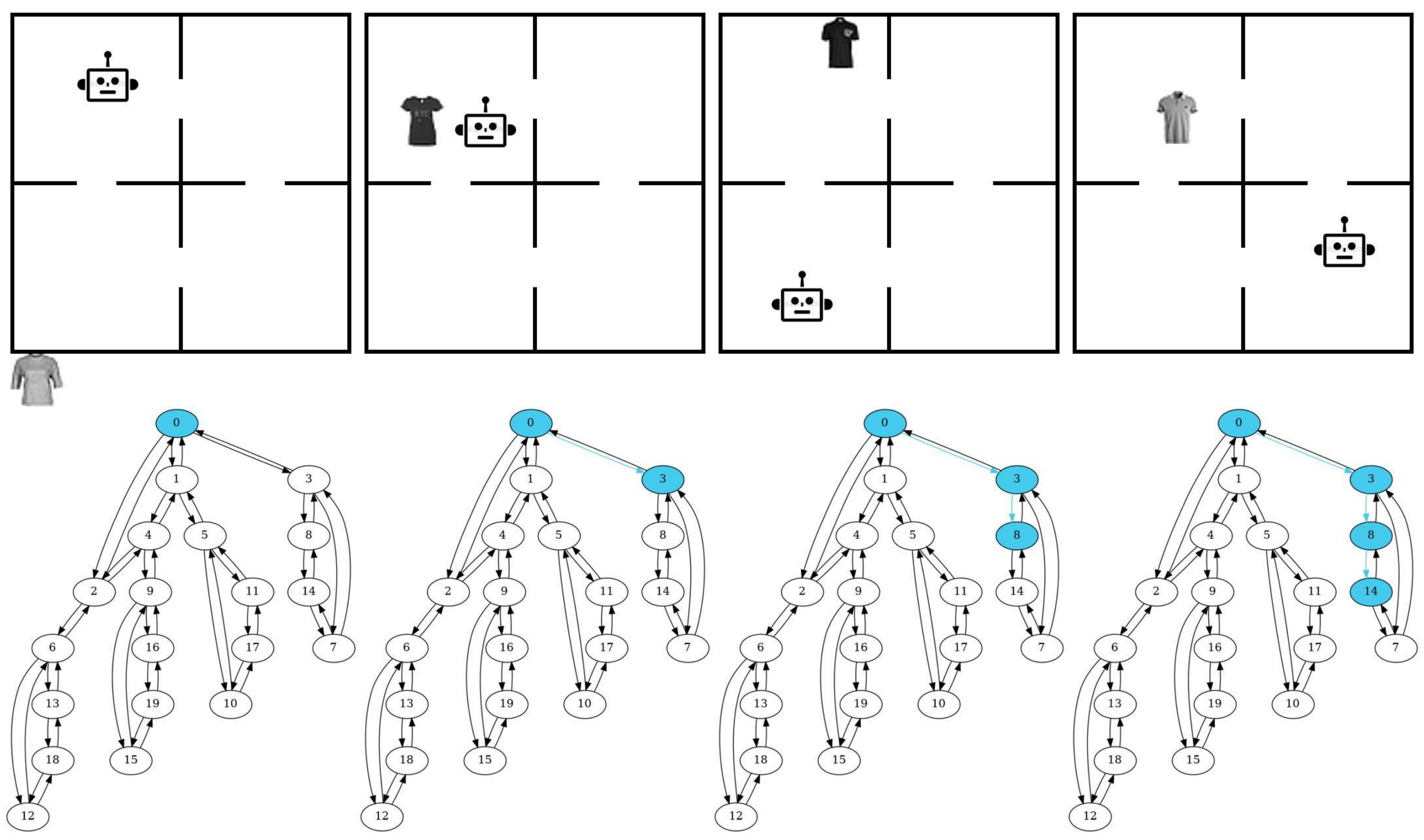
LEARNING TO RECOGNIZE REACHABLE STATES FROM VISUAL DOMAINS

Ella Morgan, Christian Muise

PROBLEM

There's a significant gap between the reasoning capabilities of humans and computers, especially when it comes to making sense of real-world processes. This work aims to bridge this gap through aligning observations from the real world with a given underlying planning model. The goal is to learn which underlying planning state is being represented in a given image, where these observations may have visual variations that may result in them looking visually different, but still represent the same underlying state. Furthermore, we want to take advantage of the given underlying structure to identify when a prediction made is incorrect, to boost prediction accuracy and usability in real life.



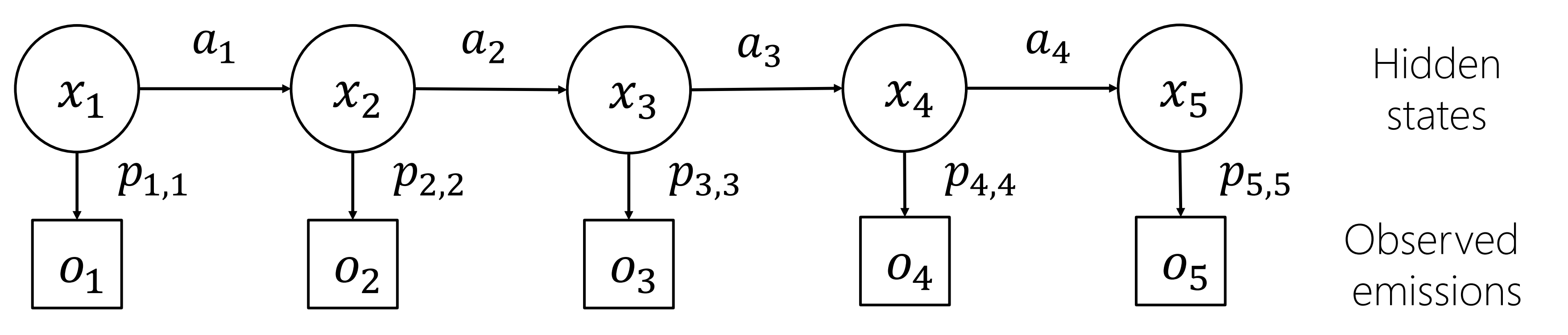
The top of the figure displays four images of visualized states from the grid domain. Possible actions include moving the robot between adjacent rooms, and picking up or placing an object in the room the robot is located in. In the first state, the robot is holding a t-shirt, which is represented by an icon in the bottom left corner. In the next states, the item is placed in the current room, and then the robot moves down and then right. The state space is shown in the graphs below the images, where nodes represent states and edges represent valid actions between states, with the current and previous states highlighted. First, a model is trained to predict the states from the images. Afterwards, we use the structure of the state space to align the sequence of the predictions to a path in the state space graph, maximizing the probability of the sequence of state predictions aligning with the state space.

PLANNING DOMAINS

We formalize a planning state space as $P = \langle S, A, f \rangle$ where S is a finite and non-empty set of states, A is the set of actions, $A(s) \subseteq A$ denotes the actions applicable in each state $s \in S$, and $f(a, s) \in S$ denotes a state transition function for all $s \in S$ and $a \in A(s)$.

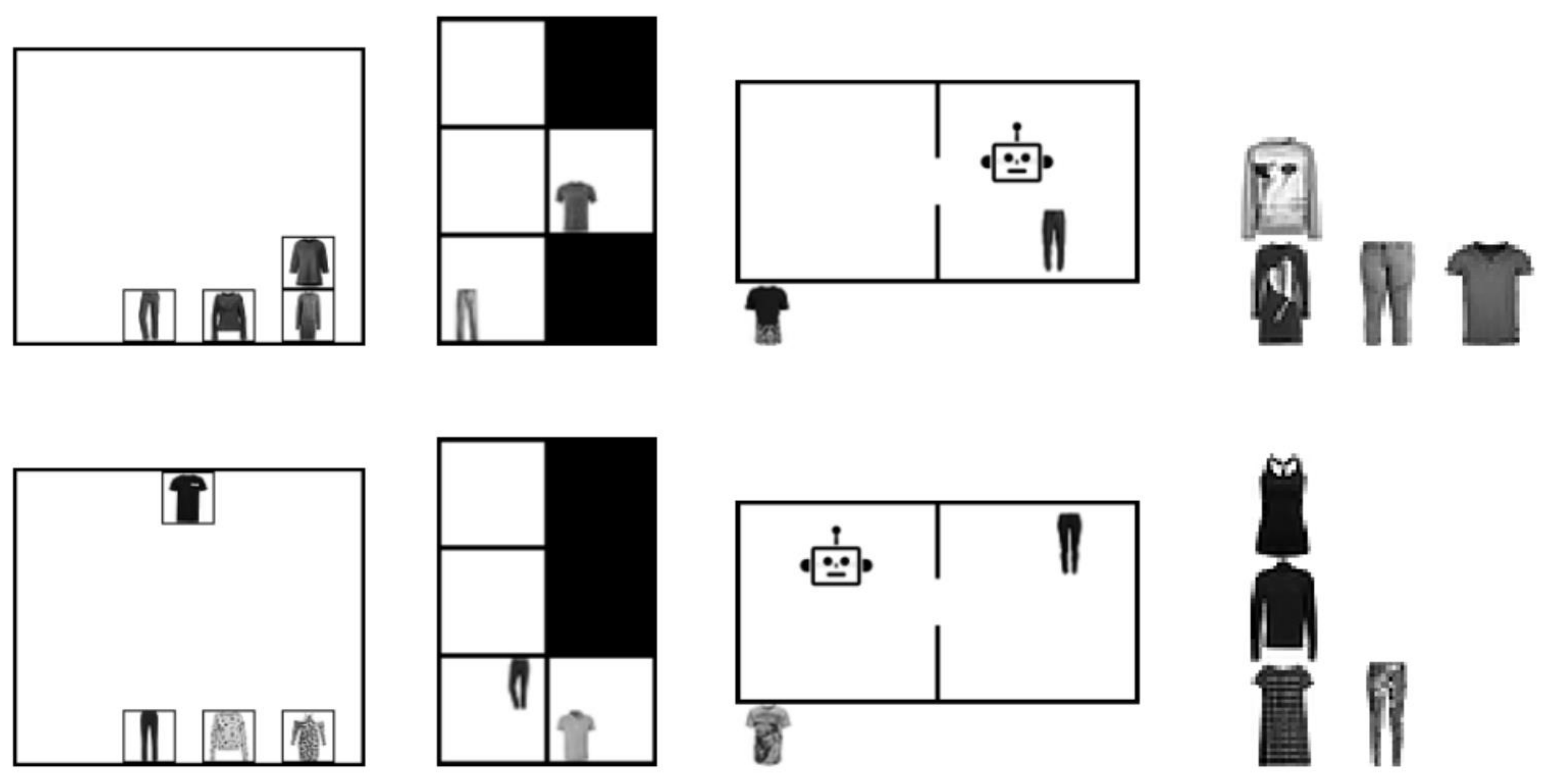
HIDDEN MARKOV MODELS

This problem can be modelled as a hidden Markov model (HMM). A HMM consists of set of states X which produce observations from a set of emissions Y. The states are hidden, and they can only be reasoned about from the sequence of observed emissions. The model is governed by state transition and state emission dynamics.


x_1 x_2 x_3 x_4 x_5
a_1 a_2 a_3 a_4
$p_{1,1}$ $p_{2,2}$ $p_{3,3}$ $p_{4,4}$ $p_{5,5}$
o_1 o_2 o_3 o_4 o_5
Hidden states
Observed emissions

Above is an interpretation of a hidden Markov model to represent planning domains. The sequence $(x_1, x_2, x_3, x_4, x_5)$ represents the transitions undergone between states, where each sequential state x_i is the result of taking an action a_{i-1} in the previous state x_{i-1}. These underlying states are hidden as we must reason about them from the sequence of emitted observations $(o_1, o_2, o_3, o_4, o_5)$, which represent the visualized planning states. We use a trained state prediction model to obtain the emission probabilities of the observations. This model outputs a probability distribution over all states for a given observation, giving the probability $p_{i,j}$ of observation o_i being emitted from state x_j for all $x_j \in S$.

DATA GENERATION



Examples of transitions between pairs of states, where each image on the bottom row is the result of applying an action in the state shown in the top row. From left to right the domains are Blocks World, Elevator, Grid, and Towers of Hanoi, and the actions applied respectively are: picking up a block, moving the elevator down one floor, moving the robot to the room to the left, and moving a disk (represented as an article of clothing to increase the difficulty of the domain) from one peg to another. Objects are samples from fashion-MNIST, and appear in random locations in the rooms in the Elevator and Grid domains to increase variation in the generated images.

ALIGNING ALGORITHMS

Greedy Align

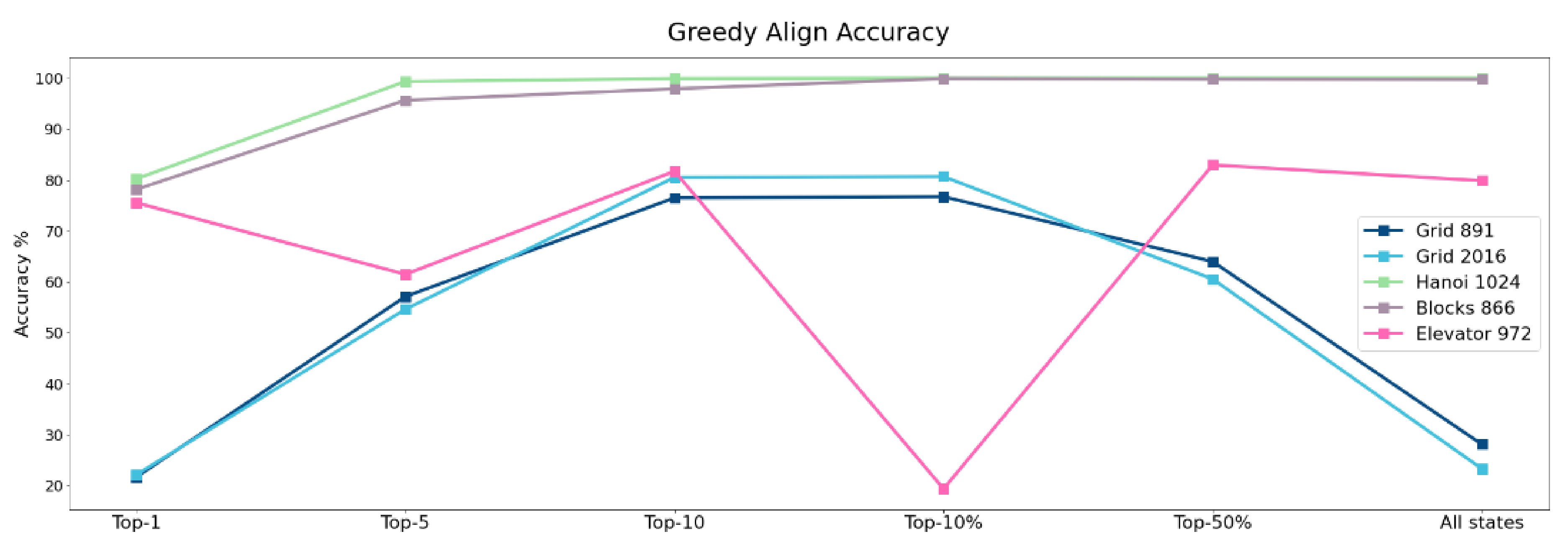
The greedy method constructs the full trace by finding edges in the graph one at a time, starting from the beginning of the trace. It begins by finding the best edge between the first two observations, taking into account the probabilities of both states. This procedure is called if the current state is unknown. A state is unknown if it is either the first state or an edge failed to be found in the previous step. Once we find an edge, we try and continue the path by selecting the next most likely state which continues the connected path.

Viterbi Algorithm

Viterbi's algorithm finds the most probable state sequence by finding the path that maximizes the joint probability of the observed sequence and the sequence of states. It takes in a hidden Markov model and works by computing, for each possible state at each time step, the probability of the most likely path to that state from the initial state, given the observations up to that time step. These probabilities are calculated recursively using the previous probabilities and the transition probabilities between states.

RESULTS

	Grid		Hanoi	Blocks	Elevator
Number of states	891	2016	1024	866	972
Top-1 accuracy	21.68%	22.16%	80.23%	78.15%	75.53%
Top-5 accuracy	85.70%	86.40%	99.92%	99.56%	98.97%
Greedy accuracy	76.72%	80.66%	100%	99.90%	82.96%*
Viterbi accuracy	95.06%	98.38%	100%	95.06%	99.79%
Greedy time	0.47s	1.12s	0.31s	0.26s	0.66s
Viterbi time	254.47s	1329.6s	552.18s	254.47s	265.37s


Greedy Align Accuracy
Accuracy %
Top-1
Top-5
Top-10
Top-10%
Top-50%
All states
Grid 891
Grid 2016
Hanoi 1024
Blocks 866
Elevator 972