Intelligent Go-Explore: Standing on the Shoulders of Giant Foundation Models

Cong Lu[1,2]
conglu@cs.ubc.ca

Shengran Hu[1,2]
srhu@cs.ubc.ca

Jeff Clune[1,2,3]
jclune@gmail.com

[1]University of British Columbia
[2]Vector Institute
[3]Canada CIFAR AI Chair

Abstract

Go-Explore is a powerful family of algorithms designed to solve hard-exploration problems built on the principle of archiving discovered states, and iteratively returning to and exploring from the most promising states. This approach has led to superhuman performance across a wide variety of challenging problems including Atari games and robotic control, but requires manually designing heuristics to guide exploration (i.e. determine which states to save and explore from, and what actions to consider next), which is time-consuming and infeasible in general. To resolve this, we propose INTELLIGENT GO-EXPLORE (IGE) which greatly extends the scope of the original Go-Explore by replacing these handcrafted heuristics with the intelligence and internalized human notions of interestingness captured by giant pretrained foundation models (FMs). This provides IGE with a human-like ability to instinctively identify how interesting or promising any new state is (e.g. discovering new objects, locations, or behaviors), even in complex environments where heuristics are hard to define. Moreover, IGE offers the exciting and previously impossible opportunity to *recognize and capitalize on serendipitous discoveries that cannot be predicted ahead of time*. We evaluate our algorithm on a diverse range of language-based tasks that require search and exploration. In Game of 24, a problem testing multistep mathematical reasoning, IGE reaches 100% success rate 70.8% faster than the best classic graph search baseline. Next, in BabyAI-Text, a challenging partially observable gridworld where an agent has to follow language instructions, IGE exceeds the previous state-of-the-art with orders of magnitude fewer online samples. Finally, in TextWorld, a rich text game, we show the unique ability of IGE to succeed in settings requiring long-horizon exploration where prior state-of-the-art FM agents like Reflexion completely fail. Overall, INTELLIGENT GO-EXPLORE combines the tremendous strengths of FMs and the powerful Go-Explore algorithm, opening up a new frontier of research into creating more generally capable agents with impressive exploration capabilities. All our code is open-sourced at: `https://github.com/conglu1997/intelligent-go-explore`.

1 Introduction

Foundation models (FMs, [5, 7, 32, 37, 39]) trained on giant internet-scale datasets have demonstrated strong general capabilities in reasoning [41] and understanding [9]. As such, these models have been increasingly employed as autonomous agents [4, 28, 34, 40, 43, 44] in decision-making tasks, showcasing the ability to adapt in-context [12, 31] to unseen tasks. However, a significant challenge remains: foundation model agents often struggle in environments that require deep exploration over extended time horizons [28]. Overcoming this limitation would enable us to realize their potential

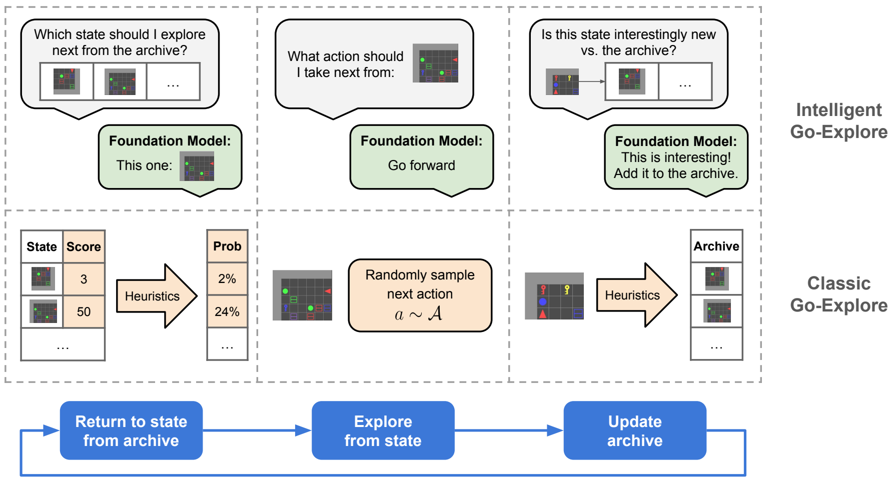
Which state should I explore next from the archive?
…
Foundation Model: This one:
What action should I take next from:
Foundation Model: Go forward
Is this state interestingly new vs. the archive?
…
Foundation Model: This is interesting! Add it to the archive.
Intelligent Go-Explore
State
Score
3
50
…
Heuristics
Prob
2%
24%
…
Randomly sample next action $a \sim \mathcal{A}$
Heuristics
Archive
…
Classic Go-Explore
Return to state from archive
Explore from state
Update archive

Figure 1: INTELLIGENT GO-EXPLORE (IGE) integrates the intelligence and internalized human notions of interestingness from giant pretrained FMs into all stages of the Go-Explore [13, 14] algorithm, enabling FM agents to robustly explore in complex environments. **Bottom:** Classic Go-Explore solved hard exploration problems by archiving novel discovered states, resetting to promising ones via domain-specific heuristics, and then performing random exploration. **Top:** Our approach, INTELLIGENT GO-EXPLORE, enables Go-Explore to tackle virtually any type of problem that is representable in the context of a large language or multimodal model. Instead of manually defining heuristics, we query the foundation model at all stages, enabling our approach to automatically catch and return to serendipitous discoveries, and harness the power of FM agents to explore. The environment shown is the BabyAI-Text game used in Section 4.2.

as autonomous assistants in more open-ended domains like scientific discovery and innovation [20]. This paper introduces INTELLIGENT GO-EXPLORE (IGE), a novel approach that combines the intelligence foundation models with the powerful Go-Explore [13, 14] framework to substantially increase the exploration capabilities of FM and reinforcement learning (RL, [35]) agents.

Go-Explore is a popular family of algorithms in deep RL based on maintaining an archive of "interestingly new" discovered states and then iteratively returning to and exploring from the most promising states (see Figure 1 for an overview of the three stages). This framework has led to superhuman performance in a range of hard-exploration problems, including long-horizon Atari games and robotic control. However, the algorithm's success *largely relies on carefully hand-designed heuristics* at all three stages to guide exploration [29, 30]. For example, in Montezuma's Revenge [2], an Atari game that was the previous grand challenge of exploration in deep RL, **(1)** saved states in the archive were returned to with probability proportional to factors like the number of times a state has been sampled before, **(2)** exploration was purely via random action sampling, and **(3)** the criteria for which states were considered interestingly new enough to be added to the archive depended on domain-specific factors like whether the agent visited a new location, or did so with more keys. Without this pre-specified knowledge, the quality of discovered trajectories is typically significantly worse [14].

These rigid, domain-specific choices are in stark contrast to human-like exploration of a new game, where players can often intuitively judge the value or interestingness of any particular state [10]. More importantly, *it is often impossible to know what is interesting or possible ahead of time* in complex domains. In the words of Isaac Asimov—*The most exciting phrase to hear in science, the one that heralds new discoveries, is not "Eureka!" but "That's funny."*. With this motivation, IGE stands on the shoulders of giant foundation models, and uses their intelligence to **(1)** act as a judge to identify the most promising states to return to and explore from, **(2)** to select the best actions to take to explore from a selected state, and **(3)** to identify serendipitous discoveries when they happen (e.g. finding new objects, locations, or other novelties) and decide whether a new state is interestingly new enough to be added to the archive as a stepping stone for future exploration (Figure 1, top).

We demonstrate IGE's ability to reliably improve the exploration capabilities of FM agents on a diverse range of language-based tasks that require search and exploration. These settings include tasks that require *commonsense reasoning, long-term planning and memory, and handling partial observability*. IGE integrates well with various agent strategies, including few-shot and chain-of-thought-based prompting; and *will only get better* as the capabilities of foundation models improve further. While IGE performs strongly all-around, some highlights from our evaluation include: IGE reaches 100% success rate on Game of 24 [43], a standard mathematical reasoning and search problem, 70.8% faster than classic graph search. Moreover, on the TextWorld [11] Coin Collector domain, IGE is the only algorithm that succeeds in discovering long-horizon optimal solution paths, where prior state-of-the-art FM agent frameworks like Reflexion [34] fail.

INTELLIGENT GO-EXPLORE simultaneously empowers foundation model agents to *reliably explore*, and reimagines the scope of Go-Explore to tackle virtually any type of problem, without being limited to hand-designed heuristics. These abilities will substantially improve our ability to develop more generally capable agents, and increase the range of tasks they can learn how to solve.

2 Background

2.1 Go-Explore for Hard-Exploration Problems

Go-Explore [13, 14] is a family of algorithms designed to solve hard-exploration [24] problems based on the principle of remembering and returning reliably to promising states. The classic setting builds an "archive" of novel states it discovers in an environment, where similar states are grouped in a single "cell". These cells are defined by heuristics like having the same visual observation when downsampled to low resolution. In the beginning, the archive only contains the initial state. We describe the overall structure of the algorithm in the same order as Figure 1 (bottom): At each iteration, **(1)** promising states are selected from the archive through domain-specific heuristics, e.g. probabilistically sampling states proportional to their progress through the environment or potential to lead to new states. The agent returns to that state, by resetting using the simulator or via a goal-conditioned policy, and **(2)** a sequence of random actions is taken to explore from that state. **(3)** All discovered states deemed interestingly new by the cell representation heuristics are added to the archive, and the process repeats. The strength of Go-Explore is due to addressing two critical impediments to exploration: forgetting how to reach previously visited states (detachment) and failing to first return to a state before exploring from it (derailment) [13].

This approach leads to a collection of high-return trajectories being discovered, which may then be fed into an imitation learning [19] algorithm to produce a policy that generalizes and is robust to stochasticity. We adopt similar assumptions as the original setting, by assuming an agent can return to a previously discovered state by restoring in the simulator. This assumption may readily be relaxed by training a policy to return to a given state, or in the foundation model case, by simply prompting the model with a past trajectory.

2.2 Large Language and Multimodal Foundation Models

The combination of model scaling and training over internet-scale data has resulted in a wide variety of foundation models [5] that exhibit generalist capabilities. In this paper, we consider autoregressive large language models (LLMs, [7, 32, 39]) which learn to generate text completions by modeling the conditional probability of a new token given the preceding tokens, $p(x_t|x_{<t};\theta)$. This framework enables LLMs to not only generate coherent text but crucially also exhibit human-like abilities, including on commonsense knowledge questions [36] and complex reasoning tasks [41]. These models may also be extended to other input modalities such as images by tokenizing these inputs into the same space as the text [49]. When prompting an FM with an instruction, the user may decide to do so with no related examples (zero-shot), with a few successful examples in related problems (few-shot, [7]), or ask for a chain of reasoning (chain-of-thought, [41]) before responding.

3 Driving Exploration with Giant Foundation Models

In this section, we propose INTELLIGENT GO-EXPLORE (IGE) which reimagines the classic Go-Explore algorithm as described in Section 2 with the intelligence of giant pretrained foundation models. Specifically, we introduce FM intelligence to selecting which archived state to return to and explore from, which action to take from each state, and deciding whether a state is interestingly new and should be archived. IGE's use of foundation models is closely related to FM-as-a-judge [48], which shows that foundation models are a good proxy for human judgment to evaluate the output of generative models. Here, instead of judging synthetic output, the foundation model makes choices to determine the best way to explore an environment. We illustrate our resultant algorithm at the top of Figure 1 and provide full pseudocode in Algorithm 1.

Wherever we query the foundation model, we introduce the overall strategy of Go-Explore alongside a brief description of the current environment in the "system message" (high-level directive) displayed below. The brief descriptions for each environment we evaluate on in Section 4 are listed in Appendix B. In the following sections, we detail our prompting techniques at each stage of IGE. The previous prompt history is visible to the agent, which enables each component of IGE to communicate with each other. We provide precise details on how we parse responses in Appendix C.1.

System Prompt.

[Brief Description Of Environment]
You will be prompted to perform systematic exploration in the style of Go-Explore. An archive will be maintained of interesting states found. You will be prompted to:
- Select a state from the archive that is the most promising, i.e., likely to lead to a solution or more novel states.
- Explore from states intelligently, by picking new actions.
- For each new state, determine if the state is interestingly new and should be added to the archive.

3.1 Select State From Archive

The power to easily store and return to promising discovered states is crucial to Go-Explore's ability to reliably solve long-horizon exploration problems. IGE leverages the foundation model's internalized notions of interestingness [45] to select the most promising state to return to from the archive (Figure 1, left). This is far more flexible than classic Go-Explore, which relied on hardcoded hand-crafted heuristics to determine cell sampling probabilities. An example prompt is shown below.

Examples of the discovered states are given in Table 1. We assign indices to these states in a list and ask the FM to select a numerical index. We define a budget of N_{state} "state-expansions". Each state expansion is followed by a sequence of exploratory actions, which we describe in the next section.

State Selection Prompt.

Current state archive:
[Discovered states]

Select the most promising state.

3.2 Explore From State

In order to effectively explore from a state selected in the previous section, we leverage the power of foundation model agents [18, 28] to choose how to act in an environment. This vastly improves on the original Go-Explore's use of random action sampling. One of the key strengths of IGE is that it is **a strict improvement** on top of any FM agent framework, *including zero-shot, few-shot, or even chain-of-thought-based prompting* [44]. We demonstrate this flexibility in Section 4.

One point of departure from the classic Go-Explore is that we additionally maintain a state-conditional action history for each archived state, so that IGE can avoid repeating previously tested options. While this information may already be available in the entire history, this helps *avoid any recency bias that can occur with longer contexts* [46]. The action history can be easily reiterated in the prompt, or the prompt could display the remaining untested actions. We define a budget of exploratory actions per state expansion N_{action}, which is typically far shorter than the full horizon of the environment and represents a small number of trial actions. An example prompt is shown here.

Action Selection Prompt.

[Agent-Specific Prompt]
Current state:
[Current State]
Previously tried actions:
[Previous Action History]

Output the next action.

3.3 Update Archive

IGE queries the foundation model to judge whether any newly discovered state is interestingly new and sufficiently different from prior states to qualify to be added to the archive. Intuitively, we should only save the most relevant stepping stones, and discard those that are unlikely to lead to new discoveries. Whilst the original Go-Explore required extensive domain knowledge to determine interestingness, IGE avoids this requirement and manual labor, critically gaining the ability to

Table 1: We show that INTELLIGENT GO-EXPLORE can efficiently explore over a diverse set of environments with increasing difficulty. The problems we consider all use text-based observations, but in general, IGE may be extended to virtually any environment that can be embedded in the context of a multimodal foundation model. For each environment, we provide an example observation, samples from the action space, and the horizon of the task in the environment.

	Game of 24	**BabyAI-Text**	**TextWorld**
Problem Type	mathematical reasoning and search	partially observable gridworld with language instructions	partially observable game requiring long-term memory and planning, exploration, and common sense
Text Observation	"Current state: (2 8 8 14)"	"Goal: unlock the red door. You see a wall 4 steps forward, You see a yellow box 2 steps left."	"You arrive in a pantry... You see a shelf. The shelf is wooden. On the shelf you can see flour..."
Next Actions	- 2 + 8 = 10 Next: (8 10 14) - 8 / 2 = 4 Next: (4 8 14) - 14 + 2 = 16 Next: (8 8 16) ...	- turn left - turn right - go forward ...	- go east - cook potato with oven - unlock door with key ...
Task Horizon	3	64 or 128	25, 40 or 80

recognize serendipitous discoveries that could not have been predicted ahead of time. In practice, we propose two options to filter discovered states after a sequence of exploratory actions. The first is to iterate through every new state and ask whether each one is interestingly new and should be added to the archive. The second is to first add all states, and then ask the foundation model to remove the uninteresting states. We discuss this choice later in Section 4.3; the second form is preferable in larger environments where there is more need to explicitly deprecate earlier discoveries that have become irrelevant to not overload the archive. An example prompt for the first option is shown below.

By default, IGE implements the foundation model at all three stages of Go-Explore, but we rigorously analyze the relative importance of each component in Section 5. In this paper, we focus on the discovery of solutions to hard-exploration problems. However, these solutions could easily then subsequently be used for downstream reinforcement learning or even improve the foundation model in the next task by in-context learning—thus allowing an agent to bootstrap its own learning indefinitely.

Archive Filtering Prompt.

Current state archive: [State Archive]
New state: [Current State]

Is this state interestingly new (relevant to the task or could lead to further stepping stones, and not close to the existing states) and should be added to the archive?

4 Empirical Evaluation

In this section, we evaluate INTELLIGENT GO-EXPLORE across a diverse set of text environments that require search and exploration. We demonstrate IGE's ability to handle partially observable and complex observation spaces, discover solutions involving long chains of actions, and effectively improve the ability of FM agents to explore. For all our experiments, we use GPT-4 [32], one of the current SOTA LLMs, as our foundation model. We compare IGE to random action sampling, a naïve LLM baseline, and two SOTA FM agents, ReAct [44] and Reflexion [34]. All methods use the same amount of environment steps and receive the same observations for a fair comparison. Naïve LLM simply queries the LLM for an action conditional on the interaction history. ReAct prompts the agent to output its reasoning before making a decision. Based on ReAct, Reflexion further conditions the agent on the previous attempted episode, asking the agent to learn from its mistakes. We provide an overview of our environments in Table 1. Full hyperparameters are detailed in Appendix D.

4.1 Game Of 24

We first demonstrate the effectiveness of IGE in a mathematical reasoning task, Game of 24 [43]. The goal is to perform basic arithmetic operations $(+, -, \times, /)$ starting from 4 numbers to obtain 24. For example, given input $(4, 9, 10, 13)$, a possible solution could be $(10 - 4) \times (13 - 9) = 24$. We formulate the problem as an MDP [35], where actions represent a reduction of two numbers by an arithmetic operation—i.e., the above solution would be represented as the sequence of state transitions $(4, 9, 10, 13) \xrightarrow{10-4=6} (6, 9, 13) \xrightarrow{13-9=4} (6, 4) \xrightarrow{6\times4=24} (24)$. Therefore, IGE uses the

FM to iteratively expand possible solution paths and archive promising ones to return to. The action space is the range of possible next operations, displayed in the same manner as in Yao et al. [43].

We evaluate IGE across 100 hard test problems in Figure 2, and additionally include the standard (unweighted) graph search algorithms depth-first search (DFS) and breadth-first search (BFS) as reference. Since the combinatorial complexity of the problem is at most $\binom{4}{2} \cdot \binom{3}{2} \cdot 4^3 = 1152$, graph search is guaranteed to find a solution within that many actions. The system prompts for both IGE and the LLM baselines contain *few-shot examples* with correct calculations on different starting numbers. IGE rapidly reaches 100% success rate, on average 70.8% quicker than the next best baseline, depth-first search (DFS)— this improvement is statistically significant (χ^2 test, $p < 0.05$) at 150 operations, where IGE has solved all problems. This success may be attributed to the fact that language models have internalized mathematical intuition and are likely to be able to identify promising pairs like $(6, 4)$ that could easily be multiplied together for a solution.

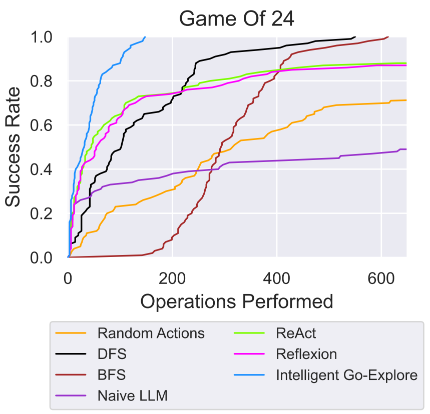

Figure 2: IGE explores the Game of 24 with the intelligence of FMs and reaches 100% success rate on average 70.8% quicker than DFS, the next best baseline. IGE completes all problems within 150 environment operations. Our use of archiving and intelligent action selection allows us to greatly outperform prior LLM agents with an equal number of operations performed. The success rate is computed over 100 test problems.

All LLM agent baselines (naïve LLM, ReAct, Reflexion) eventually plateau and even get beaten by the unintelligent DFS. This highlights the need for diverse action selection, which IGE enables. A final point of comparison we make is to Tree of Thoughts (ToT, [43]) which achieved 74% on Game of 24 within their evaluation budget. We emphasize that our evaluation setting is very different as IGE selects from the list of valid options rather than doing the math in context. However, we note the key difference to our method is that ToT evaluates and expands multiple reasoning paths following a tree structure, whereas IGE can easily jump around the search space—this is a crucial advantage in more complex environments (like those in the following sections), where it takes many coordinated actions to get from one state to another interesting state.

4.2 BabyAI-Text

Next, we show that IGE scales to the BabyAI-Text domain from Carta et al. [8], which is a procedurally-generated partially-observable 2D gridworld with text-based observations. The agent is given a textual goal instruction which could correspond to one or more instructions in a sequence, e.g. "pick up X and then go to Y". As we can see from the observations in Table 1, the task is challenging even for humans to complete and requires forming a model of the world from partial text descriptions. This kind of state observation would make it *hard to define heuristics to determine how good any particular state is*, as in classic Go-Explore. The optimal path to a solution may include moving blocking objects as well as finding keys to open doors. We consider 5 different task families of increasing difficulty: "go to", "pick up", 'pick up then go to", "open door", and "put next to", which are described fully in Appendix B.2.

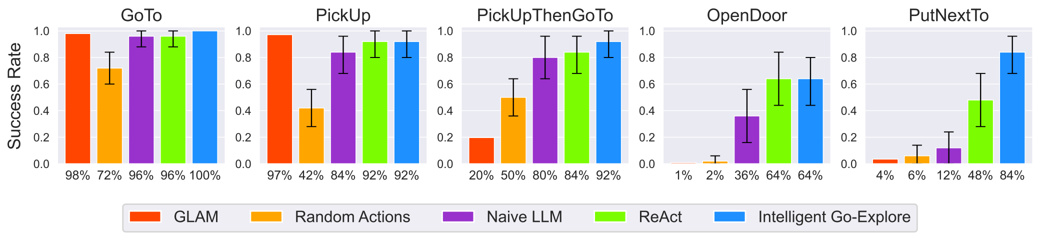

Figure 3: IGE can find solutions to challenging tasks in the BabyAI-Text environment more effectively and with orders of magnitude fewer online steps than prior RL-trained baselines (GLAM, [8]). Task types are in order of difficulty. As tasks become more difficult, the performance gap of IGE v.s. the LLM baselines grows. We show the mean and 95% bootstrap confidence interval [50] over 25 seeds per environment type. *Here, and elsewhere, confidence intervals are obtained by bootstrapped resampling 10,000 times.*

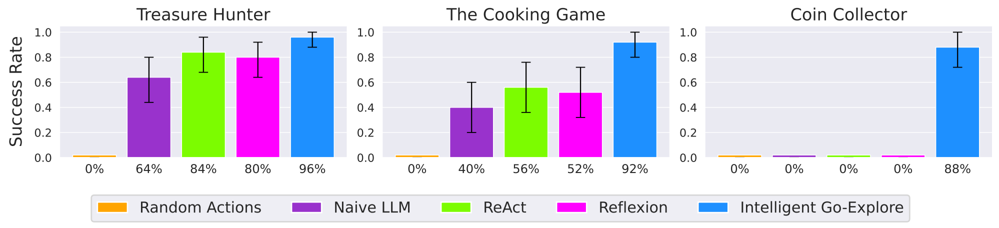

Figure 4: IGE outperforms state-of-the-art FM agents in three challenging text games in TextWorld. These results illustrate the powerful capabilities of planning, commonsense reasoning, and exploration of IGE (Section 4.3). Notably, in the Coin Collector game where hard exploration is required, we observe BFS-like search behavior emerge in IGE, enabling it to find the most efficient solution where all other approaches exhaust the environment horizon. We show the mean and 95% bootstrap confidence interval over 25 seeds for each game.

We omit the Reflexion baseline in this environment due to the high cost of querying GPT-4 with 128-step episodes in the context. Due to the complexity of this environment, we use *chain-of-thought prompting in all three components* of IGE. This allows the FM to deliberate on the state of the game before making decisions. We show that IGE can find solutions to these problems with only a tiny budget of 250 environment steps per task (divided into rollouts of 10 exploratory actions each) and visualize the final performance in Figure 3. IGE and ReAct vastly outperform the prior RL-trained language model approach, GLAM [8], with orders of magnitude fewer samples (GLAM used 1.5M online steps) and requiring no training whatsoever. IGE achieves the best or close to the best performance in every task. The gap between IGE and the second-best method grows with task difficulty, with a statistically significant 36% improvement (χ^2 test, $p < 0.05$) on ‘put next to’.

4.3 TextWorld

Finally, we show IGE’s ability to tackle tasks requiring long-horizon memory and planning, exploration, and commonsense in TextWorld [11], a classic text-based agent benchmark. We consider three challenging games in Textworld: Treasure Hunter, The Cooking Game, and Coin Collector. In each game, the agent needs to complete the task while navigating a maze of different rooms, while only seeing the current room’s description in text. The agent interacts with the world using free-form natural language commands, such as “go east” or “cook potato with oven.” In Treasure Hunter [33], the agent has to find a specific item by exploring, finding keys, and unlocking doors and containers. In The Cooking Game, the agent must find a recipe, locate and process (e.g., dice, cut, chop) ingredients, and cook them according to the recipe using various kitchen appliances (e.g., oven, pan). In Coin Collector, the agent must find a coin randomly located in the maze, testing its navigation and exploration skills. We set each game to hard difficulty, details on game customizations are provided in Appendix B.3. As in the previous section, we use chain-of-thought prompting in all three components of IGE. Because the state archive in this environment grows significantly, we implement rejection-based archive filtering, which we describe in Appendix C.2.

We present success rates achieved on the three games using IGE and the baselines in Figure 4. We observe that IGE outperforms all other baselines, with a statistically significant (χ^2 test, $p < 0.05$) performance gap between IGE and the second-best method in the harder Cooking Game and Coin Collector. In The Cooking Game, IGE outperforms the second-best agent, ReAct, by a large margin of 36%, demonstrating IGE’s advantage in hard-exploration problems. In Coin Collector, IGE *is the only method that can find the solution in the maze*, with all other methods completely failing. Interestingly, we observe that IGE exhibits BFS-like behavior, and intelligently selects rooms with unexplored directions and iteratively removes rooms with exhausted directions. This results in IGE almost always finding the shortest path to the target, while other methods fail to navigate the maze.

We highlight that Reflexion does not improve over ReAct in all the games we tested. Although Reflexion should in theory be an improvement over ReAct with the experience from previous attempts, it tends to decrease performance. We hypothesize that in long-horizon environments, the history becomes too long after the initial episode, and prevents Reflexion from effectively utilizing knowledge from the previous episode. In contrast, IGE uses the FM to iteratively filter interesting states in the archive, which ends up *controlling the context length*. This helps IGE truly make use of the cumulative knowledge gained through exploration.

Table 2: We rigorously ablate the design choices in IGE and analyze the importance of intelligent selection at each stage. Game of 24 performance is taken at 150 environment steps, over 100 evaluation seeds. BabyAI-Text performance is taken at 250 environment steps, over 25 evaluation seeds. TextWorld performance is taken at 240 environment steps, over 25 evaluation seeds. 'Standard' is mirrored from Section 4. We show the mean and 95% bootstrap confidence interval for the success rate.

Variant of IGE		**Success Rate (%)**	
	Game of 24	**BabyAI (PN)**	**TextWorld (CG)**
Standard	**100 ± 0.0**	**84 ± 14**	**92 ± 10**
✗ Intelligent action selection	68 ± 9.0	24 ± 16	0 ± 0
✗ Intelligent state selection	96 ± 3.5	48 ± 20	76 ± 16
✗ Intelligent archive filtering	93 ± 5.0	64 ± 20	64 ± 20
✗ All 3 above	61 ± 9.5	4 ± 6	0 ± 0
✗ State-conditional action history	33 ± 9.0	72 ± 16	72 ± 16

5 Analysis

In this section, we analyze (1) the importance of FM intelligence for each of the three key components of Go-Explore, (2) how IGE's selectivity produces a smaller (and thus more efficient) archive, and (3) how IGE's performance improves as the FM's size/intelligence increases. We take a representative sample of environments from the previous section of Game of 24, Put Next To (PN) from BabyAI-Text, and The Cooking Game (CG) from TextWorld. Hyperparameters are listed in Appendix D.

How Important is Foundation Model Intelligence at Each Step? First, we analyze the impact of FM intelligence on each component of INTELLIGENT GO-EXPLORE. We ablate replacing state and action selection with uniform random sampling, archive filtering with saving everything to the archive, and not maintaining a state-conditional action history. We use these unintelligent choices, as *it would be very time-consuming to attempt to design the right heuristics* based on the rich text observations in Table 1. In Table 2, we observe that where the intelligence of FMs is more valuable varies by environment. Since the environment horizon is only 3 in the Game of 24, the most important factor is ensuring that the actions tried are diverse and intelligently selected. This hypothesis is confirmed: the largest performance drops occur when removing either FM action selection or the action history. Different IGE components are most helpful in both of the longer-horizon BabyAI-Text and TextWorld environments: intelligent state selection and archive filtering make a big impact, showcasing the strength of enabling IGE to return to promising discovered states. There are smaller performance drops when removing the action history; likely because in larger environments, many more unique states are discovered, so there is less gain from preventing taking the same actions from frequently returned to states. In both environments, we also observe a drastic decrease when switching to random actions, as in classic Go-Explore. This underscores the substantial benefits IGE provides in harnessing FMs for action selection.

Finally, we elucidate the need for intelligent archive filtering across all our environments. Not only does archive filtering improve performance, but it also drastically cuts down the number of uninteresting states in the archive, as shown in Table 3 (left). As we use rejection-based archive filtering on TextWorld, we quote the average size of the archive throughout each episode. In BabyAI-Text, we observe the archive becoming around 8× larger without filtering. These metrics demonstrate IGE's innate ability to capture promising discoveries as they occur and focus attention on them, without the need for any manual heuristics.

Table 3: **Left:** We show that intelligent filtering in IGE can drastically reduce the size of the archive, and help the algorithm focus on the most interesting states. Without intelligent filtering in BabyAI-Text and TextWorld, the archive becomes over 5× larger. **Right:** We show that IGE scales with the capabilities of the underlying foundation model. We follow the same settings with Table 2 and show the mean and 95% bootstrap confidence interval for the number of states and success rate.

Archive Filtering		**Number of States**	
	Game of 24	**BabyAI (PN)**	**TextWorld (CG)**
No Filter	18.5 ± 3.2	203.5 ± 56.7	22.4 ± 15.3
With Filter	**15.6 ± 2.3**	**25.5 ± 5.2**	**4.4 ± 2.8**

Foundation Model		**Success Rate (%)**	
	Game of 24	**BabyAI (PN)**	**TextWorld (CG)**
GPT-4	**100 ± 0**	**84 ± 14**	**92 ± 10**
GPT-3.5	57 ± 10	0 ± 0	0 ± 0

What is the Effect of Foundation Model Choice? We also analyze the dependence of our algorithm on the strength of the foundation model by replacing GPT-4 with an earlier variant, GPT-3.5 in

Table 3 (right). There is a considerable difference between the two, which suggests that our environments are non-trivial to solve, and that future advancements in foundation models are likely to readily scale the performance of IGE to even harder problems.

6 Related Work

FM-as-judge. We employ FM guidance at all stages of IGE to drive exploration. FMs as judges [6, 47] have already seen use in decision-making tasks: OMNI [45] considers FM guidance in multi-task settings to select the most promising next task to train on. However, focusing on the broader task could miss out on interesting behavior that happens at a more granular level, and thus IGE greatly expands on the integration of FM intelligence into decision-making. RL from AI Feedback [1, 21, 25] considers training RL agents using reward functions derived from FM preferences. This similarly guides agents towards preferred states, but without the intelligence of FMs for action selection.

FM Agents. One of the key strengths of IGE is that it is agnostic to the precise agent formulation and thus strictly additive on top of a wide variety of strategies. A common strategy is chain-of-thought-based methods [17, 44], which prompts the FM to output a set of reasoning steps before the answer. We integrate this into the FM guidance in our experiments in Sections 4.2 and 4.3. Reflexion [34] enables an agent to improve over multiple episodes by asking it to reflect on the previous attempted episode, and learn from its mistakes. However, we show this can break down in tasks with long horizons, whilst IGE proposes a more efficient way to filter out the vast majority of uninteresting interactions. Another set of agent frameworks that are related to the idea of exploring diverse solution paths via state-connectivity is Tree of Thoughts [43] and Graph of Thoughts [3]. In contrast, IGE can exploit search strategies that are not tied to any connectivity between states and can readily jump across the archive of promising saved states. This is particularly important for long-horizon tasks with larger state spaces, as we show in Sections 4.2 and 4.3.

Closely related to exploration, FM agents have also begun to see use in search-based tasks. Stream of Search [16] considers a similar mathematical reasoning task to the Game of 24, and seeks to initially clone the actions of graph search algorithms, then use RL to self-improve. In contrast, IGE already greatly outperforms classic graph search—an exciting future direction could be to first clone the exploratory behavior of Go-Explore and then self-improve with RL, enabling the FM to learn to select better. Lehnert et al. [26] analogously train a language model to mimic the A^* algorithm. Finally, Krishnamurthy et al. [22] also consider bootstrapping exploration with an externally summarized action-history in bandit problems; our focus is more on the detection of interesting states.

Go-Explore. The original Go-Explore [13, 14] framework enabled superhuman performance in a variety of hard-exploration problems, including applications as diverse as automated game testing [29]. Gallouédec and Dellandréa [15] propose Latent Go-Explore which similarly aims to address the difficulty of designing exploration heuristics by automatically learning a latent representation and sampling states with a low latent density. However, this requires periodic retraining and could easily miss out on rare serendipitous discoveries. HuGE [38] guides Go-Explore with humans in the loop by asking for pair-wise feedback on which goal to select. On the other hand, we take humans out entirely and apply intelligent FM guidance at all components of Go-Explore.

7 Conclusion and Limitations

In this paper, we demonstrate a new approach to robust exploration in complex environments, INTELLIGENT GO-EXPLORE, reimagining Go-Explore in the era of giant foundation models. We show that IGE can drive exploration for a diverse set of FM agents, including few-shot and chain-of-thought prompting, across a variety of challenging text-based games. While we only evaluate IGE on simulated text-based environments in this paper, a particularly exciting direction for future work would be domains with multimodal search spaces. This could unlock applications as wide as scientific discovery in synthetic biology (designing novel drugs or proteins) or material science. IGE could be readily adapted to these areas, as there is already precedent for multimodal FMs as judges [42]. A further direction that could break the limits of the current state-of-the-art in autonomous decision-making is the (hitherto unsolved by intelligent agents) dungeon crawler, NetHack [23]. NetHack requires the discovery of complex strategies, deep game knowledge, and coherent behavior over an extremely long horizon. Küttler et al. [23] noted that for NetHack, classic Go-Explore's "heuristic of downsampling images of states to measure their similarity to be used as an exploration bonus will likely not work for large symbolic and procedurally generated environments." IGE represents a sharp departure from these limitations, by replacing hard-coded and inflexible exploration heuristics with the dynamic intelligence of giant foundation models.

There remain exciting opportunities to improve IGE's capabilities to explore vast state spaces. For example, we currently recall and compare against the entire archive whenever we discover a new state. This could be made much more efficient by using techniques like retrieval augmented generation [27] and only comparing to the closest previously discovered states. As we consider IGE for real-world settings, we should take steps to ensure the responsible deployment of FMs [5]. Our approach opens up the road to **safe and interpretable exploration**: through careful prompt engineering or techniques like constitutional AI [1], we could steer the agent away from unsafe behaviors. Furthermore, if we ask or train the FM to explain its choices in each part of IGE, we could gain insight into its rationale for exploring particular paths through an environment [17, 41]; improving safety, interpretability, and perhaps one day even our own understanding of how best to explore.

Acknowledgments and Disclosure of Funding

This work was supported by the Vector Institute, the Canada CIFAR AI Chairs program, grants from Schmidt Futures and Open Philanthropy, an NSERC Discovery Grant, the Center for AI Safety Compute Cluster, and a generous donation from Rafael Cosman. We thank Aaron Dharna, Ben Norman, and Jenny Zhang from our lab at the University of British Columbia for insightful discussions for insightful discussions and/or feedback on early drafts of this work. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsors.

References

[1] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022.

[2] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, June 2013. ISSN 1076-9757. doi: 10.1613/jair.3912. URL http://dx.doi.org/10.1613/jair.3912.

[3] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690, March 2024. ISSN 2159-5399. doi: 10.1609/aaai.v38i16.29720. URL http://dx.doi.org/10.1609/aaai.v38i16.29720.

[4] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[5] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, S. Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen A. Creel, Jared Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano

Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas F. Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, O. Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir P. Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Benjamin Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, J. F. Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Robert Reich, Hongyu Ren, Frieda Rong, Yusuf H. Roohani, Camilo Ruiz, Jack Ryan, Christopher R'e, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishna Parasuram Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei A. Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *ArXiv*, 2021. URL `https://crfm.stanford.edu/assets/report.pdf`.

[6] Herbie Bradley, Andrew Dai, Hannah Teufel, Jenny Zhang, Koen Oostermeijer, Marco Bellagente, Jeff Clune, Kenneth Stanley, Grégory Schott, and Joel Lehman. Quality-diversity through ai feedback, 2023.

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[8] Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 3676–3713. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/carta23a.html`.

[9] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

[10] Seth Cooper. *A framework for scientific discovery through video games*. Morgan & Claypool, 2014.

[11] Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. Textworld: A learning environment for text-based games. *CoRR*, abs/1806.11532, 2018.

[12] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

[13] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth Stanley, and Jeff Clune. First return, then explore. *Nature*, 590:580–586, 02 2021. doi: 10.1038/s41586-020-03157-9.

[14] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems, 2021.

[15] Quentin Gallouédec and Emmanuel Dellandréa. Cell-free latent go-explore, 2023.

[16] Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. Stream of search (sos): Learning to search in language, 2024.

[17] Shengran Hu and Jeff Clune. Thought Cloning: Learning to think while acting by imitating human thinking. *Advances in Neural Information Processing Systems*, 36, 2024.

[18] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents, 2022.

[19] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2), apr 2017. ISSN 0360-0300. doi: 10.1145/3054912. URL `https://doi.org/10.1145/3054912`.

[20] Minqi Jiang, Tim Rocktäschel, and Edward Grefenstette. General intelligence requires rethinking exploration. *Royal Society Open Science*, 10(6):230539, 2023.

[21] Martin Klissarov, Pierluca D'Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. Motif: Intrinsic motivation from artificial intelligence feedback, 2023.

[22] Akshay Krishnamurthy, Keegan Harris, Dylan J. Foster, Cyril Zhang, and Aleksandrs Slivkins. Can large language models explore in-context?, 2024.

[23] Heinrich Küttler, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. *Advances in Neural Information Processing Systems*, 33:7671–7684, 2020.

[24] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, 2022.

[25] Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. RLAIF: Scaling reinforcement learning from human feedback with AI feedback, 2024. URL `https://openreview.net/forum?id=AAxIs3D2ZZ`.

[26] Lucas Lehnert, Sainbayar Sukhbaatar, DiJia Su, Qinqing Zheng, Paul Mcvay, Michael Rabbat, and Yuandong Tian. Beyond a*: Better planning with transformers via search dynamics bootstrapping, 2024.

[27] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf`.

[28] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents, 2023.

[29] Cong Lu, Raluca Georgescu, and Johan Verwey. Go-explore complex 3-d game environments for automated reachability testing. *IEEE Transactions on Games*, 16(1):235–240, 2024. doi: 10.1109/TG.2022.3228401.

[30] Andrea Madotto, Mahdi Namazifar, Joost Huizinga, Piero Molino, Adrien Ecoffet, Huaixiu Zheng, Alexandros Papangelis, Dian Yu, Chandra Khatri, and Gokhan Tur. Exploration based language learning for text-based games. *arXiv preprint arXiv:2001.08868*, 2020.

[31] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

[32] OpenAI. Gpt-4 technical report, 2024.

[33] Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=Bk9zbyZCZ`.

[34] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.

[35] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL `http://incompleteideas.net/book/the-book-2nd.html`.

[36] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL `https://aclanthology.org/N19-1421`.

[37] Gemini Team. Gemini: A family of highly capable multimodal models, 2024.

[38] Marcel Torne Villasevil, Max Balsells I Pamies, Zihan Wang, Samedh Desai, Tao Chen, Pulkit Agrawal, and Abhishek Gupta. Breadcrumbs to the goal: Goal-conditioned exploration from human-in-the-loop feedback. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 63222–63258. Curran Associates, Inc., 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/c7c7cf10082e454b9662a686ce6f1b6f-Paper-Conference.pdf`.

[39] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[40] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):1–26, 2024.

[41] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[42] Tianhe Wu, Kede Ma, Jie Liang, Yujiu Yang, and Lei Zhang. A comprehensive study of multimodal large language models for image quality assessment. *arXiv preprint arXiv:2403.10854*, 2024.

[43] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate problem solving with large language models, 2023.

[44] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=WE_vluYUL-X`.

[45] Jenny Zhang, Joel Lehman, Kenneth Stanley, and Jeff Clune. OMNI: Open-endedness via models of human notions of interestingness. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=AgM3MzT99c`.

[46] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. PMLR, 2021.

[47] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc., 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf`.

[48] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

[49] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

[50] Abdefihak M Zoubir and D Robert Iskandler. Bootstrap methods and applications. *IEEE Signal Processing Magazine*, 24(4):10–19, 2007.

Supplementary Material

Table of Contents

A Algorithm Pseudocode

We provide full pseudocode for INTELLIGENT GO-EXPLORE in Algorithm 1. This complements the discussion in Section 3.

Algorithm 1 INTELLIGENT GO-EXPLORE

```
1:  Hyperparameters: no. state expansions $N_{\text{state}}$, no. exploratory actions $N_{\text{action}}$, foundation model $\mathcal{M}$
2:  Initialize: archive of states $\mathcal{S}_{\text{archive}} = \emptyset$, state-conditional action history $\mathcal{A}(\cdot) = \emptyset$
3:  $\mathcal{S}_{\text{archive}} \leftarrow s_0$                                          ▷ Add initial state to archive
4:  for $i = 1, \dots, N_{\text{state}}$ do
5:      Query $\mathcal{M}$ for the next state $s_{i,1}$, from $\mathcal{S}_{\text{archive}}$              ▷ See Section 3.1
6:      for $j = 1, \dots, N_{\text{action}}$ do
7:          Query $\mathcal{M}$ for the next action $a_{i,j}$ from $s_{i,j}$ conditional on $\mathcal{A}(s_{i,j})$   ▷ See Section 3.2
8:          $s_{i,j+1} \sim P(s_{i,j}, a_{i,j})$, $\mathcal{A}(s_{i,j}) \leftarrow a_{i,j}$           ▷ Take action and update history
9:          if $\mathcal{M}$ determines that $s_{i,j+1}$ is interesting w.r.t $\mathcal{S}_{\text{archive}}$ then   ▷ See Section 3.3
10:             $\mathcal{S}_{\text{archive}} \leftarrow s_{i,j+1}$
11:         end if
12:     end for
13: end for
14: Return best discovered trajectory
```

B Further Details on Environments

We provide further details for each of the environments used in the empirical evaluation in Section 4, and the environment-specific descriptions appended to the system prompts. Each environment description may include high-level information about the task and a description of the action space.

B.1 Game of 24

We use the environment and set of evaluation tasks from `https://github.com/princeton-nlp/tree-of-thought-llm` which is released under the MIT License. We include the environment-specific prompt that is appended to the system prompt in Section 3 below. The system prompt contains examples of correct reasoning paths on different problems (few-shot prompting).

Environment Description.

You are given 4 numbers and must use basic arithmetic operations (+ - * /) to obtain 24. At each step, you are only allowed to choose two of the remaining numbers to obtain a new number. A correct answer is one that uses each input exactly once and no other numbers. Reaching 24 before the last step does not count as a correct answer. Follow the convention that division is integer division, and never by zero.
Some examples of correct reasoning traces are as follows:
Initial state: (4 4 6 8)
Steps:
4 + 8 = 12. Next: (4 6 12)
6 - 4 = 2. Next: (2 12)
2 * 12 = 24. Next: (24)
Answer: (6 - 4) * (4 + 8) = 24
Initial state: (2 9 10 12)
Steps:
12 * 2 = 24. Next: (9 10 24)
10 - 9 = 1. Next: (1 24)
24 * 1 = 24. Next: (24)
Answer: (12 * 2) * (10 - 9) = 24
Initial state: (4 9 10 13)
Steps:
13 - 10 = 3. Next: (3 4 9)
9 - 3 = 6. Next: (4 6)
4 * 6 = 24. Next: (24)
Answer: 4 * (9 - (13 - 10)) = 24

The action space at each step is all the valid arithmetic operations, presented in an analogous way as the 'propose' step in Yao et al. [43].

B.2 BabyAI-Text

The BabyAI-Text [8] environment comes with five task types, which we list here and visualize in order in Figure 5:

- Go to <object>, a simple navigation task that requires reasoning abilities to choose the right plan given the object's position;
- Pick up <object>, a reasoning task that combines navigation tasks;
- Pick up <object A> then go to <object B> and Go to <object B> after pickup <object A>, both serving to test reasoning abilities on temporal sequences;
- Unlock <door>, a task that includes inferring that a key is needed to unlock the door, finding the right key (i.e. the one colored as the door), and eventually using the toggle action with the key on the door;
- Put <object A> next to <object B>, which requires first reaching <object A>, picking it up, reaching <object B> and finally dropping <object A> next to <object B>.

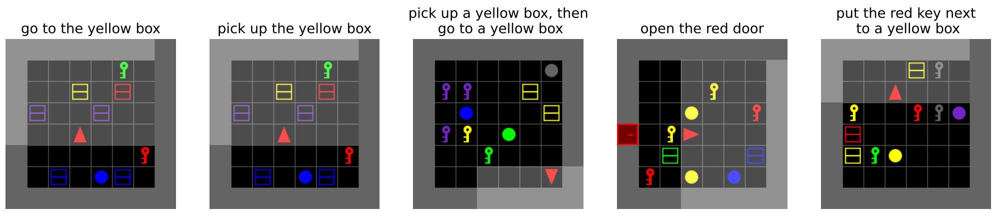
go to the yellow box
pick up the yellow box
pick up a yellow box, then go to a yellow box
open the red door
put the red key next to a yellow box

Figure 5: We visualize the 5 types of tasks that BabyAI-Text consists of for our evaluation in Section 4.2. IGE receives only partial text-based observations corresponding to the view in the figure.

We use the codebase from `https://github.com/flowersteam/Grounding_LLMs_with_online_RL` which is released under the MIT License. The action space is discrete and composed of 6 possible actions: turn left, turn right, go forward, pick up, drop, and toggle. The 'go to' and 'pick up' tasks have a shorter environment horizon of $H = 64$, whereas the rest have a horizon of $H = 128$. We include the environment-specific prompt that is appended to the system prompt in Section 3 below.

Environment Description.

You are an agent in an 8x8 partially-observable 2D text-based environment. You see the 6x6 grid in front of you, and can face north, south, east, or west. The possible actions are turn left, turn right, go forward, pick up, drop, and toggle. At each turn, you will receive a history of the last observations and actions. Your aim is to complete the task described in the goal. Each tile in the grid can only contain at most one object. Objects cannot be crossed, and may need to be bypassed or moved. You can only move onto an empty tile or on a tile containing an open door. You can only hold one object at a time, using pick up when they are one step in front. Objects are dropped one tile in front and cannot be dropped when there is another object in front. Doors are unlocked with keys of the same color using the toggle action. Actions are deterministic, do not repeat actions if they have no effect. You have H steps to complete the task.

B.3 TextWorld

We evaluate IGE on 'Treasure Hunter', 'The Cooking Game', and 'Coin Collector' from the TextWorld [11] domain. We use the environment code from `https://github.com/microsoft/TextWorld` which is released under the MIT License.

B.3.1 Treasure Hunter

For Treasure Hunter, we set the 'level' option to the maximum value of 30, resulting in a maze with 20 rooms. Locked doors and containers are added, which may need to be unlocked and opened to find the target object. To further increase the difficulty, we remove the solution description from the original game and filter out tasks that can be completed with 20 steps in the optimal solution. We include the environment-specific prompt that is appended to the system prompt in Section 3 below.

Environment Description for Treasure Hunter.

You are an agent playing TextWorld, a text-based adventure game where you are in a randomly generated maze and must find a specific object. You need to explore different rooms to find the target object.
Here are the available commands: look: describe the current room. goal: print the goal of this game inventory: print the player's inventory go <dir>: move the player north, east, south, or west. You can only go in the direction indicated with an exit or a door. open ...: open a door or a container. You need to open a closed door before you want to go through it. drop ...: drop an object on the floor take ...: take an object that is visible. Make sure the object is visible to take. put ... on ...: place an object on a supporter take ... from ...: take an object from a container or a supporter insert ... into ...: place an object into a container unlock ... with ...: unlock a door or a container with a key. You need to unlock a locked door with a matched key in your inventory before you want to open it.
- The target object might be located in a closed or locked container. - The adjective is useful for determining whether the key is matched with the lock (e.g. non-euclidean keycard is matched with non-euclidean safe). Make sure it is matched to unlock! - The key required to unlock the door may be in another room or locked inside a container. - Take the key whenever you can. - After unlocking a locked door or container, it will remain closed. You will then need to open it.
You have 40 steps to complete the task. Restarting is forbidden.

B.3.2 The Cooking Game

In The Cooking Game, we set the number of ingredients to a maximum of 5 and the number of rooms to 13. We enable all challenging additional options: doors need to be opened, food must be processed (e.g., cut, diced, chopped with a knife), and cooked (e.g., grilled with a BBQ, fried on a stove, roasted in an oven). We include the environment-specific prompt that is appended to the system prompt in Section 3 below.

Environment Description for The Cooking Game.

You are an agent playing TextWorld, a text-based adventure game where you navigate through different rooms, interact with objects, and solve puzzles. Your goal is to first find the recipe, find and prepare food according to the recipe, and finally prepare and eat the meal.
Here are the available commands: look: describe the current room goal: print the goal of this game inventory: print player's inventory go <dir>: move the player north, east, south or west. You can only go to directions indicated with an exit or a door. examine ...: examine something more closely eat ...: eat edible food open ...: open a door or a container. You need to open a closed door before you can go through it. drop ...: drop an object onto the floor take ...: take an object that is visible put ... on ...: place an object on a supporter take ... from ...: take an object from a container or a supporter insert ... into ...: place an object into a container lock ... with ...: lock a door or a container with a key unlock ... with ...: unlock a door or a container with a key cook ... with ...: cook cookable food with something providing heat slice ... with ...: slice cuttable food with something sharp chop ... with ...: chop cuttable food with something sharp dice ... with ...: dice cuttable food with something sharp prepare meal: combine ingredients from inventory into a meal. You can only prepare meals in the Kitchen.
- You can examine the cookbook to see the recipe when it is visible. - The BBQ is for grilling things, the stove is for frying things, the oven is for roasting things. Cooking ingredients in the wrong way will lead to a failure of the game. - Once you have got processed ingredients and the appropriate cooking tool ready, cook all of them according to the recipe. - There are two conditions to correctly cook something (grill/fry/roast): a) the ingredient you want to cook is in your inventory and b) there is a suitable cooking tool in the room, and then use 'cook ... with ...' command. - When you need to chop/slice/dice ingredients, you need to take the knife and the ingredient in your inventory and then 'slice/chop/dice ... with knife' - Make sure to first process the food (chop/slice/dice) before you try to cook them. - When you have all the ingredients (that got processed or cooked according to the menu), you can 'prepare meal' in the kitchen and then 'eat meal' to win the game. - The ingredients should EXACTLY match the color in the recipe, but if the recipe doesn't specify color, any color would be fine. When you 'take ... with ...', use the EXACT name you see. - You don't need to examine the container/supporter (e.g. toolbox) when it says something like "there isn't a thing on it"/"has nothing on it"
You have 80 steps to complete the task. Restarting is forbidden.

We show a successful example trajectory found by IGE below, from our evaluation in Section 4.3.

Example trajectory found by IGE in The Cooking Game.

You are hungry! Let's cook a delicious meal. Check the cookbook in the kitchen for the recipe. Once done, enjoy your meal!
-= Backyard =- You find yourself in a backyard. A normal kind of place. You can see a patio table. But oh no! there's nothing on this piece of garbage. This always happens! You make out a patio chair. But oh no! there's nothing on this piece of trash. You can make out a BBQ, so there's that. There is a closed barn door leading north. There is a closed sliding patio door leading west. There is an exit to the east. Don't worry, there is no door.
> open sliding patio door
You open sliding patio door.
> go west
-= Kitchen =- You find yourself in a kitchen. An ordinary kind of place. The room is well lit. You can make out a fridge. You hear a noise behind you and spin around, but you can't see anything other than an oven. Were you looking for a table? Because look over there, it's a table. On the table you see a cookbook. You can't wait to tell the folks at home about this! Look out! It's a- oh, never mind, it's just a counter. Now why would someone leave that there? You see a red apple, a raw purple potato, a red onion, a yellow apple and a knife on the counter. You can see a stove. You shudder, but continue examining the stove. Unfortunately, there isn't a thing on it. Hm. Oh well There is an open sliding patio door leading east. There is a closed plain door leading north. You don't like doors? Why not try going west, that entranceway is not blocked by one.
> examine cookbook
You open the copy of "Cooking: A Modern Approach (3rd Ed.)" and start reading:
Recipe #1 ——— Gather all following ingredients and follow the directions to prepare this tasty meal. Ingredients: block of cheese carrot orange bell pepper red apple tomato Directions: chop the block of cheese fry the block of cheese dice the carrot grill the carrot chop the orange bell pepper fry the orange bell pepper chop the red apple grill the red apple slice the tomato grill the tomato prepare meal
> take red apple
You take the red apple from the counter.
Your score has just gone up by one point.
> open fridge
You open the fridge, revealing a block of cheese, a carrot, a white onion and a yellow bell pepper.
> take block of cheese
You take the block of cheese from the fridge.
Your score has just gone up by one point.
> take carrot
You take the carrot from the fridge.
Your score has just gone up by one point.
> examine counter
The counter is solid.
On the counter are a raw purple potato, a red onion, a yellow apple and a knife.
> go west
-= Livingroom =- You arrive in a livingroom...
... (skip 30 steps)
> cook tomato with BBQ
You grilled the tomato.
Your score has just gone up by one point.
> go west
-= Kitchen =- You find yourself in a kitchen...
> prepare meal
Adding the meal to your inventory.
Your score has just gone up by one point.
> eat meal
You eat the meal. Not bad.
Your score has just gone up by one point.
*** The End ***
You scored 17 out of a possible 17, in 44 turns.

B.3.3 Coin Collector

In Coin Collector, we set the number of rooms to 40 and allow distractor rooms to be added along the way. Similar to Treasure Hunter, we remove the solution description from the original game, and the optimal path from the agent's starting point to the target is set to 20 steps. We include the environment-specific prompt that is appended to the system prompt in Section 3 below.

Environment Description for Coin Collector.

You are an agent playing TextWorld, a text-based adventure game where you are in a randomly generated maze and must find the coin. You need to explore different rooms to find the target object.
Here are the available commands: goal: print the goal of this game go <dir>: move the player north, east, south, or west. You can only go in the direction indicated with something like an exit or a door. take coin: 2in the game by ‘take coin’ if you see the coin in the room
The only action you can do is go <dir> to explore the maze and ‘take coin’ when you see the coin in the room.
You have 25 steps to complete the task. Restarting is forbidden.

C Further Prompt Discussion

C.1 Extracting Choices

By default, in Section 4.1, we prompt the FM to return a JSON object containing just the numerical index of the choice. We choose this because of the ease of parsing the response and validating it lies within the correct bounds. An example prompt is displayed below.

JSON Choice Prompt.

Reply concisely and exactly with the following JSON format:
{“choice”: X}
where X is the index of the desired choice.

When using chain-of-thought as in Section 4.2, we use the following prompt:

JSON Choice Prompt (Chain of Thought).

First, briefly reason about your plan.
Reply concisely and exactly with the following JSON format:
{“thought”: X, “choice”: Y}
where X is your reasoning and Y is the index of the desired choice. Make sure Y is a parsable integer.

For the TextWorld environment in Section 4.3, since the action space is much larger, we ask the FM to directly output a text action that we automatically parse.

Decision Making Prompt.

Please briefly reason about your plan and then output the command in the format ‘> command’. Ensure only one command is included.

We use the regex “> (.*?)(?:İ$)” (in Perl notation) to parse the command. We note that the failure rate for both of these options is very low, less than 0.1% across our evaluation. Despite this, we include a failsafe that returns a random choice in case of an invalid output.

C.2 Rejection-based Archive Filtering

The ‘acceptance-based’ archive filter in Section 3.3 iterates through every new state and asks whether each one is interestingly new and should be added to the archive. This can break down in larger environments where there is more need to explicitly deprecate earlier discoveries that have become irrelevant to not overload the archive, for example, in Section 4.3. In this environment, we use an alternate version of the prompt which first adds all states, and then asks the foundation model to remove the uninteresting states. An example prompt is shown below.

Rejection-based Archive Filtering Prompt.

Current state archive:
[State Archive]
Remove outdated states that are no longer relevant to the task, have had all interesting explorations attempted, or have similar states in the archive that show more progress.

D Hyperparameters

In this section, we provide the hyperparameters for our empirical evaluation in Section 4. We list the hyperparameters for IGE in Table 4. We choose the values for exploratory rollout length based on the average number of steps needed to make 'reasonable progress' in the environment.

Table 4: IGE Sampling Parameters. TH, TCG, and CC are abbreviations for Treasure Hunter, The Cooking Game, and Coin Collector in TextWorld.

Hyperparameter	Value(s)				
	Game of 24	BabyAI-Text	TH	TCG	CC
No. state expansions, N_{state}	50	25	24	48	125
No. exploratory actions, N_{action}	3	10	5	5	1

We list the sampling parameters for GPT-4 [32] passed via the OpenAI API in Table 5.

Table 5: GPT-4 Sampling Parameters

Hyperparameter	Value		
	Game of 24	BabyAI-Text	TextWorld
Temperature	0.7	0.7	0.3
Max new tokens	1000	1000	1000
Response format	JSON Object	JSON Object	Text
Version	Turbo-2024-04-09	o-2024-05-13	o-2024-05-13

We used GPT-4-Turbo for Game of 24 and GPT-4o for BabyAI and TextWorld. This was purely done to select the version of GPT-4 that was available and the cheapest at the time of running the experiments. The version of GPT-4 is consistent per environment. We use a reduced temperature for the TextWorld domain to reduce the possibility of generating malformed responses, as actions are output in free-form natural language. In our ablations in Section 5, we use the 'turbo-0125' variant of GPT-3.5.

D.1 Cost of Experiments

We provide the average cost per task for our algorithm per environment (the number of seeds is specified in Section 4):

Table 6: Per task API cost for IGE using GPT-4 listed in USD.

Environment	API Cost (USD)
Game of 24	1.04
BabyAI-Text	2.01
TextWorld	1.28

We note that the price per token of the 'o-2024-05-13' option is half that of 'Turbo-2024-04-09', so we could expect to achieve the same level of results on the Game of 24 with half the price. The total cost of API access required to perform the final experiments in this paper was under 2,000 USD. During development, we iterated on IGE with a smaller number of seeds, which represents a small fraction of this cost added on top.