Exploiting Contextual Structure to Generate Useful Auxiliary Tasks

Benedict Quartey
Department of Computer Science
Brown University

Ankit Shah
Department of Computer Science
Brown University

George Konidaris
Department of Computer Science
Brown University

Abstract

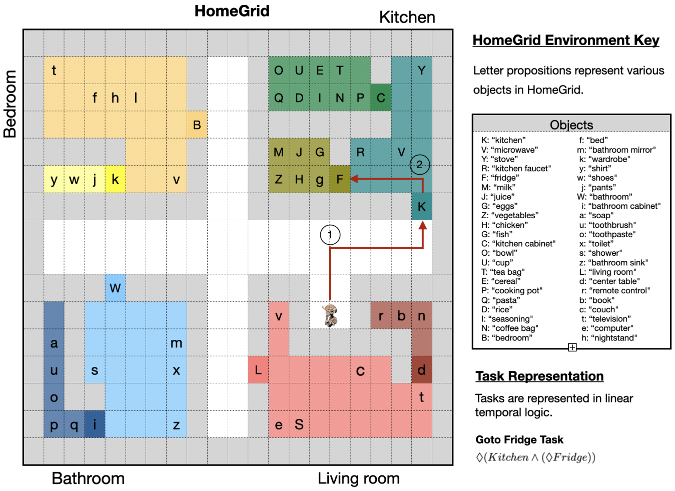
Reinforcement learning requires interaction with environments, which can be prohibitively expensive, especially in robotics. This constraint necessitates approaches that work with limited environmental interaction by maximizing the reuse of previous experiences. We propose an approach that maximizes experience reuse while learning to solve a given task by generating and simultaneously learning useful auxiliary tasks. To generate these tasks, we construct an abstract temporal logic representation of the given task and leverage large language models to generate context-aware object embeddings that facilitate object replacements. Counterfactual reasoning and off-policy methods allow us to simultaneously learn these auxiliary tasks while solving the given target task. We combine these insights into a novel framework for multitask reinforcement learning and experimentally show that our generated auxiliary tasks share similar underlying exploration requirements as the given task, thereby maximizing the utility of directed exploration. Our approach allows agents to automatically learn additional useful policies without extra environment interaction.

1 Introduction

Reinforcement learning (RL) is a general-purpose paradigm that models agents interacting with environments. It has proven to be a robust approach to sequential decision-making and has recently seen several exciting successes (1; 2; 3). However, to learn valuable behaviors that accomplish tasks, an agent must explore by repeatedly interacting with its environment, which can be prohibitively expensive, especially in robotics. Therefore it is imperative to make efficient use of experience data. One approach is to exploit the fact that, while exploring to solve any given task, an agent acquires environmental experiences that could be valuable for learning to solve many closely related tasks. Consider the task of learning to make tea in an unfamiliar house. To solve this task, a simple sequence of goals could be: *get to the kitchen, get a cup*, *get a teabag*, *get water from the faucet*, and *get milk from the fridge*. Once an agent discovers the location of a fridge while learning to solve this task, it should be able to reuse that experience to solve a related task, such as *get a drink from the fridge*.

Prior experience replay works (4; 5; 6) introduce effective methods of reusing previous experiences, and off-policy learning algorithms (7; 8; 9) also enable cross-task learning; the agent can use data generated by a *behavior policy* to learn to perform a different *target policy*. However, these approaches are limited in their ability to specify auxiliary tasks that maximally benefit from counterfactual experience reasoning and off-policy learning, in part because they do not make any assumptions about task structure.

NeurIPS 2023 Workshop on Generalization in Planning (GenPlan 2023).



HomeGrid Environment Key

Letter propositions represent various objects in HomeGrid.

Objects

K: "kitchen"	f: "bed"
V: "microwave"	m: "bathroom mirror"
Y: "stove"	k: "wardrobe"
R: "kitchen faucet"	y: "shirt"
F: "fridge"	w: "shoes"
M: "milk"	j: "pants"
J: "juice"	W: "bathroom"
G: "eggs"	i: "bathroom cabinet"
Z: "vegetables"	a: "soap"
H: "chicken"	u: "toothbrush"
G: "fish"	o: "toothpaste"
C: "kitchen cabinet"	x: "toilet"
O: "bowl"	s: "shower"
U: "cup"	z: "bathroom sink"
T: "tea bag"	L: "living room"
E: "cereal"	d: "center table"
P: "cooking pot"	r: "remote control"
Q: "pasta"	b: "book"
D: "rice"	c: "couch"
I: "seasoning"	t: "television"
N: "coffee bag"	e: "computer"
B: "bedroom"	h: "nightstand"

Task Representation

Tasks are represented in linear temporal logic.

Goto Fridge Task

$\Diamond(Kitchen \wedge (\Diamond Fridge))$

Figure 1: HomeGrid, a deterministic discrete grid-world domain. Agents in this world complete tasks by visiting grid locations corresponding to objects in the environment. Tasks are specified with LTL formulae that represent the sequence/ordering of subgoals necessary for completing a given task. Satisfying these tasks involves visiting relevant grid cells in an acceptable order determined by the task specification. As an example, the numbered arrows in the diagram indicate a policy for satisfying the **Goto Fridge** task.

We posit that task structure, as well as semantic and contextual structure in object-centric environments, can be exploited to generate auxiliary tasks. We propose a new method TaskExplore, that given a target task, uses temporal logic expressions as a means for generating contextually similar auxiliary tasks, by swapping objects using context-aware embeddings generated by large language models. In this setting, we can leverage counterfactual reasoning and off-policy methods to simultaneously learn these auxiliary tasks while learning the given task, with a behavior policy only conditioned on the given task. Our approach maximizes the utility of directed exploration experience, particularly in complex environments that require agents to constrain/direct their exploration. We show empirically that auxiliary tasks generated by TaskExplore maximally leverage the directed experience of a single-task curriculum.

In summary, we present two main contributions in this paper:

1. We present a method of using context-aware object embeddings and abstract temporal logic task representations to generate useful auxiliary tasks that share underlying exploration requirements with a given target task.
2. We demonstrate empirically that this class of generated tasks results in better experience transfer than randomly generated tasksets and uniquely benefits from directed exploration on the primary task.

2 Background

In this work, we focus on object-centric environments and leverage linear temporal logic (10) to describe temporally extended tasks involving objects. These tasks are learned using off-policy learning for linear temporal logic (9).

2.1 Linear Temporal Logic

Linear Temporal Logic (LTL) (10), presents an expressive grammar to specify temporal behavior. LTL expressions/formulae are composed of atomic propositions, the logical connectives: negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), implication ($\rightarrow$); and a set of temporal operators: next ($\bigcirc$), until (U), always ($\Box$) and eventually($\Diamond$). The minimal syntax of an LTL formula is defined below:

$$\varphi := p \mid \neg\varphi_1 \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc \varphi \mid \varphi_1 \mathrm{U} \varphi_2 \tag{1}$$

where p is an atomic proposition, a boolean literal that captures a property of the environment; φ_1 and φ_2 are valid LTL formulas. LTL formulas present an alternative, often more expressive and natural way of specifying reward objectives in the reinforcement learning setting (11). They allow the expression of explicit specifications that characterize the successful execution of a task. Consider a sequential navigation task of visiting a kitchen and then visiting a fridge, where *Kitchen* and *Fridge* are Boolean atomic propositions that can be observed. The LTL formula below can be used to specify this task:

$$\Diamond(Kitchen \wedge (\Diamond Fridge)) \tag{2}$$

LTL formulas can be progressed given a sequence of truth assignments of propositions (12) to determine which parts of the formula have been satisfied by the states seen so far and which parts remain. This is particularly useful in the reinforcement learning domain as this provides a method of tracking non-Markovian objectives. In fact, a class of reinforcement learning algorithms such as geometric-LTL (G-LTL) (11), Q-learning for Reward machines (Q-RM) (8) and LPOPL (9) leverage this to construct and solve a product MDP using the environment state space and automaton representation of LTL specifications.

2.2 Off-policy Learning with LTL

Off-policy RL methods address the setting where an agent learns a desired *Target Policy* while interacting in the environment using a different *Behavior Policy*. LTL Progression for Off-Policy Learning (LPOPL) (9) adapts the Q-learning off-policy algorithm (13) to simultaneously learn policies for multiple LTL tasks during the same environment interaction.

Given a set of LTL task specifications ϕ, LPOPL first extracts subtasks from each specification via LTL progression (12), ie. progressing every given formula over all possible truth assignments of the set of environment propositions. LPOPL then iteratively performs a series of episodes and learns a separate Q-value function for each task and extracted subtasks (also expressed as LTL formulae). At each iteration, a task is selected from ϕ and used as the objective of the episode and actions are selected with an epsilon greedy behavior policy conditioned on that task. However, all Q-value functions are updated at each step via off-policy updates, allowing the agent to make progress on tasks that may not be its current objective.

To help describe LPOPL, consider an example with the given set of tasks $\phi = \{\Diamond(a \wedge \Diamond b), \Diamond(c \wedge \Diamond b)\}$. Progressing both tasks would result in extracting the subtask $\Diamond b$. LPOPL will then initialize three Q-value functions $Q_{\Diamond(a\wedge\Diamond b)}$, $Q_{\Diamond(c\wedge\Diamond b)}$ and $Q_{\Diamond b}$. To run an episode, a task will be selected from ϕ, say $\Diamond(a \wedge \Diamond b)$, then actions will be selected epsilon greedy on $Q_{\Diamond(a\wedge\Diamond b)}$. Once the proposition *a* becomes *true* during the episode, the behavior policy becomes epsilon greedy on $Q_{\Diamond b}$, since progressing $\Diamond(a \wedge \Diamond b)$ with an assignment of True for the proposition *a* transforms the LTL formula to $\Diamond b$. This leads to a desirable property of LPOPL: task specifications that share progressed LTL forms can share subtask policies.

We borrow LPOPL's subtask extraction and Q-value function update strategies to accelerate off-policy learning of multiple tasks. However, our approach differs as we do not learn from a curriculum of tasks. We instead use a behavior policy conditioned on a *single* given task and apply counterfactual reasoning on experiences from this task to simultaneously solve generated auxiliary tasks.

3 Related Work

Experience replay methods consider single-task curriculum problems where the cardinality of the set of tasks used in extracting samples for transfer is one (1) and includes only the target task (14). The focus in these works is discovering optimal methods of organizing and training on the experience acquired from single tasks. Prioritized experience replay (5) improves on Experience Replay (4), which uniformly sampled from a replay memory, by prioritizing important transitions so they are sampled more frequently. Hindsight experience replay (HER) (6) employed exploration as an implicit curriculum and introduced learning from alternate realizations from experiences in the replay memory, by relabelling experiences based on goals that were actually achieved rather than what the agent was aiming to achieve. HER's counterfactual experience reuse is limited to singular goal states, and so cannot encode expressive temporally extended behaviors such as reaching a goal state while

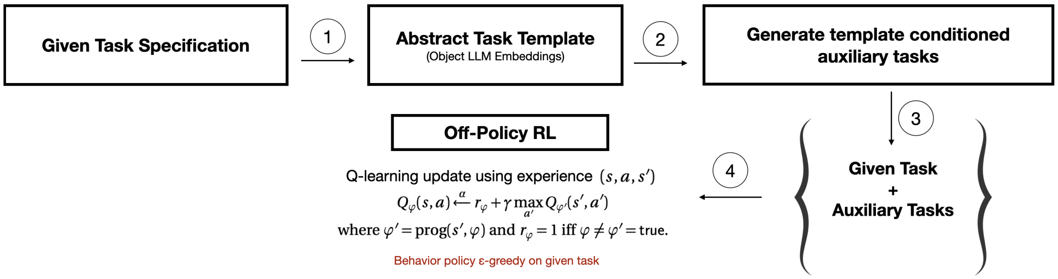

Given Task Specification
1
Abstract Task Template
(Object LLM Embeddings)
2
Generate template conditioned auxiliary tasks
3
Given Task
+
Auxiliary Tasks
4
Off-Policy RL
Q-learning update using experience (s, a, s')
$Q_\varphi(s,a) \xleftarrow{\alpha} r_\varphi + \gamma \max_{a'} Q_{\varphi'}(s', a')$
where $\varphi' = \text{prog}(s', \varphi)$ and $r_\varphi = 1$ iff $\varphi \neq \varphi' = \text{true}$.
Behavior policy ε-greedy on given task

Figure 2: This figure depicts the TaskExplore framework. Given a task specified in linear temporal logic, we construct an abstract task template that replaces instance object propositions in the given formula with large language model embeddings of their descriptions, capturing various relevant attributes of each object. We then generate auxiliary tasks by selecting objects from the environment for each proposition node in our abstract task template using the cosine similarity metric. We initialize policies (Q-value functions) for the given task and all auxiliary tasks and perform RL where actions are selected ϵ-greedy on only the given task, gathering directed experiences necessary for solving the given task. At each learning step, all Q-value functions are updated via off-policy Q-learning updates.

encountering specific intermediary states (7). Additionally, HER's alternative goals are intermediate samples of the target task and not distinct alternative tasks.

Other works such as Counterfactual experiences for reward machines (CRM) (7), Q-learning for Reward Machines (8) and LPOPL (9) introduce algorithms for applying off-policy updates to simultaneously learn multiple action-value functions. They address HER's limitation in applying counterfactual experience to temporally extended alternative tasks. However, in these works, the alternative tasks that benefit from counterfactual reasoning are assumed to be known or given. Additionally, behavior policies that dictate environment interactions from which counterfactual experiences are generated are conditioned on curricula consisting of multiple tasks–typically including the alternative tasks that benefit from these synthetic experiences.

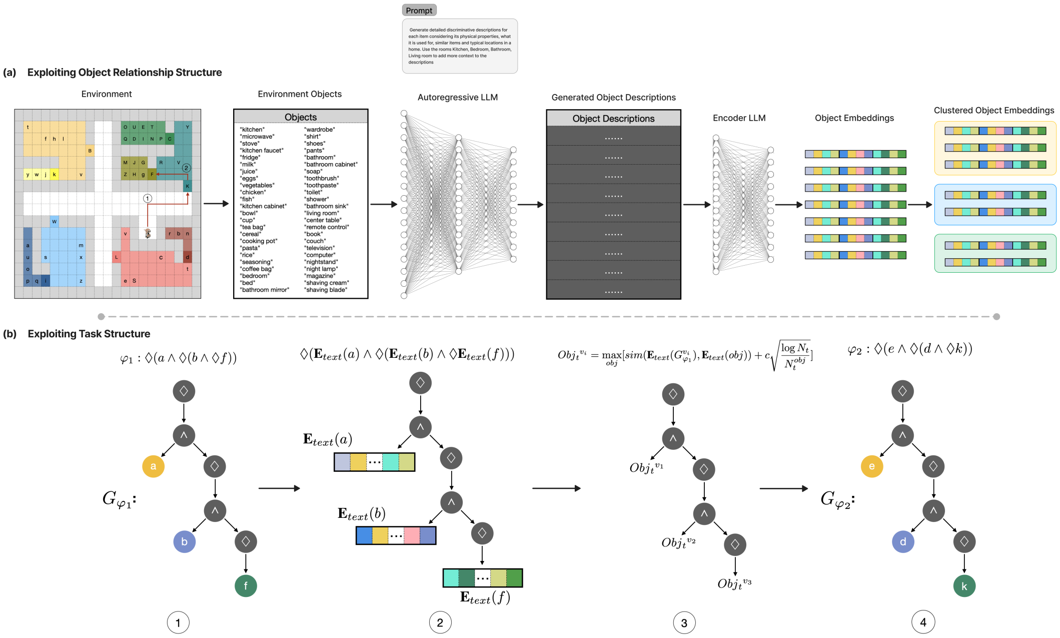
Our work seeks to present a solution to how agents might automatically generate expressive temporally extended auxiliary tasks—-in contrast to intermediate tasks—-that can maximally leverage the directed experience of a single-task curriculum in object-centric environments. TaskExplore is distinct from the limited task generation works in curriculum learning literature (15; 16), where tasks are not manually designed. The goal of our approach is not to generate good intermediate tasks to obtain experience samples, as is the goal in task generation for curriculum learning (14). Our focus is rather to generate distinct auxiliary tasks that maximally leverage directed experience from single-task curricula, which is more akin to life-long learning contexts, where agents learn to generalize from very small or constrained datasets (17).

4 Problem Definition

To define our problem formally we propose Object Oriented Non-Markovian reward decision process (OO-NMRDP), combining ideas from the Object-Oriented Markov decision process (OOMDP) (18) and Non-Markovian reward decision process (NMRDP) (19; 20; 21) formalisms. An OO-NMRDP is an 8-tuple $M = <O, C, L, S, A, T, R_\varphi, \gamma>$, where O is a set of Boolean propositions representing objects present in the environment and detectable by a labeling function $L : S \rightarrow 2^O$ that maps states to these Boolean propositions, specifying which propositions are true in which states. C is the set of object classes, S is the set of states, A is a set of actions, $\gamma \in [0, 1]$ is the discount factor and $T : S \times A \times S \rightarrow [0, 1]$ represents the transition dynamics of the environment. Unlike regular MDPs the reward function R_φ is defined over state histories, where the agent receives a reward of 1 if and only if the sequence of seen states in a given episode satisfies the LTL formula φ.

$$R_\varphi(\langle s_0, ..., s_n \rangle) = \begin{cases} 1 & \text{if } \delta_{0:n-1} \not\models \varphi \text{ and } \delta_{0:n} \models \varphi \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $\delta_{i:j} = \langle L(s_i), ..., L(s_j) \rangle$. The learning agent does not have access to either the set of object classes C or the transition dynamics of the environment T. We express sequential (11) or soft ordering constraint tasks (22) as LTL formulas over the set of propositions O.



Generate detailed discriminative descriptions for each item considering its physical properties, what it is used for, similar items and typical locations in a home. Use the rooms Kitchen, Bedroom, Bathroom, Living room to add more context to the descriptions

Figure 3: This figure depicts how TaskExplore constructs and leverages context-aware object embeddings and abstract task representations/templates. In **Figure a**, we use an autoregressive LLM to generate detailed descriptions for the list of objects in our environment and use an encoder language model to encode these generated descriptions into a 768-dimensional vector for each object. We then cluster these description embeddings, discovering object classes that capture the semantic and contextual similarity between objects. In **Figure b**, our approach constructs a task template by representing proposition nodes in the abstract syntax graph of a given LTL formula with embeddings of corresponding objects. With this task template, we can create new contextually similar tasks by selecting objects from the environment based on their cosine similarity, balancing selections between highly correlated objects and relevant yet unseen objects.

5 Exploiting contextual structure to generate auxiliary tasks

Humans can exploit the structure inherent in object-centric environments, in terms of objects and their relationships with each other, and the compositional structure of tasks. With this structure, they are able to learn tasks while thinking of alternate ways in which the experience gathered may be useful for other tasks. This ability allows humans to gain multiple useful skills when learning any one specific thing. We present how abstract temporal logic representations of tasks and context-aware embeddings of objects in an environment could be used to equip RL agents with this ability.

Figure 2 illustrates the high-level steps in our method. Given a sequential task specified in linear temporal logic φ_1, our framework leverages the compositional syntax of LTL and real-world contextual relationships between objects to develop a set of auxiliary tasks $Aux_{\varphi 1}$= $\{a_1, ..., a_n\}$ that possess similar underlying exploration requirements as φ_1, via object swaps. We then initialize a policy bank with Q-value functions for φ_1 and each task in the generated auxiliary task set $Aux_{\varphi 1}$. We simultaneously learn a policy for φ_1, and all the auxiliary tasks using off-policy updates akin to LPOPL (9). The agent always follows an ϵ-greedy policy with respect to φ_1. A key distinction between our learning approach and LPOPL is the absence of a multi-task curriculum of target tasks and our behavior policy which is conditioned on just the given task, constraining exploration in the environment.

In section 5.1 we present a detailed look into how we construct context-aware object embeddings from our environment, these embeddings are used to determine relevant objects for swaps. Section 5.2 looks at how we use these object embeddings to construct abstract task templates from which we generate auxiliary tasks. Figure 3 visualizes these two processes. Finally, in Section 5.3 we explain

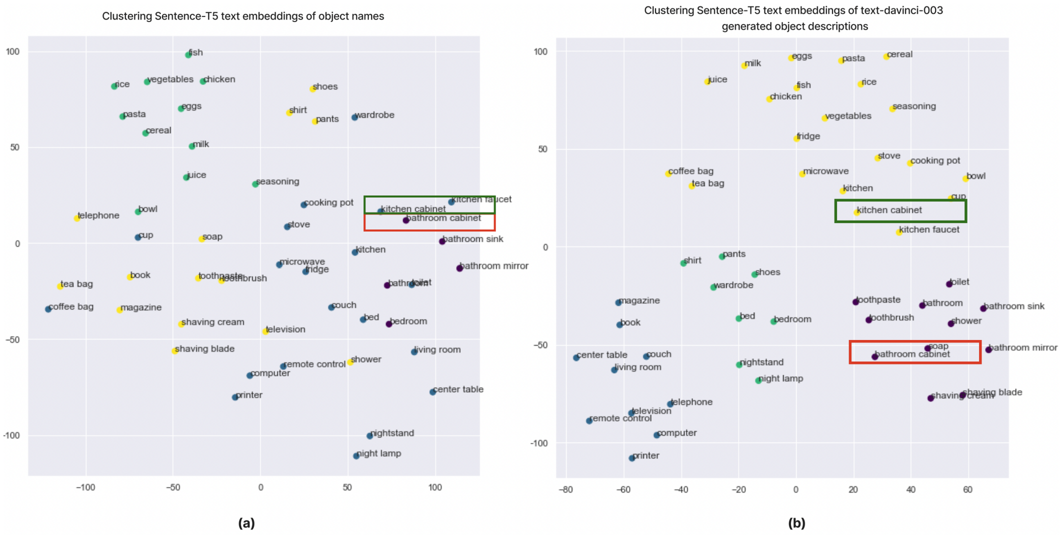


Figure 4: This figure depicts the results of performing k-means clustering on 768-dimensional embedding vectors for each environment object, results are visualized in a 2D latent space. Embeddings for each object in Figure (a) are generated by encoding the shown object name using the Sentence-T5 model. Conversely, embeddings in Figure (b) are generated by Sentence-T5 encoding text descriptions of each object generated by text-davinci-003. The number of clusters used in the k-means algorithm was four(4) based on the number distinct exploration zones in HomeGrid. **Note that embeddings generated from LLM object descriptions improved the separation of emergent cluster boundaries, and desirably increased the distance in latent space between similar yet contextually different objects such as Kitchen Cabinet and Bathroom Cabinet.**

how counterfactual reasoning and off-policy learning are used to simultaneously learn policies for the generated auxiliary tasks.

5.1 Exploiting Structure in Object Relationships

Large language models are trained on large text corpora and encode useful common-sense and context-aware human knowledge, as such act as good priors for structuring relationships between objects in object-centric environments. Our method leverages this class of models to generate discriminative context-aware embeddings of objects. Autoregressive language models such as GPT (23) are adept at language generation as they are trained to maximize the likelihood of the next token given previous tokens. Encoder-decoder and encoder-only large language models such as T5 (24) and BERT (25) on the other hand are adept at generating compact representations that effectively capture sentence context and semantics. We use the InstructGPT *text-davinci-003* model (26) to generate detailed descriptions for the list of objects in our environment and use the *Sentence-T5* encoder model (27) to encode these generated descriptions into a 768-dimensional vector for each object. We then cluster these description embeddings using the K-means algorithm (28; 29) with K-means++ initialization (30).

As an ablation, we investigate Sentence-T5's ability to generate context-aware embeddings for clustering based on object name alone, shown in Figure 4a. Figure 4b alternatively shows the improvement in clustering results using descriptions generated by the *text-davinci-003* model, presenting insights into the benefits of using large language models for context-aware data augmentation for downstream tasks.

5.2 Exploiting Structure in Task Composition

We leverage the inherent compositionality of LTL to create generalizable representations or templates of tasks given an instance LTL formula. This task template allows us to generate related auxiliary tasks. Similar to the representation format used in prior works (31) we parse a given formula φ_1 into its abstract syntax tree and represent it as a directed graph $G_{\varphi_1} = (V_{\varphi_1}, E_{\varphi_1})$. Edges E_{φ_1} in this graph connect parent operators to their subformulas and Vertices V_{φ_1} represent nodes that are either operators or atomic propositions, as shown in Figure 3b.1. This representation allows us to efficiently exploit the structure of the given task in constructing auxiliary tasks.

We create an abstract task template by traversing G_{φ_1} and replacing instance proposition nodes––which represent objects in our environment—with embeddings of their descriptions as shown in Figure 3b.2. With this abstract task template, we generate x new auxiliary tasks by swapping relevant objects from the environment at each proposition node. We select objects whose embedding lies in the same embedding cluster and prioritize those with the highest cosine similarity with the template embedding node being considered. We introduce a simple value-dependent object selection metric based on upper confidence bounds (32) that balances out selecting high cosine similarity objects and relevant but unseen objects.

Equation 4 governs object selection for each embedding node in a given template.

$$Obj_t{}^{v_i} = \max_{obj}[sim(\mathbf{E}_{text}(G^{v_i}_{\varphi_1}), \mathbf{E}_{text}(obj)) + c\sqrt{\frac{\log N_t}{N^{obj}_t}}] \tag{4}$$

where $Obj_t{}^{v_i}$ is the chosen object proposition for template node i at trial t; $sim(\mathbf{E}_{text}(G^{v_i}_{\varphi_1}), \mathbf{E}_{text}(obj))$ is the cosine similarity between the embedding of object obj and the embedding at node v_i of the template G_{φ_1}; **c** is a tuneable parameter that balances selecting high-cosine similarity objects vs relevant but unseen objects; N_t is the total number of object selection trails; N^{obj}_t is the number of trails where object obj was selected.

5.3 Off-policy Updates via Counterfactual Experience

Concerning off-policy updates, when an episode is run with an epsilon greedy behavior policy on the given LTL formula, the Q-value function of each auxiliary task will also be updated as if their corresponding formula was the current objective. Assuming an action *a* is taken in state *s* resulting in a new state $s^{'}$, to update a specific Q_φ, the reward that would have been observed during that transition if the agent's objective was φ is computed. To achieve this, φ is progressed through the new state $s^{'}$; if the resulting formula $\varphi^{'}$ is *true* the reward is 1, and 0 otherwise. Q_φ is then updated using the following rule:

$$Q_\varphi(s,a) \leftarrow Q_\varphi(s,a) + \alpha(r + \gamma \max_{a^{'}} Q_{\varphi'}(s^{'}, a^{'}) - Q_\varphi(s,a)) \tag{5}$$

As shown in LPOPL (9) learning is globally optimal as $Q_\varphi(s,a)$ is updated with the maximum of every action $a^{'}$ from its progressed subtask $Q_{\varphi'}(s^{'}, a^{'})$.

6 Experiments

The task specification used in our experiments was a food preparation task where the agent had to go to the **kitchen cabinet**, obtain a **cooking pot**, obtain **seasoning**, then go to the **fridge**, obtain **chicken**, and finally go to the **stove**. In HomeGrid, this task corresponds to visiting the right cells in the correct order. The LTL formula below represents this task using the atomic propositions that represent each of the relevant objects:

$$\Diamond(C \wedge \Diamond(P \wedge \Diamond(I \wedge \Diamond(F \wedge \Diamond(H \wedge \Diamond Y))))) \tag{6}$$

See Figure 1 for a description of our environment HomeGrid. We find that a good heuristic for choosing the minimum number of object clusters is the number of distinct exploration zones in the environment. For HomeGrid, this is four (4) since there are four distinct useful exploration regions namely "Kitchen", "Bathroom", "Living room" and "Bedroom". We evaluate our approach with the following three conditions:

1. **Ours**: Given our food preparation task φ_1 we generate **20 auxiliary tasks** following our approach. We learn these tasks simultaneously with φ_1 with a behavior policy epsilon greedy on φ_1, directing exploration towards more relevant experiences for φ_1.

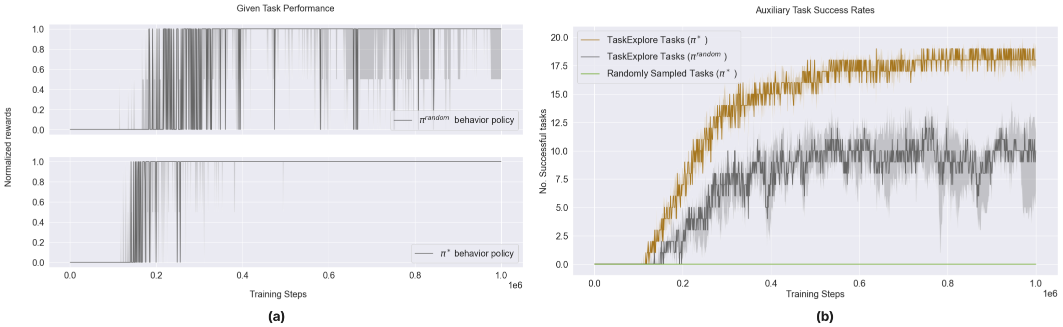


Figure 5: Figure (a) shows the normalized discounted reward obtained by the agent on the given task as it learns to solve it simultaneously with TaskExplore generated auxiliary tasks using a random behavior policy (π^{random}) and an epsilon greedy behavior policy (π^*). Figure (b) shows the task success rate on auxiliary tasks as learning progresses. Learning TaskExplore generated tasks while using epsilon greedy (π^*) behavior policy on the given task significantly outperforms all other baselines. All results are normalized over 7 different seeded runs

2. **Baseline 1**: To demonstrate that tasks generated by TaskExplore uniquely leverage the directed experience of a single-task curriculum, we repeat the approach described above, replacing the behavior policy with a random one that explores more widely.

3. **Baseline 2**: To show that tasks generated by TaskExplore more relevantly benefit from directed exploration experience than the general set of possible tasks, we generate **20 auxiliary tasks** by randomly sampling sequential tasks from the set of propositions in our environment, as typically done in prior works (31; 22). These tasks are sampled to have the same length of propositions as the given task which places them in the same level of difficulty. We learn these tasks simultaneously with φ_1, using a behavior policy epsilon greedy on φ_1.

7 Results and Discussion

Figure 5 presents the results from our experiments, which highlight several interesting properties of the auxiliary task set developed by TaskExplore. Firstly, in Figure 5a, we see that our approach which simultaneously learns to solve auxiliary tasks with a behavior policy ϵ-greedy on the given task does not adversely affect performance on the given task. However, performance on the given task deteriorates when using a random behavior policy, highlighting the relevance of directed exploration.

Intuitively, learning with a random behavior policy gathers more diverse experiences which should benefit multiple auxiliary tasks more than the experience gathered during directed exploration. However, our results in 5b show that the auxiliary tasks developed by TaskExplore maximally leverage that constrained experience and actually performs better than when using a random behavior policy. This is because the generated tasks are contextually similar to the given task and share similar underlying exploration requirements.

Figure 6 presents further insights into this phenomenon. It shows a sample exploration heatmap of a single complete run of our experiment learning the food preparation task with random and epsilon-greedy behaviour policies. In the first episode both behaviour policies explore widely, however as learning progresses the epsilon-greedy policy leads to more directed and constrained experiences that focuses on the Kitchen exploration zone. The random behaviour policy on the other hand still explores widely, not paying much attention to the kitchen exploration zone it needs to be focusing on to make progress on relevant auxiliary tasks.

Finally, the results in 5b show that developing a curriculum of randomly sampled tasks from the general distribution space of possible tasks in the environment cannot leverage directed exploration experience as well as auxiliary tasks developed by TaskExplore.

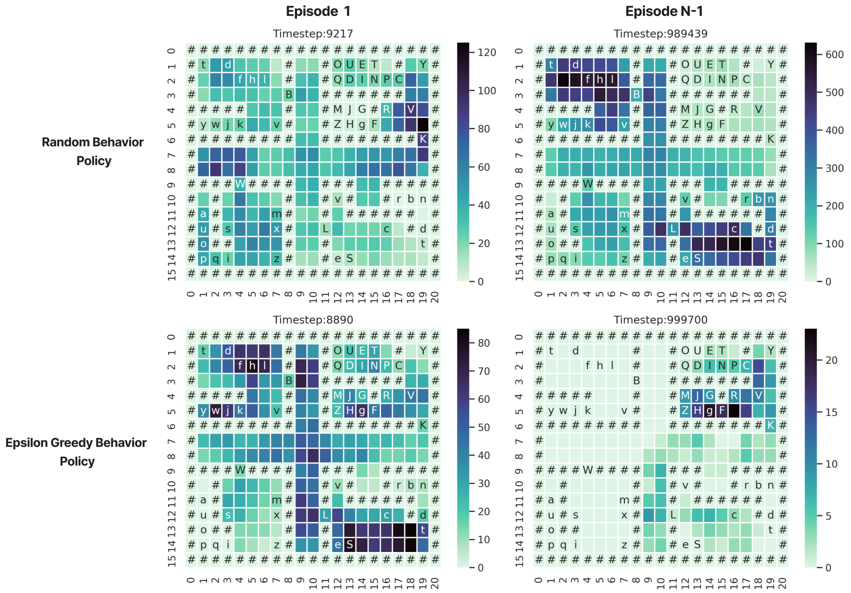


Figure 6: This figure shows a sample exploration heatmap of the random and e-greedy behavior policies while learning the given food preparation task with counterfactual updates on TaskExplore generated auxiliary tasks. The agent starts each episode in cell (x-axis=14,y-axis=13) and darker cell colors correlates to the number of times the agent visited that cell. This is an ablative diagram that helps visualize the beneficial effects of directed exploration in complex environments and how this directed experience can benefit contextually similar auxiliary tasks that share the same underlying exploration requirements, more than an exploration strategy that may produce more diverse experiences.

8 Conclusion

This paper introduced an approach to how agents might automatically generate expressive temporally extended auxiliary tasks that can maximally leverage the directed experience of a single-task curriculum in object-centric environments. This approach to auxiliary task generation is particularly valuable in the lifelong learning setting, as agents can generate and solve new tasks from constrained datasets. In the spirit of reusing computation, a policy bank of these policies can be saved and reused to accelerate learning future tasks. Modern vision-language models (VLMs) that detect open vocabulary objects in real world environemnts can be employed in future work to relax TaskExplore's dependence on a predefind set of object propositions from which tasks can be expressed and labelling functions that map states to proposition truth values,

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

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

[3] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[4] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine learning*, vol. 8, pp. 293–321, 1992.

[5] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[6] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing systems*, vol. 30, 2017.

[7] R. T. Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "Reward machines: Exploiting reward function structure in reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 73, pp. 173–208, 2022.

[8] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith, "Using reward machines for high-level task specification and decomposition in reinforcement learning," in *International Conference on Machine Learning*, pp. 2107–2116, PMLR, 2018.

[9] R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "Teaching multiple tasks to an rl agent using ltl," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, p. 452–461, International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[10] A. Pnueli, "The temporal logic of programs," in *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, p. 46–57, IEEE Computer Society, 1977.

[11] M. L. Littman, U. Topcu, J. Fu, C. Isbell, M. Wen, and J. MacGlashan, "Environment-independent task specifications via gltl," *arXiv preprint arXiv:1704.04341*, 2017.

[12] F. Bacchus and F. Kabanza, "Using temporal logics to express search control knowledge for planning," *Artificial intelligence*, vol. 116, no. 1-2, pp. 123–191, 2000.

[13] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[14] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 7382–7431, 2020.

[15] S. Narvekar, J. Sinapov, M. Leonetti, and P. Stone, "Source task creation for curriculum learning," in *Proceedings of the 2016 international conference on autonomous agents & multiagent systems*, pp. 566–574, 2016.

[16] F. L. D. Silva and A. H. R. Costa, "Object-oriented curriculum generation for reinforcement learning," in *Proceedings of the 17th international conference on autonomous agents and multiagent systems*, pp. 1026–1034, 2018.

[17] S. Thrun, "Lifelong learning algorithms.," *Learning to learn*, vol. 8, pp. 181–209, 1998.

[18] C. Diuk, A. Cohen, and M. L. Littman, "An object-oriented representation for efficient reinforcement learning," in *Proceedings of the 25th international conference on Machine learning*, pp. 240–247, 2008.

[19] R. Brafman, G. De Giacomo, and F. Patrizi, "Ltlf/ldlf non-markovian rewards," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[20] A. Camacho, O. Chen, S. Sanner, and S. A. McIlraith, "Non-markovian rewards expressed in ltl: guiding search via reward shaping," in *Tenth annual symposium on combinatorial search*, 2017.

[21] S. Thiébaux, C. Gretton, J. Slaney, D. Price, and F. Kabanza, "Decision-theoretic planning with non-markovian rewards," *Journal of Artificial Intelligence Research*, vol. 25, pp. 17–74, 2006.

[22] J. X. Liu, A. Shah, E. Rosen, G. Konidaris, and S. Tellex, "Skill transfer for temporally-extended task specifications," *arXiv preprint arXiv:2206.05096*, 2022.

[23] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, "Improving language understanding by generative pre-training," 2018.

[24] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[26] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, *et al.*, "Training language models to follow instructions with human feedback," *arXiv preprint arXiv:2203.02155*, 2022.

[27] J. Ni, G. H. Ábrego, N. Constant, J. Ma, K. B. Hall, D. Cer, and Y. Yang, "Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models," *arXiv preprint arXiv:2108.08877*, 2021.

[28] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[29] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symposium on Math., Stat., and Prob*, p. 281, 1965.

[30] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," tech. rep., Stanford, 2006.

[31] P. Vaezipoor, A. C. Li, R. A. T. Icarte, and S. A. Mcilraith, "Ltl2action: Generalizing ltl instructions for multi-task rl," in *International Conference on Machine Learning*, pp. 10497–10508, PMLR, 2021.

[32] P. Auer, "Using confidence bounds for exploitation-exploration trade-offs," *Journal of Machine Learning Research*, vol. 3, no. Nov, pp. 397–422, 2002.