Comparison of different path planning algorithms

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

13.03.2023

Contents

1 Abstract

In path planning, discovering good trajectories often requires a high-dimensional model and the evaluation of many samples. Recently, Wang et al. proposed an algorithm called **La**tent Space Latent Space Partitions for Path Planning (LaP3) claiming it outperforms existing path planning methods in 2D navigation tasks with respect to sample efficiency. In this paper, we compare the newly published LaP3 to other path-planning algorithms in several 2D environments.

extend descriptions with all information

2 Introduction

In path planning, the goal is to find the most rewarding trajectory in a given search space.

A common problem faced by approaches like Covariance matrix adaptation evolution strategy (CMA-ES) [Han16] is being trapped at local optima. Another is the exploration-exploitation tradeoff. Approaches like Voronoi Optimistic Optimization Tree (VOOT) [KLL+20] try to tackle both the local optimum problem and the exploration-exploitation tradeoff by partitioning the search space. This partitioning however is done independently from the reward function, making it less efficient.

For a more efficient search space partitioning, Wang et al. proposed LaP3 an extension of Latent Action Monte Carlo Tree Search (La-MCTS) [WFT20] where 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

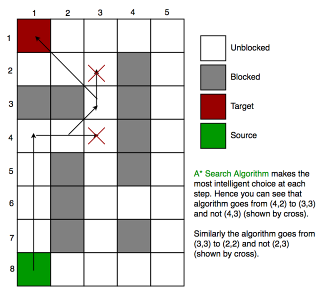
In this paper, we decided to use three different search algorithm A*, La-MCTS and LaP3.

3.1 A*

The A* algorithm, first introduced by Peter Hart and other researchers at the Stanford Research Institute in 1968, is a widely used method for pathfinding and graph traversal. [HNR68] It builds on Dijkstra's algorithm by incorporating heuristic functions and predicted costs, making it the most efficient direct search method for finding the shortest paths in a static road network and a popular heuristic for various other problems.[FBKT00]

The key component of the algorithm is the design of the valuation function, $f(n) = g(n) + h(n)$, where $g(n)$ is the actual cost from the initial node to node n, $h(n)$ is the estimated cost from node n to the target node, and $f(n)$ is the estimated cost from the initial node via node n to the target node. The A* algorithm can reach a time complexity of $O(n)$.

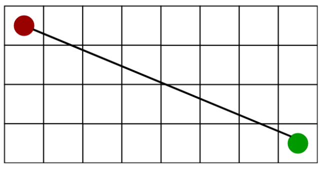
Step 1: Add the starting node to the priority queue.
Step 2: Select the node with the smallest F-value from the current priority queue and make it the current node. Use the Euclidean Distance Heuristics from figure 1b
Step 3: Mark it as visited and process its adjacent nodes.
Step 4: If the neighboring node has not been visited, add it to the queue, set the current node as its parent, and record its F, H, and G values. If the neighboring node has already been visited, check if the current node has a shorter path by comparing G values. See in figure 1a. If the current node has a smaller G value, update the parent node and G, H values of that node.
Step 5: Repeat steps 2 to 4 until the target node is marked or the priority queue is empty.
Step 6: When the path is found, trace back from the endpoint to the start node using the parent node.



A* Search Algorithm makes the most intelligent choice at each step. Hence you can see that algorithm goes from (4,2) to (3,3) and not (4,3) (shown by cross).

Similarly the algorithm goes from (3,3) to (2,2) and not (2,3) (shown by cross).

(a) Astar path



(b) Euclidean Distance Heuristics

Figure 1: Astar [R.;22]

3.2 La-MCTS

The basic idea of a La-MCTS is to recursively split the search space with every region representing 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 it into a high- and a low-reward section using K-means (k=2). An 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 by the right node.

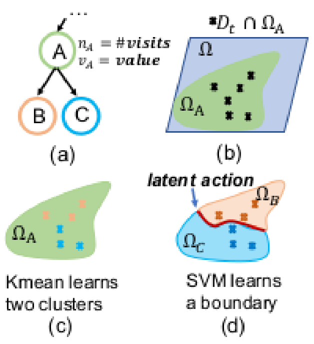


Figure 2: La-MCTS sampling and splitting [WFT20][p.3]

As shown in in figure 3, this process is now repeated by selecting a node until a leaf node is reached, taking samples along the corresponding section and splitting it again. This results in a tree structure with the leftmost node representing the section with the highest reward.

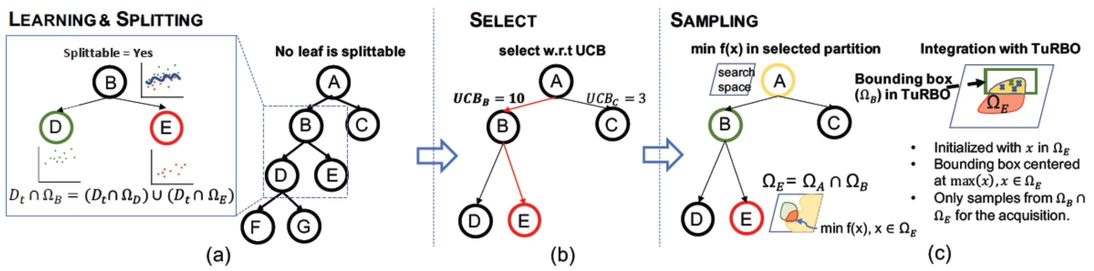

LEARNING & SPLITTING
Splittable = Yes
B
D
E
$D_t \cap \Omega_B = (D_t \cap \Omega_D) \cup (D_t \cap \Omega_E)$
No leaf is splittable
A
B
C
D
E
F
G
(a)
SELECT
select w.r.t UCB
$UCB_B = 10$
$UCB_C = 3$
(b)
SAMPLING
min f(x) in selected partition
search space
$\Omega_E = \Omega_A \cap \Omega_B$
min f(x), $x \in \Omega_E$
Integration with TuRBO
Bounding box (Ω_B) in TuRBO
Ω_E
Initialized with x in Ω_E
Bounding box centered at $\max(x), x \in \Omega_E$
Only samples from $\Omega_B \cap \Omega_E$ for the acquisition.
(c)

Figure 3: Steps of Monte Carlo Tree Search (MCTS) [WFT20][p.4]

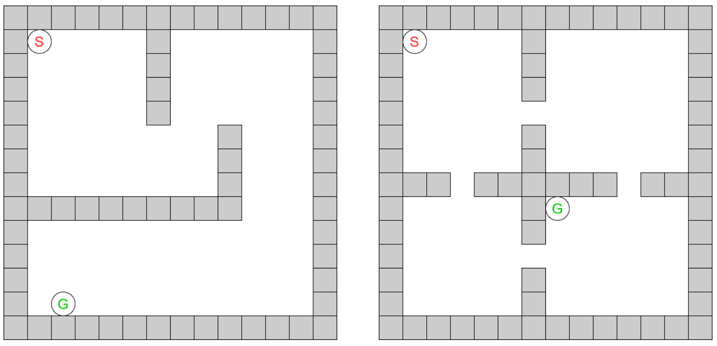
3.3 LaP3

LaP3 works similar to La-MCTS. The main difference is that after each sampling step, the whole tree is updated and not only a single leaf. Other differences are the method of determining the goodness of a node (LaP3 uses the maximum, where La-MCTS uses the mean) and the sampling method (LaP3 uses CMA-ES where La-MCTS uses Bayesian Optimization (BO)).

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. As the goal is finding a sample 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 euclidian distance to the target. It can be described by 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 environemt (b) Four Rooms

Figure 4: different environments

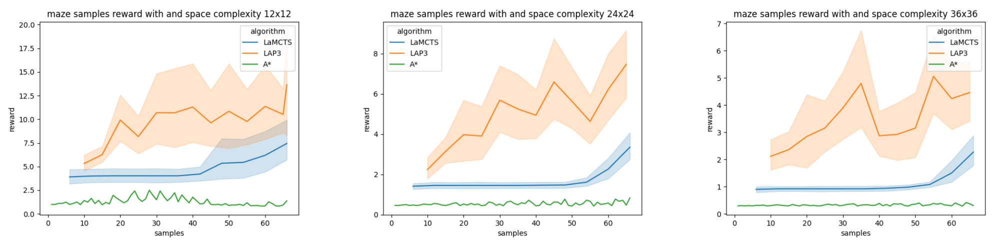
4.1 Search Efficiency

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

LaNas wird als Acronym nicht verwendet

5 Evaluation

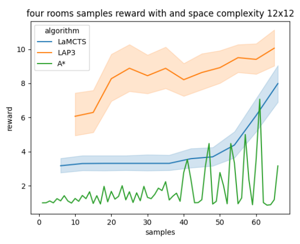
With the evaluation criteria and the different environment, the search algorithms are examined to see if the LaP3 performs better from several points of view. Figure 5 shows the number of samples drawn and the associated reward in different environments with different complexities. The environment is the maze s3 with complexity size 12x12, 24x24 and 36x36 The A* always delivers the same results, but La-MCTS and LaP3 deliver different results on hundred runs and That is why the mean value and the lower and upper bound of the 95% confidence interval are displayed in the graphs. It can be seen that the LaP3 gets a larger reward than the La-MCTS and A* with the same number of samples. Although the LaP3 has a high variance, the reward increases consistently with increasing samples. However, the LaP3 algorithm has a very large scattering of the delivered results. For the 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 from the A* fluctuates continuously, but the reward does not increase as the number of samples increases.



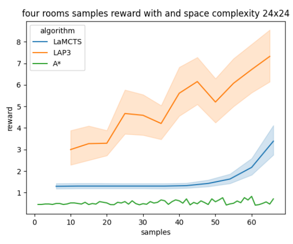
(a) space complexity 12x12 (b) space complexity 24x24 (c) space complexity 36x36

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

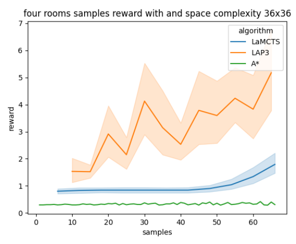
Figure 6 shows the ratio of samples drawn and the associated reward in the four rooms environment with complexity size 12x12, 24x24 and 36x36. As in the maze s3 environment, the A* generates a low reward with a low number of samples. Only in the 12x12 environent with a sample number of 40 does the reward increase and achieve a similar reward to the La-MCTS. With greater space complexity, the reward remains constantly low. The La-MCTS corresponds to a growing exponential function with constant mean. The LaP3 on the other hand has a strong growth of the rewards at the beginning. With increasing sample number, however, the growth is reduced. However, the reward of LaP3 is larger than that of La-MCTS and A* at each sample number.



(a) space complexity 12x12



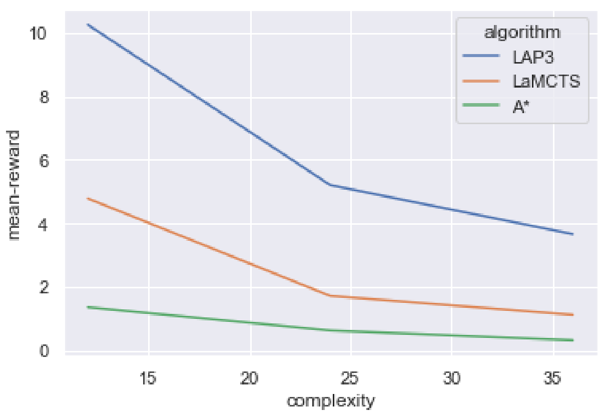
(b) space complexity 24x24



(c) space complexity 36x36

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

What is striking in both environments is that the La-MCTS has a constant reward function. In the beginning, the reward is constant and increases exponentially after 40 samples. The LaP3 exhibits significant variation in its reward data, although the specific mean values provided by the LaP3 are displayed in table 7b. As we can see in figure 7a, the complexity of the environment increases, the reward diminishes rapidly.



(a) maze s3 space complexity and mean reward $\overline{x}$

	12x12	24x24	36x36
Samples	$\overline{x}$	$\overline{x}$	$\overline{x}$
20	9.9208	3.9697	2.8462
25	8.1711	3.9062	3.1538
30	7.4318	5.6875	3.8974
35	10.6842	5.2500	4.7949
40	11.3000	4.9394	2.8718
45	9.6250	6.5938	2.9231
50	10.8485	5.6250	3.1538
55	9.7692	4.6364	5.0513
60	11.3611	4.6364	4.2368
	10.2632	5.2030	3.6588

(b) mean and 95% confidence interval of LaP3

Figure 7: first 60 samples drawn in maze s3 environment with different complexity

Aussage beantwortet, LaP3 performt immer besser

Schaubild 7 beschreiben

LaP3 beschreiben, dass man in ein locales minumu leicht reinkommt

6 Conclusions and Further Discussion

Different sampling methods

What could be explorer in the future,Manuel

functions evals?, Manuel

sample drawing in sequential action space (horiyon tto draw samples from, not whole search space), Manuel

Acronyms

LaP3	Latent Space Partitions for Path Planning
MCTS	Monte Carlo Tree Search
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.

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

[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.