# Upper Confidence Bounds in Go-Explore Cell Selection

Matthew Hattrup
mjh4dbm@virginia.edu
University of Virginia
Charlottesville, Virginia, USA

Pawan Jayakumar
pj8wfq@virginia.edu
University of Virginia
Charlottesville, Virginia, USA

## Abstract

We investigate the use of Upper Confidence Bound (UCB) cell selection in the novel exploration algorithm go-explore. Go-explore is a powerful method for exploration and learning in complex environments, but one of its limitations is its probabilistic cell selection mechanism. We show that by incorporating UCB into the cell selection process, go-explore more frequently selects promising cells, leading to improved performance and more efficient exploration of the game environment. We evaluate our approach on a relatively simple labyrinth game, and demonstrate significant improvements over the original go-explore algorithm. Our results suggest that UCB can be a valuable addition to go-explore, helping it to more effectively learn and adapt in complex and challenging environments.

***Keywords:*** go-explore, bandit algorithms, hard exploration, Atari, reinforcement learning

## 1 Introduction

Go-explore is a state of the art algorithm for game AI that is especially effective in environments with sparse or deceptive rewards [4]. Go-explore can solve every level of Montezuma's Revenge, the first and only algorithm to do so as of the time of its publication [4]. This demonstrates the exceptional capabilities of go-explore in complex and challenging environments.

Previous approaches to dealing with sparse reward environments used intrinsic motivation (IM), which provides rewards to the learning agent for discovering new states [3]. However, this approach has the drawback of causing reward detachment. Consider an agent exploring a region of state-space. At some point, its stochastic policy may lead it to explore a different area of state-space. As the agent learns to perform well in this new area, it may forget to return to the previous states due to catastrophic forgetting [4, 5]. Without the intrinsic rewards provided by those states, the agent may not be motivated to return to them. As a result, states that are only reachable from the previous states become "detached" from the agent's intrinsic reward frontier.

Go-explore avoids this by keeping an archive of all previously visited cells (a compressed representation of the environment state). At each iteration, go-explore samples cells from the archive for the agent to explore from. Exploration can be done with any method, and cells encountered during exploration are then added to or updated in archive.

One of the ways go-explore can be improved is its method of cell selection. Originally, go-explore computed a positive score for each cell using a heuristic and then normalized each score so that it became the non-zero probability of selecting the cell.

$$p_i = \frac{CellScore(i)}{\sum_j CellScore(j)} \tag{1}$$

Where $p_i$ is the probability of selecting cell $i$ in the archive. As this archive grows larger, the probability of selecting any particular cell decreases. If there are only a few promising cells in archive, the weighted sampling method will rarely pick them, resulting in slower exploration.

We hypothesize that the number of unpromising cells added to the archive will typically grow much faster than the number of promising cells, which motivates us to find a cell selection strategy that will pick promising cells more frequently. For simplicity, we consider a modified UCB algorithm for cell selection [2].

Note that cell scores are not a constant feature in go-explore. New cells are frequently added to the archive, and their heuristic scores change with time. Thus, a once promising cell can become unpromising after further exploration. Because of this, we modify the UCB algorithm to only consider the most recent cell scores and not their averages as seen in Equation 2.

## 2 Related Work

Our code is forked from Adeikalam's go-explore repository which in turn implements the original algorithm and heuristics of go-explore paper, but in a simple labyrinth environment [1, 4]. Our only significant change being to the cell selection algorithm.

In reality, the distribution of cell scores gradually changes as our agent explores the environment. Incremental and non-stationary multi-arm bandit algorithms such as sliding window UCB and limited memory deterministic sequencing exploration and exploitation (LM-DSEE), may do better our naive UCB [6].

## 3 Method

Three different methods for selecting cells were explored: uniform sampling (control), weighted sampling (the method proposed in the original go-explore paper), and UCB, the latter two of which make use of a heuristic described in the go-explore paper[4].

Equation 1 shows how the original paper selects cells where $p_i$ is the probability of picking cell $i$.[4]

$$UCB(i) = Cellscore(i) + \alpha\sqrt{\frac{log(t)}{Trials(i)}} \quad (2)$$

Equation 2 shows how the UCB score for cell $i$ is calculated. $\alpha$ is a hyper-parameter which controls how much to weigh the UCB term, $t$ is the number of cell selections that have happened so far, and $Trials(i)$ is the number of times cell $i$ has been selected. The cell with the highest UCB score is then selected.

## 4 Experiment

The three cell selection algorithms were tested in a simple labyrinth environment. Figure 1 shows the environment in three different sizes.

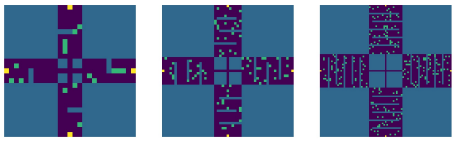

**Figure 1.** Simple maze environment with 3 different sizes: 11x 5, 21x11, 31,21 from left to right

The hyper-parameter $\alpha$ was chosen by running our UCB cell selection with three different values of $\alpha$: 0.01, 0.1, and 0.6. For each value, 10 trials were run on the medium size map. The results can be seen in Figure 2. $\alpha = 0.6$ showed the best results as it converged with the lowest variance. Attenuating the UCB, we can control the "greed" of cell selection. Too greedy (i.e. small $\alpha$), and go-explore's search space is too narrow. This leads to slower convergence as seen in Figure 2.

The evaluations for UCB selection, weighted sampling, and uniformly random sampling were tested on the large size map. Each test was run three times and the average score and variance was plotted. As shown in the figures, go-explore with UCB selection was able to converge to the optimal score the fastest of the three at around 12500 iterations compared to 20,000+ for the other two. The difference between go-explore with the weighted selection method versus the uniformly random selection method appear insignificant. We hypothesize that since the archives grows very large, the weighted sampling picks promising cells rarely, which results in similar performance to go-explore with uniform cell selection.

## 5 Limitations and Future work

One limitation this project faced was limited time and compute power. The original go-explore paper is able to solve

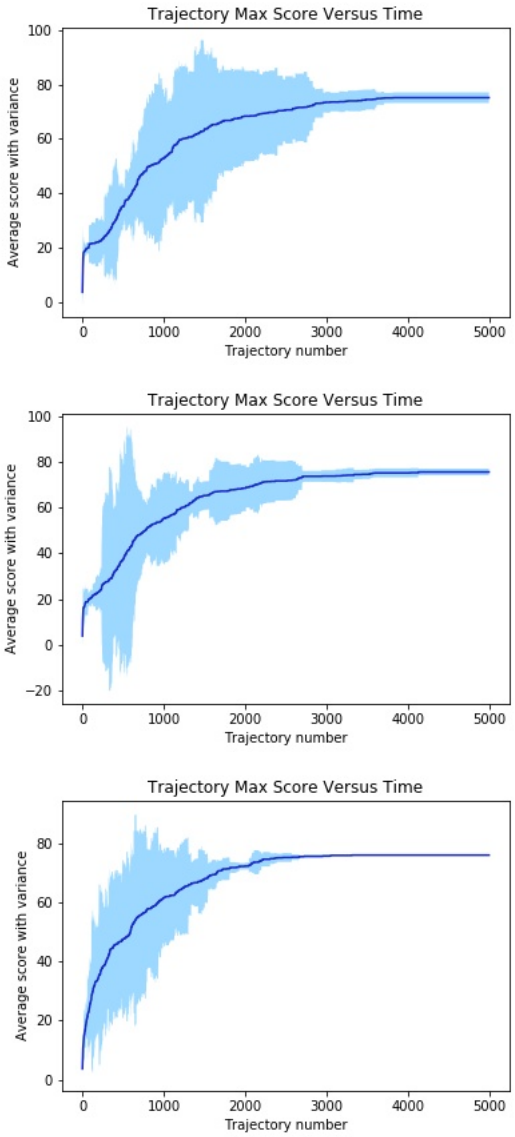


**Figure 2.** Score history of cell selection using UCB with $\alpha = 0.01$, $\alpha = 0.1$, $\alpha = 0.6$ from top to bottom

Montezuma's Revenge and Pitfall. We initially tried to also test on these environments, but even with UVA GPU servers it took too long to train. In one hour, a quad GPU server was able to compute 1 million training iterations while it requires 1.2 billion to fully train. This pace of testing was not feasible given our time constraints. In our testing, we noticed that the difference between UCB and the control was only visible on the large map. Future work could explore how large this difference is on an even harder environment like Montezuma's Revenge.

While this paper explored UCB, any cell selection algorithm which ensures all cells have a non-zero probability to be selected can work. We theorize that a selection algorithm

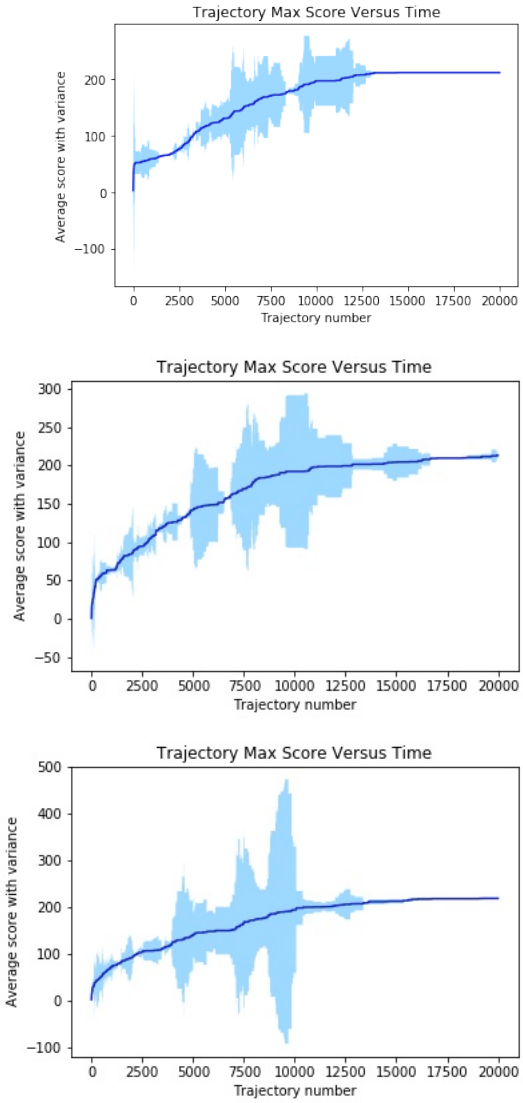


**Figure 3.** Score history of cell selection using UCB with $\alpha = 0.6$, the paper's method, and the control (uniform sampling) from top to bottom

which takes into account the gradual change in the cell score distributions will do better [6].

The paper computes cell scores using a linear combination of cell features such as the cell's average reward and the number of neighbors the cell has. Future work can investigate the relative importance among these features and whether other features should be included as well.

In conclusion, this paper shows that intelligent cell selection methods such as UCB can improve the performance of the go-explore algorithm by almost two-fold because it more frequently selects promising cells. By selecting the cell with the highest UCB score rather than sampling, it also performs well even as the archive size increases.

## References

[1] Pierre Adeikalam, Guangye Chen, Kevin Xu, and Erwan Le Pennec. 202. Go-Explore: A New Approach for Hard-Exploration Problems. https://github.com/Adeikalam/Go-Explore.

[2] Peter Auer, Nicoló Cesa-Bianchi, and Paul Fischer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. Machine Learning, Vol. 47. Springer, 235–256. https://doi.org/10.1023/A:1013689704352

[3] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying Count-Based Exploration and Intrinsic Motivation. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/afda332245e2af431fb7b672a68b659d-Paper.pdf

[4] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. 2019. Go-Explore: a New Approach for Hard-Exploration Problems. https://doi.org/10.48550/ARXIV.1901.10995

[5] Michael McCloskey and Neal J. Cohen. 1989. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. Psychology of Learning and Motivation, Vol. 24. Academic Press, 109–165. https://doi.org/10.1016/S0079-7421(08)60536-8

[6] Lai Wei and Vaibhav Srivastava. 2018. On Abruptly-Changing and Slowly-Varying Multiarmed Bandit Problems. https://doi.org/10.48550/ARXIV.1802.08380