LEARNING TO REASON ABOUT AND TO ACT ON PHYSICAL CASCADING EVENTS *

Yuval Atzmon†, Eli A. Meirom†, Shie Mannor, Gal Chechik
NVIDIA Research, Israel
{yatzmon,emeirom}@nvidia.com

ABSTRACT

Reasoning and interacting with dynamic environments is a fundamental problem in AI, but it becomes extremely challenging when actions can trigger cascades of cross-dependant events. We introduce a new learning setup called Cascade where an agent is shown a video of a simulated physical dynamic scene, and is asked to intervene and trigger a cascade of events, such that the system reaches a “counterfactual” goal. For instance, the agent may be asked to “Make the blue ball hit the red one, by pushing the green ball”. The problem is very challenging because agent interventions are from a continuous space, and cascades of events make the dynamics highly non-linear.

We combine semantic tree search with an event-driven forward model and devise an algorithm that learns to search in semantic trees in continuous spaces. We demonstrate that our approach learns to effectively follow instructions to intervene in previously unseen complex scenes. Interestingly, it can use the observed cascade of events to reason about alternative counterfactual outcomes.

1 INTRODUCTION

Cascades of events are wide-spread phenomena found in dynamical systems from biology (gene expression) and physics (meteorology) to chemistry and economics (supply chains). People can reason about such cascades and plan how to intervene to achieve a desired outcome of a system. This builds on several human capacities: inferring causal relations, reasoning in a counterfactual way about possible alternative dynamics of a system, and describing the goal in natural, semantic, language. Cascading dynamical systems are therefore a fantastic test bed for studying complex reasoning. But, how can we train computational agents to reason and intervene in a cascading dynamical system?

Here we address this question in a simulated dynamical system of a physical world, where object interactions form a cascade of events. We describe a new supervised learning setup, called Cascade (Figure 1). In this setup, an agent observes a cascade of events in a system of moving and static objects. It is then provided with a desired goal for the system described in semantic terms. The agent may intervene and manipulate one object to achieve the goal.

More concretely, we consider the following learning setup. At training time, the agent is given the initial conditions and object trajectories of an observed cascade, a semantic goal, and the initial conditions and trajectories of one possible solution. At test time, we sample a new unseen system and goal. The agent only has access to the goal and the “observed” cascade, and it should predict how to change the initial conditions to meet the goal.

This problem is very challenging for several reasons. First, the intervention space is continuous, but is fragmented into many regions, each yielding a different outcome. The region boundaries are hard and with no clear submodularity, as a slight change in movement direction can yield a qualitatively different final outcome. This “butterfly effect” is indeed common in cascading systems. As a result, it is hard to apply standard planning algorithms or gradient-based methods.

To further illustrate the difficulty of this task, consider how hard it is to even apply a “brute-force” approach. Such an approach would actually require access to a simulator of the system to test

* A detailed version of this work is described under: https://arxiv.org/abs/2202.01108
† Equal contribution.
We designed a new environment for evaluating reasoning and intervening in cascading events. The environment combines visual and physical modalities with semantics and action (Figure 1).

Scenes. Each scene describes a unique dynamical system. In these systems, several spheres move freely on a frictionless table, colliding with each other and with static pins within a confined four-
An event-driven forward model. The goal of the forward model is to uncover which semantic events may occur next. Conceptually, our model takes as input a world-state $w_i$, and outputs the next immediate semantic event. Our forward model $f(\cdot)$ is not a full fledged simulator as it does not propagate time at discrete time steps but solves a set of analytic equations to find the event time, should one occur. However, our model only provides an approximation of the dynamics of our test bed environment. These sim-to-sim differences mimic the sim-to-real problem.
Node Expansion. Suppose we decide to expand a node $u$. We apply the forward model $f(\cdot)$ to each world state $w_i \in W_u$. We then aggregate all the propagated world states which share the same immediate next semantic event $s'$ to a new node $u'$: The event sequence of the child node $u'$ is $S_{u'} = \text{concat}(S_u, s')$, the corresponding interventions are $X_{u'} = \{x_i|s' \text{ reached from } w_i\}$ and $W_{u'} = \{f(w_i)|w_i \in W_u, x_i \in X_{u'}\}$.

Expanding the tree can be viewed as a tessellation refinement of the intervention space $\mathcal{X}$. At each step, we pick one cell and split it into multiple cells, where each cell represents a different event that occurs after a shared sequence of events, represented by the parent cell.

If the tree is fully expanded, it covers all possible futures. However, expanding the whole tree is expansive, because the branching ratio, or the number of events, grows quadratically with the number of objects. In the next subsection, we discuss how we learn a scoring function and use it to guide an efficient tree search.

Learning the value function: To find a node that satisfies the goal, we prioritize which node to expand by learning a value function that is conditioned on the instruction $g$. We take a principled probabilistic approach for setting the value function, and set it to the likelihood that the sequence of events of $x \in X_u$ will satisfy the goal $g$, $V(u) = \text{Pr}(x \text{ satisfies } g, x \in X_u)$. This probabilistic perspective allows us to take a maximum-likelihood approach at inference time.

A model for the value function: The model takes as inputs the instruction $g$ and sequence of events $S_u$ that define the node $u$, and predicts a scalar value. We propose to transform each sequence to a Directed Acyclic Graph (DAG) that captures relations in the cascade of events. A node in this DAG is an event that involves some objects (a collision), and each edge represents a dynamic (moving) object shared by two subsequent events (Figure 2). We chose to use a popular message passing GNN model (1) that maintains learnable node, edge and global graph representations.

Inference: Our agent searches the tree for the maximum valued node $u_{M,A,X}$. Then, it randomly selects an intervention from its intervention subset $x \in X_{u_{M,A,X}}$. We consider two variants.

Interventional search: The agent performs a tree search for the highest valued node. At any given step, the agent stores a sorted list of nodes together with their values, it then picks the highest valued node from this list and expands it. The node children are then added to the list with their predicted values, and the agent resorts the list.

Counterfactual search: To improve the interventional search, we use the observed cascade. Consider the case where the sequence of the solution is complex (long sequence) and the observed sequence diverges from the solution at a late point. In this case, it is likely that the observed will be informative about the solution. Therefore, we start the search by expanding the nodes along the observed sequence: For each observed node, we add its children to the list described above together with their predicted values. We then continue the search as described by the “Interventional search”.

4 EXPERIMENTS

We sampled ~46K scenes, each includes 4-6 moving balls, 0-2 pins, and 4 walls and up to 5 semantic instructions (~4.25 on average). An episode is a pair of a scene and one instruction.

Comparison Methods. Our full-fledged approach is ROSETTE (Reasoning On SEManTic TreeEs): Search uses the “counterfactual” variant of the tree search (Section 3), by first expanding the nodes along the “observed” sequence. ROSETTE-IV: Like ROSETTE, but using “Interventional search” (Section 3) without using the “observed” sequence. SEQUENTIAL: Using a sequential representation for a tree chain, instead of a DAG. Deep Sets regression: Embedding the instruction and the initial world state to predict a continuous intervention. Random: Sample intervention at random from an estimated distribution of ground-truth interventions. Brute force: We train a classifier to detect goal satisfaction given a sequence of collisions. Then, search using the event-driven forward model over $10^9$ initial conditions.

Figure 2: Illustrating how a sequence of events (top) is transformed to a DAG (bottom). It corresponds to the video in Figure 1 bottom.
Figure 3: (a) Success rate of our approach and baselines. (b) Tree success rate for variants of the value function. (c) Comparing “Counterfactual search” (ROSETTE) with “Interventional” search (ROSETTE-IV) for easy and hard instructions. (d) As in c, but based on the starting point of the pivot in the sequence. Tail means that the first pivot collision happens after more than 5 events. ROSETTE performs significantly better, showing improvement of 10.7% and 13% respectively.

**Evaluation metrics.** We consider two metrics. **Simulator success rate:** The success rate when rolling out the predicted intervention using a physical simulator (2). This metric mimics experimenting in the real world. **Tree success rate:** We compare the sequence of semantic events from the selected node in our event tree with the events and constraints specified by the instruction. This metric allows us to evaluate the performance of the value function model and tree search, independently from the errors that may be introduced due to the event-driven forward model.

We measured the tree success rate by conditioning on properties of the instruction and scene. (1) **Condition on instruction complexity:** Instructions with 2 or more constraints are marked as “Hard”. (2) **Condition on pivot first collision:** The starting point of the first collision of the pivot in the ground-truth “counterfactual” video. We aggregate the success rate over “Head” cases, where “Pivot 1st collision” ∈ {1 . . . 5}, covering ~80% of the test data, and the remaining “Tail” cases.

**Results:** Table (a) in Figure 3 describes the Tree and the Simulator success rates of ROSETTE and compared methods. ROSETTE achieves the highest success rate for both the “Tree” success rate (61%) and the “Simulated” success rate (48.9%). When evaluating the brute force approach with same computational budget as ROSETTE, expanding 80 events, it performs largely worse than ROSETTE (33.5%). However, when allowing a ×20 computational budget, expanding 4000 nodes (4.5 min/episode vs 13 sec/episode), it reaches a similar success rate as ROSETTE (Tree = 61.9%, Simulated = 53.8%).

We carried out a set of ablation experiments that quantify (1) The benefits of using the “Counterfactual search” (2) Comparing with various value functions: (1) Figure 3(c, d) quantifies the benefit gained by using “Counterfactual” (ROSETTE) over “Interventional” (ROSETTE-IV) search (Section 3). (2) Table (b) in Figure 3 shows the advantage of the probabilistic formulation of the value function (ROSETTE-IV), compared to the several heuristics described in Section 4.

**Qualitative Examples:** Here we provide links to qualitative examples we uploaded to YouTube, best viewed in ×0.25 slow motion. We compare ROSETTE successes with ROSETTE-IV failures. link #1, link #2, link #3, link #4, link #5, link #6. ROSETTE followed the observed cascade along the part of the path that was useful to satisfy the instruction. It diverged from the path when necessary, and found a solution when long cascades were essential, while ROSETTE-IV struggled.

5 **DISCUSSION**

We presented a new learning setup, called Cascade, where an agent observes a cascade of events in a dynamical system and is asked to intervene and changes its initial conditions to meet a given semantic goal. The problem we try to solve is inherently not a standard planning problem. An action is taken essentially only once and there is no way to change the ensuing cascade of events using additional actions (“Fire and forget”). This problem is a complex search problem which one can conceivably try to solve as such using brute-force methods or using some metheuristic approaches.

We use an event tree representation and a principled probabilistic value function for searching efficiently over the space of interventions. In addition, we show that “hot starting” the tree search using the observed cascade improves the success rate.

**Related work** CLEVRER, CRAFT, CATER, IntPhys and CoPhy (3; 4; 5; 6; 7) are video understanding benchmarks, exploring reasoning over observed temporal and causal structures. They focus on understanding, question answering and tracking rather than taking an action as Cascade do. (8; 9; 10; 11), learn physical forward models that focus on object interactions, using a fixed time step, rather than an adaptive time step like we do.
REFERENCES


