Crafting a Pogo Stick in Minecraft with Heuristic Search (Extended Abstract)

Yarin Benyamin[1], Argaman Mordoch[1], Shahaf Shperberg[1], Wiktor Piotrowski[2], Roni Stern[1]

[1]Ben-Gurion University of the Negev
[2]Palo Alto Research Cener, SRI
bnyamin@post.bgu.ac.il, mordocha@post.bgu.ac.il, shperbsh@bgu.ac.il, wiktor.piotrowski@sri.com, roni.stern@gmail.com

Introduction

Minecraft is a widely popular video game renowned for its intricate environment. The game's open-ended design allows the creation of unique tasks and challenges for the agents, providing a broad spectrum for researchers to experiment with different AI techniques and applications. Indeed, various Minecraft tasks have been posed as an AI challenge (Guss et al. 2019; Goss et al. 2023). Most AI research on Minecraft focused on either applying Reinforcement Learning (RL) to solve the problem (Tessler et al. 2017), learning an action model for planning (James, Rosman, and Konidaris 2020; Benyamin et al. 2023), or modeling the problem for a domain-independent planner (Roberts et al. 2017).

In this work, we focus on the combinatorial search aspect of solving the Craft Wooden Pogo task within the Polycraft World AI Lab (PAL) Minecraft environment (Goss et al. 2023).[1] PAL is an interface to Minecraft that provides an API for AI agents to interact with Minecraft's environment and send commands to the main character. PAL supports *symbolic* observations of the current state, making it ideal for planning algorithms, which require a symbolic model of the environment for problem-solving. Other Minecraft research frameworks such as MineRL (Guss et al. 2019), provide a visual, pixel-based representation of the game.

Problem Definition

In the Craft Wooden Pogo, the Minecraft agent, colloquially called Steve, is located in a field comprising $N \times N$ blocks and surrounded by unbreakable bedrock walls. The field includes multiple trees and a crafting table. Steve observes the entire map, including details like cell types, inventory contents (type and quantity of items), and its position. The available actions in PAL are: (1) teleport to another location; (2) break a tree and add the resulting log to the inventory; (3) craft 4 planks using a single log from the inventory; (4) craft 4 sticks using 2 planks from the inventory; (5) teleport to the crafting table, craft a tree tap using 5 planks and 1 stick from the inventory; (6) place the tree tap on a tree and collect a polyisoprene sack; and (7) teleport to the crafting table, and craft a wooden pogo stick using a polyisoprene sack, 2 planks, and 4 sticks from the inventory. Steve is tasked with crafting a pogo stick, which requires harvesting relevant resources and performing appropriate crafting actions, as illustrated in Figure 1: break trees; craft planks, sticks, and a tree tap; place tree tap and collect polyisoprene sacks; and craft a wooden Pogo stick. To introduce variability across problem instances, we developed a problem generator that generates initial states in which it randomly assigns (1) the agent's starting position on the map, (2) the quantity and placement of trees, (3) the items present in the agent's inventory, and (4) the crafting table's position. Thus, different sequences of actions are needed to solve different problem instances.

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1]https://github.com/PolycraftWorld/PAL

Action-Based Novelty Heuristics

The Craft Wooden Pogo is naturally modeled as a numeric planning problem and defined with PDDL2.1. For example, the preconditions of crafting a stick are to have at least 2 planks in the inventory, and the effects are to decrease 2 planks and add one stick to the agent's inventory. Most PDDL2.1 domain-independent planners are based on heuristic search. Yet, existing heuristics fail to guide the search effectively in the Craft Wooden Pogo domain on large maps. Novelty-based search algorithms (Lipovetzky and Geffner 2017), which rely only on features of the state, are also not expected to be effective in our domain, since they have no way to prefer moving to one grid cell over the other. We propose domain-independent heuristics that prioritize nodes based on the *novelty of the actions* that lead to them and the novelty of the actions they enable.

For a generated node n, we associate a tuple (g, h, a, s), where g is the cost of reaching n from the start state, h is the heuristic value of n, a is the (lifted) action used to generate n, and s is the state represented by n. The first heuristic

Figure 1: A plan to accomplish the Craft Wooden Pogo task.

we consider prioritizes nodes that enable more actions to be applied. Formally, given a node $n = (g, h, a, s)$ this heuristic is computed as $1/\text{AA}(s)$ where $AA(s)$ is the number of actions (lifted) applicable in state s. The h_{AA} heuristic is agnostic to CLOSED, depending only on the state the search node represents. The other heuristics analyze CLOSED and consider the following measures of *action novelty*:

$$\text{E-AN}(n, \text{CLOSED}) = |\{a = a' | (g', h', a', s') \in \text{CLOSED}\}|$$

$$\text{A-AN}(n, \text{CLOSED}) = \frac{1}{\sum_{a \in AA(s)} \frac{1}{|\{a=a' | (g',h',a',s') \in \text{CLOSED}\}|}}$$

The E-AN of a node n reflects how often the action leading to n was used so far to generate nodes that were expanded during the search. The A-AN of a node n is based on the number of times the actions that are applicable in n were used so far to generate nodes that were expanded during the search. Based on these definitions of Action Novelty, we introduce the following heuristic functions.

$$h_{\text{E-AN}}(n, \text{CLOSED}) = \text{E-AN}(n, \text{CLOSED})$$

$$h_{\text{A-AN}}(n, \text{CLOSED}) = \text{A-AN}(n, \text{CLOSED})$$

$$h_{\text{EA-AN}}(n, \text{CLOSED}) = \text{E-AN}(n, \text{CLOSED}) + \text{A-AN}(n, \text{CLOSED})$$

this heuristic aims to balance between the novelty of the action leading to a node and the novelty of applicable actions from that node. We explored other combinations of E-AN and A-AN, and this sum worked best in our experiments.

Experimental Results

We conducted experiments on problems in maps of size $6 \times 6, 10 \times 10, 15 \times 15, 30 \times 30$, and 45×45. For every map size, we created 50 problems, which differ in the initial content of the inventory and the number of trees in the map. We implemented our heuristics into NYX (Piotrowski and Perez 2024), a versatile, Python-based, domain-independent planner. As a baseline, we run experiments with two blind searches, namely Depth-First Search (DFS) and Breadth-First Search (BFS). Then, we run experiments with Greedy Best-First Search (GBFS) using each of our newly introduced heuristics. We also experimented with two state-of-the-art numeric planners: Metric-FF (Hoffmann 2003) and ENHSP (Scala et al. 2016). We experiment with different configurations of these planners and report only the best configuration for each baseline.

Map Size	6×6	10×10	15×15	30×30
NYX_{BFS}	118,665	2,130,514	4,231,802	-
NYX_{DFS}	65,857	348,076	1,821,880	-
NYX_{AA}	17,247	838,806	2,057,879	-
$\text{NYX}_{\text{E-AN}}$	72,100	257,726	-	-
$\text{NYX}_{\text{A-AN}}$	19,763	670,296	387,674	-
$\text{NYX}_{\text{EA-AN}}$	443	1,086	2,525	10,212

Table 1: Avg. # of expanded states for different map sizes.

Results and Trends The results of our comparison with Metric-FF and ENHSP showed that Nyx with $h_{\text{EA-AN}}$ heuristics scales better. While Metric-FF managed to solve the three smallest maps and ENHPS the four smallest maps, neither could solve any problem in the 45×45 map except for Nyx with our $h_{\text{EA-AN}}$ that solved all of them.

Additionally, Table 1 shows, as expected, the number of nodes expanded by Nyx with $h_{\text{EA-AN}}$ is significantly smaller than all other cases. Interestingly, using either $h_{\text{E-AN}}$ or $h_{\text{A-AN}}$ yields unimpressive results, while their combination thrived.

Conclusions and Future Work

We studied the Craft Wooden Pogo in Minecraft as a combinatorial search problem and proposed domain-independent action-novelty-based approaches that notably enhance planners' efficacy in solving this task, enabling scaling to large maps with numerous objects. The action-novelty heuristics we proposed are domain-independent, in the sense that they do not include any Minecraft-specific element. Future research will integrate action-novelty and goal-oriented heuristics to achieve a better balance between exploring new actions, novelty of states, and progress towards the goal.

References

Benyamin, Y.; Mordoch, A.; Shperberg, S. S.; and Stern, R. 2023. Model Learning to Solve Minecraft Tasks. In *PRL Workshop in ICAPS*.

Goss, S. A.; Steininger, R. J.; Narayanan, D.; Olivença, D. V.; Sun, Y.; Qiu, P.; Amato, J.; Voit, E. O.; Voit, W. E.; and Kildebeck, E. J. 2023. Polycraft World AI Lab (PAL): An Extensible Platform for Evaluating Artificial Intelligence Agents. *arXiv preprint arXiv:2301.11891*.

Guss, W. H.; Houghton, B.; Topin, N.; Wang, P.; Codel, C.; Veloso, M.; and Salakhutdinov, R. 2019. Minerl: A large-scale dataset of minecraft demonstrations. *arXiv preprint arXiv:1907.13440*.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *Journal of Artificial Intelligence Research*, 20: 291–341.

James, S.; Rosman, B.; and Konidaris, G. 2020. Learning Object-Centric Representations for High-Level Planning in Minecraft. In *Workshop on Object-Oriented Learning at ICML*.

Lipovetzky, N.; and Geffner, H. 2017. Best-first width search: Exploration and exploitation in classical planning. In *AAAI Conference on Artificial Intelligence*, volume 31.

Piotrowski, W.; and Perez, A. 2024. Real-World Planning with PDDL+ and Beyond. *arXiv preprint arXiv:2402.11901*.

Roberts, M.; Piotrowski, W.; Bevan, P.; Aha, D.; Fox, M.; Long, D.; and Magazzeni, D. 2017. Automated planning with goal reasoning in Minecraft. In *ICAPS workshop on Integrated Execution of Planning and Acting (IntEx)*.

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramirez, M. 2016. Interval-based relaxation for general numeric planning. In *European Conference on Artificial Intelligence (ECAI)*, 655–663.

Tessler, C.; Givony, S.; Zahavy, T.; Mankowitz, D.; and Mannor, S. 2017. A deep hierarchical approach to lifelong learning in minecraft. In *AAAI conference on artificial intelligence*, volume 31.