The Journal of Open Source Software

HierarchyCraft: A Benchmark Builder for Hierarchical Reasoning

Mathis Fédérico[1,2], **Shang Wang**[1], **Yuxuan Li**[1,3], **and Matthew E. Taylor**[1,3]

1 Department of Computing Science, University of Alberta, Canada **2** CentraleSupelec, University of Paris-Saclay, France **3** Alberta Machine Intelligence Institute (Amii), Canada

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Tristan Miller

Reviewers:

- @lwu9
- @Christopher-Henry-UM
- @inpefess

Submitted: 14 January 2024
Published: unpublished

Summary

Hierarchical reasoning poses a fundamental challenge in the field of artificial intelligence (Botvinick & Weinstein, 2014). Existing methods may struggle when confronted with hierarchical tasks (Bacon et al., 2017; Heess et al., 2016; Nachum et al., 2018), yet despite the importance of understanding how the structure of an underlying hierarchy affects task difficulty, there is a lack of suitable environments or benchmarks to facilitate this exploration.

We introduce **HierarchyCraft**, a software package that allows researchers to create custom environments based on their hierarchical structures, enabling the study of hierarchical reasoning.

To isolate hierarchical behavior and ensure compatiblilty with classical planning algorithms, we exclude unstructured data like images and text, avoiding the complexity of feature extraction, and allowing comparisons between classical planning and reinforcement learning.

HierarchyCraft simplifies the creation of diverse hierarchical environments from a list of a single building block as showcased by the set of pre-defined environments.

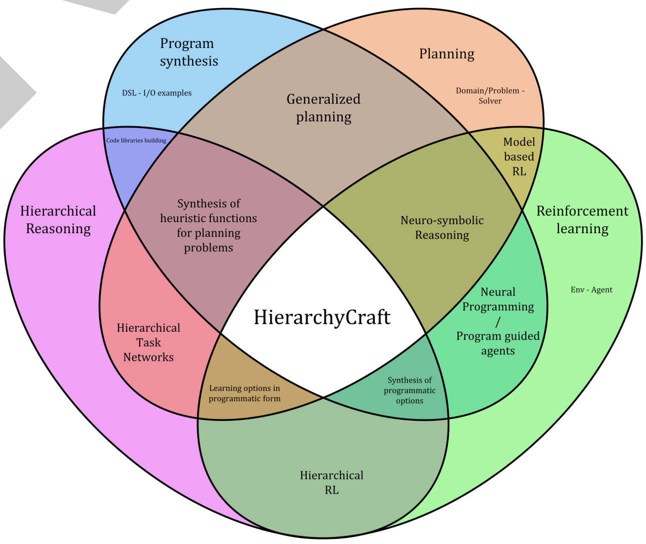

Figure 1: HierarchyCraft is at the intersection of reinforcement learning, planning, hierarchical reasoning and program synthesis.

Statement of need

HierarchyCraft is designed as a user-friendly Python library for constructing environments tailored to the study of hierarchical reasoning in the contexts of reinforcement learning, classical planning, and program synthesis as displayed in Figure 1.

Analysis and quantification of the impacts of diverse hierarchical structures on learning agents is essential for advancing hierarchical reasoning. However, current hierarchical benchmarks often limit themselves to a single hierarchical structure per benchmark, and present challenges not only due to this inherent hierarchical structure but also because of the necessary representation learning to interpret the inputs.

HierarchyCraft is a benchmark builder (not a benchmark) designed to study how different hierarchical structures impact the performance of classical planners and reinforcement learning algorithms. It includes several examples, detailed in the documentation, that can serve as initial benchmarks. These examples include basic parametrised hierarchical structures (Random, Recursive, Tower) and fixed structures imitating other environments (MineHCraft for Minecraft tasks, MiniHCraft for Minigrid tasks). Researchers are encouraged to create custom hierarchical environments to explore structures of their choice.

We argue that arbitrary hierarchical complexity can emerge from simple rules. To the best of our knowledge, no general frameworks currently exist for constructing environments dedicated to studying the hierarchical structure itself. We next highlight five related benchmarks, underscoring the necessity for the development of a tool like HierarchyCraft.

GridWorld

GridWorld, a general class of 2D grid-based environments, is frequently utilised in hierarchical reinforcement learning research, notably within the options framework (**?**).

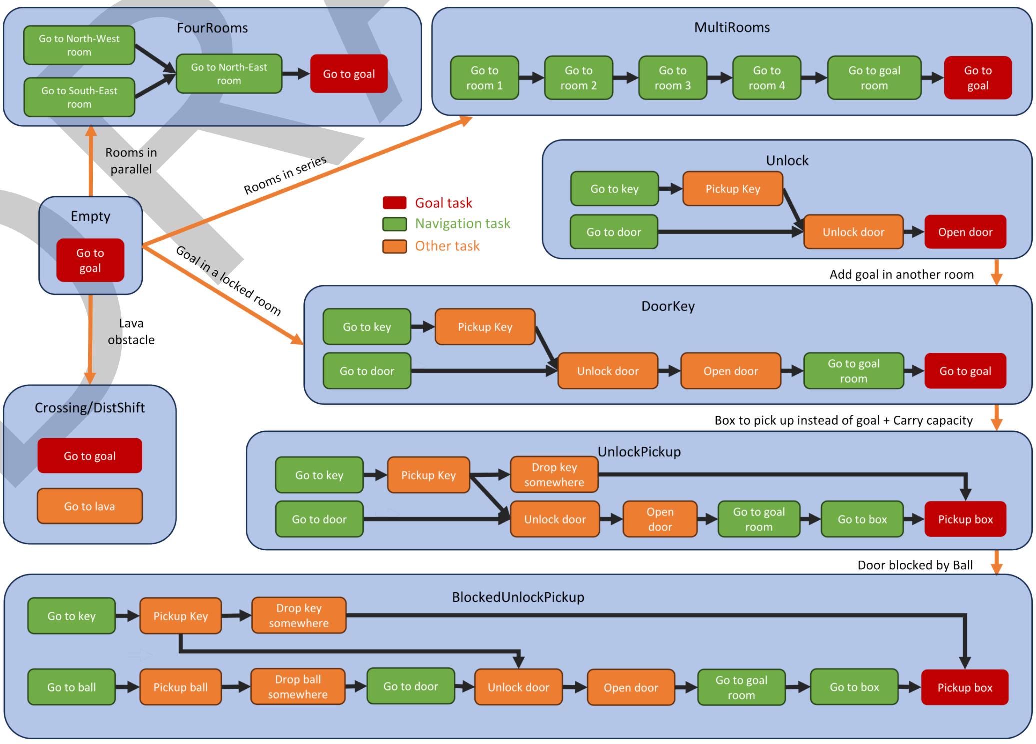

Figure 2: Example of Minigrid environments hierarchical structures and their relationships. There are only a few possible subtasks and most of them are navigation tasks (in green).

Minigrid (Chevalier-Boisvert et al., 2023) is a user-friendly Python library that not only implements a GridWorld engine but also expands its capabilities. This allows researchers to create more intricate scenarios by introducing additional rooms, objectives, or obstacles.

As illustrated in Figure 2, GridWorld environments only exhibit **limited and similar hierarchical structures** that primarily focus on navigation tasks, making these hierarchies mostly sequential.

Minecraft

A good example of a hierarchical task is the collection of diamonds in the popular video game Minecraft, as showcased in the MineRL competition (Guss et al., 2021), where hierarchical reinforcement learning agents have dominated the leaderboard (Milani et al., 2020).

Due to sparse rewards, the difficulty of exploration, and long time horizons in this procedurally generated sandbox environment, DreamerV3 (Hafner et al., 2023) recently became the first algorithm to successfully collect diamonds in Minecraft without prior training or knowledge. Unfortunately, DreamerV3 required training on an Nvidia V100 GPU for 17 days, gathering roughly 100 million environmental steps. Such **substantial computational resources are unavailable to many researchers**, impeding the overall progress of research on hierarchical reasoning.

Moreover, although Minecraft has an undeniably complex hierarchical structure, **this underlying hierarchical structure is fixed** and cannot be modified without modding the game, a complex task for researchers.

Crafter

Crafter (Hafner, 2022) presents a lightweight grid-based 2D environment, with game mechanics akin to Minecraft and poses similar challenges (exploration, representation learning, rewards sparsity and long-term reasoning) at much lower compute cost.

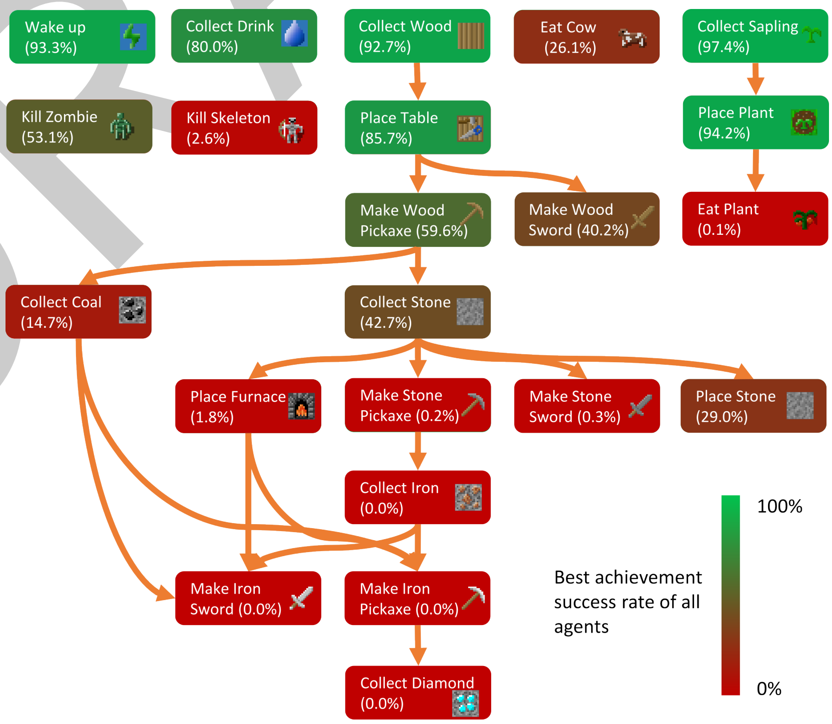

Figure 3: Hierarchical structure of the Crafter environment as presented by the authors of Crafting with their success rates. Inspired by Figure 4 of (Hafner, 2022).

Although Crafter offers 22 different tasks displayed in Figure 3, the **underlying hierarchical structure is fixed**, restricting how researchers can investigate the impacts of changes to the hierarchical structure.

Moreover, the tasks considered by the authors do not encompass various navigation-related subtasks (such as finding water, locating a cow, waiting for a plant to grow, or returning to a table), nor do they include certain optional but beneficial subtasks (for example, using swords or the skill of dodging arrows can make the task of defeating skeletons easier).

These omissions results in abrupt drops in success rates within the hierarchy, rather than a more gradual progression in difficulty. This highlights that the hierarchy presented by the authors is incomplete, as it fails to capture the full range of subtasks in Crafter and the necessary or helpful interactions between them for successfully completing higher-level tasks.

Arcade Learning Environment (Atari)

The arcade learning environment (Bellemare et al., 2015) stands as a standard benchmark in reinforcement learning, encompassing over 55 Atari games. However, **only a few of these games, such as Montezuma's Revenge and Pitfall, necessitate hierarchical reasoning**.

Each Atari game has a **fixed hierarchy that cannot be modified** and agents **demand substantial computational resources** to extract relevant features from pixels or memory, significantly slowing down experiments.

PDDLGym

PDDLGym (Silver & Chitnis, 2020) is a Python library that automatically constructs Gym environments from Planning Domain Definition Language (PDDL) domains and problems. PDDL (Drew McDermott, 1998) functions as a problem specification language, facilitating the comparison of different symbolic planners.

However, constructing PDDL domains and problems with a hierarchical structure is challenging and time-consuming, especially for researchers unfamiliar with PDDL-like languages.

Additionally, PDDLGym is **compatible only with PDDL1** and does not support numeric-fluents introduced in PDDL 2.1 that are required to represent quantities in the inventories of HierarchyCraft environments.

Abstraction and Reasoning Corpus (ARC)

The Abstraction and Reasoning Corpus (ARC) (Chollet, 2019), is both hierarchical and diverse, as each task exhibits its own implicit hierarchical structure. However, these hierarchical structures are not explicitly provided within the dataset, such as with shorter programs for each task. Making these hierarchical structures explicit would also contribute significantly to the development of hierarchical reasoning like what HierarchyCraft is trying to achieve.

Much like Gridworld, ARC tasks require feature extraction from 2D grids, leveraging priors related to their spatial nature, which bias the nature of the tasks on that specific data structure.

Additionaly ARC tasks do not emphasise long-term reasoning as they are relatively short compared to tasks within other benchmarks like Minecraft or even Gridworld; this makes underlying hierarchical structures shallow for each task and more wide than deep for the whole corpus.

Partitioning those underlying hierarchical structures and classifying them relatively to the difficulty of finding a solution, independently of the solution's nature, is at the core of HierarchyCraft's motivation.

Design goals

HierarchyCraft aims to be a fruitful tool for investigating hierarchical reasoning, focussing on achieving the following four design goals.

1. Hierarchical by design

The action space of HierarchyCraft environments consists of subtasks, referred to as *Transformations*, as opposed to detailed movements and controls. But each *Transformations* has specific requirements to be valid (e.g. have enough of an item, be in the right place), and these requirements may necessitate the execution of other *Transformations* first, inherently creating a hierarchical structure in HierarchyCraft environments. This concept is visually represented by the *Requirements Graph* depicting the hierarchical relationships within each HierarchyCraft environment. The *Requirements Graph* is directly constructed from the list of *Transformations* composing the environement, as illustrated in Figure 4. Requirements Graphs should be viewed as a generalisation of previously observed graphical representations from related works, including Figure 3 and Figure 2.

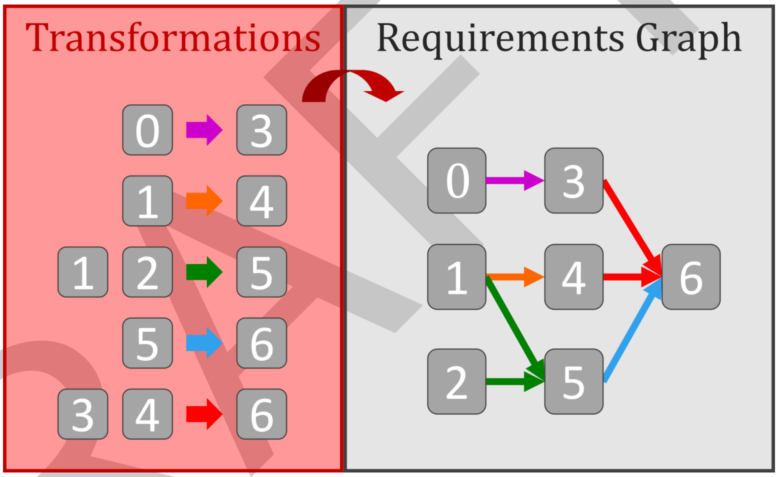

Figure 4: How subtasks build a hierarchical structure.

2. Easy to use and customise

HierarchyCraft is a versatile framework enabling the creation of diverse hierarchical environments. The library is designed to be simple and flexible, allowing researchers to define their own hierarchical environments with detailed guidance provided in the documentation. To showcase the range of environments possible within HierarchyCraft, multiple examples are provided.

3. No feature extraction needed

In contrast to benchmarks that yield grids, pixel arrays, text, or sound, HierarchyCraft directly provides a low-dimensional representation that does not require additional feature extraction, as depicted in Figure 5.

The Journal of Open Source Software

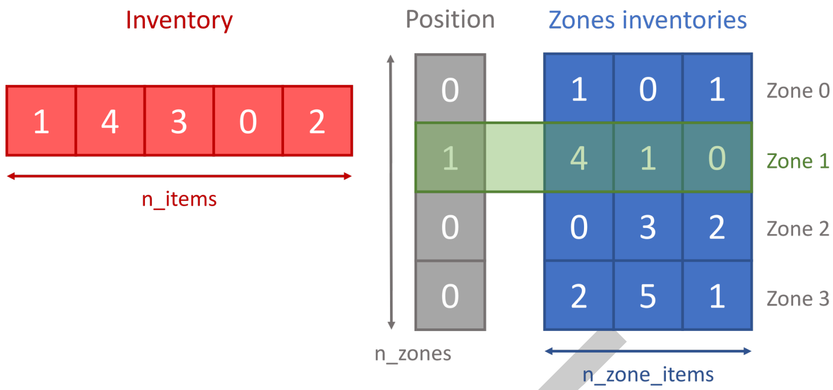

Figure 5: HierarchyCraft compact state representation.

This not only saves computational time but also enables researchers to concentrate on hierarchical reasoning, allows the use of classical planning frameworks such as PDDL (Drew McDermott, 1998) or ANML (Smith et al., 2007), and enables the creation of any arbitrary complex custom environment from a list of *Transformations*, nothing more.

4. Compatible with multiple frameworks

HierarchyCraft environments are directly compatible with both reinforcement learning through Gymnasium (Towers et al., 2024) and planning through the Unified Planning Framework ("Unified Planning Framework," 2023) (see Figure 6). This compatibility facilitates usage by both the reinforcement learning and planning communities.

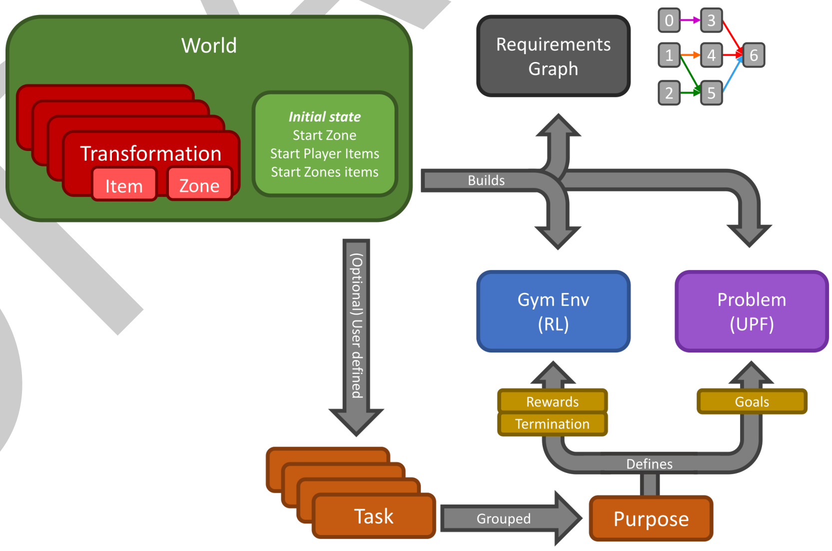

Figure 6: HierarchyCraft pipeline into different representations.

Acknowledgements

This work was made possible by the research program of the engineering cursus at CentraleSupelec, University of Paris-Saclay, France.

The research was conducted at the Intelligent Robot Learning (IRL) Lab, University of Alberta, which is supported in part by research grants from the Alberta Machine Intelligence Institute (Amii); a Canada CIFAR AI Chair, Amii; Compute Canada; Huawei; Mitacs; and NSERC.

The Journal of Open Source Software

Special thanks to Laura Petrich and other members of the IRL Lab for their assistance in finding and describing related works, their critical thinking on the project, and their contributions to the revisions of the documentation and report.

References

Bacon, P.-L., Harb, J., & Precup, D. (2017). The option-critic architecture. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 1726–1734.

Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2015). The arcade learning environment: An evaluation platform for general agents. *Proceedings of the 24th International Conference on Artificial Intelligence*, 4148–4152. ISBN: 9781577357384

Botvinick, M., & Weinstein, A. (2014). Model-based hierarchical reinforcement learning and human action control. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *369*(1655), 20130480.

Chevalier-Boisvert, M., Dai, B., Towers, M., Perez-Vicente, R., Willems, L., Lahlou, S., Pal, S., Castro, P. S., & Terry, J. (2023). Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *Advances in Neural Information Processing Systems 36, New Orleans, LA, USA*. https://github.com/Farama-Foundation/Minigrid

Chollet, F. (2019). On the measure of intelligence. *ArXiv*, *abs/1911.01547*. https://api.semanticscholar.org/CorpusID:207870692

Drew McDermott. (1998). PDDL - the planning domain definition language. *Artificial Intelligence Planning Systems, TR98003/DCS TR1165*.

Guss, W. H., Castro, M. Y., Devlin, S., Houghton, B., Kuno, N. S., Loomis, C., Milani, S., Mohanty, S. P., Nakata, K., Salakhutdinov, R., Schulman, J., Shiroshita, S., Topin, N., Ummadisingu, A., & Vinyals, O. (2021). The MineRL 2020 competition on sample efficient reinforcement learning using human priors. *CoRR*, *abs/2101.11071*. https://arxiv.org/abs/2101.11071

Hafner, D. (2022). Benchmarking the spectrum of agent capabilities. *International Conference on Learning Representations*. https://openreview.net/forum?id=1W0z96MFEoH

Hafner, D., Pasukonis, J., Ba, J., & Lillicrap, T. (2023). Mastering diverse domains through world models. *arXiv Preprint arXiv:2301.04104*.

Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., & Tassa, Y. (2016). Learning continuous control policies by stochastic value gradients. *Advances in Neural Information Processing Systems (NeurIPS)*, *29*, 2944–2952.

Milani, S., Topin, N., Houghton, B., Guss, W. H., Mohanty, S. P., Nakata, K., Vinyals, O., & Kuno, N. S. (2020). Retrospective analysis of the 2019 MineRL competition on sample efficient reinforcement learning. In H. J. Escalante & R. Hadsell (Eds.), *Proceedings of the NeurIPS 2019 competition and demonstration track* (Vol. 123, pp. 203–214). PMLR. https://proceedings.mlr.press/v123/milani20a.html

Nachum, O., Gu, S., Lee, H., & Levine, S. (2018). Data-efficient hierarchical reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, *31*, 3303–3313.

Silver, T., & Chitnis, R. (2020). PDDLGym: Gym environments from PDDL problems. *CoRR*, *abs/2002.06432*. https://arxiv.org/abs/2002.06432

Smith, D. E., Frank, J., & Cushing, W. (2007). *The ANML language*. https://api.semanticscholar.org/CorpusID:14116191

Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., & others. (2024). Gymnasium: A standard interface for reinforcement learning environments. *arXiv Preprint arXiv:2407.17032*.

Unified planning framework. (2023). In *GitHub repository*. GitHub. https://github.com/aiplan4eu/unified-planning