Causality-driven Hierarchical Structure Discovery for Reinforcement Learning – Appendix

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

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 [14], 2d-Minecraft [13], 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 [9, 13, 6, 2, 5] commonly use them as the standard inputs. For example, LESSON [9] assumes that each input state dimension represents an independent feature. DSC [2] 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[15] and DEAR [11].

(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 [13, 10, 12].

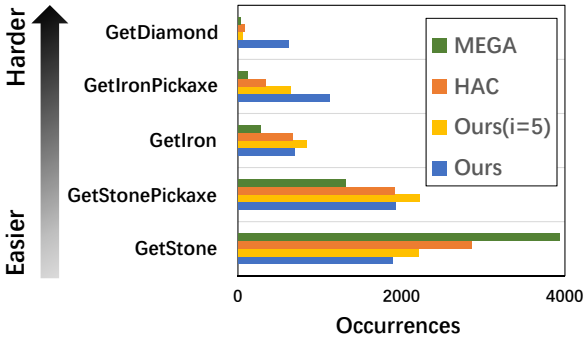
(4) Why we employ EV: One reason is the acquisition of EV is easy in state vector-based observation. Another important reason is described in section 4.1 of the paper. How to discover

† Corresponding author.

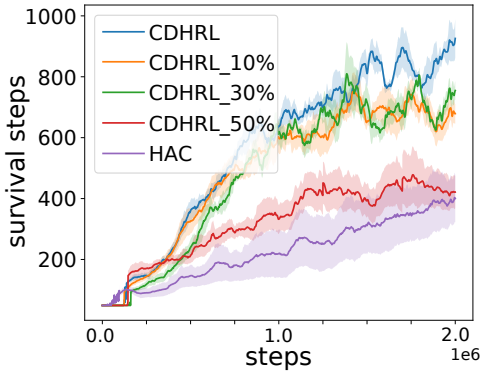
36th Conference on Neural Information Processing Systems (NeurIPS 2022).

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.

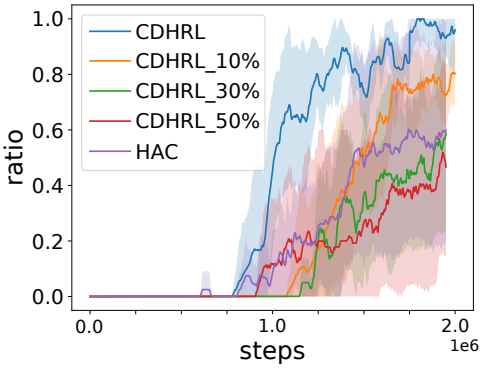
A.2 Sensitivity analysis on Environment Variables (EV)



(a) exploration in 2D-Minecraft



(b) Eden



(c) 2D-Minecraft

Figure 1: Sensitivity analysis on environment variables. (a): Exploration capability comparison. (b) and (c): Performance of CDHRL under different missing ratio of EVs

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 1(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 1(b) and Figure 1(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.

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.

Algorithm 1 CDHRL

Parameter Threshold ϕ_{causal}

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

B.2 Causality Discovery

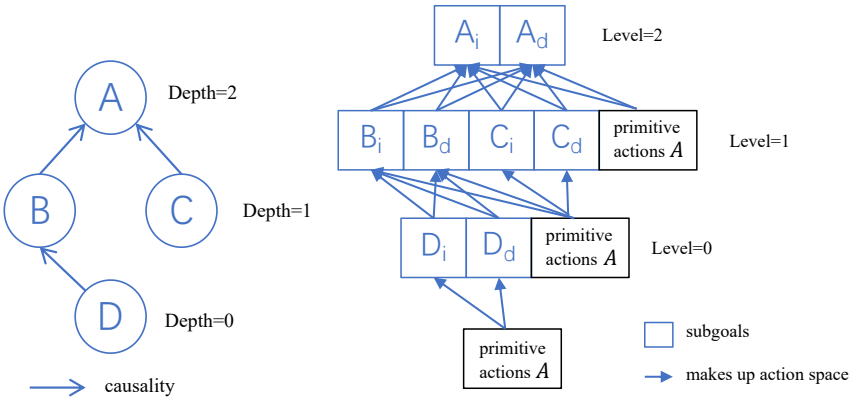
We adopt the classic SCM-based casual discovery methodology to learn the causality between environment variables. As show in the structure equation,

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

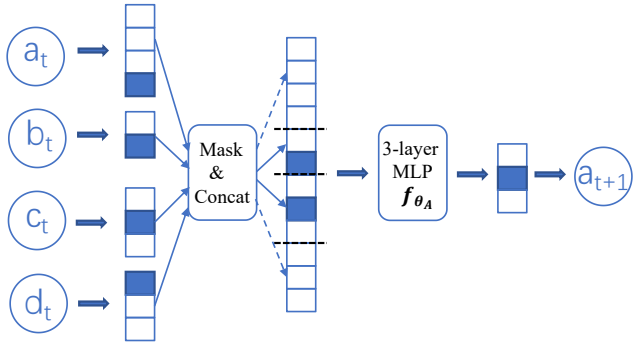
SCM over M variables consists of two sets of parameters: **structural parameters** $\eta \in R^{M \times M}$ models the causality graph C and **functional parameters** θ model the generating functions f_i. The matrix $\eta \in R^{M \times M}$ is the soft adjacency matrix of the directed acyclic causality 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 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 2(c) shows the generation function f_{θ_A} of variable A under the causality graph C in figure 2(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 [8] to alternately



(a) Causality Graph (b) Subgoal Hierarchy



(c) Generation Function

Figure 2: 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}.

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. [3]. 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:

$$g_{ij} = \sum_k (\sigma(\eta_{ij}) - c_{ij}^{(k)}) \left(\frac{L_{C,i}^{(k)}(X)}{\sum_k L_{C,i}^{(k)}(X)} \right), \ \forall i \in \{0, \ldots, M-1\}, \tag{2}$$

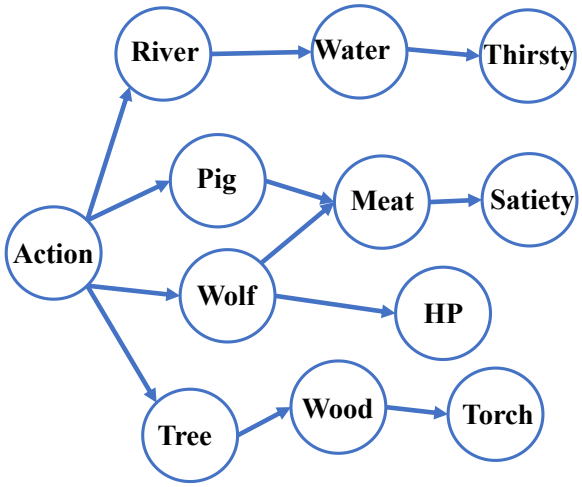
where k represents the k-th draw of hypothesis configuration C under the current η. $L_{C,i}^{(k)}(X)$ represents the X_i's likelihood of the intervention data based on $C^{(k)}$ (see lines 13-20 in algorithm 2). The two optimization phase cycle until convergence. In practice, we train T iterations and return the causalities whose estimated probability exceeds 80% (see line 25 in algorithm 2).

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};

while T times iteration **do**
  while $f_{idx} < Fs$ **do**
    for variable $i = 0$ to $Size(S_{IV})$ **do**
      $X \sim D_I[i]$
      $C \sim Ber((\sigma(\eta)))$
      $L = -\log P(X|C;\theta_i)$
      $\theta_i \leftarrow Adam(\theta_i, \nabla_{\theta_i} L)$
    end for
    $f_{idx} = f_{idx} + 1$
  end while
  while $q_{idx} < Qs$ **do**
    for variable $j = 0$ to $Size(S_{IV})$ **do**
      while $k < K$ **do**
        $X \sim D_I[j]$
        $C^{(k)} \sim Ber((\sigma(\eta)))$
        $L_C^{(k)} = -\log P(X|C;\theta)$
        $k = k + 1$
      end while
      $g_{:,j} = \sum_k (\sigma(\eta_{:,j}) - c_{:j}^{(k)}) \left(\frac{L_C^{(k)}}{\sum_k L_C^{(k)}} \right)$
      $\eta_j \leftarrow \eta_j + g_j$
    end for
    $q_{idx} = q_{idx} + 1$
  end while
end while
return $C = \{\mathbf{1}_{\sigma(\eta_{ij})>0.8}\}$

The causality graph discovered in 2d-Minecraft has been shown in the results section, that discovered in Eden is shown in the figure 3(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 hierar-

chy 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 2(a) and 2(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.

Algorithm 3 SubgoalTraining

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

1: Change Function Set $F = \{f_i, f_d\}$
2: Candidate goals $G_C = \{(X, y) | X \in S_{CC}, y \in F\}$
3: $k = MaxDepth(C)$
4: **if** $k > \pi_h.levels$ **then**
5: $\quad BuildNewLevel(k, \pi_h)$
6: **end if**
7: $InsertSubgoal(G_C, \pi_h)$
8: $k_{min} = MinDepth(G_C)$
9: $k_{max} = MaxDepth(G_C)$
10: **for** $k_{idx} = k_{min}$ **to** k_{max} **do**
11: $\quad$ Trainning goals $G_T = \{g | g.depth = k_{idx}\}$
12: $\quad$ Trained steps $t = 0$
13: $\quad$ **while** $t < T$ **do**
14: $\quad\quad$ Random goal $g = RandomSelect(G_T)$
15: $\quad\quad$ Execute g and put $trajectory$ into replay buffer D
16: $\quad\quad$ Train π_h using D
17: $\quad\quad$ $t = t + Length(trajectory)$
18: $\quad$ **end while**
19: **end for**
20: Controllable variables set $S_C = \{X | SuccessRatio((X, y)) > \phi_{causal}, y \in F\}$
21: **return** S_C

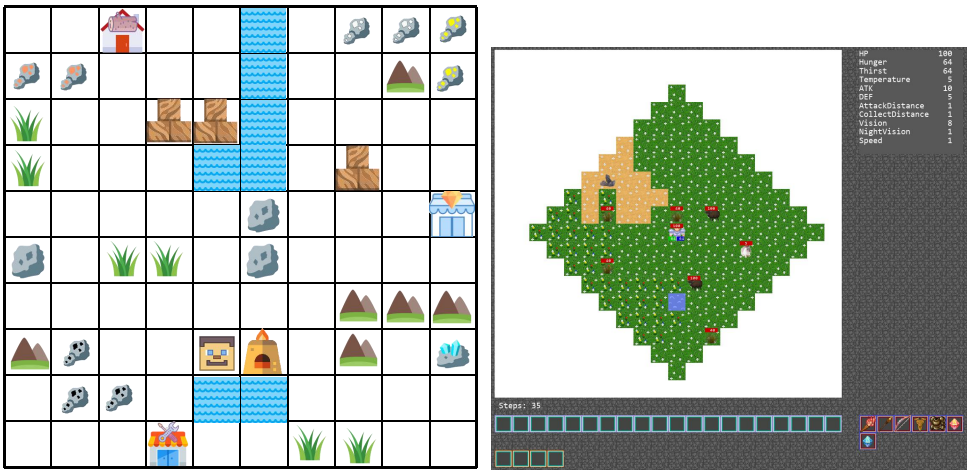
B.4 Implementation Parameters

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.
- **Goal horizon** H**:** The max steps to achieve a goal, $H_{eden} = 10$, $H_{2D-Minecraft} = 15$
- **Goal policy Gamma:** $\gamma_{eden} = 0.9$, $\gamma_{2D-Minecraft} = 0.9$



(b) 2D-Minecraft (c) Eden

Figure 3: Environments

- **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 [13] is a simplified 2D version of the famous Minecraft [7] with a 10×10 grid map as shown in Figure 3(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 4(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

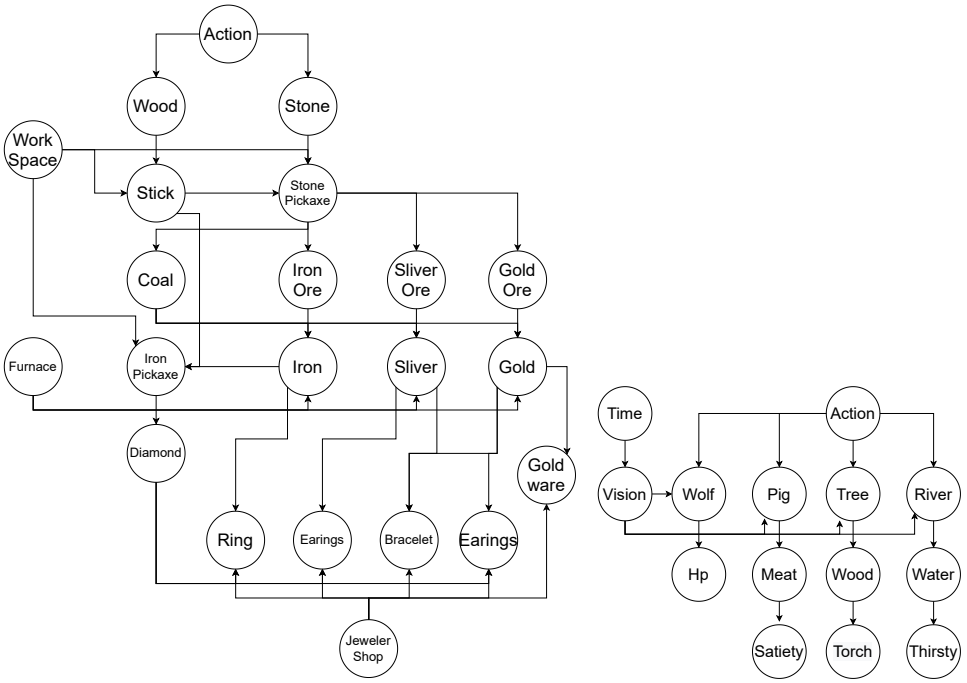
Eden [4] is a survival game which is similar to "Don't Starve" as shown in Figure 3(c) To make a living in a 40×40 grid map, the agent with 8×8 vision range must chase animals to obtain food to maintain satiety, find rivers to get water to prevent being thirsty, and collect materials to craft tools to be survived in the night. The agent can only get a negative reward when its satiety or thirst value 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

Table 1: 2D-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

Table 2: Eden Environment variables



(a) 2D-Minecraft (b) Eden

Figure 4: Causality graphs

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

Table 1 and Table 2 shows Environment variables (EV) of the two environments. The ground truth causality graphs are as shown in Figure 4(a) and 4(b).

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

[3] 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`.

[4] 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.

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

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

[7] 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`.

[8] 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.

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

[10] 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.

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

[12] 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.

[13] 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.

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

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