language models. In: Proc. 36th Int. Conf. Neural Inf. Proc. Syst. Curran As. Inc. (2022)
21. Wilt, C.M., Ruml, W.: When does weighted A* fail? In: Borrajo, D., Felner, A., Korf, R.E., Likhachev, M., López, C.L., Ruml, W., Sturtevant, N.R. (eds.) Proc. of the 5th Annual Symp. on Combinatorial Search. pp. 137–144. AAAI Press (2012)