Can language model plan in extrapolated environments?: Casestudy in textualized Gridworld

Anonymous Author(s)
Affiliation
Address
email

Abstract

While language models have demonstrated impressive capabilities across generalized language tasks, their ability to extrapolate in a certain task is highly unknown. We first introduce the optimal path planning task in a textualized Gridworld environment as a valid probe for estimating the extrapolability of language models. We show that the mere next token prediction inherently fails to extrapolate in solving the task. Inspired by human cognition, we claim that language models should construct an internal simulation that explores the environment, i.e. cognitive map before actually interacting with the given environment. We demonstrate that autoregressive generation of cognitive map and planning sequence can significantly enhance the performance of the planning power even in extrapolated environments, suggesting the necessity of cognitive map for language models as a path forward.

1 Introduction

Language models have recently demonstrated remarkable proficiency in a variety of complex tasks (Brown et al., 2020; Touvron et al., 2023; Chowdhery et al., 2023; Chen et al., 2021), from natural language understanding to code generation, primarily through the training objective of next token prediction. This training paradigm enables language models to excel at general planning tasks by leveraging their extensive learned knowledge and pattern recognition capabilities (Ahn et al., 2022; Liang et al., 2023; Song et al., 2023). However, language model also falter in scenarios that require robust, long-horizon planning tasks (Dziri et al., 2024). It is in contrast that humans naturally employ model-based planning, internally construct models to simulate outcomes and guide optimal decision-making, as extensively documented in cognitive science literature (Daw et al., 2005). This distinction aligns with the dual-process theory of reasoning: System 1 processes are fast, automatic, and pattern-based, akin to model-free planning, while System 2 processes are slower, deliberative, and involve explicit reasoning (Daniel, 2017).

The Chain of Thought (CoT) approach (Wei et al., 2023) is a prominent method used by language models to emulate the System 2 cognitive process, optimizing the reasoning pathway. While CoT has significantly improved reasoning and planning, existing methods focus solely on generating intermediate steps from the initial state to the goal (Wei et al., 2023; Nye et al., 2021). This approach deviates from the concept of "simulation," which involves analyzing real-world systems and predicting outcomes. In contrast, human System 2 cognition typically engages in iterative simulations, refining steps until reaching the goal while avoiding dead-end states. After reaching the goal, humans often work backward to determine the optimal actions leading to that state. This process, which forms a "cognitive map," is controlled by the prefrontal cortex (Daw et al., 2005), allowing for deliberate, complex decision-making (Doll et al., 2012).

Submitted to NeurIPS 2024 Workshop on Compositional Learning: Perspectives, Methods, and Paths Forward. Do not distribute.

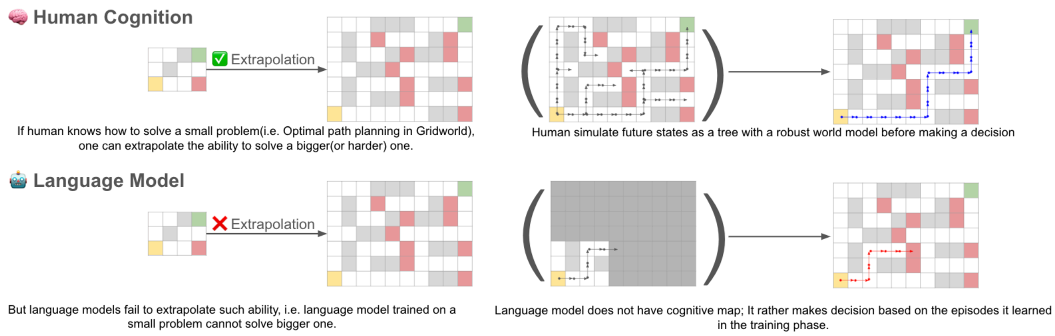


Figure 1: Difference between human cognition and language models: When both humans and language models are taught to path plan in small Gridworld environments, humans can successfully extrapolate to larger environments after a few demonstrations. This ability is due to the construction of a mental map of future states, known as a cognitive map. In contrast, language models struggle with extrapolation, primarily because they rely solely on the training data. We propose that language models should develop the capability to internally simulate future states, organized as a decision tree, to improve performance on tasks requiring extrapolation.

We argue that the lack of a cognitive map in current language models is a primary reason they fail to adapt learned behaviors from training into extrapolated environments, as depicted in Figure 1. To support this claim, we introduce an optimal path-planning task in a textualized Gridworld environment as a benchmark for assessing language model extrapolability. Our results demonstrate that language models struggle to extrapolate in such environments when trained purely through imitation learning and even with CoT fine-tuning. As a solution, we insist that language models also need human-like cognitive maps. To support the idea, we propose training language models using datasets augmented with a simple cognitive map(See Figure 2). We find that auto-regressive generation of cognitive maps and planning sequences can significantly improve the model's ability to plan effectively, even in unfamiliar, extrapolated environments. These results indicate that integrating cognitive maps into language models may be a promising step toward achieving human-like cognition and enhancing their capacity to generalize to more composite and unseen environments.

2 Experimental setup

2.1 Basic setup

In this paper, we set textualized Gridworld (Brown, 2015) as the main task. Gridworld is a task that involves path planning from the start state to the goal state while avoiding the pit, wall, and grids outside the world. Especially, we ensure that there is only one path from start to goal for each environment. We prompt a textualized instruction of the Gridworld to the model and set a textworld environment that receives an action(either up, down, left, or right) and outputs the corresponding transition state(See Appendix B.1 for textualized input instruction).

We choose Gridworld because of following reasons: First, Gridworld requires minimal knowledge. We can probe the extrapolability of language models with board game such as Einstein's puzzle (Brainzilla, 2017) or Blocksworld (Valmeekam et al., 2022), but with assessing minimal world knowledge in order to probe the pure extrapolability of the model. Gridworld only requires 4 actions(up, down, right and left) and the transition of each action is simple and explicit, which is a good fit. Second, we can generate Gridworld environment of an arbitrary size. After training with few demonstrations, we need to find a larger environment which was unseen during the training phase in order to test the extrapolability of the model. Gridworld, unlike other planning tasks such as coding or math, can always generate a board of a bigger size that was unseen during the training phase. This makes train and validation much easier. Last, Gridworld is outside TC^0 complexity class, making it impossible to solve with mere next token prediction. Existing works implies that the task outside TC^0 class cannot be solved directly with next token prediction with fixed precision transformer architecture (Merrill and Sabharwal, 2023).

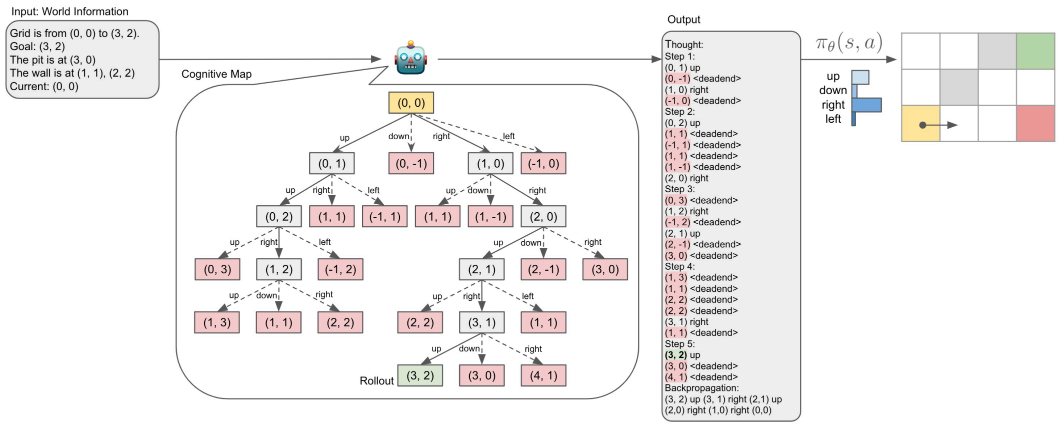


Figure 2: Optimal path-planning with cognitive map: 1. We initialize the world environment which the model is going to interact. 2. The textualized input instruction containing information about the environment(u) is fed into the model. 3. Before interacting with the world, the model constructs a textualized cognitive map(m). Our construction of m is a tree-structured verbal representation of the world model, which is represented in a sequential manner. 4. With the constructed map, the agent interacts with the environment. We analyze the power of cognitive map in generating both the optimal plan without further observation. We show that the cognitive map deduces optimal plan, and it shows human-cognitive characteristics such as generalization to extrapolated environments(Section 3).

For each observation, we give the information of the current state along with possible moves that do not directly result the deadend state along with its corresponding states. For example, if the current state s_t is (11, 4) and there the possible actions are right or left, the observation o_t is "Current:\n(11, 4)\nPossible:\n(10, 4)\nleft\n(12, 4)\nright".(See Appendix B.2 for details)

2.2 Baselines

We name NONE and COT as our baseline experiments representing imitation-based planning. NONE implicitly learns how to conduct path planning, while COT learns a verbalized backward trace as a Chain of Thought(CoT) demonstration. For example, the COT reasoning of the Figure 1 is "(3, 2)up\n(3, 1)\nright\n(2, 1)\nup\n(2, 0)\nright\n(1, 0)\nright\n(0, 0)", while we do not verbalize anything for NONE. See Appendix E.1 for the whole construction.

2.3 Cognitive map design choices

The design of the cognitive map involves three key processes: **Sampling**, **Propagation**, and **Backtracking**. The model begins by sampling plausible actions at each state and then propagates forward by exploring new states that result from these actions. This process of sampling and propagating continues until the model reaches the goal. Once the goal is achieved, the model backtracks from the goal state to the starting state to refine the optimal path(See Figure 2).

Our construction of the cognitive map is straightforward; We sample all 4 possible directions("up", "down", "right", and "left") as actions, and we propagate each action iteratively starting from the goal state until reaching the start state. After reaching the start state, we backtrack until reaching the goal state. We comprise these steps as Sampling, Propagation and Backtracking stage. See Appendix C for more details. We See Appendix D for ablations of different cognitive map constructions.

2.4 Experiment scheme

We observe the capability of the language model when generating the optimal plan without any further observations for each experiments, NONE, COT and COGNITIVE MAP. We evaluate the optimality of the trajectory generated by $a_1, a_2, \ldots, a_n \sim \pi_\theta(\cdot|u, m) \in \mathcal{A}^n$(For preliminary notation, see Appendix A). The only difference among the experiments is the generation of m. For NONE, we do not generate m. For COT, we generate a backward trace from the goal to the start. For COGNITIVE MAP, we generate a simulation process from goal to the start.

Given the instruction u and generated (action, state) trajectory k, we first define optimal path k^* as an element of $\arg\min_{k \in \{k | r(u,k) = \text{success}\}} |k|$. Since we ensure there is only one possible k^*, we only need to check whether the generated trajectory is k^*. Now we can define $R(u, k)$ as follows:

- $R(u, k) =$ SUCCESS if $k = k^*$
- $R(u, k) =$ FAIL otherwise

We compute the mean value of $R(u, k)$ across each size of the Gridworld to probe the extrapolability of the language model. If we get a high value beyond the training boundary, that means that the model has a good extrapolability, otherwise it cannot extrapolate into bigger environments.

3 Result and Analysis

3.1 Cognitive map induces extrapolability

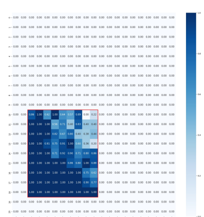
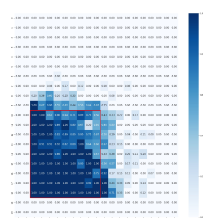
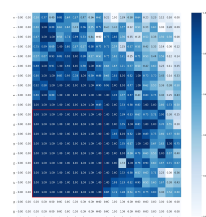
Implicit baseline	Explicit baseline	**Cognitive map**
		
NONE: 19.0%	COT: 26.5%	**COGNITIVE MAP: 76.5%**

Table 1: Qualitative comparison of optimal plan generation rate between baselines and our methods. The degree of the darkness at (x, y) coordinate of each plot denotes the performance of the corresponding model in Gridworld of size x×y. The red box denotes the boundary of the training data. We provide visualization of all the experiments in Appendix F.

As stated in Section 2.1, we train the model on world sizes of up to 10x10 and test it on world sizes of up to 20x20 world to investigate the extrapolation ability. Extrapolation success is defined as the ability to succeed in rollouts beyond the 10x10 boundary. As shown in Table 1, the darkness outside the boundary of the red box shows that **the cognitive map helps in planning on the extrapolated data.** We also observe a consistent tendency for experiments with ALL inclusion, which have a wider coverage of success rate in the extrapolated data. These results align with the result discussed in Section 3.1. We analyze ablation results of cognitive map in Appendix F.

3.2 Why does cognitive map inducde extrapolation?

A language model can be viewed as a statistical model trained to minimize the KL divergence between its predicted logits and the training objective. Given the training data distribution P_{train}, the model is optimized to fit this distribution, enabling it to predict interpolated data points. However, the model's ability to extrapolate beyond the training data is less understood and lacks a clear theoretical explanation. So it is not surprising for NONE and COT not having extrapolability. But what is the key difference of cognitive map that enables such ability? Our current insight is as follows:

Human demonstrations are often abstract and entangled, with the human policy π_{human} being intertwined with the observed replication policy $\pi_{\text{human demo}}$. The replication policy tends to be more complex due to hidden variables and factors influencing human demonstrations. We hypothesize that the human policy π_{human} is relatively simple and can be modeled using a combination of three key functions: sampling (S), propagation (T), and backtracking (T^{-1}). By leveraging these simple, well-defined functions, the model can effectively handle path-planning tasks and demonstrate robust extrapolative abilities beyond the training data.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. 2022. Do as i can, not as i say: Grounding language in robotic affordances.

Brainzilla. 2017. Einstein's riddle. Accessed on September 10, 2024.

Brandon Brown. 2015. Q-learning with neural networks: Learning gridworld with q-learning. Accessed on June 19, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. Journal of Machine Learning Research, 24(240):1–113.

Kahneman Daniel. 2017. Thinking, fast and slow.

Nathaniel D Daw, Yael Niv, and Peter Dayan. 2005. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. Nature neuroscience, 8(12):1704–1711.

Bradley B Doll, Dylan A Simon, and Nathaniel D Daw. 2012. The ubiquity of model-based reinforcement learning. Current opinion in neurobiology, 22(6):1075–1081.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. 2024. Faith and fate: Limits of transformers on compositionality. Advances in Neural Information Processing Systems, 36.

Mehran Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. 2023. Lambada: Backward chaining for automated reasoning in natural language.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023. Code as policies: Language model programs for embodied control.

Ilya Loshchilov and Frank Hutter. 2017. Sgdr: Stochastic gradient descent with warm restarts.

William Merrill and Ashish Sabharwal. 2023. The parallelism tradeoff: Limitations of log-precision transformers.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. 2021. Show your work: Scratchpads for intermediate computation with language models. arXiv preprint arXiv:2112.00114.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M. Sadler, Wei-Lun Chao, and Yu Su. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971.

Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2022. Large language models still can't plan (a benchmark for llms on planning and reasoning about change). arXiv preprint arXiv:2206.10498.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. 2023. Pytorch fsdp: Experiences on scaling fully sharded data parallel.

A Planning task interacting with world model

Planning task with environment feedback in language model can be formalized as a Markov decision process with instruction space $\mathcal{U}$, state space $\mathcal{S}$, action space $\mathcal{A}$, observation space $\mathcal{O}$, metric space $\mathcal{C}$, transition function $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, and metric function $R : \mathcal{U} \times (\mathcal{S} \times \mathcal{A})^n \to \mathcal{C}$ where n is the trajectory length. Note that for language model domain, $\mathcal{U}$, $\mathcal{A}$, and $\mathcal{O}$ are given as sequences of language tokens.

Given an instruction $u \in \mathcal{U}$, the model θ first generates the action $a_1 \sim \pi_\theta(\cdot|u) \in \mathcal{A}$ according to its policy π_θ. For each state $s_t \in S$ and its observation $o_t \in \mathcal{O}$, the agent generates the corresponding action in the $t+1$ step $a_{t+1} \sim \pi_\theta(\cdot|u, a_1, o_1, \ldots, a_t, o_t) \in \mathcal{A}$, which concludes to a new state $s_{t+1} = T(s_t, a_t)$ and its observation o_{t+1}. The interaction loop repeats until the task has terminated for some reason(succeeded, failed, number of steps exceeded maximum value, etc.), and the action trajectory is denoted as:

$$e = (u, a_1, o_1, \ldots, o_{n-1}, a_n, o_n) \sim \pi_\theta(e|u),$$

$$\pi_\theta(e|u) = \prod_{j=1}^{n} \pi_\theta(a_j|u, a_1, o_1, \ldots, o_j),$$

Finally, we define the (action, state) trajectory $k = ((a_1, s_1), \ldots, (a_n, s_n))$ accordingly. The final reward is computed based on the metric function $R(u, k)$.

Note that we can apply reasoning before deciding the action with so-called a "thinking" process. Namely, we have a thinking space $\mathcal{M}$(of language subset) which generates $m \sim \pi_\theta(\cdot|u) \in \mathcal{M}$, then generate $a_1 \sim \pi_\theta(\cdot|u, m) \in \mathcal{M}$ upon the generated thought.

B Experimental Details

B.1 Input detail

Table 2 describes a sample input of the model, describing the instruction of the Gridworld and the specific world information.

Common Prompt	human: You are given a rectangular gridworld, where you can move up, down, left, or right as long as each of your x, y coordinates are within 0 to the x, y size of the grid. If you move up, your y coordinate increases by 1. If you move down, your y coordinate decreases by 1. If you move left, your x coordinate decreases by 1. If you move right, your x coordinate increases by 1.\n\nYou will interact with the gridworld environment to reach the goal state, while avoiding the pit and the wall. You cannot move through the wall or move outside the grid. If you fall into the pit, you lose. If you reach the goal, you win. For each of your turn, you will be given the possible moves.\n\nYou should respond your move with either one of 'up', 'down', 'left', or 'right'. gpt: OK human: Grid is from (0, 0) to (3, 2). Goal: (3, 2)\nCurrent: (0, 0)\nThe pit is at (3, 0). The wall is at (1, 1), and (2, 2).\nCurrent:\n(0, 0)\nPossible:\n(0, 1)\nup\n(1, 0)\nright

Table 2: The prompt for all cases

B.2 Experimental setup details

We train the model for 1 epoch with 50K samples of size 10×10 at largest. After training, we test the model with 3K samples with each grid being 20×20 at largest. We utilize Llama-3-8B model[1] throughout the whole experiments. In each turn, we set the maximum token length of the model to be 8192.

We use one 8 Nvidia A100 node for both training and inference. For the training steps, we use FSDP framework (Zhao et al., 2023) and cosine annealing learning rate scheduler (Loshchilov and Hutter, 2017) for 1 epoch. We utilize bfloat16 floating-point format and a warmup ratio of 0.03. We set the weight decay as 0. We set the batch size of 2 for each GPUs, so the effective batch size is 16 per step. We train each model for $50000/16 = 3125$ steps. For inference, we use VLLM framework (Kwon et al., 2023).

While exploring the pure planning ability of the language model, we did not want the model to refuse to explore extrapolated data only because it has never seen the coordinate. To handle the bias, we adjust the starting position of the grid using uniform sampling while ensuring that the entire grid, with its x and y coordinates, fits within the range of 0 to 19. This method, illustrated in Figure 3, minimizes bias related to the unseen x and y coordinates by randomizing the starting point within the defined bounds.

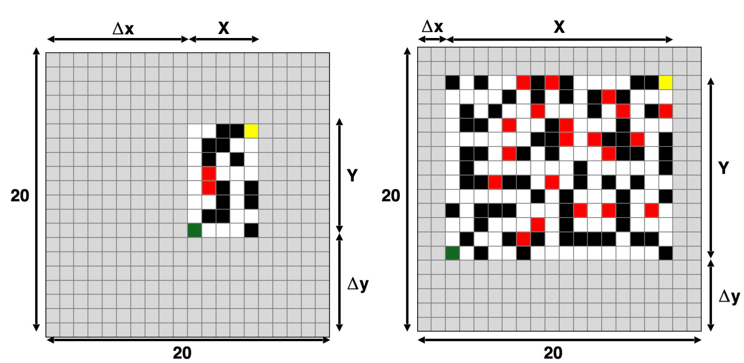


Figure 3: Visualization of configuring Gridworld instance for train(left) and test(right) dataset. To evaluate the extrapolation ability, we set the size of the grid as $X, Y \sim Unif(2, 10)$ for train and $X, Y \sim Unif(2, 20)$ for test. Also we set the starting point of the grid as $\Delta x \sim Unif(0, 20 - X)$, $\Delta y \sim Unif(0, 20 - Y)$ for both train and test.

[1]https://llama.meta.com/llama3/

C Cognitive map for language model

Our proposed process can be divided into three different stages: sampling, propagation, and backtrack. The sampling stage can be defined as the process where the model expects potential actions that can be applied for each state. The propagation stage explores potential outcomes of the actions sampled from the sampling stage. The backtrack stage is the process of tracing back from the goal state to refine and select the optimal path based on the outcomes of the simulation

C.1 Sampling

During the sampling stage, the model samples possible actions in $S(s) \subset A$ regarding the current state s, which leads to a new state that was not propagated before. This sampling process is repeated until reaching the desirable goal. This iterative approach enables the cognitive map of the agent to explore the various states of the given world model.

C.2 Propagation

During the propagation stage, the model expects a transited state $T(s, a) \in \mathcal{S}$ for each action a sampled from the current state s. For example, in the Gridworld, the model simulates different paths by considering movements in four directions (up, down, left, right) from each cell. The goal is to explore various routes to reach the target cell, considering obstacles and the grid's boundaries. This stage is crucial for understanding the global representation of the world model by expecting the future consequences of each action before committing to the actual decision.

C.3 Backtrack

Once the sampling and propagation stages have concluded by reaching a desirable goal state, we need to backtrack through the simulated paths to determine the most efficient route taken to achieve the goal. This involves assessing the propagated paths and identifying the one that reaches the goal. Backtrack search identifies $T^{-1}(s, a) \in \mathcal{S}$ for each state-action pair to ensure that the selected path is valid. By refining the decisions made during the simulation stage, backtracking ultimately guides the model to make informed, strategic choices based on the simulated outcomes. Since the propagation stage deduces only one path with minimal steps, we can guarantee the optimality of the generated path.

C.4 Training how to generate cognitive map

To sum up, constructing the cognitive map m is a sequential application of sampling(S), propagation(T), and backtrack(T^{-1}). We train the language model θ via supervised learning method so that the model can successfully construct the cognitive map without any external interaction, as shown in Figure 2.

D Experiment Ablation

D.1 Reachable planning analysis: multi-turn setting

We also investigate whether the cognitive map can deduce a reachable plan(which doesn't require optimal planning) in a multi-turn setting. We evaluate the reachability of the trajectory $a_1, a_2, \dots, a_n$ generated by $a_{t+1} \sim \pi_\theta(\cdot|u, m, a_1, o_1, \dots, a_t, o_t) \in \mathcal{A}$. There are multiple interactions between the model and the environment in a muli-turn setting, so failure cases can be divided into three cases(deadend, max step, and invalid). Especially, given the instruction u and generated (action, state) trajectory k, we define $R(u, k)$ as follows:

- $R(u, k) = \text{SUCCESS}$ if $k[-1][1] = goal$
- $R(u, k) = \text{DEADEND}$ if $k[-1][1] \in P$
- $R(u, k) = \text{MAX STEP}$ if $|k| > max$
- $R(u, k) = \text{INVALID}$ if $\exists a \in k[:][0] \mid a \notin A$

[2] In the paper, we set the maximum steps $max = 200$ for each environment. We denote P as the set of deadend states, i.e. set of pits and walls in the grid, and all the states outside the grid.

For each observation, we give the information of the current state along with possible moves that do not directly result the deadend state along with its corresponding states. For example, if the current state s_t is (11, 4) and there the possible actions are right or left, the observation o_t is "Current:\n(11, 4)\nPossible:\n(10, 4)\nleft\n(12, 4)\nright".

D.2 Cognitive map design choices

Our construction of the cognitive map is straightforward; We sample all 4 possible directions("up", "down", "right", and "left") as actions, and we propagate each action iteratively until reaching the goal state. After reaching the goal, we backtrack until reaching the start state. For ablation experiments, we denote ALL as sampling all 4 actions and BACKTRACK as appending backtrack trace after reaching the goal. We experiment effects of excluding each component when designing the cognitive map. First, we see the effect of excluding the backtrack stage(denoted as "w.o. BACKTRACK"). Also, we observe the effect when we only sample possible moves instead of all moves(denoted as "w.o. ALL"). See Appendix E.1 , E.2 for examples.

Marking deadend In this paper, we variate two different cognitive maps when marking the deadend state. We can either verbalize all samples and mark the actions resulting deadend state with a special token(*<deadend>*)(denoted as "MARKING deadend"), or just verbalize all the actions including the deadend states(denoted as "w.o. MARKING"). For example, the verbalized cognitive map at state (0, 0) in Figure 2 is as follows:
MARKING deadend: "(0, 1) up (0, -1) *<deadend>* (1, 0) right (-1, 0) *<deadend>*"
W.O. MARKING: (0, 1) up (0, -1) down (1, 0) right (-1, 0) left".
See Appendix E.1 for the whole construction.

Backward cognitive map construction: BWD Backward chaining is a powerful approach that simplifies complex problems by focusing on the desired outcome and systematically working back to the starting point. LAMBADA (Kazemi et al., 2023) shows that backward chaining helps in reasoning tasks. We adopt the intuition to see if constructing the cognitive map in a backward manner enhances the planning ability of the language model.

In this paper, we define two types of construction, FWD and BWD. For FWD, the construction from the start to the goal is identical to the procedure stated in Appendix C. For BWD, we build the reversed cognitive map starting from the goal state to the start state. We have the sampling function identical to FWD(S). The main experiments are set to BWD. For the propagation stage, we expect a reverse transition state $T^{-1}(s, a) \in \mathcal{S}$ for a given state s and action a. For the backtracking stage, we backtrack the path from start to goal by iteratively searching $T(s, a) \in \mathcal{S}$. See Appendix E.2 for the example.

[2]Although our main analysis consists only of success rate, we provide plots for different types of fails in AppendixF.3

E Cognitive map description

E.1 Cognitive map example: FWD

Table 3 describes a sample of FWD cognitive map construction for each experiment.

Design choice	Cognitive map example
NONE	
COT	Thought:\nStep 1:\nStep 2:\nStep 3:\nStep 4:\nStep 5:\nBacktrack:\n(3, 2)up\n(3, 1)\nright\n(2, 1)\nup\n(2, 0)\nright\n(1, 0)\nright\n(0, 0)
w.o. ALL BACKTRACK	Thought:\nStep 1:\n(0, 1)\nup\n(1, 0)\nright\nStep 2:\n(0, 2)\nup\n(2, 0)\nright\nStep 3:\n(1, 2)\nright\n(2, 1)\nup\nStep 4:\n(3, 1)\nright\nStep 5:\n(3, 2)\nup
w.o. ALL	Thought:\nStep 1:\n(0, 1)\nup\n(1, 0)\nright\nStep 2:\n(0, 2)\nup\n(2, 0)\nright\nStep 3:\n(1, 2)\nright\n(2, 1)\nup\nStep 4:\n(3, 1)\nright\nStep 5:\n(3, 2)\nup\nBacktrack:\n(3, 2)up\n(3, 1)\nright\n(2, 1)\nup\n(2, 0)\nright\n(1, 0)\nright\n(0, 0)
w.o. MARKING BACKTRACK	Thought:\nStep 1:\n(0, 1)\nup\n(0, -1)\ndown\n(-1, 0)\nleft\n(1, 0)\nright\nStep 2:\n(0, 2)\nup\n(0, 0)\ndown\n(-1, 1)\nleft\n(1, 1)\nright\n(1, 1)\nup\n(1, -1)\ndown\n(0, 0)\nleft\n(2, 0)\nright\nStep 3:\n(0, 3)\nup\n(0, 1)\ndown\n(-1, 2)\nleft\n(1, 2)\nright\n(2, 1)\nup\n(2, -1)\ndown\n(1, 0)\nleft\n(3, 0)\nright\nStep 4:\n(1, 3)\nup\n(1, 1)\ndown\n(0, 2)\nleft\n(2, 2)\nright\n(2, 2)\nup\n(2, 0)\ndown\n(1, 1)\nleft\n(3, 1)\nright\nStep 5:\n(3, 2)\nup\n(3, 0)\ndown\n(2, 1)\nleft\n(4, 1)\nright
w.o. MARKING	Thought:\nStep 1:\n(0, 1)\nup\n(0, -1)\ndown\n(-1, 0)\nleft\n(1, 0)\nright\nStep 2:\n(0, 2)\nup\n(0, 0)\ndown\n(-1, 1)\nleft\n(1, 1)\nright\n(1, 1)\nup\n(1, -1)\ndown\n(0, 0)\nleft\n(2, 0)\nright\nStep 3:\n(0, 3)\nup\n(0, 1)\ndown\n(-1, 2)\nleft\n(1, 2)\nright\n(2, 1)\nup\n(2, -1)\ndown\n(1, 0)\nleft\n(3, 0)\nright\nStep 4:\n(1, 3)\nup\n(1, 1)\ndown\n(0, 2)\nleft\n(2, 2)\nright\n(2, 2)\nup\n(2, 0)\ndown\n(1, 1)\nleft\n(3, 1)\nright\nStep 5:\n(3, 2)\nup\n(3, 0)\ndown\n(2, 1)\nleft\n(4, 1)\nright\nBacktrack:\n(3, 2)up\n(3, 1)\nright\n(2, 1)\nup\n(2, 0)\nright\n(1, 0)\nright\n(0, 0)
w.o. BACKTRACK	Thought:\nStep 1:\n(0, 1)\nup\n(0, -1)\ncut\n(-1, 0)\ncut\n(1, 0)\nright\nStep 2:\n(0, 2)\nup\n(0, 0)\ncut\n(-1, 1)\ncut\n(1, 1)\ncut\n(1, 1)\ncut\n(1, -1)\ncut\n(0, 0)\ncut\n(2, 0)\nright\nStep 3:\n(0, 3)\ncut\n(0, 1)\ncut\n(-1, 2)\ncut\n(1, 2)\nright\n(2, 1)\nup\n(2, -1)\ncut\n(1, 0)\ncut\n(3, 0)\ncut\nStep 4:\n(1, 3)\ncut\n(1, 1)\ncut\n(0, 2)\ncut\n(2, 2)\ncut\n(2, 2)\ncut\n(2, 0)\ncut\n(1, 1)\ncut\n(3, 1)\nright\nStep 5:\n(3, 2)\nup\n(3, 0)\ncut\n(2, 1)\ncut\n(4, 1)\ncut
MARKING deadend	Thought:\nStep 1:\n(0, 1)\nup\n(0, -1)\ncut\n(-1, 0)\ncut\n(1, 0)\nright\nStep 2:\n(0, 2)\nup\n(0, 0)\ncut\n(-1, 1)\ncut\n(1, 1)\ncut\n(1, 1)\ncut\n(1, -1)\ncut\n(0, 0)\ncut\n(2, 0)\nright\nStep 3:\n(0, 3)\ncut\n(0, 1)\ncut\n(-1, 2)\ncut\n(1, 2)\nright\n(2, 1)\nup\n(2, -1)\ncut\n(1, 0)\ncut\n(3, 0)\ncut\nStep 4:\n(1, 3)\ncut\n(1, 1)\ncut\n(0, 2)\ncut\n(2, 2)\ncut\n(2, 2)\ncut\n(2, 0)\ncut\n(1, 1)\ncut\n(3, 1)\nright\nStep 5:\n(3, 2)\nup\n(3, 0)\ncut\n(2, 1)\ncut\n(4, 1)\ncut\nBacktrack:\n(3, 2)up\n(3, 1)\nright\n(2, 1)\nup\n(2, 0)\nright\n(1, 0)\nright\n(0, 0)

Table 3: FWD cognitive map example for each experiment

E.2 Cognitive map example: BWD

Table 4 describes a sample of BWD cognitive map construction for each experiment.

Design choice	Cognitive map example
NONE	
COT	Thought:\nStep 1:\nStep 2:\nStep 3:\nStep 4:\nStep 5:\nBacktrack:\n(0, 0)\nright\n(1, 0)\nright\n(2, 0)\nup\n(2, 1)\nright\n(3, 1)\nup\n(3, 2)
W.O. ALL BACKTRACK	Thought:\nStep 1:\n(3, 1)\nup\nStep 2:\n(2, 1)\nright\nStep 3:\n(2, 0)\nup\nStep 4:\n(1, 0)\nright\nStep 5:\n(0, 0)\nright
W.O. ALL	Thought:\nStep 1:\n(3, 1)\nup\nStep 2:\n(2, 1)\nright\nStep 3:\n(2, 0)\nup\nStep 4:\n(1, 0)\nright\nStep 5:\n(0, 0)\nright\nBacktrack:\n(0, 0)\nright\n(1, 0)\nright\n(2, 0)\nup\n(2, 1)\nright\n(3, 1)\nup\n(3, 2)
W.O. MARKING BACKTRACK	Thought:\nStep 1:\n(3, 3)\ndown\n(3, 1)\nup\n(2, 2)\nright\n(4, 2)\nleft\nStep 2:\n(3, 2)\ndown\n(3, 0)\nup\n(2, 1)\nright\n(4, 1)\nleft\nStep 3:\n(2, 2)\ndown\n(2, 0)\nup\n(1, 1)\nright\n(3, 1)\nleft\nStep 4:\n(2, 1)\ndown\n(2, -1)\nup\n(1, 0)\nright\n(3, 0)\nleft\nStep 5:\n(1, 1)\ndown\n(1, -1)\nup\n(0, 0)\nright\n(2, 0)\nleft
W.O. MARKING	Thought:\nStep 1:\n(3, 3)\ndown\n(3, 1)\nup\n(2, 2)\nright\n(4, 2)\nleft\nStep 2:\n(3, 2)\ndown\n(3, 0)\nup\n(2, 1)\nright\n(4, 1)\nleft\nStep 3:\n(2, 2)\ndown\n(2, 0)\nup\n(1, 1)\nright\n(3, 1)\nleft\nStep 4:\n(2, 1)\ndown\n(2, -1)\nup\n(1, 0)\nright\n(3, 0)\nleft\nStep 5:\n(1, 1)\ndown\n(1, -1)\nup\n(0, 0)\nright\n(2, 0)\nleft\nBacktrack:\n(0, 0)\nright\n(1, 0)\nright\n(2, 0)\nup\n(2, 1)\nright\n(3, 1)\nup\n(3, 2)
W.O. BACKTRACK	Thought:\nStep 1:\n(3, 3)\ncut\n(3, 1)\nup\n(2, 2)\ncut\n(4, 2)\ncut\nStep 2:\n(3, 2)\ncut\n(3, 0)\ncut\n(2, 1)\nright\n(4, 1)\ncut\nStep 3:\n(2, 2)\ncut\n(2, 0)\nup\n(1, 1)\ncut\n(3, 1)\ncut\nStep 4:\n(2, 1)\ncut\n(2, -1)\ncut\n(1, 0)\nright\n(3, 0)\ncut\nStep 5:\n(1, 1)\ncut\n(1, -1)\ncut\n(0, 0)\nright\n(2, 0)\ncut
MARKING deadend	Thought:\nStep 1:\n(3, 3)\ncut\n(3, 1)\nup\n(2, 2)\ncut\n(4, 2)\ncut\nStep 2:\n(3, 2)\ncut\n(3, 0)\ncut\n(2, 1)\nright\n(4, 1)\ncut\nStep 3:\n(2, 2)\ncut\n(2, 0)\nup\n(1, 1)\ncut\n(3, 1)\ncut\nStep 4:\n(2, 1)\ncut\n(2, -1)\ncut\n(1, 0)\nright\n(3, 0)\ncut\nStep 5:\n(1, 1)\ncut\n(1, -1)\ncut\n(0, 0)\nright\n(2, 0)\ncut\nBacktrack:\n(0, 0)\nright\n(1, 0)\nright\n(2, 0)\nup\n(2, 1)\nright\n(3, 1)\nup\n(3, 2)

Table 4: BWD cognitive map example for each experiment

	Implicit baseline	Explicit baseline	Cognitive map	
Optimal	NONE	COT	MARKING deadend	W.O. MARKING
BWD	0.190	0.265	0.705	**0.765**
FWD	0.190	0.252	0.585	0.618
Reachable	NONE	COT	MARKING deadend	W.O. MARKING
BWD	0.321	0.287	**0.885**	0.724
FWD	0.321	0.339	0.854	0.816

Table 5: Optimal and reachable rate of generated plans via single- and multi-turn settings: The first two columns(NONE and BACKTRACK) are the baselines for imitation-based learning, and the rest are different design choices of constructing the cognitive map. Also BWD constructs the map starting from the goal state, while FWD starts from the start state. See Appendix E for actual prompts.

	W.O. ALL		with ALL	
Optimal	W.O. ALL BACKTRACK	W.O. ALL	W.O. MARKING	MARKING deadend
BWD	0.296	0.277	**0.765**	0.705
FWD	0.295	0.290	**0.618**	0.585
Reachable	W.O. ALL BACKTRACK	W.O. ALL	W.O. MARKING	MARKING deadend
BWD	0.394	0.283	0.724	**0.885**
FWD	0.416	0.345	0.816	**0.854**

Table 6: Planning performance with ALL exclusion

	W.O. BACKTRACK		with BACKTRACK	
Optimal	W.O. MARKING BACKTRACK	W.O. BACKTRACK	W.O. MARKING	MARKING deadend
BWD	0.406	0.423	**0.765**	0.705
FWD	0.528	0.516	**0.618**	0.585
Reachable	W.O. MARKING BACKTRACK	W.O. BACKTRACK	W.O. MARKING	MARKING deadend
BWD	0.739	0.852	0.724	**0.885**
FWD	0.672	0.624	0.816	**0.854**

Table 7: Planning performance with BACKTRACK exclusion

F Additional results

F.1 Additional investigations

Cognitive map compared with baselines Table 5 shows both the optimal and reachable rate of plans generated by each experiment. As shown in the table, w.o. MARKING shows the best performance among the experiments both for BWD and FWD cognitive map construction in the optimal planning. (76.5% and 61.8% for BWD and FWD construction, respectively). Unlike in the optimal planning setting, MARKING deadend shows the best performance both for BWD and FWD cognitive map construction(88.5% and 85.4% for BWD and FWD construction, respectively). **Our experiments show that cognitive maps improve performance in the Gridworld path planning task.** It boosts the optimal planning performance by up to 57.5% and reachable planning by up to 56.4% with implicit fine-tuning baseline, and enhancing optimal planning by up to 50% and reachable planning by up to 54.6% with CoT fine-tuning baseline. We also qualitatively show the planning performance in both optimal and reachable settings with respect to the size of the grid in Table 1.

Cognitive map further enhances reachability A notable observation is the performance gap between optimal and reachable planning. Reachable planning experiments show significant improvements compared to optimal planning across all configurations, except for ALL BACKTRACK in FWD cognitive map construction. For instance, w.o. BACKTRACK from FWD map construction more than doubled its score (42.3% to 85.2%). This implies that **although the cognitive map was designed to find the optimal plan, it also substantially enhances reachable planning capability.**

BWD approach enhances the performance of the cognitive map As shown in Table 5, both the optimal and reachable planning got the highest performance with BWD cognitive map construction. This aligns with the findings in LAMBADA (Kazemi et al., 2023), where they show that backward chaining helps in reasoning tasks.

Effect of ALL inclusion Both cognitive map constructions w.o. ALL (w.o. ALL BACKTRACK: 29.6% and 29.5% for BWD and FWD construction, respectively; w.o. ALL: 27.7% and 29.0% for BWD and FWD construction, respectively) suffer at generating the optimal plan, achieving under 30% success rate. However, every cognitive map construction with the inclusion of ALL shows better performance, with FWD MARKING deadend being the lowest among them(58.5%).

We observe a similar trend in the reachable planning test. While both baseline methods w.o. ALL performed at best 41.6% (FWD w.o. ALL BACKTRACK), the lowest performance among every cognitive map construction was 72.4% (BWD w.o. MARKING). This implies that the **inclusion of ALL significantly enhances the planning capability, leading to more successful and efficient pathfinding.**

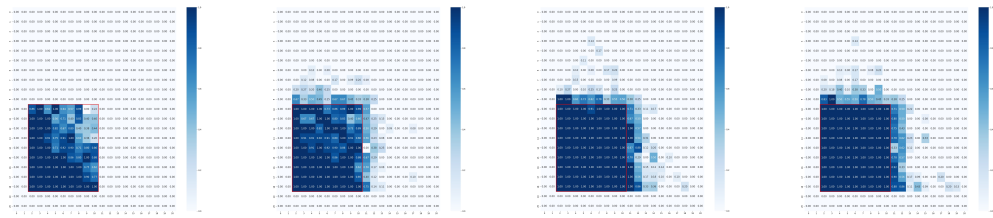
Effect of BACKTRACK inclusion Both cognitive map constructions w.o. BACKTRACK (w.o. MARKING BACKTRACK: 40.6% and 52.8% for BWD and FWD construction, respectively; w.o. BACKTRACK: 42.3% and 51.6% for BWD and FWD construction, respectively) suffer at generating the optimal plan. However, every cognitive map construction with the inclusion of BACKTRACK shows better performance, with FWD MARKING deadend being the lowest among them(58.5%).

The analysis in the reachable planning test was slightly blurry, yet there was an obvious trend. For each experiment, adding backtracking enhanced the performance of the planning in most settings(except BWD construction w.o. MARKING). This implies that the **inclusion of BACKTRACK slightly enhances the planning capability.**

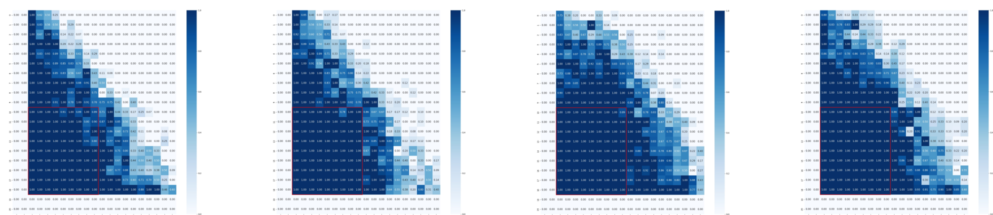
F.2 Vizualization for optimal planning experiments

For optimal planning, we have only success or failure cases. Hence we only provide the success rate for each experiment.

FWD construction See Figure 4 for success rate of each experiment.



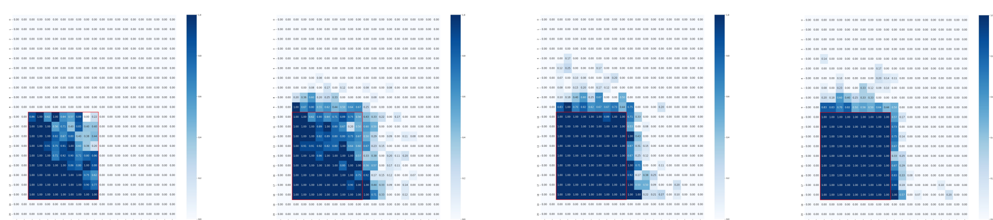
(a) NONE: 19% (b) COT: 25% (c) w.o. ALL BACKTRACK: 29% (d) w.o. ALL: 29%



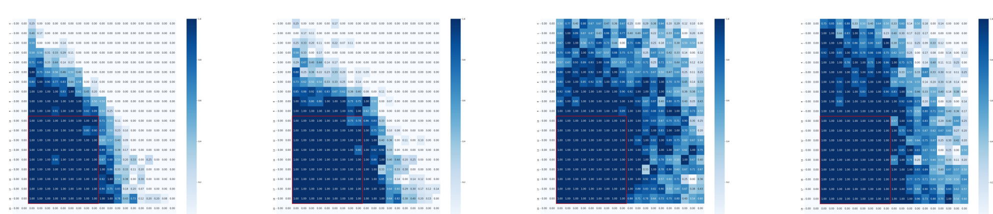
(e) w.o. MARKING BACKTRACK: 53% (f) w.o. BACKTRACK: 52% (g) **w.o. MARKING**: 62% (h) **MARKING deadend: 59%**

Figure 4: Success rate for optimal planning, FWD construction

BWD construction See Figure 5 for success rate of each experiment.



(a) NONE: 19% (b) COT: 27% (c) w.o. ALL BACKTRACK: 30% (d) w.o. ALL: 28%



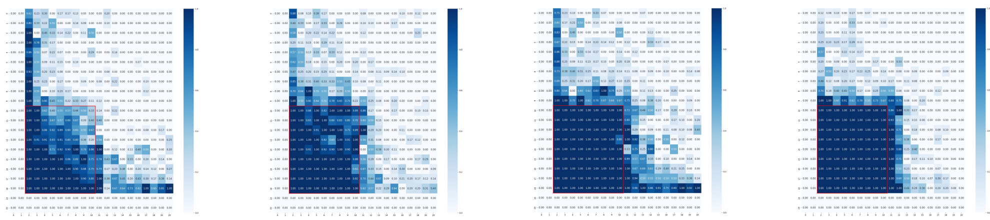
(e) w.o. MARKING BACKTRACK: 41% (f) w.o. BACKTRACK: 42% (g) **w.o. MARKING: 76%** (h) **MARKING deadend: 71%**

Figure 5: Success rate for optimal planning, BWD construction

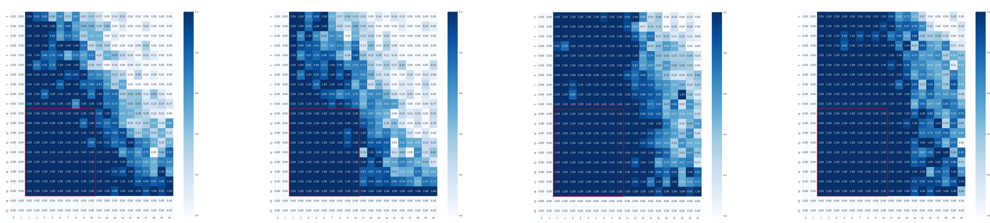
F.3 Visualization for reachable planning experiments

For reachable planning, we have one success cases and three different failure cases(deadend, max step, and invalid). Hence we provide the visualization of all 4 cases.

FWD construction For success rate, see Figure 6. For failure cases, see Figure 7 for deadend, Figure 8 for max step, and Figrue 9 for invalid rate.

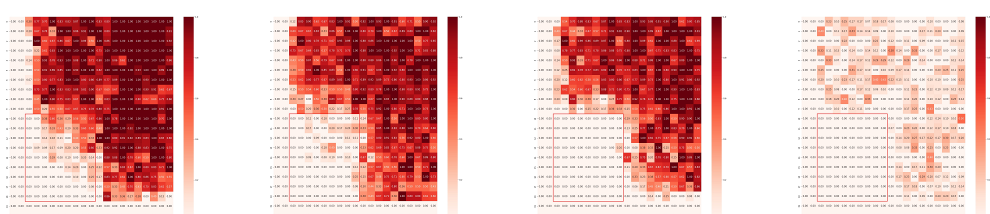


(a) NONE: 32% (b) COT: 34% (c) w.o. ALL BACKTRACK: 42% (d) w.o. ALL: 35%

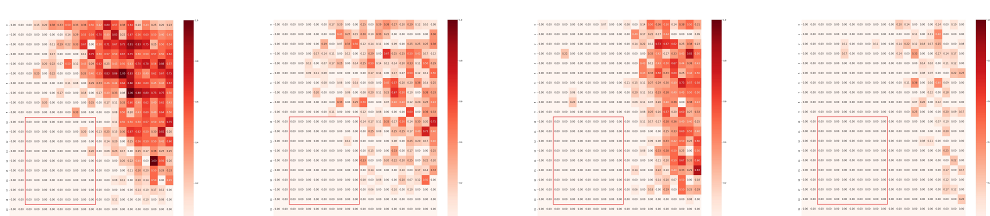


(e) w.o. MARKING BACKTRACK: 67% (f) w.o. BACKTRACK: 62% (g) **w.o. MARKING: 82%** (h) **MARKING deadend: 85%**

Figure 6: Success rate for reachable planning, FWD construction

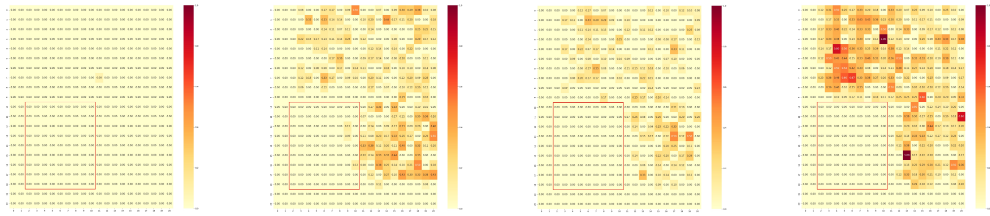


(a) NONE: 64% (b) COT: 58% (c) w.o. ALL BACKTRACK: 51% (d) w.o. ALL: 8%

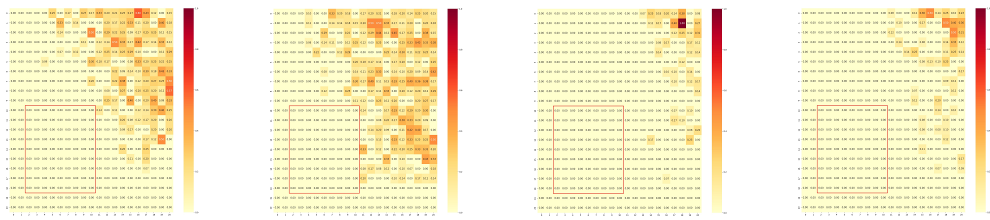


(e) w.o. MARKING BACKTRACK: 29% (f) w.o. BACKTRACK: 10% (g) **w.o. MARKING: 12%** (h) **MARKING deadend: 2%**

Figure 7: Deadend rate for reachable planning, FWD construction

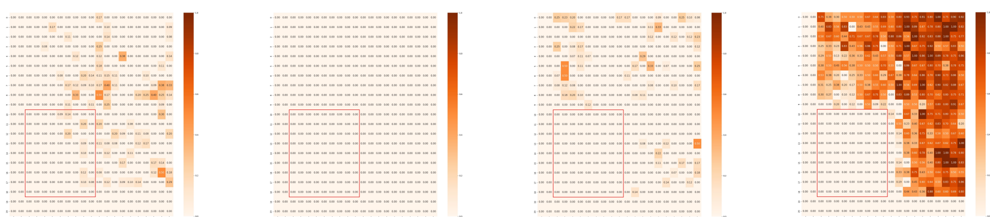


(a) NONE: 0% (b) COT: 8% (c) w.o. ALL BACKTRACK: 5% (d) w.o. ALL: 15%

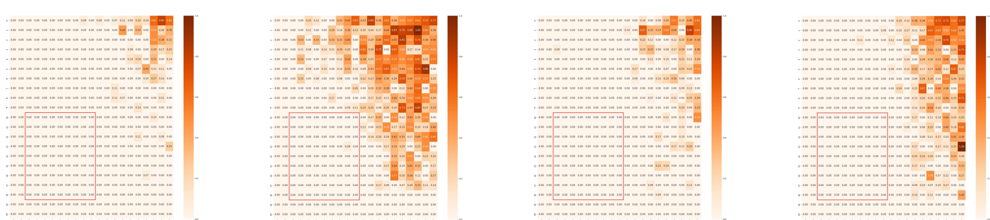


(e) w.o. MARKING BACKTRACK: 7% (f) w.o. BACKTRACK: 10% (g) **w.o. MARKING: 2%** (h) **MARKING deadend: 3%**

Figure 8: Max step rate for reachable planning, FWD construction



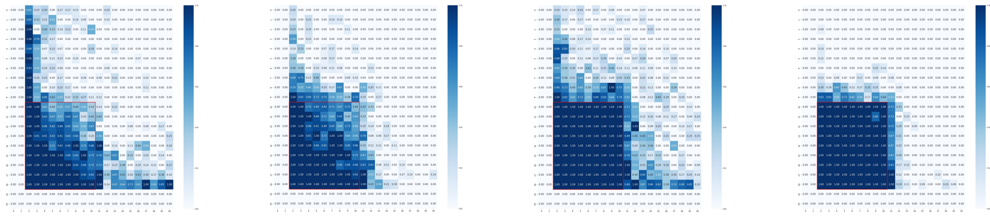
(a) NONE: 3% (b) COT: 0% (c) w.o. ALL BACKTRACK: 3% (d) w.o. ALL: 43%



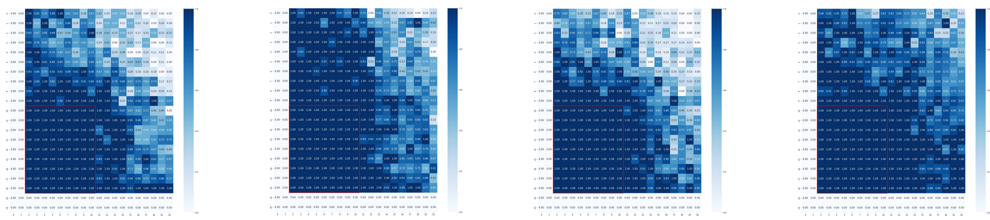
(e) w.o. MARKING BACKTRACK: 3% (f) w.o. BACKTRACK: 18% (g) **w.o. MARKING: 4%** (h) **MARKING deadend: 10%**

Figure 9: Invalid rate for reachable planning, FWD construction

BWD construction For success rate, see Figure 10. For failure cases, see Figure 11 for deadend, Figure 12 for max step, and Figrue 13 for invalid rate.

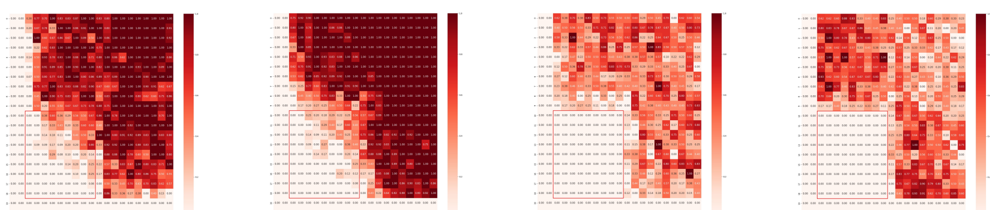


(a) NONE: 32% (b) COT: 29% (c) w.o. ALL BACKTRACK: 40% (d) w.o. ALL: 28%

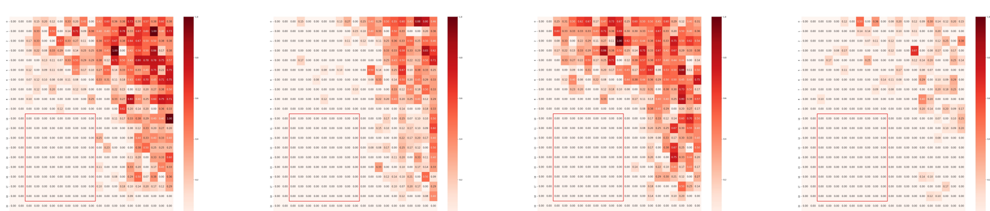


(e) w.o. MARKING BACKTRACK: 74% (f) w.o. BACKTRACK: 85% (g) **w.o. MARKING: 72%** (h) **MARKING deadend: 89%**

Figure 10: Success rate for reachable planning, BWD construction

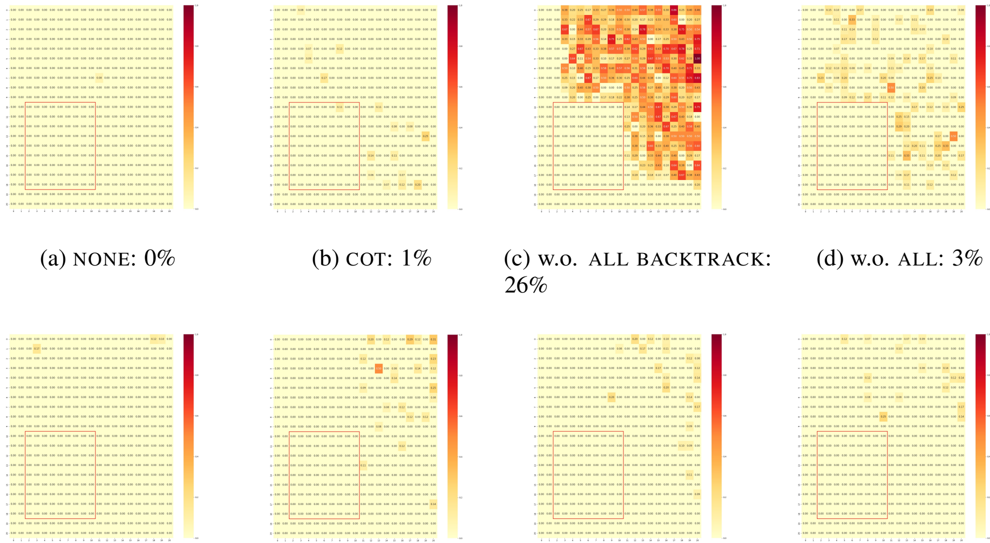


(a) NONE: 64% (b) COT: 70% (c) w.o. ALL BACKTRACK: 33% (d) w.o. ALL: 37%



(e) w.o. MARKING BACKTRACK: 19% (f) w.o. BACKTRACK: 10% (g) **w.o. MARKING: 20%** (h) **MARKING deadend: 3%**

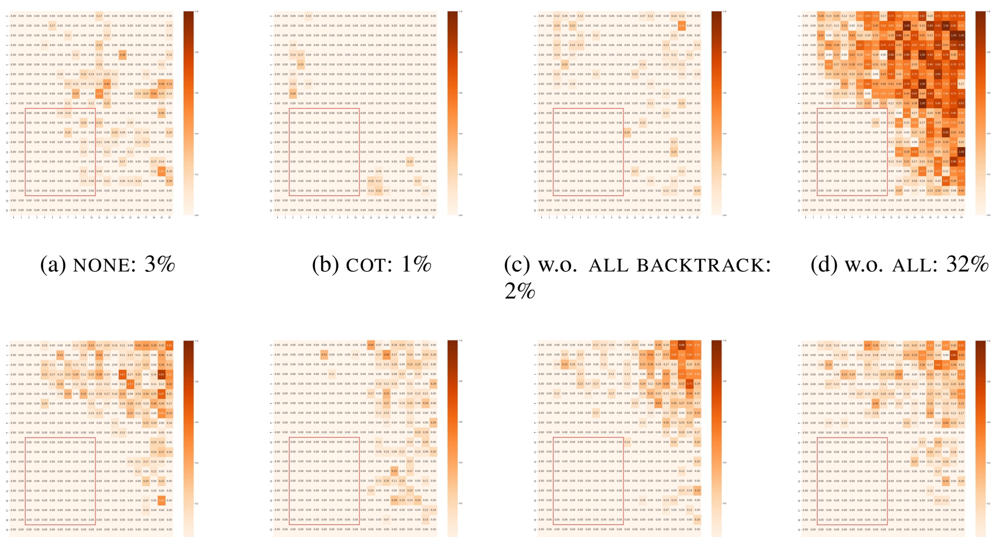
Figure 11: Deadend rate for reachable planning, BWD construction



(a) NONE: 0% (b) COT: 1% (c) w.o. ALL BACKTRACK: 26% (d) w.o. ALL: 3%

(e) w.o. MARKING BACKTRACK: 0% (f) w.o. BACKTRACK: 1% (g) **w.o. MARKING: 1%** (h) **MARKING deadend: 1%**

Figure 12: Max step rate for reachable planning, BWD construction



(a) NONE: 3% (b) COT: 1% (c) w.o. ALL BACKTRACK: 2% (d) w.o. ALL: 32%

(e) w.o. MARKING BACKTRACK: 7% (f) w.o. BACKTRACK: 4% (g) **w.o. MARKING: 7%** (h) **MARKING deadend: 8%**

Figure 13: Invalid rate for reachable planning, BWD construction