Causality-driven Hierarchical Structure Discovery for Reinforcement Learning

Shaohui Peng[1,2,3] Xing Hu[1] Rui Zhang[1,3] Ke Tang[4]
Jiaming Guo[1,2,3] Qi Yi[1,3,6] Ruizhi Chen[2,5] Xishan Zhang[1,3]
Zidong Du[1,3] Ling Li[2,5] Qi Guo[1] Yunji Chen[1,2,†]
[1]SKL of Processors, Institute of Computing Technology, CAS
[2]University of Chinese Academy of Sciences [3]Cambricon Technologies
[4]Department of Computer Science and Engineering, Southern University of Science and Technology
[5]SKL of Computer Science, Institute of Software, CAS
[6]University of Science and Technology of China
{pengshaohui18z, huxing, zhangrui, guojiaming18s,
zhangxishan, duzidong, guoqi, cyj}@ict.ac.cn,
tangk3@sustech.edu.cn, yiqi@mail.ustc.edu.cn,
{ruizhi, liling}@iscas.ac.cn

Abstract

Hierarchical reinforcement learning (HRL) effectively improve agents' exploration efficiency on tasks with sparse reward, with the guide of high-quality hierarchical structures (e.g., subgoals or options). However, how to automatically discover high-quality hierarchical structures is still a great challenge. Previous HRL methods can hardly discover the hierarchical structures in complex environments due to the low exploration efficiency by exploiting the randomness-driven exploration paradigm. To address this issue, we propose CDHRL, a causality-driven hierarchical reinforcement learning framework, leveraging a causality-driven discovery instead of a randomness-driven exploration to effectively build high-quality hierarchical structures in complicated environments. The key insight is that the causalities among environment variables are naturally fit for modeling reachable subgoals and their dependencies and can perfectly guide to build high-quality hierarchical structures. The results in two complex environments, 2D-Minecraft and Eden, show that CDHRL significantly boosts exploration efficiency with the causality-driven paradigm.

1 Introduction

Reinforcement learning (RL) enables the intelligence of the agents by learning to take actions in the environments for maximum rewards [38, 26, 24]. For complex tasks with large state spaces and sparse delayed rewards [17], hierarchical reinforcement learning (HRL) extends RL methods' successes by discovering beneficial hierarchical structures, which helps the agent train multiple levels of policies and explore efficiently using high-level actions [29].

The major challenge of HRL methods is how to discover high-quality hierarchical structures (e.g., subgoals, skills, or options)[1]. A straightforward method is to manually define the hierarchical structures based on task-specific prior knowledge. However, in observing that the hand-crafted

† Corresponding author.

[1]In this paper, the hierarchical structure refers to the subgoal hierarchy in the environment.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

hierarchical structures require human expertise and yield in low generality, some researchers propose methods to discover subgoals automatically [27, 22, 45, 28]. These methods focus on how to discover subgoals from experience but follow an inefficient randomness-driven exploration paradigm to obtain experience. Specifically, the agent explores the environment through the randomness of the policy, which is implemented by raising the entropy of the policy or adding random noise to actions, and then discovers subgoals from the collected experience. Randomness-driven exploration paradigm hardly discovers the high-quality hierarchical structure in complicated environments, such as Minecraft, because of low exploration efficiency, thus limiting the HRL method's performance.

To address the above issue, we propose a Causality-Driven Hierarchical Reinforcement Learning (CDHRL) framework which leverages the causality in the environment as the guidance to discover high-quality hierarchical structures. The key insight is that causality in a system, which explains the dependencies among objects, states, events, or processes, can be naturally used to describe a final goal with sub-goals structurally. Therefore, in CDHRL, we use the causalities among environment variables to guide the discovery of the subgoal hierarchy. Such a causality-driven paradigm avoids the inefficient random exploration process, and significantly enhances the efficiency of discovering hierarchical structures in complicated environments. Moreover, the generated high-quality hierarchical structure, which consists of subgoals and their dependency, is more instructive and efficient during training than the traditional fully-connected hierarchical structure.

In implementing CDHRL, we utilize an iterative boosting framework to progressively discover the causality and construct the hierarchical structure, since it is difficult to discover all causalities in complicated environments directly. The proposed CDHRL consists of two processes, causality discovery and subgoal hierarchy construction. Specifically, causality discovery refers discovery of causality between environment variables. In subgoal hierarchy construction, we find reachable subgoals, which are changes of controllable environment variables, and leverage causality to construct dependency of subgoals. The above two processes can iteratively boost each other: On one side, we can transform discovered causality between environment variables into the subgoal hierarchy. On the other side, we can utilize trained subgoal-based policies to enhance the efficiency of causality discovery. Taking a simplified example in Minecraft play, stonepickaxe (SP) and ironore(IO) are environment variables. After the agent learned subgoals of getting SP, CDHRL could control SP's distribution to discover causality from SP to IO, which indicates that the number of stonepickaxes affects the probability of acquiring ironore. Then, CDHRL adds subgoals about the effect variable IO to the subgoal hierarchy and trains corresponding subgoal-based policies. After that, we in turn discover further causality with IO as cause efficiently by utilizing subgoals about IO to change IO's distribution and collect intervention data. Such a causality-driven discovery paradigm makes the two processes, causality discovery and subgoal hierarchy construction, iteratively promote each other.

We verified our method in two typical complex tasks, including 2d-Minecraft [36] and simplified sandbox survival games Eden [6]. The results show that CDHRL discovers high-quality hierarchical structures and significantly enhances exploration efficiency. Compared to the state-of-the-art HRL method HAC [22] and HRL enhanced by curriculum learning [31], our method significantly outperforms existing studies in terms of both final performance and learning speed.

2 Related Works

2.1 Hierarchical Reinforcement Learning

Discovery of hierarchical structure plays an essential role in the Hierarchical Reinforcement Learning (HRL) algorithm. Some methods reduce the difficulty of discovering the hierarchical structure by adding artificial prior, such as sub-tasks dependency graph [36] and manually defined subgoals [14, 39]. Others may manually restrict the form of hierarchical structure to reduce the search space [16, 15, 9]. By introducing prior information, these methods perform well on specific tasks. However, it not only requires solid expert knowledge but also sacrifices versatility. Our work does not introduce downstream task-specific information, and thus has a broader application.

Some researchers propose methods to discover subgoals automatically. Vezhnevets et al. [41] and Nachum et al. [27] restrict the goal space as state space and train goal-based hierarchical policy end-to-end. Levy et al. [22] enable the multi-level hierarchical structure with hindsight corrections. Some methods discover hierarchical structure by optimizing additional objectives. Zhang et al. [45] assume that the subgoals should be near to the current state. Nachum et al. [28] claim that the optimal

policy based on subgoals should have minimal loss than that based on primitive actions. Pitis et al. [31] consider the training progress for finding the next desired subgoal. Existing methods focus on discovering subgoals from data that is randomly collected. They are often applied in easy-to-explore environments like Maze [4] or Robot [13], but hard work in a complicated environment due to the low exploration efficiency of their randomness-driven exploration paradigm. CDHRL follows the causality-driven exploration paradigm, which is more suitable for complicated environments.

2.2 Causal Reinforcement Learning

Causal reinforcement learning is a research direction that combines the causal effect with reinforcement learning. Pitis et al. [32] leverages local causal structures to improve the sample efficiency of off-policy RL. Méndez-Molina et al. [25] makes a trade-off between exploration and exploitation based given causal graph. Corcoll and Vicente [10] distinguishes the controllable effect of the agent by conducting counterfactual detection. Sontakke et al. [37] and Seitzer et al. [33] discover simple causal influences to improving the efficiency of reinforcement learning. Most of them utilize predefined causality graph as prior knowledge, or detect one-step causality to enhance RL method. However, none of them automatically discover complex causality graph in the environment to guide the exploration of hierarchical structure. Causal discovery methods [18, 19, 21] discover causality between variables through the data, which is an important issue in causal learning. Our method introduces a causal discovery-based exploration dramatically enhances the discovery efficiency of hierarchical structures in complex environments.

2.3 Relational Reinforcement Learning

Relational Reinforcement Learning (RRL) represents entities, actions, and policy by relational language to combine relation learning or reasoning with RL[12], which shares similar motivation with our method. Zambaldi et al. [44] adopt self-attention mechanism upon entities-centric representation to implements relation learning . Wang et al. [42] learns particular relations using a graph neural network to promote zero-shot performance on continuous control tasks. Chitnis et al. [7] and Kokel et al. [20] both learn relation model and planner in first-order language described environment. RRL aims to combine relation induction and reasoning with RL to promote policy generalization, which often works in structural environments with a clear definition of logical predicates. Our method adopts causality, which is more simple and common than logical relation, to guide the direction of subgoals discovery and thus enhance its efficiency in complicated environments.

3 Preliminary

3.1 Subgoal-based Markov Decision Process (MDP)

Hierarchical reinforcement learning that adopts subgoals as hierarchical structure extends the MDP framework by introducing the subgoal space $\mathcal{G}$. Subgoal-based MDP is a six-tuple, $(\mathcal{S}, \mathcal{G}, \mathcal{A}, T, r, \gamma)$, including a state set $\mathcal{S}$, a subgoal space $\mathcal{G}$, and an action set $\mathcal{A}$. The transition probability function $T(s, a, s')$ represents the probability of transition from s to s' with executing action a. $r(s, g, a, s')$ represents the goal-reaching reward function that indicates whether the agent achieves the subgoal g in transition (s, a, s'). And γ is the discount rate. The target of reinforcement learning is to find a subgoal-based policy $\pi : \mathcal{S}, \mathcal{G} \rightarrow \mathcal{A}$ that maximizes the accumulated rewards $R_\pi(s, g) = E[\sum_{n=0}^{\infty} \gamma^n R_{t+n+1} | s_t = s, g_t = g]$. Our proposed framework aims to pre-train the subgoal-based policy for downstream sparse reward tasks.

3.2 Causal Discovery

Causal discovery aims to infer causality through variables data under causal modeling and assumptions. There are two key parts of it: how to represent causality and how to obtain the data for discovering causality.

Structural Causal Model (SCM) is a kind of model that describes the system's causality. SCM over a finite number M of random variables X_i includes a directed acyclic graph (DAG) $C \in \{0, 1\}^{M \times M}$ as the causality graph, and a set of generating functions f_i of variables:

$$X_i := f_i(X_{pa(i,C)}, N_i), \ \ \forall i \in \{0, \ldots, M-1\}. \tag{1}$$

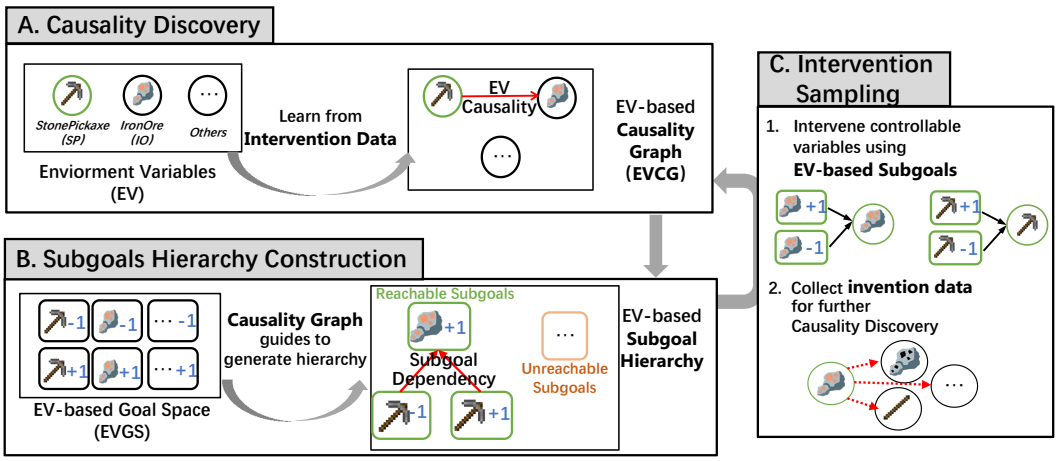


Figure 1: Overview of CDHRL framework. **A. Discover causality between Environment Variables (EVs):** Taking EV in Minecraft as a simplified example, we discover the causality between StonePickaxe (SP) and IronOre (IO) from SP's intervention data. **B. Build EV-based subgoal hierarchy for HRL subgoal training.** Guided by causality from SP to IO, we add reachable subgoals of increasing/decreasing IO values to the subgoal hierarchy and train corresponding subgoal-based policies. **C. Obtaining intervention data for causality discovery in the new round.** After getting subgoal-based policies about IO, the agent can further change IO's distribution and collect intervention data required in causality discovery (Step A) of the next iteration. (The pseudo-code of the framework is in Appendix B.1)

$c_{ij} = 1$ indicates that X_j is the parent of X_i, which means that X_j is the direct cause of X_i. N_i is the jointly-independent noise and $Pa(i, C)$ is the collection of the parent nodes (direct cause variables) of X_i in causality graph C.

Intervention sampling is a typical operation in causal discovery. Unlike standard sampling, it sets the distribution of certain variables in the system as fix value or uniform distribution, then samples to get intervention data. Intervention data has more causal information than observation data. In RL environments, we cannot do intervention directly because the variable generation mechanisms in the environment are unavailable. However, since the agent's behavior can affect the state of the environment, we train subgoal-based policies in CDHRL that can increase or decrease the values of variables to conduct the intervention.

4 Causality-Driven Hierarchical Reinforcement Learning

In this section, we present our Causality-Driven Hierarchical Reinforcement Learning (CDHRL) framework that leverages the causality of Environment Variables for high-efficient hierarchical structure discovery. Figure 1 shows the architecture of CDHRL. To enable the causality discovery and the subgoal hierarchy construction to promote each other, we first adopt environment variables to build a causality-based hierarchical structure that bridges causality discovery and hierarchical reinforcement learning (Section 4.1). Then, we provide the causality discovery methodology in the context of MDP (Section 4.2 & Section 4.4). Additionally, we propose the subgoal policy training methodology based on the causality-based hierarchical structure (Section 4.3).

4.1 Causality-based Hierarchical Structure

To bridge causality discovery and subgoal hierarchy construction, and to make them collaboratively promote each other, we introduce the main components of these two processes, causality graph and goal space, both based on environment variables. The **Environment Variables** (EV) are disentangled factors of the environment observation, which can include noise or lack some key information of the true environment state. In other words, EV does not mean the perfect abstract representation of the environment state that the experts provide. Obtaining EV is relatively convenient for environments providing state vector-based observation since information of different dimensions is disentangled.

For environments with purely image-based observation, finding EV has been largely studied, like CausalVAE [9] and DEAR [10]. Environment Eden and 2d-Minecraft both offer state vector-based observation. Considering that CDHRL focuses on the hierarchical structure discovery guided by the causality, we assume that EV can be obtained from observation by a function, $O : \mathcal{S} \rightarrow \mathcal{X}$, which intercepts part of observation as EV vector. We also clarify EV in detail and conduct a sensitivity analysis on EV in Appendix A to verify the reasonableness of the function O.

Environment Variable-based Causality Graph (EVCG). The causality graph describes the system's causality through a directed acyclic graph (DAG), where the source nodes of directed edges represent cause variables and destination nodes represent effect variables. The causality graph is shown in Figure 1.A. The process learns the environment variable-based causality graph (EVCG) of the RL environment, where nodes consist of environment variables and edges represent causality. Environment Variables are key factors of environment state and the fundamental element of causality. Taking Minecraft as an example, environment variables consist of the number of tools (like StonePickaxe), materials (like IronOre), and surrounding objects (like Pigs). The causality represented by the arrow from StonePickaxe to IronOre means that the number of stone pickaxes the agent-owned affects the probability of acquiring iron ore.

Environment Variable-based Goal Space (EVGS). To leverage the discovered causality between the environment variables, we introduce environment variable-based goal space (EVGS) and construct the subgoal hierarchy upon EVGS, as shown in Figure 1.B. Different from prior studies that adopt the state or latent spaces as subgoal space [22, 28], EVGS consists of goals which are to change the value of environment variables. Thus the subgoal-based policies can be used to change the distribution of environment variables or complete downstream tasks. Formally, we introduce two fundamental changes, Increase (F_i) and Decrease (F_d), for variable values:

$$\begin{aligned} F_i(x_{i,t}, x_{i,t+1}) &= \mathbf{1}_{x_{i,t+1} > x_{i,t}} \\ F_d(x_{i,t}, x_{i,t+1}) &= \mathbf{1}_{x_{i,t+1} < x_{i,t}}. \end{aligned} \tag{2}$$

where $X_{i;t}$ is the value of X_i in state s_t. The EVGS is the Cartesian product of the environment variable set $\mathcal{X}$ and the changes set $\mathcal{F}_{change} = \{F_i, F_d\}$:

$$\mathcal{G}_{EVGS} = \{(X_i, F) | X_i \in \mathcal{X}, F \in \mathcal{F}_{change}\}. \tag{3}$$

The corresponding goal-reaching function $r(s_t, g, a, s_{t+1})$ is:

$$r(s_t, (X_i, F), a, s_{t+1}) = F(O(s_t)_i, O(s_{t+1})_i), \tag{4}$$

where O is the mapping from state to variable values.

Collaboration between Causality Discovery and Subgoal Hierarchy Construction. With the EV-based causality graph and EV-based goal space, we can naturally transform the causality graph into a subgoal hierarchy. The subgoal hierarchy contains reachable subgoals selected from $\mathcal{G}_{EVGS}$ through the causality graph and are organized by the causality between variables. Step A and B in Figure 1 show an example. Specifically, we divide the environment variables into *controllable* and unknown ones for the agent. The controllable variables are variables whose related subgoals have been added into the subgoal hierarchy (green ones, e.g., StonePickaxe) so that the agent can change their distribution to obtain their intervention data. Hence, in each iteration, the agent can discover causality from controllable variables (e.g., StonePickaxe) to the unknown variables (e.g., IronOre). Then, some unknown variables, which as the effect of the causality, are considered as new controllable variables for the agent, because the agent's behavior can affect them and thus recognize their cause variables. The subgoals related to new controllable environment variables are considered to be *reachable*. We then train reachable subgoal-based policies and add them to the subgoal hierarchy. In the next iteration, the agent can change the distribution of new controllable environment variables through subgoal-based policies and discover new causality which takes them as the cause variable. The iteration converges when there is no new causality, and the remained isolated variables are uncontrollable for the agent. Subgoals of uncontrollable variables will not be added to the subgoal hierarchy.

In this way, causality discovery and subgoal hierarchy construction can promote each other. On the one hand, the discovered causality between environment variables guides the agent to find available subgoals accurately and construct the subgoal hierarchy upon EVGS. On the other hand, the agent can utilize trained policies of subgoals to do intervention on environment variables. More complex causality between variables can be discovered along with the sampling data from poor to rich. The alternating iterative optimization above can address the challenge of discovering hierarchical structure.

4.2 Causality Learning

Causality in the Context of Markov Decision Process (MDP). We first introduce two key points of causality discovery in the context of MDP: what causality to discover and how to get learning data. 1) ***Causality is discovered within adjacent steps.*** According to the properties of MDP, the state s_{t+1} is only determined by state s_t and action a_t. Hence, we redefine the causality equation 7 as $X_{i,t+1} := f_i(X_{pa(i,C),t}, N_i)$, which means the variable values at step $t+1$ are only affected by that at step t. In other words, we aim to learn the causality from transition data of adjacent steps in the agent's trajectory. 2) ***Intervention data is obtained through subgoal-based policy.*** As mentioned in the preliminary, it is hard to do intervention in the RL environment directly since the generation mechanisms are unavailable. Therefore, we control the variables' distribution through the agent's behavior to sample intervention data for causal discovery. Specifically, with the application of subgoals in EVGS, the agent can change the distribution of controllable environment variables and collect the intervention data subsequently. More details about the intervention process based on subgoal-based policy are introduced in Section 4.4.

Structural Causal Model Learning. The structural Causal Model (SCM) is a kind of model that describes the system's causality. We adopt a causal discovery methodology similarly to SDI [19] and intervention data collected from the agent's trajectories to learn the environment variables' SCM. As the equation 7 shown, SCM over M variables can be described by two sets of parameters: **structural parameters** $\eta \in R^{M\times M}$ model the causality graph C and **functional parameters** θs model the generating functions f.

The parameters η and θ are optimized through a two-phase iterative and continuous process. In the first function learning phase, we want to fix the input variables of generating function and optimize the functional parameters θ. So we sample hypothesis configuration C of the causality graph from Bernoulli distribution $Ber(\sigma(\eta))$, where σ represents the Sigmoid function. And then, we optimize θ by maximizing the likelihood of the intervention data. In the second structure learning phase, we want to learn the causality graph under fixed generating functions f. We estimate the gradient of η through fixed θ and a REINFORCE-like predictor proposed by Bengio et al. [5]. For the causality from variable X_j to X_i, the gradient g_{ij} of η_{ij} is estimated from X_j's intervention data set $\{X\}$ by the following formula:

$$g_{ij} = \sum_k (\sigma(\eta_{ij}) - c_{ij}^{(k)}) \left(\frac{L_{C,i}^{(k)}(X)}{\sum_{k'} L_{C,i}^{(k')}(X)} \right), \ \forall i \in \{0, \dots, M-1\}, \tag{5}$$

where k represents the k-th draw of hypothesis configuration C. $L_{C,i}^{(k)}(X)$ represents the X_i's likelihood under the intervention data. The two optimization phases cycle until convergence. (Algorithm details are in Appendix B.2).

4.3 Subgoal-based Policy Training

As described in section 4.1, when new causalities are discovered, the subgoals related to new effect variables in the EVGS are added to the subgoal hierarchy by the guidance of new causalities. Figure 1.B shows the effect variable IO's subgoals are added to the subgoal hierarchy. Specifically, we first assumed the new effect variable X_i is controllable, and subgoals of X_i are reachable. We then train X_i's subgoal-based policies $\pi(a|s, (X_i, F))$. To induce an efficient training, the action space of $\pi(a|s, (X_i, F))$ consists of subgoals related to X_i's cause variables and primitive action set $\mathcal{A}$:

$$a \in \{(X_j, F') | X_j \in X_{pa(i,C)}, F' \in \mathcal{F}_{change}\} \bigcup \mathcal{A}. \tag{6}$$

Besides, to verify whether newly associated variables are controllable as supposed, we compare the final training success rates of the new subgoals with a preset threshold ϕ_{causal} before adding them to the subgoal hierarchy. When the corresponding subgoal-based policies are successfully trained, variables will be confirmed as controllable variables and can be intervened to obtain data in the next iteration. Along with the causality discovered from shallow to deep, the subgoal-based policies are trained layer by layer and finally consist of a multi-level hierarchy. After the above process converges to no new causality, subgoals related to remaining isolated variables are taken as unreachable ones. For details on subgoal-based policy training, please refer to the training algorithm in Appendix B.3.

Distinguishing reachable subgoals related to controllable environment variables and filtering unreachable ones during policy training can reduce the goal space reasonably. Meanwhile, through

the dependency of subgoals suggested by causality between environment variables, we construct the action space for the subgoal-based policies, which is more efficient than primitive action space. The above two improvements in policy training lead to a more data-efficient learning process, verified in experiment results in Section 6.2).

4.4 Intervention Sampling

Considering that environment variables in RL environments cannot be directly intervened, we propose a methodology that adopts subgoal-based policies to obtain the intervention data. Benefit from variable-based goal space, the subgoal-based policy could effectively change the distribution of controllable variables.

The detailed intervention sampling includes the following steps: **(1)** For controllable variable X_i, sample the desired value x_i from the uniform distribution, and then use subgoals (X_i, F) to drive the agent to change the value of X_i to x_i; **(2)** Collect the follow-up trajectory until X_i's value changes, and then obtain intervention data of X_i from all variables' values of adjacent steps. At the beginning of the optimization, there are no trained subgoal-based policies. Therefore, we set the agent's action as one of the variables first and do intervention on it to initiate the iterative process.

5 Experiment Setup

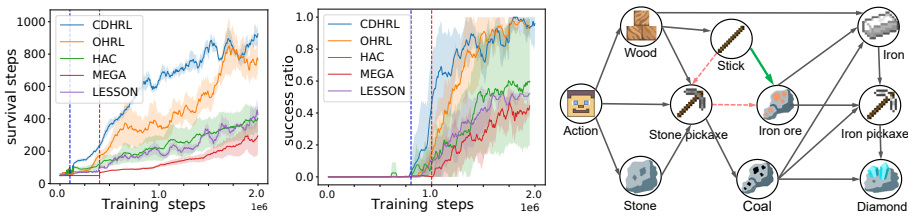
Environment To verify the effectiveness of CDHRL, we choose two environments with discrete action space, namely 2d-Minecraft [36] and Eden [6] (more details of environments are introduced in Appendix C.1. They have complex hierarchical structures and very sparse rewards. **2D-Minecraft** [36] is a simplified 2D version of the famous Minecraft [17]. In one episode with limited steps, the agent must navigate the map, pick up various materials, and craft tools until obtaining the diamond. The agent only can get a positive reward after getting the diamond. **Eden** [6] is a survival game which is similar to "Don't Starve." To make a living in a 40×40 grid world, the agent with 8×8 limited vision range must chase animals to obtain food for maintaining satiety, find rivers to get water for preventing being thirsty, and collect materials to craft tools for surviving in the night. The agent takes survival as the unique target and cannot get any positive rewards except a negative reward after it dies and ends the episode.

Implementation We implement the agent based on multi-level DQN with HER [2], considering the action space is discrete. The training is divided into two stages, pre-training and adaptation. In the pre-training stage, we train multi-level subgoal-based policies until no new causality is discovered. In the adaptation stage, we train an upper controller on the pre-trained subgoals to maximize the task reward. The SCM is implemented like SDI [19]. The environment supports the function O from environment state to variable values.

Baselines We compare CDHRL with the following three methods for validation. For a fair comparison, we use the environment variable-based goal space $\mathcal{G}_{EVGS}$ to replace the goal space in these algorithms to ensure that the performance promotion is because of the difference in hierarchical structure discovery methods. **Oracle HRL (OHRL)** is implemented as a two-level DQN with HER and an *oracle goal space*. The goal space is a subset of $\mathcal{G}_{EVGS}$ after artificial eliminating unreachable and useless subgoals. **HAC** [22] is a powerful goal-conditioned HRL that discovers subgoals with a randomness-driven exploration paradigm. We implement a two-layer HAC **with subgoal space** $\mathcal{G}_{EVGS}$. **LESSON** [23] is a modified goal-conditioned HRL method based on HAC that discovers subgoals from slowly changed features. **MEGA** [31] is a kind of goal-conditioned HRL enhanced by curriculum learning, which pre-train subgoals in the order of their training progress. We set the initial subgoal distribution as the uniform distribution on $\mathcal{G}_{EVGS}$ and pre-trains subgoals with enough steps. After that, like the adaption stage in CDHRL, we train an upper controller on the trained subgoals.

6 Results

We first carry out extensive experiments to compare the performance of CDHRL and the state-of-the-arts in complex environments with a hard-to-explore hierarchical structure. Then, we dig deep ablation studies to show why CDHRL triumphs over the existing HRL paradigm. At last, we showcase the discovered causality-based hierarchical structure, which can be reasonably interpreted.



(a) Eden (b) 2D-Minecraft (c) Causality Graph in Minecraft

Figure 2: (a) Agent's survival time in Eden. (b) Success ratio in 2D-Minecraft. The vertical dotted lines indicate the end of pre-training of CDHRL and MEGA. Results are derived from average data in 8 trials. (c) The causal graph of 2D-Minecraft discovered by the agent. Some uncontrollable variables that unlinked are ignored here.

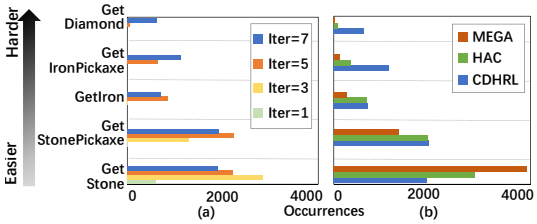


Figure 3: Exploration capability comparison. We select five explore milestones in 2D-Minecraft from easy-to-explore to hard-to-explore and record the occurrences in 10K test episodes to compare the agent's exploration capability. More occurrences on the hard-to-explore milestone represent higher exploration capability. (a) CDHRL's exploration capability iteratively increases along with the construction of hierarchical structures. (b) CDHRL shows much better exploration capability than HAC and MEGA. All compared methods are tested after trained 800K steps.

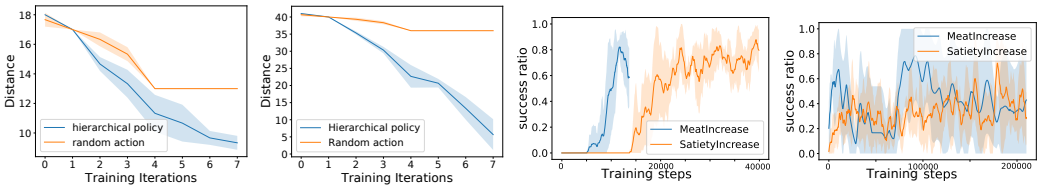
6.1 Comparative Analysis

We compare the performance on Eden and 2D-Minecraft of CDHRL with other methods. Figure 2 shows that CDHRL learns faster and achieves the best performance even though all the evaluated methods have the same subgoal space. Concretely, OHRL, HAC and LESSON do not have the pre-training stage and start learning directly. However, the success ratios of OHRL, HAC and LESSON in the first 1e6 steps of 2(b) keep nearly zero since they seldomly obtain the success trajectory data of getting diamonds. CDHRL learns much faster even if there is pre-training time overhead. MEGA has the pre-training stage for curriculum learning to identify the near-term and long-term subgoals. However, it fails to identify the effective hierarchical structures in these two environments with relatively large sub-goal space. The hierarchical structure discovered by CDHRL not only significantly enhances the agent's exploration capability in sparse reward, but also identifies the reachable subgoals that promotes the learning speed of subgoal-based policies. We will explain these phenomenons further in Section 6.2.

6.2 Insights

Further, we evaluate the detailed exploration capability and causal discovery efficiency and show how CDHRL outperforms existing HRL paradigms. We have two observations from the experiment results:
1) The causality-driven exploration paradigm enhances the exploration capability significantly. We experimentally evaluate the agent's exploration capability in 2D-Minecraft based on the achieved times of exploration milestones in the environment. Figure 3(b) shows the exploration capability comparison of different methods. The CDHRL's achieved times of hard-to-explore milestones are much higher than HAC and MEGA, indicating a much better exploration capability of CDHRL. Figure 3(a) shows the exploration capability increases along with iterations of causality discovery in CDHRL. The reason is that the agent's causality-driven exploration becomes more efficient as more subgoal-based policies and controllable variables, which other HRL methods cannot achieve. In conclusion, with the help of the causality-based exploration paradigm, the agent can achieve better exploration efficiency.

2) The subgoal hierarchy construction and causality discovery iteratively promote each other. On the one hand, the hierarchical structure makes the causality graph more accurate. We train the



(a) SHD (b) SID (c) CDHRL (d) MEGA

Figure 4: (a) Structure Hamming Distance (SHD) and (b) Structural Interventional Distance (SID) between learned and ground-truth causality graph. The intervention data sampled by the assistance of hierarchical policy make the causality graph more accurate. The ground-truth causality graphs of Eden and 2D-Minecraft are in Appendix C.2.(c) and (d) are learning curve of two subgoals.

causal model in 2D-Minecraft using intervention data collected by hierarchical policy and random action, respectively. We record the causality graph of each iteration and compute their *Structure Hamming Distance* (SHD) and *Structural Interventional Distance* (SID) relative to the ground truth, where SHD and SID are typical metrics for the accuracy of causality graphs. The lower distance means the more accurate causality graph. Figure 4(a) and 4(b) show the result. The accuracy of the causality graph learned using the hierarchical policy is much better than using random action, especially for SID. Such results indicate that utilizing the hierarchical policy to obtain intervention data is essential for causality discovery.

On the other hand, the construction of the subgoal hierarchy is more efficient and reasonable. We compare the convergence speed and stableness of CDHRL and MEGA to achieve two subgoals in Eden. For these two subgoals, $MeatIncrease = (X_{Meat}, F_i)$ and $SatietyIncrese = (X_{Satiety}, F_i)$, the former is the prerequisite goal of the latter one and should be learnt by agent first. Figure 4(c) and 4(d) show that CDHRL achieves a faster and more stable learning process than MEGA. We attribute the poor performance of MEGA to the fact that it cannot identify the dependency of the subgoals and is unable to learn subgoals in order, resulting in much lower performance, especially when the environment is complex with a complicated dependency of subgoals. In contrast, since X_{Meat} is the cause variable of $X_{Sariety}$, CDHRL infers the learning order of subgoals based on discovered causality and achieves stable and fast learning progress. Additionally, the reduced action space of multi-level policy transformed from causality also improves training efficiency.

6.3 An Showcase of Causality-based Hierarchical Structure

We showcase the established hierarchical structure based on discovered causality in 2D-MineCraft from the causality graph view. The learned causality graph can be reasonably interpreted. As shown in Figure 2(c), most of the causality in the graph meets with human cognition. We also find an interesting phenomenon that CDHRL sometimes may discover long-term causality and ignore corresponding short-term causality, e.g. “Stick” to “Iron re” (green edge) instead of “Stick” to “Stone pickaxe” to “Iron ore” (red edges) in Figure 2(c).CDHRL may benefit from such phenomenons because cutting unimportant causality details can improve the training speed and the generalization of hierarchical structure. Although there is some deviation between the discovered and the ground-truth causality graph, the causality-based hierarchical structure still improves the exploration efficiency.

7 Conclusion

Hierarchical reinforcement learning methods can hardly work well in complicated environments because of the low exploration efficiency of the randomness-driven exploration paradigm when discovering the hierarchical structure. We propose a causality-driven hierarchical reinforcement learning framework (CDHRL) to address the challenge, which autonomously builds high-quality hierarchical structures in an iterative boosting way. In two complicated environments with very sparse rewards, 2d-Minecraft and Eden, our method discovers a high-quality subgoal hierarchy and significantly enhances exploration efficiency. Limited in causal discovery algorithm capability and the complexity of variable changes set, our method can only be applied in environments with discrete environment variables. Moreover, CDHRL needs an extra disentangled encoder in purely image-based environments for finding environment variables. We leave broadening the application scope of CDHRL as our future work.

Acknowledgements

This work is partially supported by the National Key Research and Development Program of China(under Grant 2018AAA0103300), the NSF of China(under Grants 61925208, 62002338 62102399, , U19B2019, 61906179, 61732020), Strategic Priority Research Program of Chinese Academy of Science (XDB32050200), Beijing Academy of Artificial Intelligence (BAAI) and Beijing Nova Program of Science and Technology (Z191100001119093) , CAS Project for Young Scientists in Basic Research(YSBR-029), Youth Innovation Promotion Association CAS and Xplore Prize.

References

[1] Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R. Devon Hjelm. Unsupervised state representation learning in atari. In *NeurIPS*, pages 8766–8779, 2019.

[2] Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *NIPS*, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/453fadbd8a1a3af50a9df4df899537b5-Abstract.html`.

[3] Akhil Bagaria and George Konidaris. Option discovery using deep skill chaining. In *ICLR*, 2020. URL `https://openreview.net/forum?id=B1gqipNYwH`.

[4] Akhil Bagaria and George Dimitri Konidaris. Option discovery using deep skill chaining. In *ICLR*, 2020.

[5] Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Nan Rosemary Ke, Sébastien Lachapelle, Olexa Bilaniuk, Anirudh Goyal, and Christopher J. Pal. A meta-transfer objective for learning to disentangle causal mechanisms. In *ICLR*, 2020. URL `https://openreview.net/forum?id=ryxWIgBFPS`.

[6] Ruizhi Chen, Xiaoyu Wu, Yansong Pan, Kaizhao Yuan, Ling Li, TianYun Ma, JiYuan Liang, Rui Zhang, Kai Wang, Chen Zhang, Shaohui Peng, Xishan Zhang, Zidong Du, Qi Guo, and Yunji Chen. Eden: A unified environment framework for booming reinforcement learning algorithms, 2021.

[7] Rohan Chitnis, Tom Silver, Joshua B. Tenenbaum, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Glib: Efficient exploration for relational model-based reinforcement learning via goal-literal babbling. In *AAAI*, 2021.

[8] Caleb Chuck, Supawit Chockchowwat, and Scott Niekum. Hypothesis-driven skill discovery for hierarchical deep reinforcement learning. In *IROS*, pages 5572–5579. IEEE, 2020.

[9] Caleb Chuck, Supawit Chockchowwat, and Scott Niekum. Hypothesis-driven skill discovery for hierarchical deep reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5572–5579. IEEE, 2020. doi: 10.1109/IROS45743.2020.9340891. URL `https://doi.org/10.1109/IROS45743.2020.9340891`.

[10] Oriol Corcoll and Raul Vicente. Disentangling causal effects for hierarchical reinforcement learning. *CoRR*, abs/2010.01351, 2020.

[11] Oriol Corcoll and Raul Vicente. Disentangling causal effects for hierarchical reinforcement learning. *CoRR*, abs/2010.01351, 2020. URL `https://arxiv.org/abs/2010.01351`.

[12] Saso Dzeroski, Luc De Raedt, and Hendrik Blockeel. Relational reinforcement learning. In Jude W. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*, pages 136–143. Morgan Kaufmann, 1998.

[13] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *ArXiv*, abs/1802.06070, 2019.

[14] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. In *ICLR*, 2017. URL https://openreview.net/forum?id=B1oK8aoxe.

[15] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*. PMLR, 2018. URL http://proceedings.mlr.press/v80/florensa18a.html.

[16] Sébastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *CoRR*, abs/1708.02190, 2017. URL http://arxiv.org/abs/1708.02190.

[17] William H. Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. In *IJCAI*. ijcai.org, 2019. doi: 10.24963/ijcai.2019/339. URL https://doi.org/10.24963/ijcai.2019/339.

[18] Amin Jaber, Murat Kocaoglu, Karthikeyan Shanmugam, and Elias Bareinboim. Causal discovery from soft interventions with unknown targets: Characterization and learning. In *NIPS*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6cd9313ed34ef58bad3fdd504355e72c-Abstract.html.

[19] Nan Rosemary Ke, Olexa Bilaniuk, Anirudh Goyal, Stefan Bauer, Hugo Larochelle, Chris Pal, and Yoshua Bengio. Learning neural causal models from unknown interventions. *CoRR*, abs/1910.01075, 2019. URL http://arxiv.org/abs/1910.01075.

[20] Harsha Kokel, Arjun Manoharan, Sriraam Natarajan, Balaraman Ravindran, and Prasad Tadepalli. Reprel: Integrating relational planning and reinforcement learning for effective abstraction. In *ICAPS*, 2021.

[21] Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu, and Simon Lacoste-Julien. Gradient-based neural DAG learning. In *ICLR*, 2020. URL https://openreview.net/forum?id=rklbKA4YDS.

[22] Andrew Levy, George Dimitri Konidaris, Robert Platt Jr., and Kate Saenko. Learning multi-level hierarchies with hindsight. In *ICLR*, 2019. URL https://openreview.net/forum?id=ryzECoAcY7.

[23] Siyuan Li, Lulu Zheng, Jianhao Wang, and Chongjie Zhang. Learning subgoal representations with slow dynamics. In *ICLR*. OpenReview.net, 2021.

[24] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR*, 2016. URL http://arxiv.org/abs/1509.02971.

[25] Arquímides Méndez-Molina, Ivan Feliciano-Avelino, Eduardo F. Morales, and Luis Enrique Sucar. Causal based q-learning. *Res. Comput. Sci.*, 149(3):95–104, 2020. URL http://rcs.cic.ipn.mx/2020_149_3/Causal%20Based%20Q-Learning.pdf.

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, and et al. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015. doi: 10.1038/nature14236. URL https://doi.org/10.1038/nature14236.

[27] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *NIPS*, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/e6384711491713d29bc63fc5eeb5ba4f-Abstract.html.

[28] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. In *ICLR*, 2019. URL https://openreview.net/forum?id=H1emus0qF7.

[29] Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine. Why does hierarchy (sometimes) work so well in reinforcement learning? *CoRR*, abs/1909.10618, 2019. URL http://arxiv.org/abs/1909.10618.

[30] Vihang P. Patil, Markus Hofmarcher, Marius-Constantin Dinu, Matthias Dorfer, Patrick M. Blies, Johannes Brandstetter, José Antonio Arjona-Medina, and Sepp Hochreiter. Align-rudder: Learning from few demonstrations by reward redistribution. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 17531–17572. PMLR, 2022.

[31] Silviu Pitis, Harris Chan, Stephen Zhao, Bradly C. Stadie, and Jimmy Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *ICML*. PMLR, 2020. URL http://proceedings.mlr.press/v119/pitis20a.html.

[32] Silviu Pitis, Elliot Creager, and Animesh Garg. Counterfactual data augmentation using locally factored dynamics. *ArXiv*, abs/2007.02863, 2020.

[33] Maximilian Seitzer, Bernhard Schölkopf, and Georg Martius. Causal influence detection for improving efficiency in reinforcement learning. In *NeurIPS*, pages 22905–22918, 2021.

[34] Xinwei Shen, Furui Liu, Hanze Dong, Qing Lian, Zhitang Chen, and Tong Zhang. Disentangled generative causal representation learning. *ArXiv*, abs/2010.02637, 2020.

[35] Alexey Skrynnik, Aleksey Staroverov, Ermek Aitygulov, Kirill Aksenov, Vasilii Davydov, and Aleksandr I. Panov. Forgetful experience replay in hierarchical reinforcement learning from expert demonstrations. *Knowl. Based Syst.*, 218:106844, 2021.

[36] Sungryull Sohn, Junhyuk Oh, and Honglak Lee. Hierarchical reinforcement learning for zero-shot generalization with subtask dependencies. In *NIPS*, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/018dd1e07a2de4a08e6612341bf2323e-Abstract.html.

[37] Sumedh A. Sontakke, Arash Mehrjou, Laurent Itti, and Bernhard Schölkopf. Causal curiosity: RL agents discovering self-supervised experiments for causal representation learning. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 9848–9858. PMLR, 2021.

[38] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*. The MIT Press, 1999. URL http://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation.

[39] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J. Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *AAAI*. AAAI Press, 2017. URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14630.

[40] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pages 5026–5033. IEEE, 2012.

[41] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *ICML*. PMLR, 2017. URL http://proceedings.mlr.press/v70/vezhnevets17a.html.

[42] Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *ICLR*, 2018.

[43] Mengyue Yang, Furui Liu, Zhitang Chen, Xinwei Shen, Jianye Hao, and Jijie Wang. Causalvae: Disentangled representation learning via neural structural causal models. *CVPR*, 2021.

[44] Vinícius Flores Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David P. Reichert, Timothy P. Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew M. Botvinick, Oriol Vinyals, and Peter W. Battaglia. Deep reinforcement learning with relational inductive biases. In *ICLR*, 2019.

[45] Tianren Zhang, Shangqi Guo, Tian Tan, Xiaolin Hu, and Feng Chen. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. In *NIPS*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f5f3b8d720f34ebebceb7765e447268b-Abstract.html.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 7
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] We adopt open-source experimental environments.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We offer the code of our method in the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] There are no personally identifiable information or offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Environment Variables (EV)

A.1 Clarification of oracle Environment Variables (EV)

We first clarify environment variables (EV) from the following aspects:

(1) What is EV: EV are disentangled factors in the environment observation rather than perfect abstract representation of the environment state that experts provide. It comes from observation, and can include noise or lacks some key information of the true environment state. We conduct sensitivity analysis experiments of EV in appendix A that verify the quality of EV does not seriously hurt the performance of CDHRL.

(2) How to acquire EV in practicality: For environments providing state vector-based observation (which cover a broad category of environments, such as Atari [1], Mujoco [40], 2d-Minecraft [36], and so on), obtaining "Environment Variables" (EV) is relatively convenient. The most significant property of the EV is disentanglement. The state vector-based observation can be directly transformed into the EV vector since information of different dimensions is disentangled. Many HRL methods [23, 36, 11, 3, 8] commonly use them as the standard inputs. For example, LESSON [23] assumes that each input state dimension represents an independent feature. DSC [3] uses disentangled factors like position, orientation, linear velocity, rotational velocity, and a Haskey indicator as the state space. For environments with image-based observation, finding EV has been largely studied, like CausalVAE[43] and DEAR [34].

(3) What EVs do we use in our experiment: In our experiments, we utilize part of the observation of the environment as EV. In Eden, we use the state values of the agent and the map as EV (including much useless noise). In 2d-Minecraft, we use the backpack observation as EV (also used by other RL methods for Mineraft [36, 30, 35].

(4) Why we employ EV: One reason is the acquisition of EV is easy in state vector-based observation. Another important reason is described in section 4.1 of the paper. How to discover disentangled representation and how to exploit it are orthogonal and both important. Our paper focuses on how to leverage causality to discover a high-quality subgoal hierarchy upon disentangled EV. So we employ an oracle function. Besides, to ensure fairness and show the effect of causality-driven discovery, we run all baselines with EV to compare.

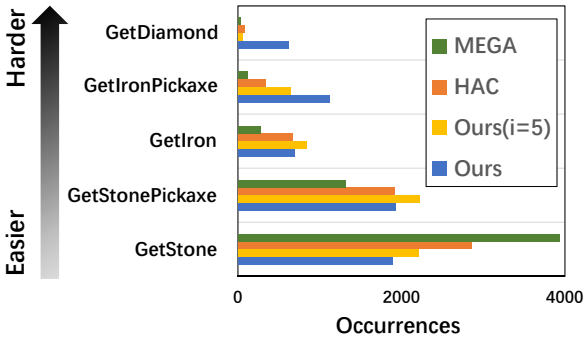
A.2 Sensitivity analysis on Environment Variables (EV)

We want to explain that CDHRL does not necessarily build the complete causality graph on a perfect EV set to take effect. We consider the following three cases:

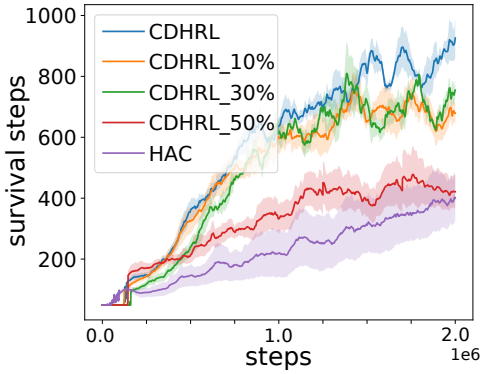
(1) Incomplete causality graph: As long as some causality is discovered, the exploration efficiency would be improved. $i = 5$ in Figure 5(a) means that CDHRL has iterated five cycles. The bars of $i = 5$ shows that CDHRL can already explore hard-to-reach subgoals more efficiently than MEGA and HAC even though the causality graph has not been converged.

(2) Existing noisy environment variables: We have already included noisy EVs in the experiment setting, like unavailable materials and tools in 2D-Minecraft, and state values that irrelevant to survival in Eden. Our methodology filters the noisy variables during causality graph generation because they are uncontrollable environment variables for the agent.

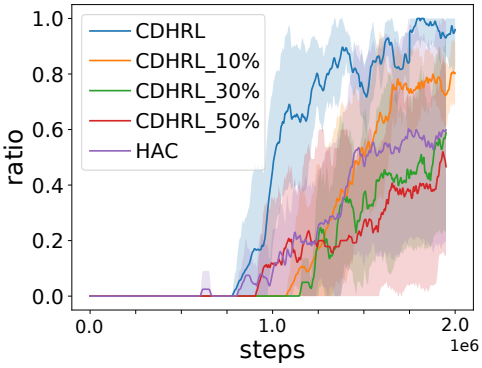
(3) Missing effective environment variables: Missing a few effective EVs (10%) does not significantly hurt the methodology. Figure 5(b) and Figure 5(c) show the performance of CDHRL under different missing ratios of environment variables. We can see CDHRL still has competitive performance compared with HAC even if missing some effective variables. This is because CDHRL can discover $A \rightarrow C$ instead $A \rightarrow B \rightarrow C$ if B is missing, and thus can guide to discover A's subgoals before C's to promote efficiency. As demonstrated in section 6.3, even though the discovered causality lost some variables and there are some long causality instead of true short causality, CDHRL still significantly outperforms existing methodologies, which also shows that CDHRL effectiveness is not essentially sensitive to EVs.



(a) exploration in 2D-Minecraft



(b) Eden



(c) 2D-Minecraft

Figure 5: Sensitivity analysis on environment variables. (a): Exploration capability comparison. (b) and (c): Performance of CDHRL under different missing ratio of EVs

B Causality-driven Hierarchical Reinforcement Learning (CDHRL) Details

B.1 CDHRL framework

The detail processing flow of the CDHRL framework is as described in Algorithm 1. As lines 6-12 shown, after causality discovery, we first select new effect variables, whose cause variables have been in the controllable intervention variables set S_{IV}, as candidate controllable variables S_{CC}. Then, we train subgoals of the candidate controllable variables. Furthermore, the subgoal training success ratios are compared with the pre-defined threshold ϕ_{causal} to select successfully trained subgoals. Finally, we add new controllable variables S_C that with successfully trained subgoals to the intervention variables set S_{IV} before the next round of intervention sampling and causality discovery.

B.2 Causality Discovery

We adopt the classic SCM-based casual discovery methodology to learn the causality between environment variables. As show in the structure equation,

$$X_i := f_i(X_{pa(i,C)}, N_i), \quad \forall i \in \{0, \ldots, M-1\}, \tag{7}$$

SCM over M variables consists of two sets of parameters: **structural parameters** $\eta \in R^{M \times M}$ models the causality graph C and **functional parameters** θ model the generating functions f_i. The matrix $\eta \in R^{M \times M}$ is the soft adjacency matrix of the directed acyclic causality graph. $\sigma(\eta_{ij})$ represents the probability that X_j is a direct cause of X_i, where $\sigma(x) = 1/(1 + \exp(-x))$. By Bernoulli sampling $Ber(\sigma(\eta))$, we can draw a hypothesis configuration C of true causality graph. θ_i are the parameters of X_i's conditional probability function f_i given X_i's parent variable set $X_{pa(i,C)}$.

In the implementation, the soft adjacent matrix η in SCM is modeled by a $M \times M$ tensor. Each variable's generating function f_{θ_i} is modeled by a 3-layer MLP network. For variable X_i's generating

Algorithm 1 CDHRL

Parameter Threshold ϕ_{causal}

initial SCM's structure parameters η, functional parameters θ; subgoal-based policies π_h
Initial the intervention variables set $S_{IV} = \{V_{action}\}$
while $True$ **do**
  Intervention data $D_I = InterventionSampling(\pi_h, S_{IV})$
  Causality graph $C = CausalityDiscovery(\eta, \theta, D_I, S_{IV})$
  Candidate controllable variables set $S_{CC} = \{V_i \mid V_i \notin S_{IV} \ and \ V_{pa(i,C)} \subset S_{IV}\}$
  if S_{CC} is empty **then**
    Break
  else
    Controllable variable set $S_C = SubgoalTraining(\pi_h, C, S_{CC}, \phi_{causal})$
    $S_{IV} = S_{IV} + S_C$
  end if
end while
initial upper policy π_t over subgoal-based policies π_h
train π_t maximizing the calculated extrinsic reward

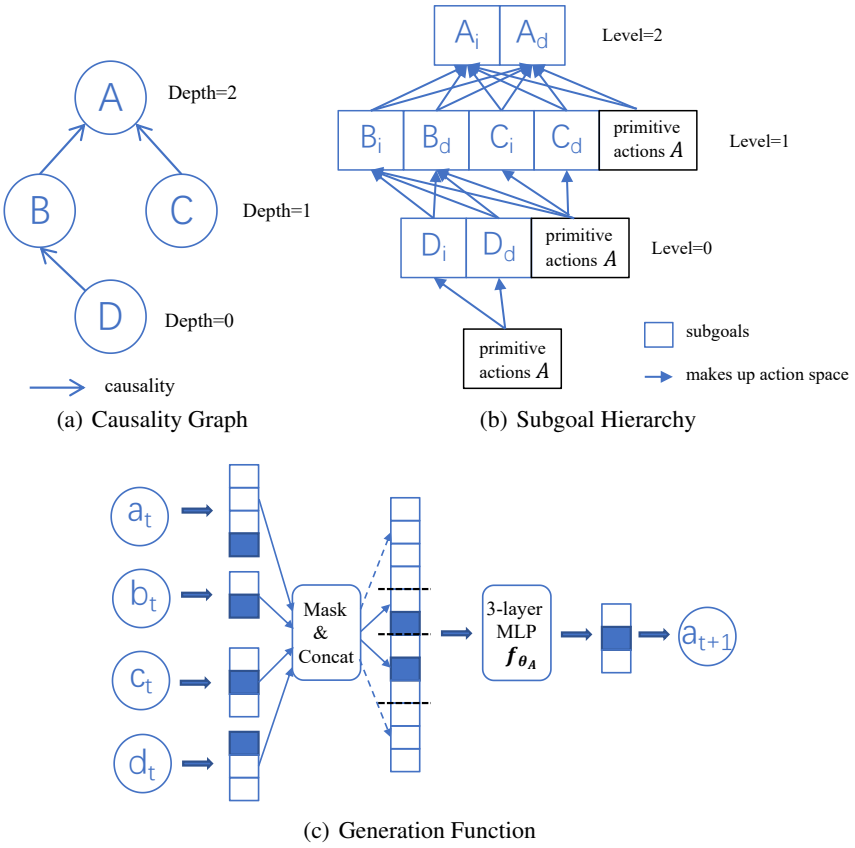


Figure 6: An implementation example. (a): A causality graph with four variables A, B, C, D. (b): Implementation of the subgoal hierarchy based on the causality graph in (a). The arrows pointing to the subgoal indicate the subgoal's action space. For example, subgoal B_i's action space consists of B's parent variable D's subgoals and primitive action space. (c): Implementation of variable A's generation function f_{θ_A}.

Algorithm 2 CausalityDiscovery

Parameter Structural parameters $\eta \in R^{M \times M}$; Functional parameters θ; Intervention data set D_I; Intervention variable set S_{IV};

1: **while** T times iteration **do**
2: **while** $f_{idx} < Fs$ **do**
3: **for** variable $i = 0 \rightarrow$Size(S_{IV}) **do**
4: $X \sim D_I[i]$
5: $C \sim Ber((\sigma(\eta)))$
6: $L = -\log P(X|C;\theta_i)$
7: $\theta_i \leftarrow Adam(\theta_i, \nabla_{\theta_i} L)$
8: **end for**
9: $f_{idx} = f_{idx} + 1$
10: **end while**
11: **while** $q_{idx} < Qs$ **do**
12: **for** variable $j = 0 \rightarrow$Size(S_{IV}) **do**
13: **while** $k < K$ **do**
14: $X \sim D_I[j]$
15: $C^{(k)} \sim Ber((\sigma(\eta)))$
16: $L_C^{(k)} = -\log P(X|C;\theta)$
17: $k = k + 1$
18: **end while**
19: $g_{:,j} = \sum_k (\sigma(\eta_{:,j}) - c_{:j}^{(k)}) \left(\frac{L_C^{(k)}}{\sum_k L_C^{(k)}} \right)$
20: $\eta_j \leftarrow \eta_j + g_j$
21: **end for**
22: $q_{idx} = q_{idx} + 1$
23: **end while**
24: **end while**
25: **return** $C = \{\mathbf{1}_{\sigma(\eta_{ij})>0.8}\}$

function f_{θ_i}, all values of variables $X_{*,t}$ are transformed to one-hot vectors and then concatenated as the input of the network. But all variables' values except X_i's cause variables $X_{pa(i,C),t}$ are masked when computing. The output of f_{θ_i} activated by a softmax layer are used to model the discrete distribution of $X_{i,t+1}$. Figure 6(c) shows the generation function f_{θ_A} of variable A under the causality graph C in figure 6(a). When computing A's distribution, variable values except for A's parents B, C are masked as zero.

Mathematically, we can optimize parameters η and θ to learn the causality between variables. We adopt a two-phase iterative and continuous optimization method similarly to SDI [19] to alternately learn the η and θ parameters. The pseudo-code of the optimization process is as described in Algorithm 2. Fs is function parameters θ's training times in one iteration, Qs is structural parameters η's training times in one iteration, and K is the sampling times to estimate the gradient of η.

In the first phase, called function learning, θ can be optimized by fixing η and maximizing the likelihood of the collected data. Specifically, we first collect intervention data of different variables. Then we sample the hypothesis configuration C of the causality graph from $\sigma(\eta)$ to control the input of the generation function f. Finally, we maximize the likelihood of the data under the configuration C to optimize θ (see lines 4-7 in algorithm 2).

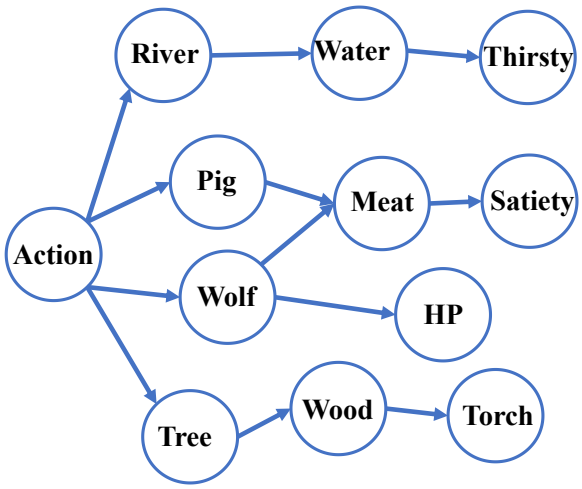
In the second phase, called structure learning, we estimate the gradient of η through a REINFORCE-like predictor proposed by Bengio et al. [5]. For the causality with the variable X_j as the cause, the gradient g_{ij} of η_{ij} is estimated from its intervention data X by the following formula:

$$g_{ij} = \sum_k (\sigma(\eta_{ij}) - c_{ij}^{(k)}) \left(\frac{L_{C,i}^{(k)}(X)}{\sum_k L_{C,i}^{(k)}(X)} \right), \forall i \in \{0, \ldots, M-1\}, \tag{8}$$

where k represents the k-th draw of hypothesis configuration C under the current η. $L_{C,i}^{(k)}(X)$ represents the X_i's likelihood of the intervention data based on $C^{(k)}$ (see lines 13-20 in algorithm 2).

The two optimization phase cycle until convergence. In practice, we train T iterations and return the causalities whose estimated probability exceeds 80% (see line 25 in algorithm 2).

The causality graph discovered in 2d-Minecraft has been shown in the results section, that discovered in Eden is shown in the figure 7(a).



(a) Causality Graph discovered in Eden

B.3 Subgoal Training

The pseudo-code of the subgoals training process is as described in Algorithm 3. The $k = MaxDepth(c)$ in line 3 shows that the max depth in the causality graph C is k. As the hierarchy of subgoals is transformed from the causality graph, we should keep that k is also the level number of subgoal-based hierarchical policy π_h. A simple example of the causality graph and the corresponding subgoal hierarchy is shown in Figure 6(a) and 6(b). The max depth of the causality graph and the level number of the subgoal-based hierarchical policy are both equal to three. As lines 3-7 show, when there is new causality, we check its depth in the causality graph first to decide whether to build a new subgoal level. Then we insert the new subgoals to the corresponding levels. Because all subgoal policies of the same level are implemented in one policy neural network, we need to ensure the agent will not forget the old subgoals when training new subgoals. Thus, whenever we train new subgoals in level k, we train all subgoals at level k together (see lines 10-16). After training, we return variables whose corresponding subgoal success ratio exceeds the pre-defined ratio ϕ_{causal} as controllable variables.

B.4 Implementation Parameters

B.4.1 Causality Discovery parameters

- **Function model:** 3-layer MLP, hidden size = 128.
- **Batch:** 256.
- **T:** 50 iterations.
- **Fs:** 1000.
- **Qs:** 100.
- **K:** 25 per cycle.
- **learning rate:** $lr_\theta = 5e-3,\ lr_\eta = 5e-2$

B.5 Subgoal Training Parameters

- **Exploration in DQN:** $\epsilon - greedy$ exploration with $\epsilon = 0.05$.
- **Batch:** 128.

Algorithm 3 SubgoalTraining

Parameter subgoal-based hierarchical policy π_h; causality graph C; Candidate controllable variables set S_{CC}; Verification threshold ϕ_{causal}

Change Function Set $F = \{f_i, f_d\}$
Candidate goals $G_C = \{(X, y) | X \in S_{CC}, y \in F\}$
$k = MaxDepth(C)$
if $k > \pi_h.levels$ **then**
  $BuildNewLevel(k, \pi_h)$
end if
$InsertSubgoal(G_C, \pi_h)$
$k_{min} = MinDepth(G_C)$
$k_{max} = MaxDepth(G_C)$
for $k_{idx} = k_{min}$ **to** k_{max} **do**
  Trainning goals $G_T = \{g | g.depth = k_{idx}\}$
  Trained steps $t = 0$
  while $t < T$ **do**
    Random goal $g = RandomSelect(G_T)$
    Execute g and put $trajectory$ into replay buffer D
    Train π_h using D
    $t = t + Length(trajectory)$
  end while
end for
Controllable variables set $S_C = \{X | SuccessRatio((X, y)) > \phi_{causal}, y \in F\}$
return S_C

- **Goal horizon H:** The max steps to achieve a goal, $H_{eden} = 10$, $H_{2D-Minecraft} = 15$
- **Goal policy Gamma:** $\gamma_{eden} = 0.9$, $\gamma_{2D-Minecraft} = 0.9$
- **Task policy gamma:** $\gamma_{eden} = 0.99$, $\gamma_{2D-Minecraft} = 0.95$
- **learning rate:** $lr = 1e-4$
- **Causal threshold:** $\phi_{causal} = 0.8$
- **Goal trained threshold:** $\phi_{trained} = 0.6$
- **Max goal trained steps:** $T = 10000$

C Environment Details

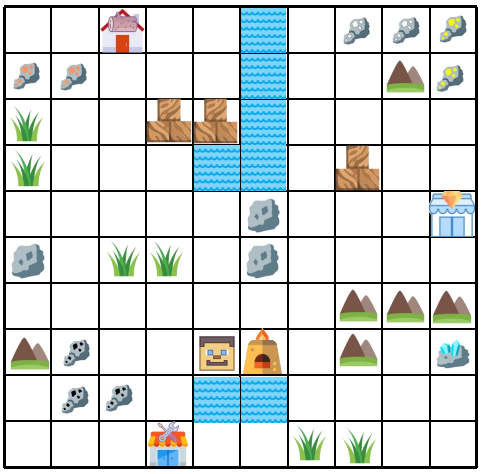
C.1 Descriptions

C.1.1 2D-Minecraft

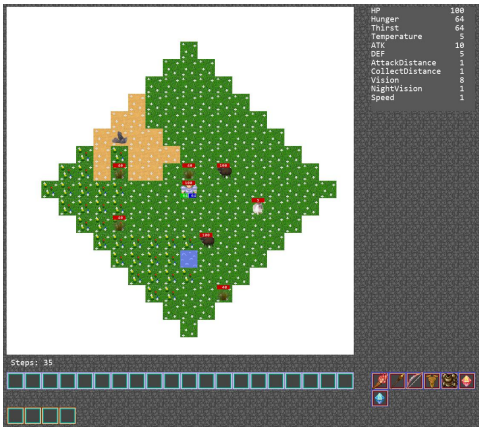
2D-Minecraft [36] is a simplified 2D version of the famous Minecraft [17] with a 10×10 grid map as shown in Figure 7(b). In one episode with limited steps, the agent needs to navigate in the map, pick up various materials, and craft tools in specific positions to obtain advanced materials until getting the diamond. There are 21 kinds of objects and complicated relationships in the environment, as shown in 8(a). The observation consists of the positions of each material and the contents of the backpack. The environment variables are the numbers of different items in the backpack (including has not acquired ones). The actions include moving, picking, and crafting. The extrinsic reward in the experiments is highly sparse since the agent can only get a positive reward when obtaining a diamond during an episode.

C.1.2 Eden

Eden [6] is a survival game which is similar to "Don't Starve" as shown in Figure 7(c) To make a living in a 40×40 grid map, the agent with 8×8 vision range must chase animals to obtain food to maintain satiety, find rivers to get water to prevent being thirsty, and collect materials to craft tools to be survived in the night. The agent can only get a negative reward when its satiety or thirst value



(b) 2D-Minecraft



(c) Eden

Figure 7: Environments

drops to zero and dies. Compared to 2D-Minecraft, the acquisition relationship between items in Eden is relatively simple. However, the observation and action space are more complicated. The observation consists of positions of different resources, various state values about the agent and environment, and contents in the backpack. The actions include flexible moving, attacking, gathering, crafting, discarding, and consuming items. The environment variables include the number of items in the backpack, distances to different resources, and state values such as satiety, thirst, and time.

EV	Ranges
Workspace	2
Furnace	2
Jeweler shop	2
Wood	2
Stone	2
Coal	2
Iron ore	2
Silver ore	2
Gold ore	2
Diamond	2
Stick	2
Stone pickaxe	2
Iron	2
Silver	2
Iron pickaxe	2
Gold	2
Earings	2
Ring	2
Goldware	2
Bracelet	2
Necklace	2
Action	6

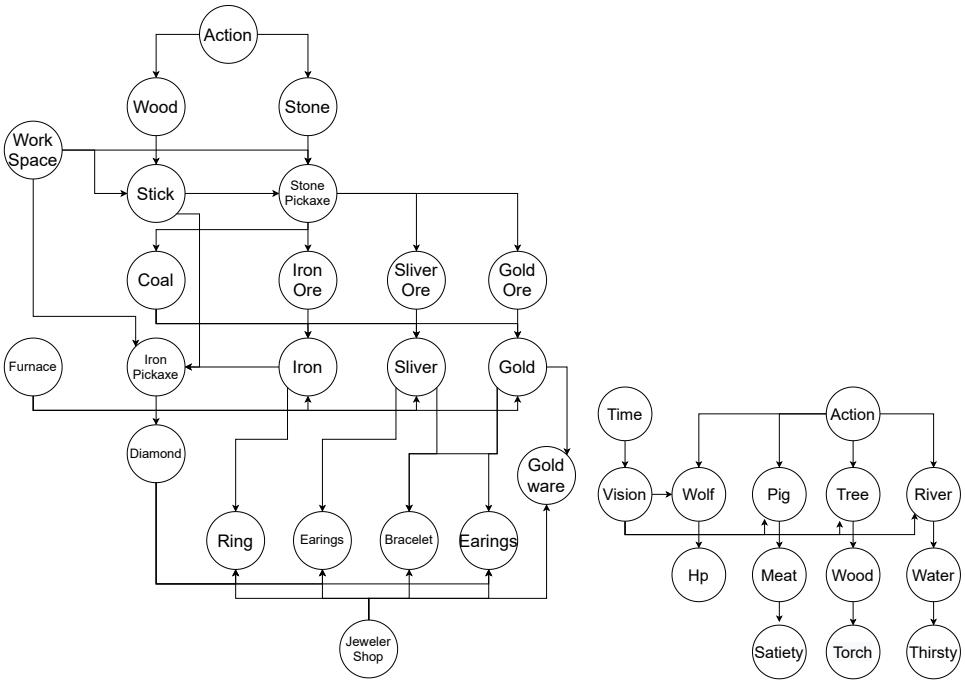
table2D-Minecraft Environment variables

EV	Ranges
Pig	3
River	3
Tree	3
Wolf	3
Meat	5
Water	5
Wood	5
Torch	5
Satiety	5
Thirsty	5
Hp	5
Vision	2
Time	12
Action	11

tableEden Environment variables

C.2 Environment Variables (EV) definitions and Causality Graph

Table C.1.2 and Table C.1.2 shows Environment variables (EV) of the two environments. The ground truth causality graphs are as shown in Figure 8(a) and 8(b).



(a) 2D-Minecraft (b) Eden

Figure 8: Causality graphs