Supplementary Material

Theory-based Causal Transfer: Integrating Instance-level Induction and Abstract-level Structure Learning

Contents

List of Tables

List of Figures

A Overview

This supplementary document provides additional formulation details, technical details, extra analysis experiments, more quantitative and qualitative test results to the main paper.[1]

Section B provides additional implementation details on our causal theory-based learning scheme; additional ablation results are presented in Section C. Section D details the reinforcement learning (RL) model including architectures and hyper-parameters, and Section E showcases additional details on our RL experimental procedure. These experiments are organized by different Reinforcement Learning algorithms, casual schemas, and reward strategies.

B Causal Theory Induction

In this section, we outline additional details on our model. In particular, we provide additional details on the agent's bottom-up instance-level learning and top-down abstract-level structure learning.

B.1 Instance-level Inductive Learning

Here, we outline additional formulation and implementation details regarding our instance-level learning scheme. This scheme combines a set of attributes with a single action but can be easily extended to include multiple actions or additional dimensions to consider for instance-level learning. This knowledge encodes a naive Bayesian view of causal events by independently examining how frequently attributes and actions were involved in causal events. First, we revisit our formulation:

$$p(\rho|c;\beta) = \prod_{c_i \in c} p(\rho_i|c_i;\beta), \tag{1}$$

where $p(\rho_i|c_i;\beta)$ is computed as

$$p(\rho_i|c_i;\beta) = p(\rho_i|\phi_{i0},\ldots,\phi_{ik},a_i;\beta) \tag{2}$$

$$= \frac{p(\phi_{i0},\ldots,\phi_{ik},a_i|\rho_i;\beta)p(\rho_i;\beta)}{p(\phi_{i0},\ldots,\phi_{ik},a_i;\beta)} \tag{3}$$

$$= \frac{p(\rho_i;\beta)p(a_i|\rho_i;\beta)\prod_{\substack{\phi_{ij}\in s_i \\ s_i\in c_i}} p(\phi_{ij}|\rho_i;\beta)}{p(a_i;\beta)\prod_{\substack{\phi_{ij}\in s_i \\ s_i\in c_i}} p(\phi_{ij};\beta)} \tag{4}$$

$$= \frac{p(\rho_i;\beta)\frac{p(\rho_i|a_i;\beta)p(a_i;\beta)}{p(\rho_i;\beta)}\prod_{\substack{\phi_{ij}\in s_i \\ s_i\in c_i}} \frac{p(\rho_i|\phi_{ij};\beta)p(\phi_{ij};\beta)}{p(\rho_i;\beta)}}{p(a_i;\beta)\prod_{\substack{\phi_{ij}\in s_i \\ s_i\in c_i}} p(\phi_{ij};\beta)} \tag{5}$$

$$= \frac{p(\rho_i|a_i;\beta)\prod_{\substack{\phi_{ij}\in s_i \\ s_i\in c_i}} p(\rho_i|\phi_{ij};\beta)}{p(\rho_i;\beta)^k} \tag{6}$$

$$\propto p(\rho_i|a_i;\beta)\prod_{\substack{\phi_{ij}\in s_i \\ s_i\in c_i}} p(\rho_i|\phi_{ij};\beta), \tag{7}$$

where k is the number of attributes of the state node s_i in c_i. We assume $p(\rho_i;\beta)$ is uniform. Note that this derivative is effectively a Naive Bayes approximation of the true joint distribution, $p(\rho_i|\phi_{i0},\ldots,\phi_{ik},a_i;\beta)$.

[1] A video demonstration of the environment and model execution can be found in the submitted supplementary materials or at `http://138.68.224.173/aaai20/video_demo.mp4`

B.2 Abstract-level Structure Learning

In this section, we provide additional details about our abstract schema learning scheme. We will begin with the belief in abstract schemas, defined as:

$$p(g^A;\gamma) = \sum_{g^M\in\Omega_{g^M}} p(g^A|g^M)p(g^M;\gamma), \tag{8}$$

where $p(g^M;\gamma)$ is the prior over atomic schemas, whose parameters are provided by the atomic schema Dirichlet distribution. The $p(g^A|g^M)$ is computed as a exponential distribution:

$$p(g^A|g^M) = \frac{1}{Z}\exp(-D(g^A,g^M)), \tag{9}$$

where $D(g^A,g^M)$ is the graph edit distance between the abstract schema g^A and the atomic schema g^M, and Z is the normalizing constant, $Z = \sum_{g^A\in\Omega_{g^A}}\exp(-D(g^A,g^M))$.

Next, we compute the belief in an instantiated schema as:

$$p(g^I|do(q);\gamma) = \sum_{g^A\in\Omega_{g^A}} p(g^I|g^A,do(q))p(g^A;\gamma), \tag{10}$$

where $p(g^I|g^A,do(q))$ is computed as a uniform distribution among all g^I that have $D(g^I,g^A)=0$ (ignoring vertex labels) *and* contain the solutions found thus far q, and 0 elsewhere. Next, the belief in a chain is computed as:

$$p(c|do(q);\gamma) = \sum_{g^I\in\Omega_{g^I}} p(c|g^I,do(q))p(g^I|do(q);\gamma). \tag{11}$$

Similarly, $p(c|g^I,do(q))$ is uniform across all $c\in g^I$ and 0 elsewhere. Finally, we compute the belief in each possible subchain as:

$$p(c_i|do(r,q);\gamma) = \sum_{c\in\Omega_C} p(c_i|c,do(\tau,q))p(c|do(q);\gamma), \tag{12}$$

where $p(c_i|c,do(\tau,q))$ is uniform across all $c_i\in c$ and 0 elsewhere.

C Ablation Results

In this section, we present additional results from our proposed method. Specifically, we show how well the model performs under two ablations: (i) top-down structure learning and (ii) bottom-up instance learning. This examination seeks to identify to what degree and how well much each model component contributes to the model's performance. In our formulation, these ablations amount to setting a probability of 1 for the ablated component in the subchain posterior; *i.e.*, the subchain posterior reduces to the remaining active model component (bottom-up during a top-down ablation and top-down during a bottom-up ablation).

Figure 1 shows the results of the ablated model. In Figure 1a and Figure 1b, the model is ablated to disable the top-down abstract structure learning. We see the agent performing with similar trends as the full model results, but with worse performance. This is due to the agent learning the bottom-up associative theory regarding which instances can be manipulated to produce a causal effect, but the agent performs worse

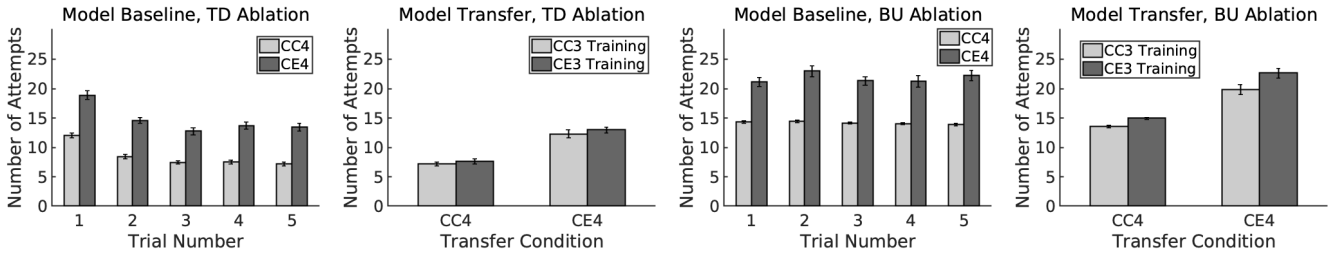

Figure 1: Results using the proposed theory-based causal transfer under ablations. (a) Proposed model baseline results under a top-down ablation (*i.e.*, only instance-level learning occurred). (b) Proposed model transfer results under a top-down ablation. (c) Proposed model baseline results under a bottom-up ablation (*i.e.*, only abstract-level structure learning occurred). (b) Proposed model transfer results under a bottom-up ablation.

due to the lack of task structure. During transfer, we see little difference (with no significance; $t(79) = 0.8; p = 0.42$ and $t(79) = 0.8; p = 0.43$ for Common Cause 4 (CC4) and Common Effect 4 (CE4) respectively) between the training groups. This is expected; an agent that learns no task structure should exhibit no difference between tasks. This agent is essentially aimlessly searching the structure space, biased towards *any* structure with subchains with a high likelihood of producing a causal event.

Figure 1c and Figure 1d show the model ablated with the bottom-up instance learning disabled. In the baseline results, we see a slight increase in performance over time for CC4; this is because the agent is becoming more confident in which structure governs the environment. However, this version of the model has no regard for whether or not an agent can interact with a particular instance (*i.e.*, it lacks the bottom-up associative theory regarding causal events). Because of this limitation, the agent must try many possible instantiations of the correct abstract structure before finding a solution. During transfer, we see the agent benefiting most from training in Common Cause 3 (CC3), which is counter-intuitive for the CE4 transfer condition.

However, we believe this is best explained from a decision tree perspective, as elaborated in the main text. Throughout all model and human experiments, we observed that Common Effect (CE) was more difficult than Common Cause (CC). From a decision tree perspective, agents that learn a CC structure will first identify the first lever in the structure; this is the only lever they can interact with initially. After identifying this lever, they can then push on either remaining lever to unlock the door. While this strategy will not work for CE directly, it may still benefit an agent only equipped with structure learning. For instance, when applying this strategy to CE, the agent may find the *first* solution faster. After finding the first solution, the space of second solutions is constrained to contain the first solution. From here, despite having learned the "wrong" structure for this task, the agent may find both remaining solutions faster. This is an unexpected phenomenon and will be examined as future work.

D Details of Reinforcement Learning Model

In the section, we will detail the OpenLock experiment for RL agents, hyper-parameters, and training procedures used for our experiments. These hyper-parameters are selected via a grid search.

D.1 Overview

In RL experiments, we want to answer:

1. Can predominate, state-of-the-art model-free RL algorithms solve the OpenLock task?
2. What transferable representations, if any, do these RL agents establish?

Notice our task definition requires agents to find *all* solutions in a trial. This requirement means that an agent that *memorizes* and biases to specific one solution will be scored badly under our evaluation method. Agents must form abstract transferable notions of the task or must memorize all possible settings of the task.

To answer the first question, we show the performances of typical RL algorithms. We try to improve their performances by providing several reward strategies. The details of algorithms, tasks, and rewards we used can be found in Section D.3.

To answer the second question, if the agents are able to establish such concepts, they can master the task with similar casual schema both better and faster than training on that task from scratch; *i.e.*, we expect to see a positive transfer. In this experiment, all the agents are first trained on 3-lever tasks, then we transfer these agents to target 4-lever tasks using fine-tuning. By comparing the results in our transfer experiments with directly training on target tasks (*i.e.*, baseline experiments), we can verify whether the agents are able to build such abstract casual concepts.

D.2 State and Action Spaces in OpenLock

In this section, we outline some specifications to OpenLock environment. Readers are encouraged to examine (Edmonds et al. 2018) for additional details.

- **State Space:** The state space consists of 16 binary dimensions: 7 for the state of each lever (*pushed* or *pulled*), 7 dimensions for the color of each lock (*grey* or *white*), 1 dimension for the state of the lock (*locked* or *unlocked*), and 1 dimension for the state of the door (*closed* or *open*).
- **Action Space:** The action space is a discrete space with 15 dimensions: each of the 7 levers has 2 actions (*push* and *pull*), and the door has one action (*push*).

D.3 Algorithms, Casual Schemas and Rewards

We select a set of predominate, state-of-the-art RL algorithms as baselines, including Deep Q-Network (DQN) (Mnih et al. 2015), DQN with prioritized experience replay (DQN-PE) (Schaul et al. 2016), Advantage Actor-Critic (A2C) (Mnih et al. 2016), Trust Region Policy Optimization (TRPO) (Schulman et al. 2015), Proximal Policy Optimization (PPO) (Schulman et al. 2017) and Model-Agnostic Meta-Learning (Finn, Abbeel, and Levine 2017). Table 1 lists all the baselines we considered. These algorithms have been applied to solve a variety of tasks including Atari Games (Mnih et al. 2015), classic control, and even complex visual-motor skills (Levine et al. 2018), and they have shown remarkable performance on these tasks when large amounts of simulated or real-world exploration data are available.

Additionally, we also include a strong baseline of Model-agnostic Meta Learning (MAML) (Finn, Abbeel, and Levine 2017). Note that the MAML does not employ a standard transfer learning setting as it requires to access the target task during the meta-learning phase, which can be more advantageous than other transfer methods. Our main goal is to verify whether the state-of-the-art meta-learning algorithm (*i.e*., MAML) can solve the OpenLock task by forming the correct causal abstraction of the task.

However, these algorithms cannot fully succeed in OpenLock even after exploring during a large number of episodes. In the experiments, we use four casual schemas: CC3, Common Effect 3 (CE3), CC4, CE4. We have two experimental settings: (i) baseline, where agents are trained in a 4-lever condition (*i.e*. CC4 or CE4) and (ii) transfer, where agents are trained in a 3-lever condition (*i.e*. CC3 or CE3) and then transfer to a 4-lever condition (*i.e*. CC4 or CE4).

Now we will discuss the reward strategies we used in baseline experiments. Rewards in the OpenLock environment are very sparse; agents must search in a large space of possible attempts (*i.e*. action sequences) of which there are 2 or 3 action sequences that achieve the task. Sparse rewards have traditionally been a challenge for RL (Sutton and Barto 1998). To overcome this, we enhance the reward by shaping it to provide better feedback for the agent; we introduce task-relevant penalties and bonuses. We utilize 6 reward strategies:

Basic (B) The agent will receive a reward for unlocking the door and will receive the largest reward for opening the door. No other rewards are granted for all other outcomes.

Unique Solution (U) Inherits from Reward B, but the agent only receives a reward when unlocking/opening the door with a new solution. There are a finite (2 for 3-lever trials and 3 for 4-lever trials) number of solutions. This reward is designed to encourage the agent to find all solutions within a trial, instead of only finding/pursuing the first solution found.

Reward B and Negative Immovable (B+N) Inherits from Reward B, but introduces an extra penalty for manipulating an immovable lever (Reward N). This is judged by whether a state change occurs after executing an action; this penalty is designed to encourage the agent to only interact with movable levers.

Reward U and Negative Immovable (U+N) This reward is a combination of Reward U and the Negative Immovable penalty (Reward N) introduced in Reward B+N.

Reward N and Solution Multiplier (N+M) This reward inherits from Reward B, but in this reward setting, we encourage the agent to find out more solutions in a slightly different way from Reward U. Instead of only providing reward when finishing the task with a new solution, the agent will receive a reward every time it unlocks/opens the door, but when the agent finds a unique solution, the reward it receives is multiplied by a fixed factor (> 1). This effectively encourages our agent to find new solutions in a more reward-dense setting. In addition, we also use the Negative Immovable penalty (Reward N) for learning efficiency.

Reward N+M and Partial Sequence (N+M+P) Inherits from Rewards B, N, and M, but adds a Partial Sequence bonus. When the executed action sequence is exactly a prefix of a solution to the current trial (no matter whether this solution has been found out or not), the agent will receive a bonus. This is a form of reward shaping to overcome the sparse reward problem.

D.4 Hyper-parameters and Training Details

Table 2 presents the hyperparameters and training details for our experiments. We select these parameters through several preliminary experiments.

E Additional Results on Reinforcement Learning Experiments

In the section, we will first introduce the RL experiments are organized, then show the results for the baseline, training and transfer experiments, and finally provide some intuitions and analysis.

E.1 Experimental Procedure

Here we describe the complete experimental procedures for RL agents. Each agent is trained for 200 *iterations*. In each iteration, there are 6 *trials* for 3-lever tasks (CC3 and CE3; referred to as the training phase) and 5 for 4-lever tasks (CC4 and CC4; referred to as the testing phase). Agents are allowed to take at most 700 *attempts* to find all of the solutions within a trial. A typical trial proceeds as follows:

1. A new trial starts.
2. Agent is allowed for taking a finite number of attempts to find all solutions. An attempt will start from the initial state of the environment, and end with opening the door or reaching the maximum action limit.
3. A trial ends either when all the solutions have been found or the agent reaches the maximum attempt limit.
4. After finding all solutions or running out of attempts, the agent is placed in the next trial with *different* lever configurations but the same casual schema during the training phase.
5. After completing all trials in the training phase, the agent is placed into a single 4-lever trial for the testing phase.

For baseline experiments (To answer Q1 in Section D.1)	For transfer experiments (To answer Q2 in Section D.1)
DQN on 3-lever task from scratch	Fine-tune DQN on 4-lever task
DQN-PE on 3-lever task from scratch	Fine-tune DQN-PE on 4-lever task
A2C on 3-lever task from scratch	Fine-tune A2C on 4-lever task
TRPO on 3-lever task from scratch	Fine-tune TRPO on 4-lever task
PPO on 3-lever task from scratch	Fine-tune PPO on 4-lever task
MAML (Meta learning with 3 and 4-lever tasks)	MAML (N shot adaption on 4-lever task)

Table 1: Baselines used in our experiments.

Table 2: Hyperparameters and training details.

Parameter	Value
Shared	
Optimizer	Adam
Learning rate	$3e^{-4}$
Discount (γ)	0.99
Architecture of policy and value networks	(128, 128)
Nonlinearity	Tanh
Batch size	2048
L2 regularization	0.001
DQN/DQN-PE	
Size of replay buffer	10000
Epsilon for exploration	0.9
Epsilon decay interval	50
Epsilon decay method	exponential
Epsilon decay ending	0.05
TRPO	
Maximum KL divergence	0.01
Damping	0.01
MAML	
Meta optimizer	TRPO
Others	
Reward multipliers	1, 10, 20 for 1st, 2nd and 3rd solution
Repeated times	10
Total number of experiments	1800

We have 6 configurations in total for 3-lever tasks and 5 for 4-lever tasks. The configuration of the lever is selected in a loop; the initial order of the configurations is randomized per agent, but each agent see the same room ordering for the entire experiment.

We evaluate the final performances after all iterations are finished. The details of the evaluation are discussed in Section E.2.

E.2 Evaluation Details

We expect an agent learning the correct abstractions and generalizations to quickly adapt to similar but slightly different circumstances. More specifically, an agent learning the correct abstractions should perform better (*i.e.* have lower attempts) as the agent encounters more trials with the same causal schema. We propose several criteria to evaluate agents. In our plots (Figure 2-8), we list 3 different curves:

- **Attempt Amount** This curve shows the number of attempts used in each trial. Since a trial terminates when all solutions have been found, an agent with better performance will have fewer attempts per trial. Moreover, the decreasing speed of this curve can also show how quickly the agent mastered finding all solutions.
- **Percentage of Found Solution** This curve shows how many solutions the agent found within a trial, *e.g.*, if the agent found all the 3 solutions (for a 4-lever task), this value will be 1 for this trial. This plot also shows how well

the agent mastered find all solutions.

- **Averaged Trial Reward** This curve shows the averaged reward in a trial (reward sum divided by the number of attempts). Since the reward strategies are varied in our experiments, this value cannot be a direct criterion to compare the performance of various experimental settings.

In Table 5-10, we list the *Averaged Attempt Amount* for all the experiments. This value is averaged by the last 30 trails to show the final performance over all the lever configurations. As mentioned above, the fewer the attempts, the better the performance.

E.3 Baseline Experiments

In baseline experiments, we want to evaluate the agents' performance on a single causal schema. The agent needs to do several trials successively. Among these trials, the causal schema is fixed, while the lever configurations and observational solutions are varied (structurally, the solutions remain the same). The goal in each trial is to find all the solutions using as few attempts as possible. We evaluate all the 5 algorithms (DQN, DQN-PE, A2C, TRPO and PPO) on four casual schemas, and the results are shown in Table 5-8 and Figure 2-5.

In general, 3-lever tasks are easier than 4-lever tasks, because there are more solutions to find in the latter case. Specifically, for rewards that do not encourage finding multiple solutions, such as Reward B and N, it is quite difficult for agents to find all the solutions, and agents are frequently biased to one specific solution. In other words, agents *memorize* a single solution instead of learning the abstract, multi-solution causal schema. As for the reward strategies that encourage finding multiple solutions, Reward U is the best for most of the agents. In addition, for some importance-sampling based policy gradient methods (PPO/TRPO), an extra penalty (Reward N) can slightly improve the stability and final results.

In the Reward N+M and Reward N+M+P strategies, we introduce some reward shaping techniques, including reward multiplier and partial sequence bonus, to mitigate the sparse reward problem. However, the results are worse and more unstable. We posit that this may be caused by the positive reward for non-unique solutions. Although the agents are encouraged to find new solutions using the multiplied reward, nothing prevents agents from being biased towards a specific solution, yielding a sub-optimal policy. To eliminate this, we may need to adjust the learning rate dynamically as solutions are found. Thus selecting hyper-parameters for the last 2 reward strategies is challenging, and the results are difficult to match expectations.

Another interesting result is the performance of value-based methods (DQN, DQN-PE). For all casual schemas and reward strategies, these methods do not perform well under any of our experiments. Since the lever settings vary between trials, it extremely difficult for the agent to build a universal value function based on discrete state-action input (Edmonds et al. 2018). The casual schema remains the same, but the value function learned is not directly based on the abstract casual state. The RL agents examined do not appear able to construct a representation capable of inferring the connection between the explicit discrete state and the abstract casual state.

E.4 Transfer Experiments

In transfer experiments, we first train our agents in a 3-lever task and them to a 4-lever task. We perform quantitative evaluations on the target 4-lever task for all the transferred models. Additionally, we also compare them with the models that trained on a 4-lever task from scratch (*i.e.*; baseline experiments). If the agents form useful abstract structural representations of tasks, we expect them to complete the 4-lever task faster than training from scratch. All 5 algorithms and 6 reward strategies are considered. The results are listed in Table 9-10 and Figure 6-8.

Reward strategies that were not effective in baseline experiments were also not effective in transfer experiments, as expected. Baseline experiments showed that policy-based methods (A2C, PPO, TRPO) with explicit encouragement to multi-solution performed better; these agents mastered most of the solutions (Table 7-8). As we mentioned above, if an agent is able to establish a concept to the corresponding casual schema, it should have comparable transfer performance regarding the performance of agent training on a 4-lever task from scratch, and it is also expected to converge faster. To verify this, we can make a comparisons between Table 7-8 and Table 9-10. However, for both CC4 and CE4 casual schemas, there is a significant gap between transfer performance and training performance. Even under the most effective reward strategies (Reward U, Reward U+N, and Reward N+M), the agents find it hard to match the corresponding training performance, indicating negative transfer.

E.5 Empirical results of MAML

Here we separately present the empirical results of MAML since it is a meta-learning approach that does not comes from the same category as other transfer learning methods (see Table 1). We conduct experiments on MAML with only the reward strategy of *unique solutions* (**Reward U**) as this strategy overall provides the best performances. All the numerical results are presented in Table 3 and Table 4 for CC4 and CE4 scenarios respectively, while the learning curves can be found in Table 8.

As the meta optimizer we use in MAML is TRPO (Schulman et al. 2015), we compare the adaption results with TRPO in transfer experiments on CC4/CE4, which can be found in the second row of Figure 6 and Figure 7. The results indicate that during few-shot adaption phase, MAML overall outperforms than fine-tuning policy previously learned on 3-lever task with TRPO, which demonstrates that the transferring, or adaption do benefit from meta-learning from both the 3 and 4-lever tasks. However, when comparing with the oracle baseline results that directly training on 4-lever tasks (see the second row of Figure 4 and Figure 5), there is still a significant performance gap, which indicates that the MAML agent cannot master the target tasks well. Namely, being similar as all the fine-tuning methods, meta-learning on the previous task with same causal schema can improve neither the performances of subsequent policy learning on target task nor the convergence properties but misleads the policy learning even

with similar causal schema. This demonstrates that the state-of-the-art meta-learning approach also may not be able to establish a useful concept toward the causal schemas among the tasks it encounters during the meta-learning phase.

References

[Edmonds et al. 2018] Edmonds, M.; Kubricht, J.; Summers, C.; Zhu, Y.; Rothrock, B.; Zhu, S.-C.; and Lu, H. 2018. Human causal transfer: Challenges for deep reinforcement learning. In *40th Annual Meeting of the Cognitive Science Society (CogSci)*.

[Finn, Abbeel, and Levine 2017] Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*.

[Levine et al. 2018] Levine, S.; Pastor, P.; Krizhevsky, A.; Ibarz, J.; and Quillen, D. 2018. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research (IJRR)* 37(4-5):421–436.

[Mnih et al. 2015] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

[Mnih et al. 2016] Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning (ICML)*, 1928–1937.

[Schaul et al. 2016] Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2016. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*.

[Schulman et al. 2015] Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, 1889–1897.

[Schulman et al. 2017] Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[Sutton and Barto 1998] Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*. MIT press.

Table 3: Summary of RL results. Averaged Attempt Amount (Averaged by last 30 trials) for CC4.

	DQN	DQN-PE	A2C	TRPO	PPO	MAML
CC4, baseline	696.47	690.93	27.23	**6.59**	8.75	610.8
CC4, transfer	698.52	699.92	**30.27**	658.45	77.46	510.8

Table 4: Summary of RL results. Averaged Attempt Amount (Averaged by last 30 trials) for CE4.

	DQN	DQN-PE	A2C	TRPO	PPO	MAML
CE4, baseline	697.64	700.00	**30.05**	36.15	121.14	363.9
CE4, transfer	694.01	697.56	657.16	**510.60**	700.00	401.9

Table 5: Averaged Attempt Amount (Averaged by last 30 trials) for CC3.

	DQN	DQN-PE	A2C	TRPO	PPO
Reward B	687.80	693.35	**538.01**	674.94	690.87
Reward U	691.59	686.67	12.36	**3.79**	4.54
Reward B+N	674.93	692.74	**610.29**	691.33	700.00
Reward U+N	673.71	691.97	12.93	**3.48**	4.05
Reward N+M	669.58	692.86	69.39	**5.61**	283.34
Reward N+M+P	673.13	684.25	**19.47**	40.38	422.75

Table 6: Averaged Attempt Amount (Averaged by last 30 trials) for CE3.

	DQN	DQN-PE	A2C	TRPO	PPO
Reward B	689.24	686.97	568.61	666.95	**560.89**
Reward U	676.01	685.63	14.97	**3.56**	4.23
Reward B+N	684.61	680.53	**489.33**	660.83	630.46
Reward U+N	684.05	684.09	13.97	**3.56**	11.00
Reward N+M	694.21	684.50	14.45	**3.68**	143.81
Reward N+M+P	691.77	691.62	15.60	**3.71**	560.75

Table 7: Averaged Attempt Amount (Averaged by last 30 trials) for CC4.

	DQN	DQN-PE	A2C	TRPO	PPO
Reward B	700.00	**699.86**	700.00	700.00	700.00
Reward U	696.47	690.93	27.23	**6.59**	8.75
Reward B+N	**672.02**	699.38	700.00	700.00	700.00
Reward U+N	688.61	700.00	67.69	283.51	**7.09**
Reward N+M	692.19	700.00	326.39	**6.43**	562.75
Reward N+M+P	686.15	697.21	490.77	**173.35**	589.14

Table 8: Averaged Attempt Amount (Averaged by last 30 trials) for CE4.

	DQN	DQN-PE	A2C	TRPO	PPO
Reward B	698.44	699.14	700.00	**667.14**	700.00
Reward U	697.64	700.00	**30.05**	36.15	121.14
Reward B+N	700.00	698.72	700.00	**679.21**	700.00
Reward U+N	700.00	693.59	**35.25**	247.35	415.17
Reward N+M	700.00	693.61	**45.39**	81.79	267.13
Reward N+M+P	698.47	691.17	**34.06**	64.90	367.52

Table 9: Averaged Attempt Amount (Averaged by last 30 trials) for CC3-CC4.

	DQN	DQN-PE	A2C	TRPO	PPO
Reward B	**698.61**	699.12	700.00	700.00	700.00
Reward U	699.87	699.85	**24.68**	644.59	85.17
Reward B+N	**686.29**	698.27	700.00	700.00	700.00
Reward U+N	684.68	700.00	60.99	700.00	**7.32**
Reward N+M	674.48	700.00	**372.51**	700.00	683.41
Reward N+M+P	686.89	698.09	**59.55**	700.00	576.51

Table 10: Averaged Attempt Amount (Averaged by last 30 trials) for CE3-CE4.

	DQN	DQN-PE	A2C	TRPO	PPO
Reward B	**687.13**	700.00	700.00	700.00	700.00
Reward U	693.39	699.67	616.62	**483.27**	700.00
Reward B+N	**698.43**	698.86	700.00	700.00	700.00
Reward U+N	700.00	694.74	**156.80**	700.00	638.09
Reward N+M	698.55	700.00	**326.29**	700.00	700.00
Reward N+M+P	693.49	700.00	**510.08**	700.00	700.00

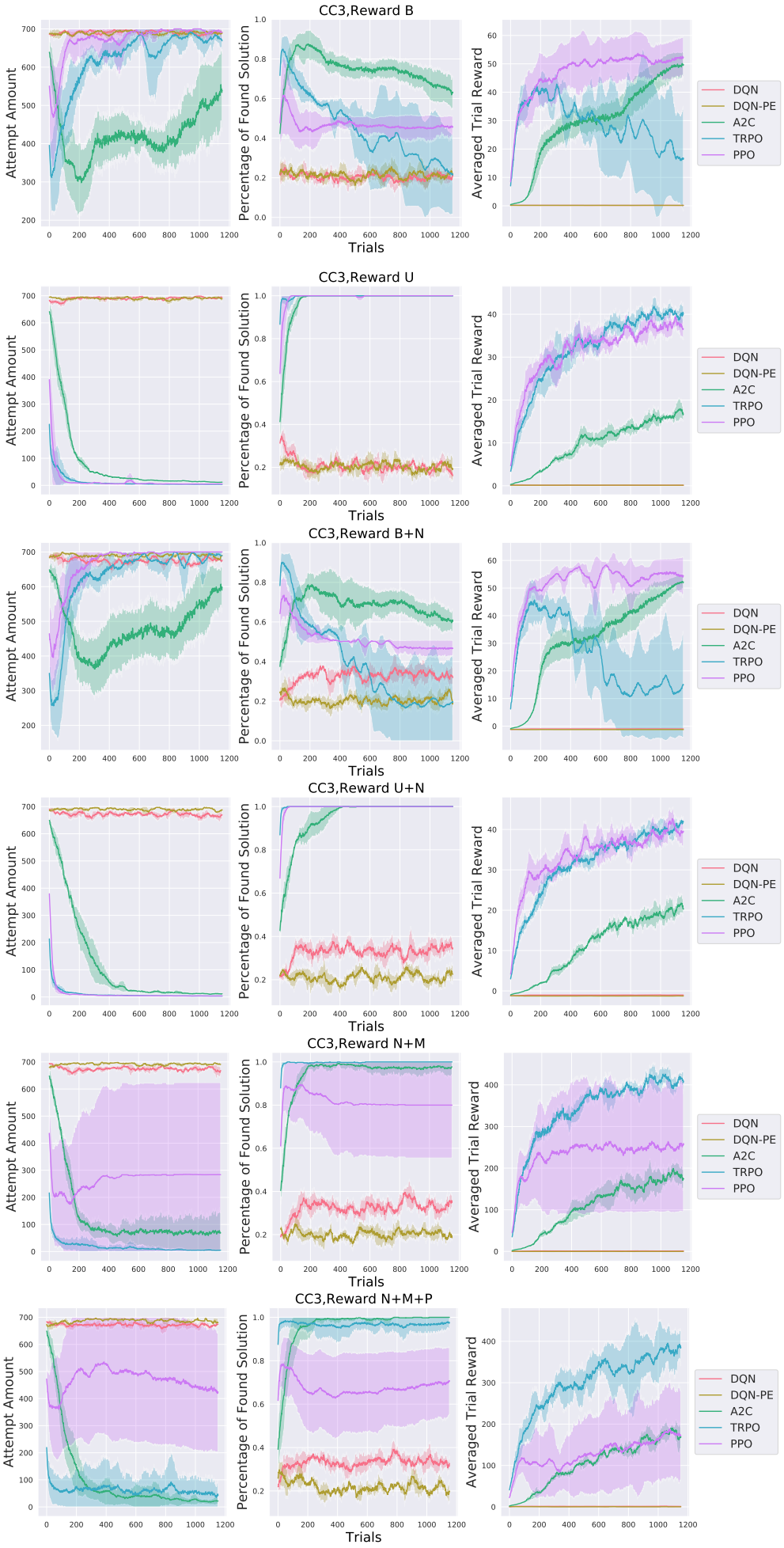

Figure 2: Results of different reward strategies on Common-Cause in 3-lever task (CC3).

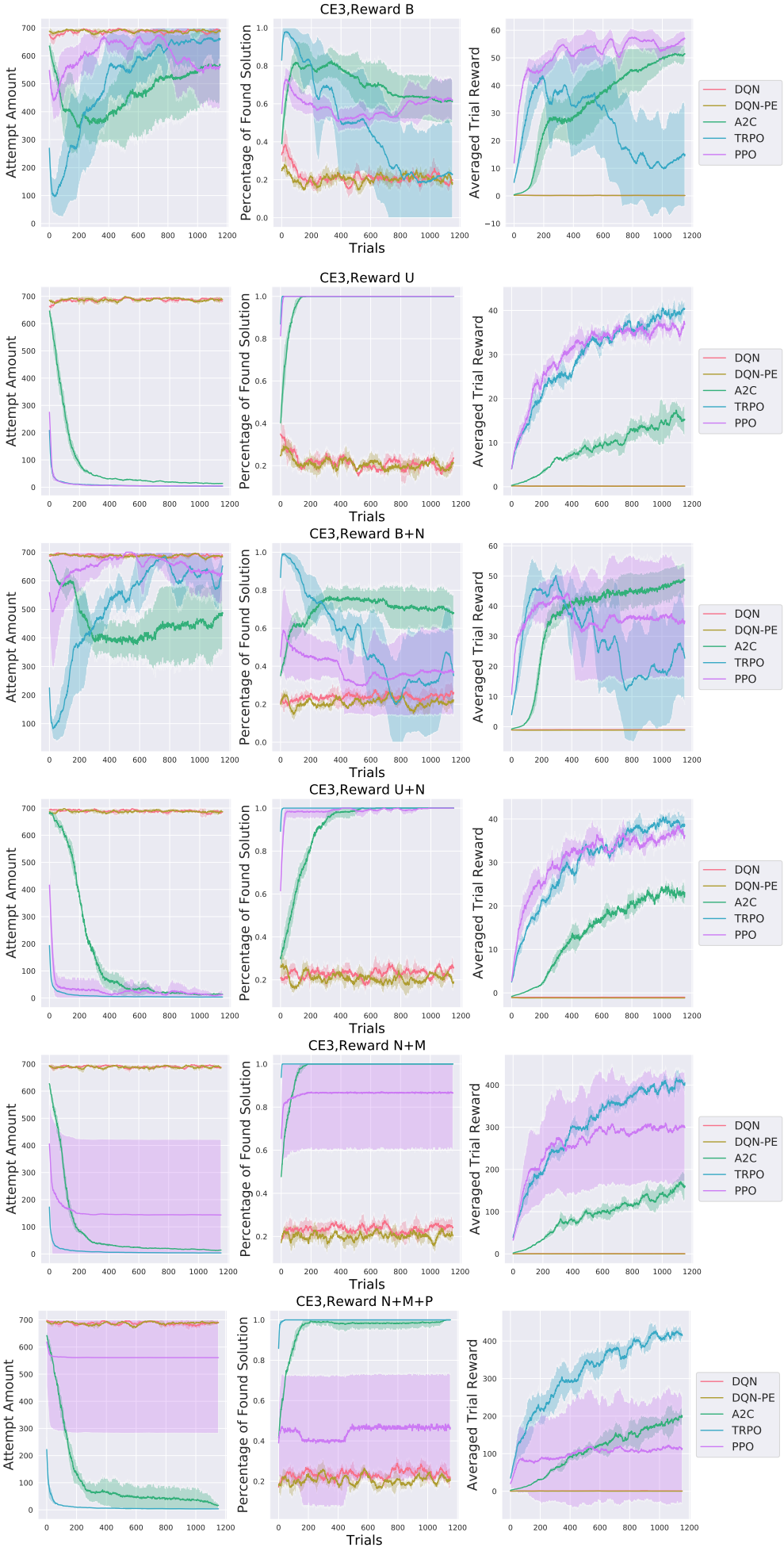

Figure 3: Results of different reward strategies on Common-Effect in 3-lever task (CE3).

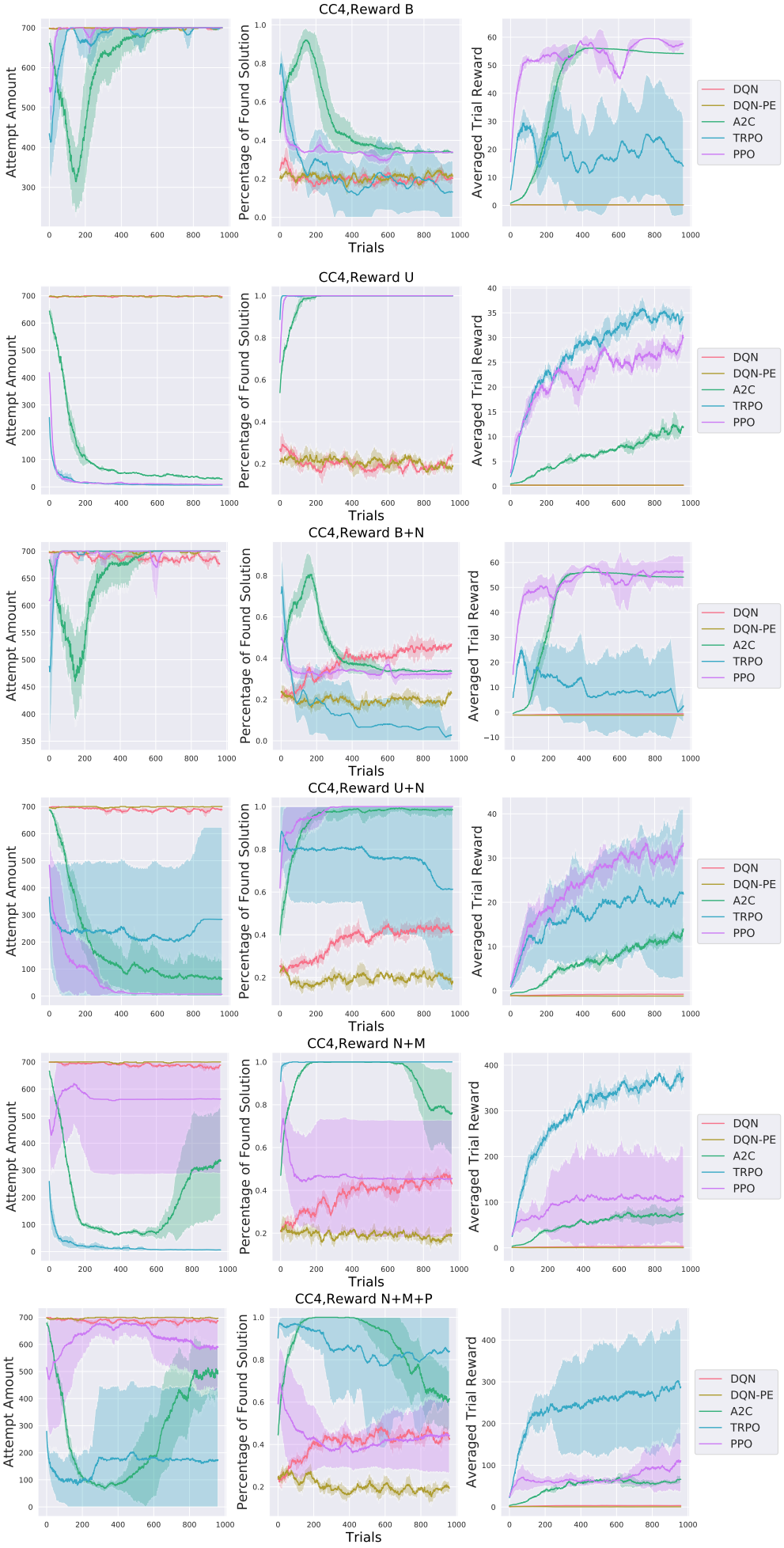

Figure 4: Results of different reward strategies on Common-Cause in 4-lever task (CC4).

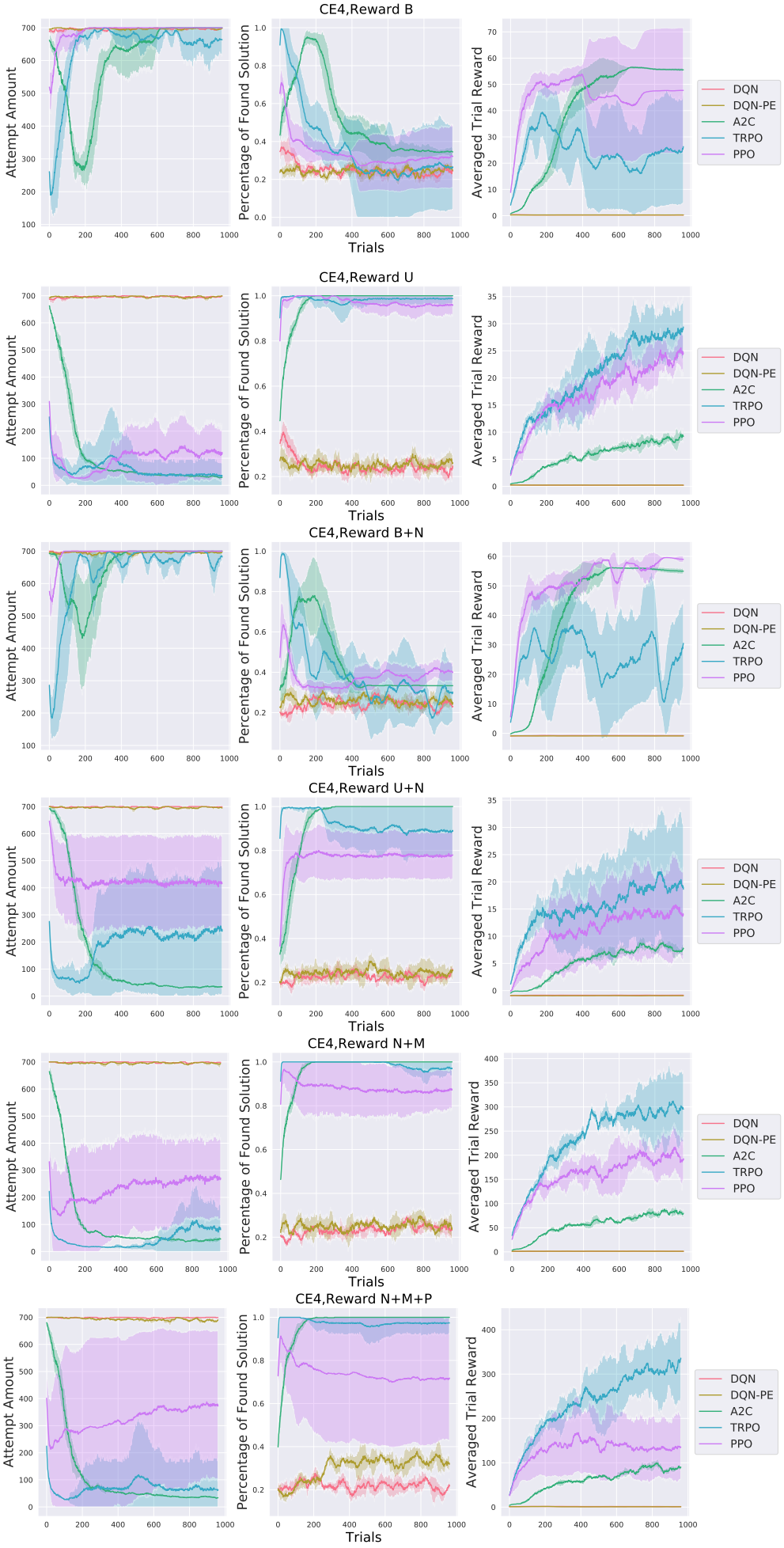

Figure 5: Results of different reward strategies on Common-Effect in 4-lever task (CE4).

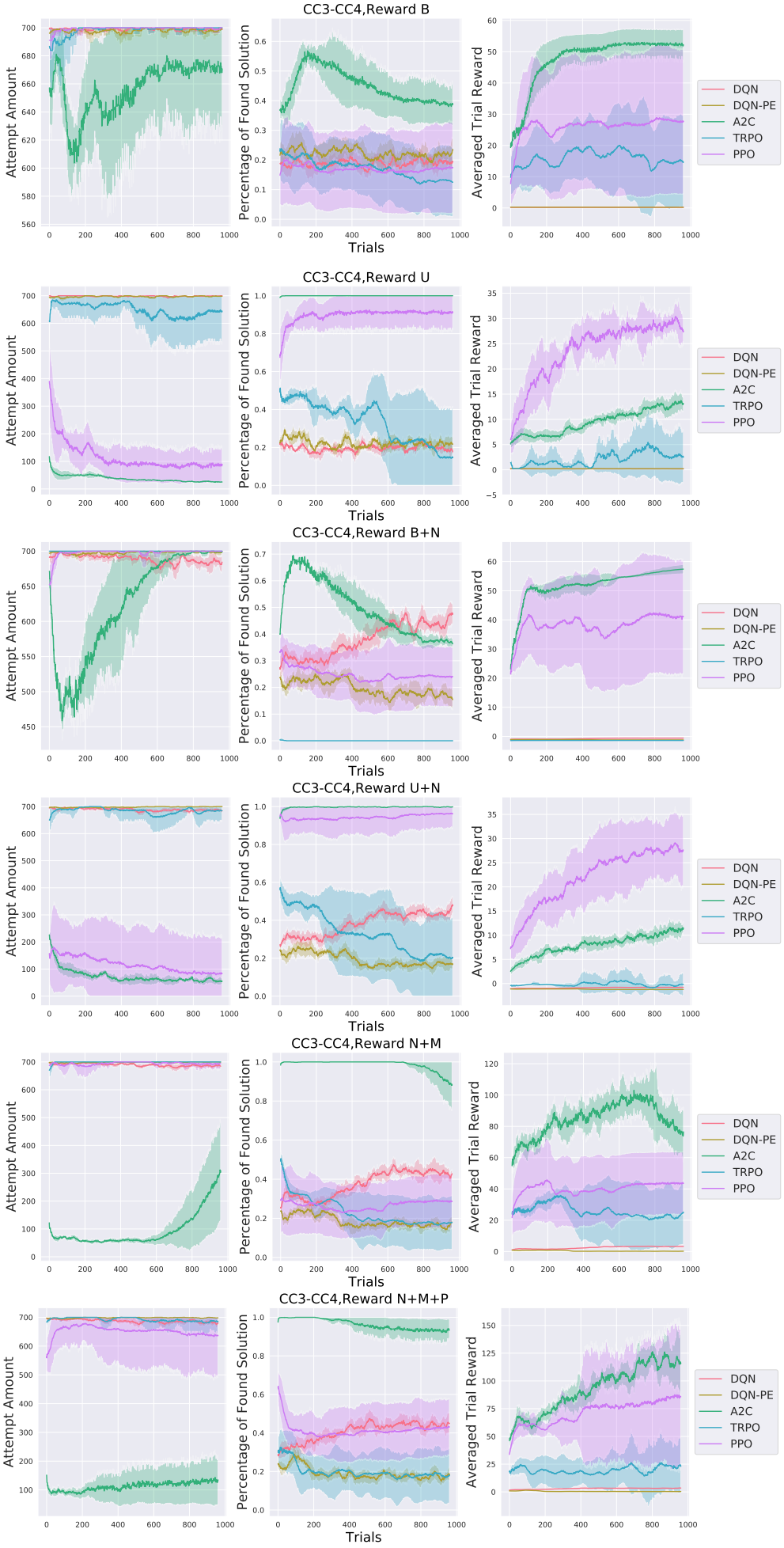

Figure 6: Results of different reward strategies on transfer to CC4 from CC3 (CC3-CC4).

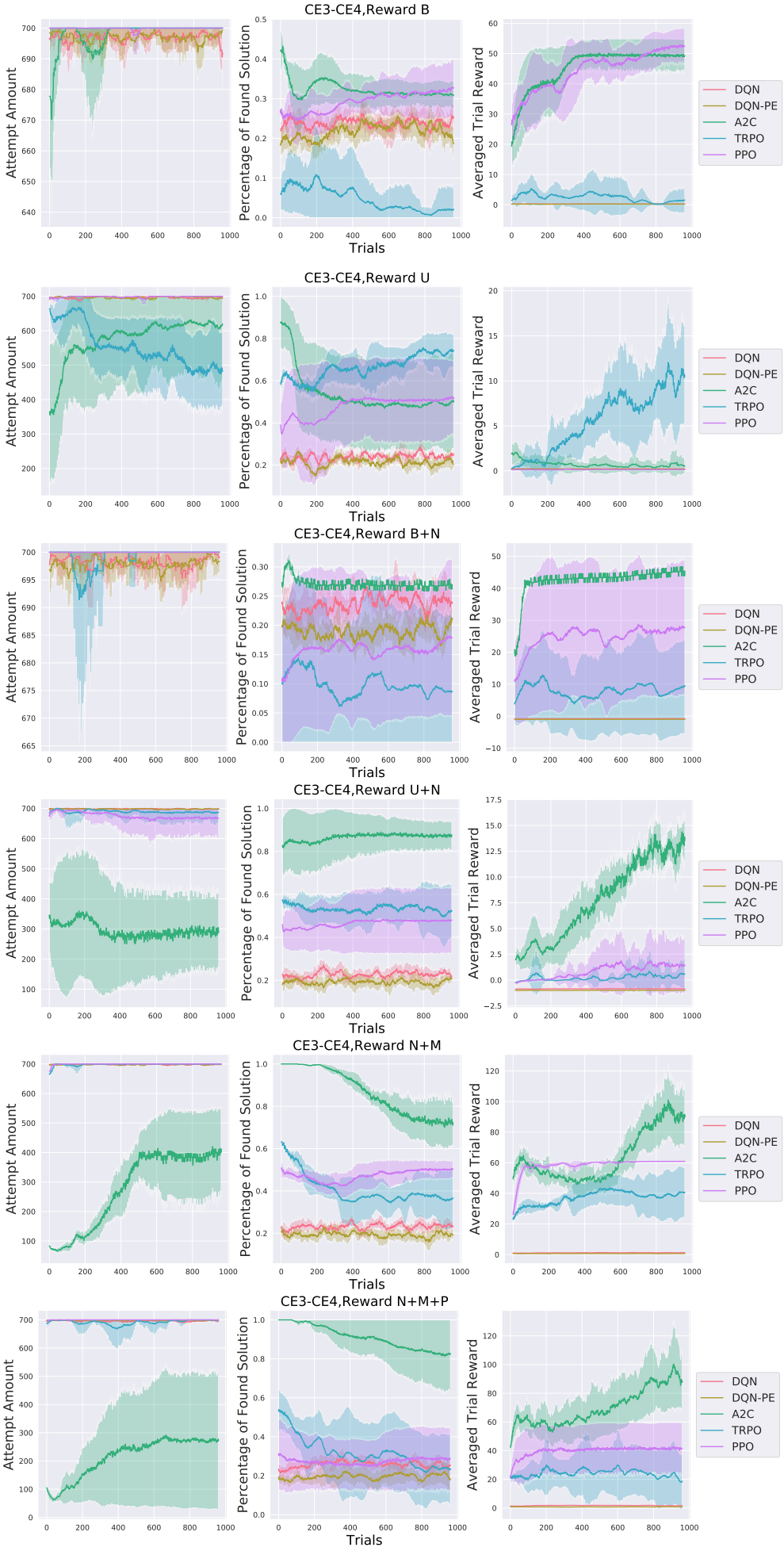

Figure 7: Results of different reward strategies on transfer to CE4 from CE3 (CE3-CE4).

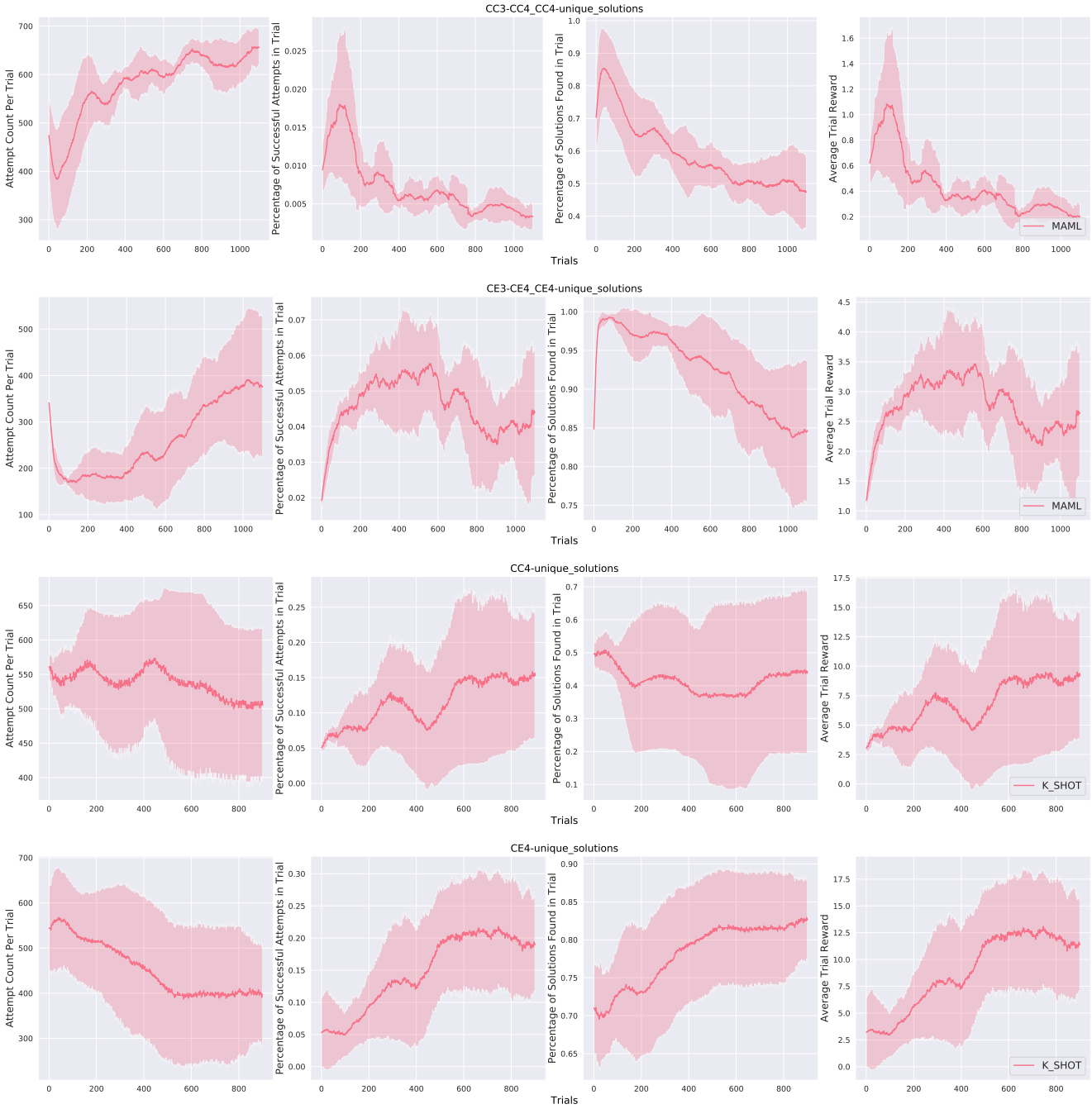

Figure 8: Results using MAML on transfer to CE4/CC4 from CE3/CC3 (CE3/CC3-CE4/CC4) with reward strategy of unique solutions. **1st** row: meta-learning on CC3 and CC4 tasks. **2nd** row: meta-learning on CE3 and CE4 tasks. **3rd** row: adaption to CC4 task with policy meta-learned on CC3 and CC4. **4th** row: adaption to CE4 task with policy meta-learned on CE3 and CE4.