HOCHSCHULE
RAVENSBURG-WEINGARTEN
UNIVERSITY
OF APPLIED SCIENCES

Comparison of different path planning algorithms

Manuel Gnannt - 34946, IN
Florian Betz - 35653, IN

11.03.2023

Contents

1 Abstract

Path planning technology relies on path planning algorithms, which have varying levels of search and planning efficiency depending on their application environment. In path planning, the discovery of good trajectories often requires a high-dimensional model and the evaluation of many samples. Recently, Wang et al. proposed an algorithm called Latent Space Partitions for Path Planning (LaP3), which they claimed outperforms existing path planning methods in 2D navigation tasks in terms of sampling efficiency. With the increasing number of path planning algorithms and the complexity of the environment, it becomes more and more difficult to fulfill the actual requirements of path planning and to achieve the desired path planning results. In this paper, we compare the newly published LaP3 with other path planning algorithms in different 2D environments.

2 Introduction

In path planning, the goal is to find the most rewarding trajectory in a given search space. There are different search algorithms because different problems may have different requirements for search strategies. Furthermore, different applications and systems may also have varying requirements for search algorithms, such as in terms of efficiency, memory requirements, or robustness to interference or change. Thus, multiple search algorithms exist to address a wide range of problems and requirements. A common problem faced by approaches like Covariance matrix adaptation evolution strategy (CMA-ES) [Han16] is that they are trapped in local optima. Another problem is the exploration- exploitation tradeoff. Approaches such as Voronoi Optimistic Optimization Tree (VOOT) [KLL+20] attempt to tackle both the local optimum problem and the exploration-exploitation tradeoff by partitioning the search space. However, this partitioning is done independently of the reward function, which makes it less efficient.

For more efficient search space partitioning, Wang et al. proposedLaP3 an extension of Latent Action Monte Carlo Tree Search (La-MCTS) [WFT20], in which the search space is partitioned into high and low reward regions thus splitting is done based on the reward function. This paper explores the topic of path planning, with a focus on three specific algorithms A*, La-MCTS, and LaP3.

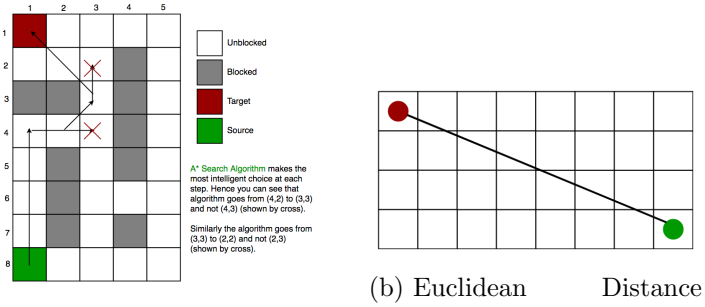
3 Path Planning Algorithms

In this paper, we decided to use three different search algorithms A*, La-MCTS and LaP3.

3.1 A*

The A* algorithm is a widely used method for heuristic path finding and graph traversal. [HNR68] [FBKT00] It builds on Dijkstra's algorithm by incorporating heuristic functions and predicted costs, making it the most efficient direct search method for finding shortest paths for various problems.[FBKT00]

An essential part of the algorithm is the development of the evaluation function $f(n) = g(n) + h(n)$ [W.;21, p. 121], where $g(n)$ is the sum of the costs from the starting point to the current node n, $h(n)$ is the expected cost from node n to the target node, and $f(n)$ is the heuristic weighting function that estimates the cost from the starting node through node n to the target node. The most common heuristic function is the Euclidean distance $d(x_1, x_2) = ||x_2 - x_1|| = \sqrt{\sum_{i=1}^{n} (x_{2_i} - x_{1_i})^2}$ The A* algorithm is complete and optimal because it always finds the solution with the lowest total cost when the heuristic h is admissible.[W.;21, p. 121] This implies that the actual costs from node n to the target are never overestimated.



(a) Astar path

(b) Euclidean Distance Heuristics

Figure 1: A* algorithm [R.;22]

Algorithm steps: [LMCX19]
step 1: Insert the starting node into a priority queue.
step 2: Choose the node with the lowest F value from the current queue and assign it as the current node.
step 3: Mark the current node as visited and process its neighbor nodes.
step 4: If the neighbor node has not been visited yet, add it to the queue, set the current node as its parent, and record its F, H, and G values. If the neighbor node has already been visited, compare the G values to determine if the current node has a shorter path. See figure 1a. If the current node has a smaller G value, update the parent node and the G and H values of that node.
step 5: Repeat steps 2 through 4 until the target node is marked or the priority queue is empty.
step 6: Once the path is discovered, follow the parent node from the endpoint to the start node.

3.2 La-MCTS

The basic idea of a La-MCTS is to recursively partition the search space, where each region represents a node in a Monte Carlo tree.[WFT20] As shown in Figure 2, this is done by taking some samples from the search space and splitting them into a high reward region and a low reward region using K-means (k=2). A support vector machine (SVM) then learns this boundary, and the two sections can be represented as nodes in a Monte Carlo tree. In our example, the high reward section is represented by the left node and the low reward section is represented by the right node.

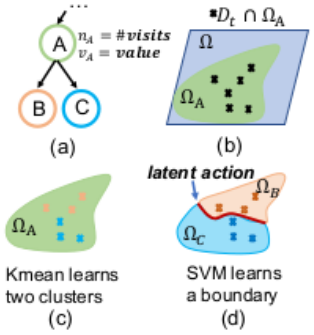


Figure 2: La-MCTS sampling and splitting [WFT20, p.3]

As shown in Figure 3, this process is now repeated by selecting a node until a leaf node is reached, sampling along the corresponding section, and splitting it again. The result is a tree structure where the leftmost node represents the section with the highest reward.

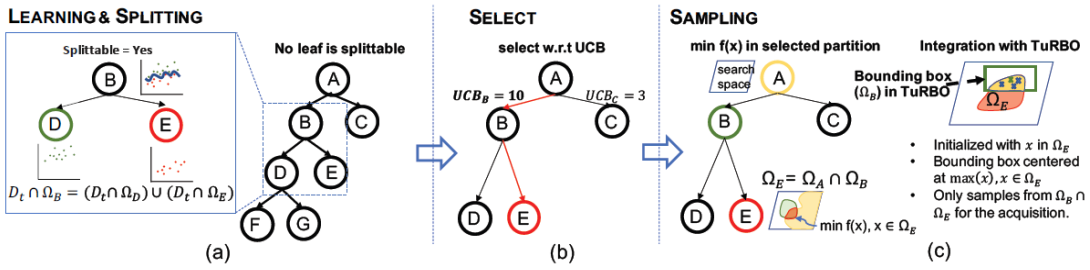


Figure 3: Steps of La-MCTS [WFT20, p.4]

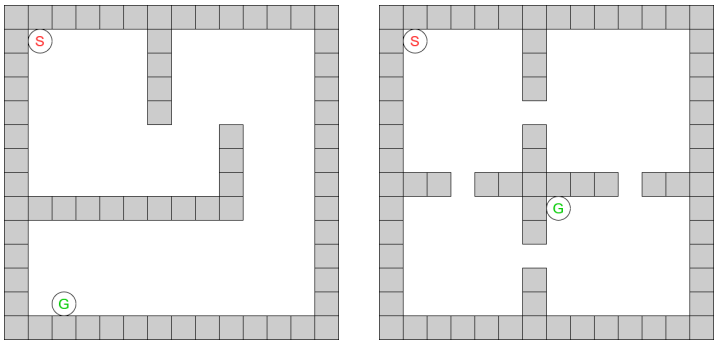
3.3 LaP3

LaP3 functions similarly to La-MCTS. The main difference is that the entire tree is updated after each sampling step, rather than a single leaf. Other differences include the method used to determine the goodness of a node (LaP3 uses the maximum, while La-MCTS uses the mean) and the sampling method (LaP3 uses CMA-ES, while La-MCTS uses Bayesian Optimization (BO)). [YZC+21] [WFT20]

4 Methodology

We ran LaP3, La-MCTS, and an A* on two different environments, with both environments containing local optima. For each environment, the experiment was run on a 12x12, a 24x24, and a 36x36 grid. Since the goal is to find a sampling efficient trajectory, the number of samples was used as a metric in relation to the reward. The reward was calculated by the inverse of the Euclidean distance to the target. It can be described by the following formula.

$$\begin{cases} r(\vec{x}) = \frac{10}{\sqrt{\Delta x_1^2 + \Delta x_2^2}} & , \Delta x_1 + \Delta x_2 \neq 0 \\ r(\vec{x}) = 15 & , \quad \Delta x_1 + \Delta x_2 = 0 \end{cases} \tag{1}$$



(a) Maze S3 (b) Four Rooms

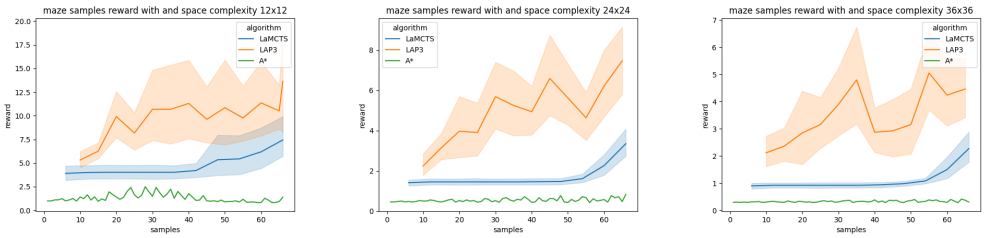
Figure 4: different environments

4.1 Search Efficiency

The goal in path planning is to find the most rewarding trajectory in the search space. Since the optimum can eventually be found by random sampling, if the number of samples is high enough, an efficient algorithm is preferred. Therefore, the search efficiency of an algorithm can therefore be described by the number of samples and the achieved reward. Since this metric is widely used in comparing different search algorithms, [WFT20], [WXL+19], [KLL+20], we also used it for comparison.

5 Evaluation

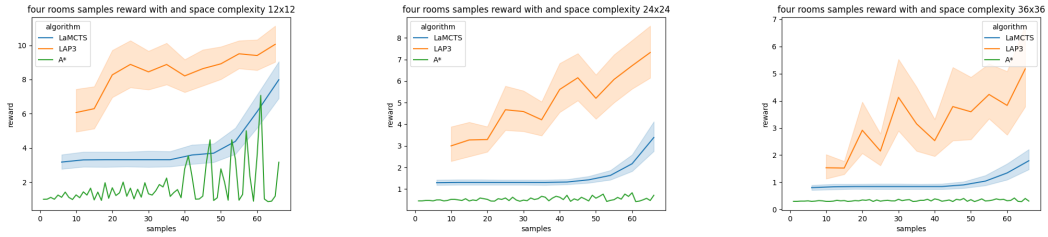
The algorithms were ran in different environments, to see if LaP3 performs better with respect to sample efficiency. Figure 5 shows the number of samples drawn and the associated reward in different environments with different complexities. The environment is maze s3 with complexity sizes 12x12, 24x24 and 36x36 A* always yields the same results, but La-MCTS and LaP3 deliver different results for a hundred runs. Therefore, the mean and the lower and upper bounds of the 95 confidence interval are shown in the graphs. It can be seen that LaP3 receives a larger reward than La-MCTS and A* with the same number of samples. Although LaP3 has a high variance, the reward increases steadily as the number of samples increases. However, the LaP3 algorithm has a very large variance in the results provided. For La-MCTS, an increase in variance does not occur until a sample size of 42 in a 12x12 environment and a sample size of 55 in a 24x24 and 36x36 environment. The reward of A* varies continuously, but the reward does not increase as the number of samples increases.



(a) space complexity 12x12 (b) space complexity 24x24 (c) space complexity 36x36

Figure 5: number of samples and reward in Maze s3 environment with different space complexity

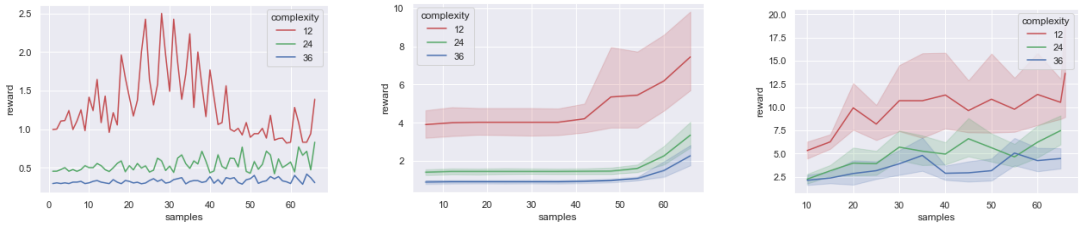
Figure 6 shows the number of samples drawn and the associated reward in the four rooms environment with complexity sizes of 12x12, 24x24, and 36x36. As in the maze s3 environment, A* produces a low reward with a low number of samples. Only in the 12x12 environment with a sample number of 40 does the reward increase and achieve a similar reward as the La-MCTS. At greater space complexity, the reward remains consistently low. The La-MCTS corresponds to a growing exponential function with constant mean. The LaP3, on the other hand, has a strong growth in rewards at the beginning. However, as the sample number, however, the growth decreases. However, LaP3 has a higher reward than La-MCTS and A* for each number of samples.



(a) space complexity 12x12 (b) space complexity 24x24 (c) space complexity 36x36

Figure 6: number of samples and reward in four rooms environment with different space complexity

What we have seen in the previous comparisons in the maze s3 and four rooms environment, the LaP3 always has a better reward per drawn samples. However, the results of the algorithms depend on the environment in which they are executed. As can be seen in Figure 7, it can also be seen that the LaP3 performs most robustly when the space complexity is increased. The reward of the A* is small with increasing complexity and almost does not increase. On the other hand, the La-MCTS shows similar behavior of rewards at different spatial complexities.



(a) A* (b) LaMCTS (c) LaP3

Figure 7: number of samples and reward in maze s3 environment

6 Conclusions and Further Discussion

The data clearly show that LaP3 outperforms other path planning algorithms in different environments, as claimed by the authors. LaP3 has a greater ability to deal with uncertainty and stochastic models. A* requires a deterministic heuristic function and LaP3 uses a probabilistic procedure to achieve the goal. This can be to the advantage of LaP3 when the algorithm is run in dynamic environments and in environments with incomplete information. LaP3 may be able to detect that a certain path leads to collisions or obstacles. However, we did not examine the single improvements of LaP3 over La-MCTS, such as the whole tree update, the different sampling methods, and the different computation of node quality and their partial impact on search efficiency. In addition, search efficiency could be measured not in absolute terms (number of samples), but based on the size of the search space they explore. This would facilitate the comparison of different space complexities (12x12, 24x24, etc.). It would also be interesting to know if LaP3 would still outperform the other algorithms by the same amount if sampling the entire search space were not possible.

Acronyms

LaP3 Latent Space Partitions for Path Planning

La-MCTS Latent Action Monte Carlo Tree Search

CMA-ES Covariance matrix adaptation evolution strategy

BO Bayesian Optimization

VOOT Voronoi Optimistic Optimization Tree

SVM support vector machine

References

[FBKT00] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 8:325–344, 06 2000.

[Han16] Nikolaus Hansen. The cma evolution strategy: A tutorial, 2016.

[HNR68] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for

the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[KLL+20] Beomjoon Kim, Kyungjae Lee, Sungbin Lim, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Monte carlo tree search in continuous spaces using voronoi optimistic optimization with regret bounds. In *AAAI Conference on Artificial Intelligence*, 2020.

[LMCX19] Chenguang Liu, Qingzhou Mao, Xiumin Chu, and Shuo Xie. An improved a-star algorithm considering water current, traffic separation and berthing for vessel path planning. *Applied Sciences*, 9(6), 2019.

[R.;22] Belwariar R.;. A* Search Algorithm. `https://www.geeksforgeeks.org/a-search-algorithm/`, 18.12.2022.

[W.;21] Ertel W.;. *Grundkurs Künstliche Intelligenz* . Springer Vieweg Wiesbaden, 2021.

[WFT20] Linnan Wang, Rodrigo Fonseca, and Yuandong Tian. Learning search space partition for black-box optimization using monte carlo tree search. *CoRR*, abs/2007.00708, 2020.

[WXL+19] Linnan Wang, Saining Xie, Teng Li, Rodrigo Fonseca, and Yuandong Tian. Sample-efficient neural architecture search by learning action space, 2019.

[YZC+21] Kevin Yang, Tianjun Zhang, Chris Cummins, Brandon Cui, Benoit Steiner, Linnan Wang, Joseph E Gonzalez, Dan Klein, and Yuandong Tian. Learning space partitions for path planning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 378–391. Curran Associates, Inc., 2021.