DOUBLY ROBUST MONTE CARLO TREE SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

We present Doubly Robust Monte Carlo Tree Search (DR-MCTS), a novel algorithm that integrates doubly robust off-policy estimation into MCTS to improve sample efficiency in computationally expensive environments. Our approach employs an adaptive hybrid estimator that dynamically balances Monte Carlo rollouts with doubly robust estimation through variance-minimizing weights computed online from empirical statistics. We provide theoretical guarantees for unbiasedness and establish conditions for variance reduction. Empirically, DR-MCTS shows consistent improvements across diverse domains: competitive game playing (9×9 Go), mathematical reasoning (GSM8K), and embodied planning (VirtualHome). While providing modest gains in traditional domains, DR-MCTS excels in LLM-augmented environments, achieving 3× higher success rates than standard MCTS on complex compositional tasks while reducing computational costs by over 50%. Notably, entropy-based methods (MENTS, BTS, DENTS) fail to complete tasks within the same computational budgets. These results highlight how variance reduction becomes increasingly valuable when simulations involve expensive language model queries, making DR-MCTS particularly suited for the growing class of LLM-guided planning applications.

1 INTRODUCTION

Decision-making in complex, partially observable environments remains a fundamental challenge in artificial intelligence. Monte Carlo Tree Search (MCTS) has emerged as a powerful approach for addressing this challenge, demonstrating remarkable success in domains ranging from game playing to robotics Browne et al. (2012). Recently, MCTS has found applications in enhancing the reasoning capabilities of Large Language Models (LLMs) Yao et al. (2023); Zhou et al. (2024), enabling more structured and coherent text generation. However, as environments become increasingly complex and the cost of sampling increases—particularly in the context of LLMs where each node expansion may involve expensive model queries Yao et al. (2023)—there is a growing need for more sample-efficient methods that can make better decisions with fewer simulations.

To address this challenge, we introduce DR-MCTS, a novel algorithm that integrates Doubly Robust (DR) off-policy estimation into the MCTS framework. Our approach employs an adaptive hybrid estimator that dynamically balances traditional MCTS rollouts with DR estimation through a variance-minimizing weighting mechanism. This mechanism computes optimal mixing coefficients online by tracking empirical variance statistics for each state-action pair, enabling the algorithm to adaptively adjust its reliance on different estimators based on their observed performance. By leveraging the strengths of both Monte Carlo sampling and off-policy evaluation, DR-MCTS achieves superior sample efficiency and decision quality in complex environments. This improvement is particularly valuable when computational resources are limited or when each evaluation carries significant cost, as is increasingly common in applications that leverage large language models, where inference costs can range from fractions of a cent to several dollars per query depending on model size and complexity Luccioni et al. (2024).

Our work makes the following key contributions:

1. We introduce DR-MCTS, a novel algorithm that incorporates Doubly Robust off-policy estimation into the MCTS framework through an adaptive variance-minimizing hybrid estimator that optimally combines Monte Carlo rollouts with DR estimation.

2. We provide theoretical guarantees proving the unbiasedness of our hybrid estimator and establish conditions under which it achieves variance reduction relative to standard MCTS, with our adaptive weighting mechanism designed to minimize the combined estimator's variance online.
3. We conduct extensive empirical evaluations across three diverse domains: the fully observable game of 9×9 Go, the GSM8K mathematical reasoning benchmark, and the partially observable VirtualHome environment. DR-MCTS demonstrates consistent improvements over baselines across all domains, with the most pronounced gains observed in LLM-guided reasoning tasks.

1.1 Related Work

Our work draws from two primary research streams: advances in Monte Carlo Tree Search and off-policy evaluation techniques in reinforcement learning.

Monte Carlo Tree Search. Since its introduction by Coulom Coulom (2006), MCTS has established itself as a fundamental algorithm for sequential decision-making, achieving notable success across diverse domains from game playing to robotics. While the algorithm has powered several breakthrough systems Silver et al. (2016; 2018); Schrittwieser et al. (2020), our focus lies in recent methodological innovations that improve MCTS's sample efficiency without requiring extensive computational infrastructure.

Several recent approaches have tackled the sample efficiency challenge from different angles. Xiao et al. Xiao et al. (2019) introduced Maximum Entropy Monte-Carlo Planning (MENTS), which applies the maximum entropy principle to balance exploration and exploitation. By computing softmax values during backpropagation, MENTS achieves exponential convergence rates compared to UCT's polynomial rates, though this can occasionally lead to suboptimal policies when the entropy objective conflicts with reward maximization. Recognizing this limitation, Painter et al. Painter et al. (2023) developed two variants: Boltzmann Tree Search (BTS), which maintains exploration benefits while targeting reward-optimal policies, and Decaying ENtropy Tree-Search (DENTS), which gradually reduces entropy's influence as search progresses, effectively transitioning from exploration to exploitation.

From a different perspective, Grosse et al. Grosse et al. (2021) proposed Probabilistic DAG Search, which improves efficiency by exploiting structural similarities between states. Their approach employs a jointly Gaussian probabilistic model to share information across the search tree, reducing the number of simulations needed to identify optimal actions. Meanwhile, Borges and Oliveira Borges & Oliveira (2021) explored how MCTS naturally generates off-policy data during exploration, proposing methods to derive off-policy targets from the search tree itself.

These diverse approaches—entropy regularization, adaptive exploration decay, probabilistic state modeling, and off-policy data utilization—all pursue the same goal of improving MCTS's sample efficiency. Our work contributes a complementary perspective by addressing the fundamental challenge of variance in value estimation.

Off-Policy Evaluation and Doubly Robust Methods. The doubly robust estimation framework, originally developed in causal inference and biostatistics Robins & Rotnitzky (1995), has proven valuable for off-policy evaluation in reinforcement learning. Traditional importance sampling Precup et al. (2000) suffers from high variance, particularly with long trajectories or significant policy mismatch. DR methods address this limitation by combining importance sampling with direct value estimation, providing consistent estimates when either component is accurate.

The adaptation of DR methods to reinforcement learning has progressed through several stages. Dudik et al. Dudík et al. (2011) first applied DR estimation to contextual bandits, demonstrating significant variance reduction. Jiang and Li Jiang & Li (2016) then extended the framework to full RL domains, establishing theoretical foundations for sequential decision problems. Subsequent work has refined these techniques: Thomas and Brunskill Thomas & Brunskill (2016) introduced weighted DR estimators that achieve better finite-sample performance, Farajtabar et al. Farajtabar et al. (2018) developed the More Robust Doubly Robust (MRDR) estimator with improved variance bounds, and Kallus and Uehara Kallus & Uehara (2020) proposed double reinforcement learning for more efficient evaluation across both state and action spaces.

Our Contribution. DR-MCTS represents the first direct integration of doubly robust estimation into the MCTS framework. Unlike previous work that either modifies the exploration strategy (MENTS, BTS, DENTS) or leverages state similarities (Probabilistic DAG Search), we address variance reduction through principled off-policy evaluation. Our approach goes beyond simply utilizing off-policy data generated by MCTS Borges & Oliveira (2021); we fundamentally restructure the value estimation process using DR techniques.

The key innovation lies in our adaptive variance-minimizing hybrid estimator, which dynamically balances Monte Carlo rollouts with DR estimation based on empirical variance statistics tracked online. As illustrated in Figure 1, while standard MCTS relies solely on Monte Carlo sampling during simulation, DR-MCTS employs this hybrid estimator to achieve more accurate value estimates with fewer samples. This integration is particularly valuable in domains where each simulation is computationally expensive, such as complex planning tasks or applications involving large language models.

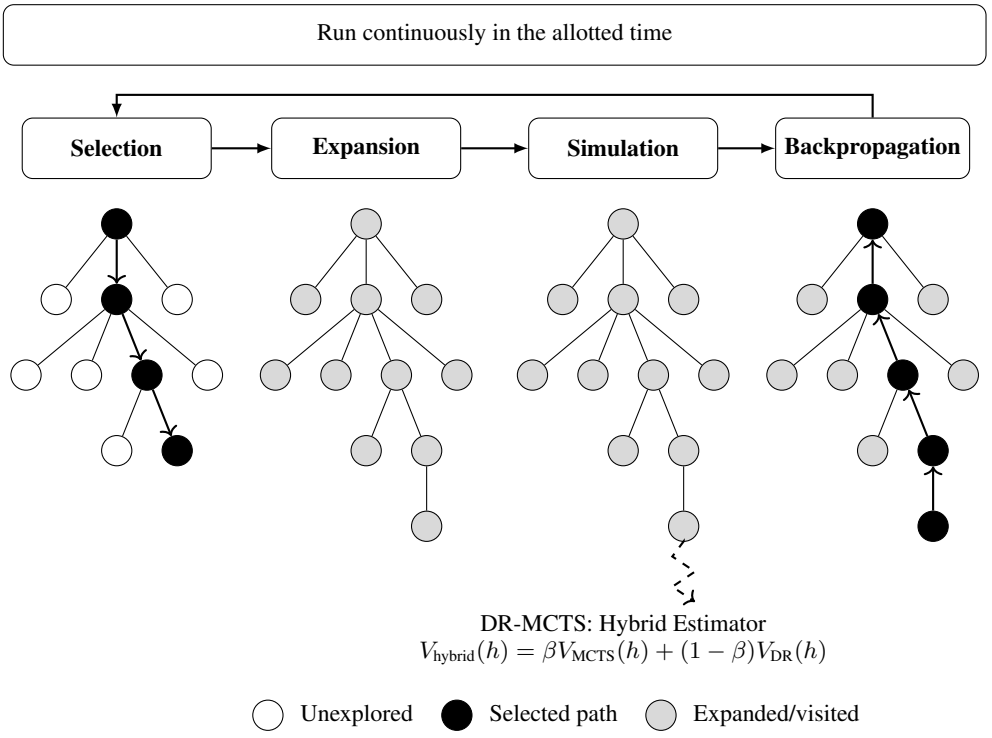

Figure 1: Monte Carlo Tree Search phases. Standard MCTS uses pure Monte Carlo rollouts in the simulation phase, while DR-MCTS employs a hybrid estimator combining MCTS rollouts with doubly robust off-policy estimation.

1.2 Background

1.2.1 Markov Decision Processes

We formalize our problem setting as a Markov Decision Process (MDP), defined by the tuple $M = \langle S, A, P, R, \gamma \rangle$. Here, S and A represent the state and action spaces respectively, while $P : S \times A \times S \to [0, 1]$ captures the transition dynamics, with $P(s'|s, a)$ denoting the probability of transitioning to state s' after taking action a in state s. The reward function $R : S \times A \to \mathbb{R}$ assigns immediate rewards $R(s, a)$ to state-action pairs, and $\gamma \in [0, 1]$ serves as the discount factor for future rewards.

A trajectory through this MDP unfolds as a sequence $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_H)$, where each reward $r_t = R(s_t, a_t)$ follows from the corresponding state-action pair. An agent's behavior is governed by a policy $\pi : S \to \Delta(A)$, which maps states to probability distributions over actions. The fundamental objective is to find a policy that maximizes the expected cumulative discounted

reward:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{H} \gamma^t r_t \,\middle|\, s_0 = s\right] \tag{1}$$

Many practical domains, including board games like Go and simulated environments like VirtualHome Puig et al. (2018), exhibit sparse reward structures where feedback occurs primarily at terminal states. This sparsity presents a significant challenge for value estimation, as intermediate actions receive no immediate signal about their quality.

1.2.2 Monte Carlo Tree Search

Monte Carlo Tree Search addresses the challenge of decision-making in large state spaces through selective sampling and incremental tree construction Browne et al. (2012). Rather than exhaustively exploring all possibilities, MCTS focuses computational resources on promising regions of the search space through an iterative four-phase process.

The algorithm begins with *selection*, traversing from the root to a leaf node by balancing exploration of uncertain branches with exploitation of promising ones. We implement this trade-off using the Polynomial Upper Confidence Trees (PUCT) criterion Silver et al. (2017):

$$a^* = \arg\max_a \left(Q(s,a) + c\pi_b(a|s)\frac{\sqrt{N(s)}}{1+N(s,a)} \right) \tag{2}$$

This formula elegantly combines the estimated action value $Q(s,a)$ with an exploration bonus that decreases as the state-action pair (s,a) is visited more frequently, where $N(s)$ and $N(s,a)$ track visit counts and c controls the exploration-exploitation balance.

Upon reaching a leaf, the *expansion* phase adds new child nodes to grow the tree. The crucial *simulation* phase then estimates the leaf's value through Monte Carlo rollouts:

$$V_{\text{MCTS}}(s) = \frac{1}{N(s)} \sum_{i=1}^{N(s)} R_i(s) \tag{3}$$

where each $R_i(s)$ represents the cumulative reward from an independent simulation starting at state s. Finally, *backpropagation* updates the statistics of all nodes along the traversed path, propagating the new information toward the root.

While this Monte Carlo approach provides unbiased estimates, it can suffer from high variance, particularly when simulations are expensive or limited.

1.3 Off-Policy Evaluation Methods

The challenge of estimating a policy's value using data collected under a different policy arises naturally in MCTS, where the tree policy used for exploration differs from the target policy we ultimately wish to evaluate. Off-policy evaluation methods provide principled approaches to this mismatch.

1.3.1 Importance Sampling

Importance Sampling (IS) corrects for the distribution mismatch between behavior policy π_b and target policy π_e through density ratios Precup et al. (2000). For each timestep, we compute the importance weight $\rho_t = \pi_e(a_t|s_t)/\pi_b(a_t|s_t)$, which reweights the observed data to match what would have been observed under the target policy. The step-wise IS estimator accumulates these weighted rewards:

$$V_{\text{step-IS}}(s) = \sum_{t=0}^{H-1} \gamma^t \rho_{1:t} r_t \tag{4}$$

where $\rho_{1:t} = \prod_{k=1}^{t} \rho_k$ represents the cumulative importance weight. While this approach provides unbiased estimates under mild conditions, the variance can become prohibitive when the behavior and target policies diverge significantly.

1.3.2 DOUBLY ROBUST ESTIMATION

Doubly Robust estimation elegantly addresses the high-variance limitation of IS by incorporating a baseline function that reduces variance without introducing bias Jiang & Li (2016). The key insight is to combine importance sampling with direct value function approximation:

$$V_{\text{DR}}(s) = \hat{V}(s) + \sum_{t=0}^{H-1} \gamma^t \rho_{1:t} \left(r_t + \gamma \hat{V}(s_{t+1}) - \hat{Q}(s_t, a_t) \right) \tag{5}$$

This formulation starts with a baseline estimate $\hat{V}(s)$ and adds a correction term that accounts for the discrepancy between observed rewards and predicted values. Crucially, the estimator remains unbiased if *either* the importance weights are correct *or* the value function approximations are accurate—hence the term "doubly robust." This robustness property makes DR estimation particularly attractive for complex domains where perfect models are unattainable.

To further reduce bias in finite-sample settings, we employ cross-validation when estimating $\hat{Q}(s_t, a_t)$ Chernozhukov et al. (2018), preventing overfitting to the limited data available within each MCTS node. This combination of robustness and practical bias reduction techniques forms the foundation of our DR-MCTS algorithm.

2 METHODS

2.1 DOUBLY ROBUST MONTE CARLO TREE SEARCH

Our DR-MCTS algorithm enhances the standard MCTS framework by introducing a variance-minimizing hybrid estimator that adaptively combines Monte Carlo rollouts with doubly robust off-policy evaluation. The core innovation lies in dynamically adjusting the mixture weights based on empirical variance statistics, allowing the algorithm to optimally balance different sources of value information.

The hybrid estimator takes the form:

$$V_{\text{hybrid}}(s) = \beta(s,a) V_{\text{MCTS}}(s) + (1 - \beta(s,a)) V_{\text{DR}}(s) \tag{6}$$

where $\beta(s,a) \in [0,1]$ determines the relative contribution of each component. Rather than using a fixed or heuristically-decaying weight, we compute $\beta(s,a)$ online to minimize the combined estimator's variance.

The key to our approach is the adaptive computation of $\beta(s,a)$ based on observed variance statistics. For each state-action pair, we maintain online estimates of the variances and covariance of the two estimators. The variance-minimizing weight is then computed as:

$$\beta^*(s,a) = \frac{\text{Var}(V_{\text{DR}}) - \text{Cov}(V_{\text{MCTS}}, V_{\text{DR}})}{\text{Var}(V_{\text{MCTS}}) + \text{Var}(V_{\text{DR}}) - 2\text{Cov}(V_{\text{MCTS}}, V_{\text{DR}})} \tag{7}$$

where the variance and covariance terms are estimated online using a sliding window of recent samples. When insufficient data is available for reliable variance estimation (typically in the first few visits), we fall back to an exponentially decaying heuristic:

$$\beta_{\text{fallback}}(s,a) = \beta_{\text{base}} \cdot \exp(-\lambda \cdot N(s,a)) \tag{8}$$

This ensures reasonable behavior during the initial exploration phase while transitioning to optimal variance-based weighting as data accumulates.

To compute the hybrid estimator in practice, we first calculate the value function estimate $\hat{V}(s)$ using Equation 10 and the Q-value estimates $\hat{Q}(s_t, a_t)$ using Equation 11. These estimates are then substituted into the doubly robust estimator $V_{\text{DR}}(s)$ in Equation 5. Finally, the hybrid estimator $V_{\text{hybrid}}(s)$ in Equation 6 combines the standard MCTS rollout value $V_{\text{MCTS}}(s)$ with the doubly robust estimate $V_{\text{DR}}(s)$ using the adaptive weighting parameter $\beta(s,a)$.

The target policy π_e for importance sampling is derived from current Q-value estimates using a softmax distribution:

$$\pi_e(a|s) = \frac{\exp(Q(s,a))}{\sum_{a'} \exp(Q(s,a'))} \tag{9}$$

The behavior policy $\pi_b(a|s)$ varies by domain as detailed in Appendix C.

We estimate the value function as a weighted average over child nodes:

$$\hat{V}(s) = \sum_a \pi_e(a|s) \cdot \frac{1}{N(s,a)} \sum_{i=1}^{N(s,a)} R_i(s,a) \tag{10}$$

where $R_i(s,a)$ denotes the i-th return observed from taking action a in state s.

For action-value estimation, we employ k-fold cross-validation to reduce overfitting bias:

$$\hat{Q}(s_t,a_t) = \frac{1}{K}\sum_{k=1}^{K} \frac{1}{|D_k|} \sum_{i \in D_k} R_i(s_t,a_t) \tag{11}$$

where the data is partitioned into K folds, with each fold D_k providing an independent estimate.

The complete DR-MCTS algorithm is provided in Algorithm 1 in the Appendix.

2.2 Theoretical Analysis

Our hybrid estimator provides strong theoretical guarantees that underpin its empirical success.

Theorem 2.1 (Unbiasedness of Hybrid Estimator)**.** *The hybrid estimator $V_{hybrid}(s)$ is unbiased for estimating the value of the target policy π_e.*

Theorem 2.1 aims to establish that despite combining two different estimators with adaptive weights, our hybrid estimator preserves the unbiasedness property. The key insight is that since both V_{MCTS} and V_{DR} are unbiased estimators of the true value function $V(s)$, any convex combination of them remains unbiased regardless of how $\beta(s,a)$ adapts over time.

Theorem 2.2 (Variance Reduction with Optimal Weighting)**.** *The hybrid estimator with variance-minimizing weight $\beta^*(s,a)$ as defined in Equation 7 has lower variance than the standard MCTS estimator when:*

$$\mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^{2t} \rho_{1:t}^2 (Q(s_t,a_t) - \hat{Q}(s_t,a_t))^2\right] = o\left(Var(V_{MCTS}(s))\right) \tag{12}$$

where $\beta^*(s,a)$ minimizes $Var(V_{hybrid}(s))$ by optimally balancing the contributions of MCTS and DR estimators.

Theorem 2.2 identifies the conditions under which DR-MCTS with optimal weighting achieves lower variance than standard MCTS. The key insight is that by choosing $\beta(s,a)$ to minimize the hybrid estimator's variance—rather than using a fixed or heuristically-decaying weight—we can guarantee variance reduction whenever the Q-value estimation errors are small relative to the Monte Carlo variance.

The optimal weight $\beta^*(s,a)$ from Equation 7 adapts to the relative reliability of the two estimators: when $\text{Var}(V_{\text{DR}})$ is large (indicating unreliable importance weights or poor value estimates), β^* increases to rely more on MCTS; conversely, when the DR estimator has low variance, β^* decreases to leverage its variance reduction benefits. This adaptive mechanism ensures that the hybrid estimator automatically adjusts to different scenarios without manual tuning.

To analyze the variance of this optimally-weighted hybrid estimator, we introduce the difference $\Delta(s) = V_{\text{DR}}(s) - V_{\text{MCTS}}(s)$, which represents the correction provided by the DR estimator. This reformulation allows us to express the hybrid estimator as $V_{\text{hybrid}}(s) = V_{\text{MCTS}}(s) + (1-\beta^*(s,a))\Delta(s)$, where the optimal weight $\beta^*(s,a)$ minimizes the total variance by balancing the variance of $\Delta(s)$ against its covariance with $V_{\text{MCTS}}(s)$.

Complete proofs are provided in Appendix B.

3 Experiments

We evaluate DR-MCTS across three diverse domains that test different aspects of planning and decision-making: the strategic game of Go, mathematical reasoning problems from GSM8K, and

complex household tasks in the VirtualHome environment. The hyperparameter configuration is provided in Appendix 4.

Our evaluation compares DR-MCTS against several state-of-the-art MCTS variants. The primary baseline is standard MCTS with pure Monte Carlo rollouts, providing a direct comparison for our variance reduction approach. We also evaluate IS-MCTS, which replaces the doubly robust estimator in our hybrid formulation with step-wise importance sampling (Equation 4). We additionally compare against three entropy-based methods: Maximum Entropy Tree Search (MENTS) Xiao et al. (2019), Boltzmann Tree Search (BTS), and Decaying Entropy Tree Search (DENTS) Painter et al. (2023). These methods represent the current state-of-the-art in exploration-driven MCTS improvements.

Go. We evaluate our approach on 9×9 Go, a domain that combines strategic depth with computational tractability. Our implementation follows standard Go rules including stone capturing and ko prevention through board state hashing. The state representation uses a 9×9 integer array with Zobrist hashing for efficient state comparison and ko detection.

The reward structure provides +1.0 for wins and 0.0 for losses at game termination, with small intermediate rewards (up to 0.5) for capturing opponent stones during play. We include a standard 6.5 point komi for White to offset Black's first-move advantage. Games terminate under three conditions: two consecutive passes (traditional Go rule), reaching a maximum of 75 moves, or when the board reaches 90% occupancy.

For evaluation, we conduct tournaments where each algorithm pair plays 30 games, alternating Black and White to ensure fairness. We report win rates with 95% confidence intervals using the Wilson score method.

GSM8K Mathematical Reasoning. The GSM8K dataset Cobbe et al. (2021) presents grade-school mathematics problems requiring multi-step reasoning, providing a challenging domain for tree search due to the combinatorial explosion of possible reasoning sequences. We frame each problem as a sequential decision-making task with a discrete action space of six reasoning operations: *identify key information*, *set up equation*, *perform calculation*, *break down problem*, *check intermediate result*, and *provide final answer*. The state representation consists of the original problem statement and the accumulated reasoning chain, where each step contains both the chosen action and corresponding LLM-generated mathematical work produced by GPT-4o-mini as the world model. To prevent trivial solutions, we enforce that *provide final answer* becomes available only after at least one reasoning step has been completed. Episodes terminate when the agent provides a final answer or after 5 reasoning steps, with rewards of +1.0 for correct solutions and 0.0 otherwise.

Our experimental evaluation uses 50 randomly selected problems from the test set, with each algorithm performing 20 MCTS simulations per decision step, under a 30-hour computational budget per algorithm to ensure fair comparison under practical resource constraints. We report accuracy rates with 95% confidence intervals, and average steps and time needed to solve one problem per algorithm.

VirtualHome Household Planning. VirtualHome Puig et al. (2018) provides a partially observable 3D household simulation environment that challenges agents with complex spatial reasoning and long-horizon planning tasks involving realistic household activities. The environment features interactive objects (furniture, appliances, containers) distributed across multiple rooms, where agents must navigate and manipulate objects to achieve high-level goals such as "prepare breakfast" or "clean the living room." Following Zhao et al. (2023), we leverage LLMs in dual roles: GPT-4o-mini serves as a world model providing commonsense knowledge about household object properties and spatial relationships, while GPT-4o acts as a policy model for high-level action selection guidance. The action space consists of primitive operations including navigation (*walk to*, *run to*), object manipulation (*grab*, *put*, *open*, *close*), and interaction commands (*sit on*, *turn on*), with state representations capturing both the agent's current location and the status of all objects in the environment.

We evaluate across three task categories designed to test different aspects of generalization and compositional reasoning. **Novel Simple** tasks (123 total) involve rearranging familiar household objects in new spatial configurations, testing the agent's ability to adapt learned object manipulation skills to novel arrangements within single rooms. **Novel Objects** tasks (34 total) introduce objects that were not present during any training phase, requiring the agent to leverage commonsense reasoning

about object properties and affordances to successfully interact with unfamiliar items. **Novel Compositional** tasks (23 total) combine multiple subtasks in previously unseen sequences, demanding hierarchical planning where agents must decompose complex goals into appropriate sequences of primitive actions while maintaining awareness of preconditions and dependencies between subtasks.

Success is measured as the percentage of tasks completed correctly within predefined step limits: 10 steps for Simple and Novel Objects categories, and 15 steps for the more complex Compositional tasks. As with GSM8K, we impose a 30-hour computational budget for each algorithm to complete all tasks within each category.

4 RESULTS

9×9 Go. Table 1 presents tournament results for DR-MCTS against baseline methods on 9×9 Go. Each algorithm pair played 30 games with alternating colors.

Table 1: Win rates of DR-MCTS against baseline methods on 9×9 Go. Each entry shows the win rate with 95% Wilson confidence intervals based on 30 games per matchup.

Opponent	Win Rate	95% CI
MCTS	0.567	[0.392, 0.726]
IS-MCTS	0.633	[0.455, 0.781]
MENTS	0.600	[0.423, 0.754]
BTS	**0.667**	[0.488, 0.808]
DENTS	0.533	[0.361, 0.698]

DR-MCTS achieved positive win rates against all baselines, with the strongest performance against BTS (66.7%), followed by IS-MCTS (63.3%) and MENTS (60.0%). The method maintained a competitive edge against DENTS (53.3%) and standard MCTS (56.7%). The improvement over IS-MCTS indicates that the doubly robust estimator provides better variance reduction than importance sampling alone in this domain.

GSM8K Mathematical Reasoning. Table 2 shows performance on 50 GSM8K problems under a 30-hour computational budget.

Table 2: Performance on GSM8K. Accuracy shows correctly solved problems with 95% Wilson confidence intervals. Statistics computed over attempted problems.

Method	Accuracy (%)	95% CI	Avg. Steps	Time/Problem
DR-MCTS	**86.0**	[73.1, 93.5]	**2.5**	**22m 41s**
MCTS[1]	82.4	[66.1, 91.9]	4.6	52m 00s
IS-MCTS[2]	80.6	[64.8, 90.3]	4.6	50m 01s

DR-MCTS completed all 50 problems while achieving the highest accuracy (86.0%) with 45% fewer reasoning steps and 56% less computation time per problem compared to MCTS. Standard MCTS and IS-MCTS exhausted the time budget after 34 and 36 problems respectively. It is important to note that our experimental setup constrains each algorithm to 20 MCTS simulations per decision step to evaluate sample efficiency under limited computational budgets—a setting distinct from specialized GSM8K methods that may use extensive prompt engineering or fine-tuning to achieve higher absolute accuracy. The focus here is on comparing tree search algorithms' ability to efficiently explore the solution space when each simulation requires expensive LLM calls, rather than achieving state-of-the-art performance on the benchmark itself. The entropy-based methods failed to solve any problems within the time constraint, suggesting their exploration strategies are incompatible with the large action spaces in LLM-based reasoning.

[1]Completed 34/50 problems within 30-hour budget.

[2]Completed 36/50 problems within 30-hour budget.

VirtualHome Household Planning. Table 3 presents success rates across three task categories under 30-hour computational budgets.

Table 3: Success rates on VirtualHome tasks. DR-MCTS shows best performance across β_{base} configurations. 95% Wilson confidence intervals provided.

Task Category	MCTS	IS-MCTS	DR-MCTS
Novel Simple (123 tasks)	85.4%	95.1%	**95.9%**
	[78.2, 90.6]	[89.6, 97.8]	[90.7, 98.3]
Novel Objects (34 tasks)	23.5%	38.2%	**41.2%**
	[12.5, 39.9]	[23.9, 54.9]	[26.3, 57.9]
Novel Compositional (23 tasks)	19.0%	34.8%	**56.5%**
	[7.7, 39.5]	[18.8, 55.1]	[36.8, 74.4]

DR-MCTS achieved the highest success rates across all categories. Performance gains were modest for Novel Simple tasks (0.8% over IS-MCTS) but increased substantially for Novel Objects (3.0% improvement) and Novel Compositional tasks (21.7% improvement). The latter represents a 62% relative improvement over IS-MCTS and nearly triple the performance of standard MCTS. As with GSM8K, entropy-based methods failed to complete any tasks within the time constraint.

5 Conclusion and Discussion

This work introduced DR-MCTS, a novel algorithm that integrates doubly robust off-policy estimation into Monte Carlo Tree Search through an adaptive variance-minimizing hybrid estimator. Our theoretical analysis established the unbiasedness of the hybrid estimator and identified conditions for variance reduction relative to standard MCTS. The core contribution demonstrates that principled variance reduction through doubly robust estimation improves MCTS performance, particularly when simulations are computationally expensive or rewards are sparse.

Our empirical evaluation reveals consistent improvements across three distinct domains with varying characteristics. In 9×9 Go, DR-MCTS achieved positive win rates against all baselines, with its advantage over IS-MCTS confirming that the doubly robust formulation provides stability beyond importance sampling alone. For GSM8K mathematical reasoning, DR-MCTS demonstrated superior sample efficiency: achieving 86.0% accuracy while requiring 45% fewer reasoning steps and 56% less computation time per problem, compared to standard MCTS which achieved 82.4% accuracy before exhausting the computational budget. The VirtualHome results revealed an interesting pattern where performance gains scaled with task complexity—from 10.5% improvement on Novel Simple tasks to nearly 3× improvement on Novel Compositional tasks. This scaling suggests that variance reduction compounds its benefits in deeper search trees where estimation errors accumulate. The inability of entropy-based methods (MENTS, BTS, DENTS) to complete tasks within computational budgets in LLM-augmented domains highlights a key insight: when node expansions are expensive, reducing value estimation variance provides greater practical benefit than exploration bonuses.

Several limitations merit consideration. The variance-minimizing weight relies on empirical estimates that may be unreliable early in the search, necessitating our fallback heuristic. Additionally, the theoretical variance reduction condition assumes Q-value approximation accuracy that may not hold initially. Future work could address these limitations through more robust online variance estimation methods and investigation of alternative baseline functions. The particular success of DR-MCTS in LLM-augmented environments suggests potential for developing variance reduction techniques tailored to the specific characteristics of language model planning, where the cost structure and error patterns differ from traditional domains. By providing an adaptive, domain-agnostic approach to balancing different value estimation sources, DR-MCTS offers a practical enhancement to MCTS that becomes increasingly valuable as applications move toward more computationally intensive simulation models.

6 REPRODUCIBILITY

Hyperparameters used to reproduce experiment results are detailed in Appendix E. Computational resources, training time, and hardware specifications needed for replication are detailed in Appendix F.1. All code and data will be made publicly available upon publication.

7 USE OF LLM

Large Language Models were only used to correct grammar errors and polish writing in this manuscript. No LLMs were used for data generation, analysis, interpretation of results, or writing of scientific content. All experimental design, methodology, and conclusions are the original work of the authors.

REFERENCES

Alexandre Borges and Arlindo Oliveira. Combining off and on-policy training in model-based reinforcement learning, 2021. URL https://arxiv.org/abs/2102.12194.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68, 01 2018. ISSN 1368-4221. doi: 10.1111/ectj.12097. URL https://doi.org/10.1111/ectj.12097.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pp. 72–83. Springer, 2006.

Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 1097–1104, 2011.

Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. More robust doubly robust off-policy evaluation. In *International Conference on Machine Learning*, pp. 1447–1456. PMLR, 2018.

Julia Grosse, Cheng Zhang, and Philipp Hennig. Probabilistic dag search. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, pp. 1424–1433. PMLR, 2021.

Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *International Conference on Machine Learning*, pp. 652–661. PMLR, 2016.

Nathan Kallus and Masatoshi Uehara. Double reinforcement learning for efficient off-policy evaluation in markov decision processes. In *International Conference on Machine Learning*, pp. 5168–5178. PMLR, 2020.

Alexandra Sasha Luccioni, Yacine Jernite, and Emma Strubell. Power hungry processing: Watts driving the cost of ai deployment? *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, pp. 85–99, 2024.

Michael Painter, Mohamed Baioumy, Nick Hawes, and Bruno Lacerda. Monte carlo tree search with boltzmann exploration. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Doina Precup, Richard S Sutton, and Satinder P Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 759–766, 2000.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8494–8502, 2018.

James M Robins and Andrea Rotnitzky. Semiparametric efficiency in multivariate regression models with missing data. *Journal of the American Statistical Association*, 90(429):122–129, 1995.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Philip Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, pp. 2139–2148. PMLR, 2016.

Chenjun Xiao, Ruitong Huang, Jincheng Mei, Dale Schuurmans, and Martin Müller. Maximum entropy monte-carlo planning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, pp. 11809–11822. Curran Associates, Inc., 2023.

Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for large-scale task planning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 31967–31987. Curran Associates, Inc., 2023.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160. PMLR, 21–27 Jul 2024.

A Appendix

B Theoretical Analysis of DR-MCTS

B.1 Unbiasedness of the Hybrid Estimator

Theorem B.1 (Unbiasedness of Hybrid Estimator)**.** *The hybrid estimator is unbiased for estimating the value of the target policy π_e.*

Proof. We know that $V_{\text{MCTS}}(s)$ is unbiased due to the properties of Monte Carlo estimation. For $V_{DR}(s)$, we can express it as:

$$V_{DR}(s) = \hat{V}(s) + \sum_{t=0}^{H-1} \gamma^t \rho_{1:t}(r_t + \gamma \hat{V}(s_{t+1}) - \hat{Q}(s_t, a_t)) \tag{13}$$

where $\rho_{1:t} = \prod_{k=1}^{t} \frac{\pi_e(a_k|s_k)}{\pi_b(a_k|s_k)}$ is the cumulative importance ratio.

Taking the expectation with respect to the behavior policy π_b:

$$\mathbb{E}_{\pi_b}[V_{DR}(s)] = \mathbb{E}_{\pi_b}[\hat{V}(s)] + \mathbb{E}_{\pi_b}\left[\sum_{t=0}^{H-1} \gamma^t \rho_{1:t}(r_t + \gamma \hat{V}(s_{t+1}) - \hat{Q}(s_t, a_t))\right] \tag{14}$$

$$= \hat{V}(s) + \sum_{t=0}^{H-1} \gamma^t \mathbb{E}_{\pi_b}[\rho_{1:t}(r_t + \gamma \hat{V}(s_{t+1}) - \hat{Q}(s_t, a_t))] \tag{15}$$

$$= \hat{V}(s) + \sum_{t=0}^{H-1} \gamma^t (Q(s_t, a_t) - \mathbb{E}_{\pi_e}[\hat{Q}(s_t, a_t)]) \tag{16}$$

$$= V(s) \tag{17}$$

The last step follows from the fact that $\mathbb{E}_{\pi_e}[\hat{Q}(s_t, a_t)] = Q(s_t, a_t)$ when $\hat{Q}$ is an unbiased estimator of Q.

Therefore, both $V_{\text{MCTS}}(s)$ and $V_{DR}(s)$ are unbiased estimators of $V(s)$. Since $V_{\text{hybrid}}(s)$ is a linear combination of these unbiased estimators, it is also unbiased:

$$\mathbb{E}[V_{\text{hybrid}}(s)] = \mathbb{E}[\beta(s,a)V_{\text{MCTS}}(s) + (1 - \beta(s,a))V_{DR}(s)] \tag{18}$$

$$= \beta(s,a)\mathbb{E}[V_{\text{MCTS}}(s)] + (1 - \beta(s,a))\mathbb{E}[V_{DR}(s)] \tag{19}$$

$$= \beta(s,a)V(s) + (1 - \beta(s,a))V(s) \tag{20}$$

$$= V(s) \tag{21}$$

Thus, the hybrid estimator remains unbiased in the DR-MCTS context. □

B.2 Variance Reduction with Optimal Beta

Theorem B.2 (Variance Reduction with Variance-Minimizing Beta)**.** *Let $V_{hybrid}(s)$ be the hybrid estimator as defined in Equation 6, with $\beta(s,a)$ chosen to minimize variance as in Equation 7. The hybrid estimator with optimal $\beta^*(s,a)$ has lower variance than the standard MCTS estimator when:*

$$\mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^{2t} \rho_{1:t}^2 (Q(s_t, a_t) - \hat{Q}(s_t, a_t))^2\right] = o\left(Var(V_{MCTS}(s))\right) \tag{22}$$

where the optimal weight is:

$$\beta^*(s,a) = \frac{Var(V_{DR}(s)) - Cov(V_{MCTS}(s), V_{DR}(s))}{Var(V_{MCTS}(s)) + Var(V_{DR}(s)) - 2Cov(V_{MCTS}(s), V_{DR}(s))} \tag{23}$$

Proof. We begin by deriving the optimal β that minimizes the variance of the hybrid estimator. The variance of $V_{\text{hybrid}}(s)$ is:

$$\begin{aligned}
\text{Var}(V_{\text{hybrid}}(s)) &= \text{Var}(\beta V_{\text{MCTS}}(s) + (1-\beta)V_{\text{DR}}(s)) && (24)\\
&= \beta^2 \text{Var}(V_{\text{MCTS}}(s)) + (1-\beta)^2 \text{Var}(V_{\text{DR}}(s)) && (25)\\
&\quad + 2\beta(1-\beta)\text{Cov}(V_{\text{MCTS}}(s), V_{\text{DR}}(s)) && (26)
\end{aligned}$$

To find the variance-minimizing β^*, we take the derivative with respect to β and set it to zero:

$$\begin{aligned}
\frac{\partial}{\partial \beta}\text{Var}(V_{\text{hybrid}}(s)) &= 2\beta \text{Var}(V_{\text{MCTS}}(s)) - 2(1-\beta)\text{Var}(V_{\text{DR}}(s)) && (27)\\
&\quad + 2(1-2\beta)\text{Cov}(V_{\text{MCTS}}(s), V_{\text{DR}}(s)) = 0 && (28)
\end{aligned}$$

Solving for β^*:

$$\begin{aligned}
&\beta^* \text{Var}(V_{\text{MCTS}}(s)) + \beta^* \text{Var}(V_{\text{DR}}(s)) - 2\beta^* \text{Cov}(V_{\text{MCTS}}(s), V_{\text{DR}}(s)) && (29)\\
&= \text{Var}(V_{\text{DR}}(s)) - \text{Cov}(V_{\text{MCTS}}(s), V_{\text{DR}}(s)) && (30)
\end{aligned}$$

Therefore:

$$\beta^*(s,a) = \frac{\text{Var}(V_{\text{DR}}(s)) - \text{Cov}(V_{\text{MCTS}}(s), V_{\text{DR}}(s))}{\text{Var}(V_{\text{MCTS}}(s)) + \text{Var}(V_{\text{DR}}(s)) - 2\text{Cov}(V_{\text{MCTS}}(s), V_{\text{DR}}(s))} \tag{31}$$

Now, substituting this optimal β^* back into the variance expression. Let $\sigma_M^2 = \text{Var}(V_{\text{MCTS}}(s))$, $\sigma_D^2 = \text{Var}(V_{\text{DR}}(s))$, and $\sigma_{MD} = \text{Cov}(V_{\text{MCTS}}(s), V_{\text{DR}}(s))$ for notational simplicity. The minimized variance is:

$$\begin{aligned}
\text{Var}(V^*_{\text{hybrid}}(s)) &= \frac{\sigma_M^2\sigma_D^2 - \sigma_{MD}^2}{\sigma_M^2 + \sigma_D^2 - 2\sigma_{MD}} && (32)\\
&= \frac{\sigma_M^2\sigma_D^2 - \sigma_{MD}^2}{(\sqrt{\sigma_M^2} - \sqrt{\sigma_D^2})^2 + 2\sqrt{\sigma_M^2}\sqrt{\sigma_D^2} - 2\sigma_{MD}} && (33)
\end{aligned}$$

For the hybrid estimator to have lower variance than standard MCTS, we require:

$$\text{Var}(V^*_{\text{hybrid}}(s)) < \text{Var}(V_{\text{MCTS}}(s)) \tag{34}$$

This is equivalent to:

$$\frac{\sigma_M^2\sigma_D^2 - \sigma_{MD}^2}{\sigma_M^2 + \sigma_D^2 - 2\sigma_{MD}} < \sigma_M^2 \tag{35}$$

Cross-multiplying and simplifying:

$$\begin{aligned}
\sigma_M^2\sigma_D^2 - \sigma_{MD}^2 &< \sigma_M^2(\sigma_M^2 + \sigma_D^2 - 2\sigma_{MD}) && (36)\\
\sigma_M^2\sigma_D^2 - \sigma_{MD}^2 &< \sigma_M^4 + \sigma_M^2\sigma_D^2 - 2\sigma_M^2\sigma_{MD} && (37)\\
-\sigma_{MD}^2 + 2\sigma_M^2\sigma_{MD} &< \sigma_M^4 && (38)\\
\sigma_{MD}(2\sigma_M^2 - \sigma_{MD}) &< \sigma_M^4 && (39)
\end{aligned}$$

Now, from the doubly robust estimator structure, we know that:

$$V_{\text{DR}}(s) = \hat{V}(s) + \sum_{t=0}^{H-1} \gamma^t \rho_{1:t}(r_t + \gamma \hat{V}(s_{t+1}) - \hat{Q}(s_t, a_t)) \tag{40}$$

The variance of $V_{\text{DR}}(s)$ can be bounded by:

$$\text{Var}(V_{\text{DR}}(s)) \leq \mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^{2t} \rho_{1:t}^2 (r_t + \gamma \hat{V}(s_{t+1}) - \hat{Q}(s_t, a_t))^2\right] \tag{41}$$

When the Q-value approximation is accurate, i.e., $\hat{Q}(s_t, a_t) \approx Q(s_t, a_t)$, and defining:

$$\epsilon = \mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^{2t} \rho_{1:t}^2 (Q(s_t, a_t) - \hat{Q}(s_t, a_t))^2\right] \tag{42}$$

The variance reduction condition becomes:

$$\epsilon = o(\text{Var}(V_{\text{MCTS}}(s))) \tag{43}$$

This means that when the Q-value estimation error (weighted by importance ratios and discount factors) is small relative to the Monte Carlo variance, the variance-minimizing hybrid estimator achieves lower variance than standard MCTS.

Furthermore, the optimal β^* automatically adapts to the relative reliability of the two estimators: when $\text{Var}(V_{\text{DR}}(s))$ is large, β^* increases (favoring MCTS); when $\text{Var}(V_{\text{DR}}(s))$ is small, β^* decreases (favoring DR). This adaptive property ensures robust performance across different scenarios. □

C Behavior Policies

C.1 Go and GSM8K Behavior Policy

For Go and GSM8K, we implement a uniform behavior policy over all available actions:

$$\pi_b(a|s) = \frac{1}{|\mathcal{A}(s)|} \tag{44}$$

where $\mathcal{A}(s)$ represents the set of legal actions available in state s. This uniform distribution ensures comprehensive exploration across the action space while serving as a simple baseline for off-policy evaluation.

C.2 VirtualHome Behavior Policy

For the VirtualHome environment, we adapt the approach of Zhao et al. Zhao et al. (2023) to leverage Large Language Models (LLMs) as a heuristic policy. Specifically, we use GPT-4o to generate the behavior policy, guiding action selection in the simulation procedure.

The LLM takes as input:

- K-shot examples from the dataset
- Goal description
- Current observation
- History of actions

All inputs are translated into English sentences. The LLM then outputs a suggested action plan. To approximate the policy distribution, we sample the LLM M times, querying it with the prompt and trajectory history h:

$$\alpha_i \sim \text{LLM}(s, \text{prompt}) \tag{45}$$

where α_i is the first action of the LLM's answer.

The prompt examples are selected based on their similarity to the current language instruction ℓ. We use sentence embeddings to calculate the cosine similarity between the current instruction and instructions ℓ_i in the dataset D:

$$\text{similarity} = \text{CosineSim}(\ell_i, \ell) \tag{46}$$

We select the top K similar instructions and use their corresponding expert trajectories as the K-shot prompt.

To ensure executability, we represent both the LLM's suggested actions and the admissible actions as embeddings and evaluate their cosine similarity. The empirical policy distribution is then formulated as:

$$\hat{\pi}_b(a|s) = \lambda \frac{1}{|A|} + (1-\lambda)\text{Softmax}\left\{\sum_{i=1}^{M} \text{CosineSim}(\alpha_i, a) - \eta\right\} \tag{47}$$

where η is the average value of $\sum_i \text{CosineSim}(\alpha_i, a)$, $|A|$ is the size of the admissible action space, and λ is a hyperparameter that adds randomness to the policy. This results in a mixture of the approximated policy from the LLM and a uniform distribution.

D DR-MCTS ALGORITHM

Algorithm 1 presents our DR-MCTS approach. The algorithm initializes a search tree with the root node representing the initial state and history. For a specified number of iterations, it traverses the tree using the PUCT selection strategy (Equation 2), balancing exploration and exploitation. When a new node is reached, it's added to the tree, and a simulation estimates its value V_{MCTS}. If the DR estimator is used, V_{DR} is calculated (Equation 5).

The hybrid estimator (Equation 6) combines these estimates using the variance-minimizing weight $\beta^*(s, a)$ from Equation 7. This weight is computed online by tracking empirical variances of both estimators and their covariance through a sliding window of recent samples. When insufficient samples are available for reliable variance estimation (typically during early visits), the algorithm falls back to the heuristic weight $\beta_{\text{base}} \cdot \exp(-\lambda \cdot N(s, a))$ as specified in Equation 8.

The resulting value is backpropagated, updating node statistics. After all iterations, the algorithm returns the action with the highest estimated value at the root node.

E EXPERIMENTAL SETTINGS AND DETAILS

E.1 9×9 GO

E.1.1 EXPERIMENTAL SETTINGS

We evaluate DR-MCTS on 9×9 Go, a domain that balances computational tractability with strategic complexity. Table 4 provides complete specifications of all hyperparameters used.

E.1.2 IMPLEMENTATION DETAILS

- **State representation**: 81-element array for 9×9 board positions
- **Action space**: Empty intersections plus pass move (maximum 82 actions)

Algorithm 1 DR-MCTS Algorithm

Input: state s_0, history h_0, iterations N
Initialize tree T with root (s_0, h_0)
Initialize variance trackers for each node
for $i = 1$ **to** N **do**
  $(s, h) \leftarrow (s_0, h_0)$
  while (s, h) is not terminal **and** (s, h) is in T **do**
    $a \leftarrow \text{argmax}_{a'}\text{PUCT}((s, h), a')$
    $s, h \leftarrow \text{Apply}(s, a), h + a$
  end while
  if (s, h) is not terminal **then**
    Add (s, h) to T
    $v_{\text{MCTS}} \leftarrow \text{Simulate}(s, h)$
    $v_{\text{DR}} \leftarrow \text{ComputeDR}(s, h, \pi_e, \pi_b, \hat{Q}, \hat{V})$
    Update variance statistics: $\text{Var}(V_{\text{MCTS}})$, $\text{Var}(V_{\text{DR}})$, $\text{Cov}(V_{\text{MCTS}}, V_{\text{DR}})$
    if $N(s, a) \geq$ min_samples **then**
      $\beta \leftarrow \frac{\text{Var}(V_{\text{DR}}) - \text{Cov}(V_{\text{MCTS}}, V_{\text{DR}})}{\text{Var}(V_{\text{MCTS}}) + \text{Var}(V_{\text{DR}}) - 2\text{Cov}(V_{\text{MCTS}}, V_{\text{DR}})}$ {Variance-min}
    else
      $\beta \leftarrow \beta_{\text{base}} \cdot \exp(-\lambda \cdot N(s, a))$ {Fallback heuristic}
    end if
    $v \leftarrow \beta v_{\text{MCTS}} + (1 - \beta) v_{\text{DR}}$ {Hybrid value}
  else
    $v \leftarrow \text{Reward}(s)$
  end if
  while (s, h) is not (s_0, h_0) **do**
    Update statistics for (s, h) in T with v
    $(s, h) \leftarrow \text{Parent}(s, h)$
  end while
end for
Return: $\text{argmax}_a Q((s_0, h_0), a)$

- **Reward structure**: +1.0 for wins, 0.0 for losses at game termination, with small intermediate rewards (up to 0.5) for capturing opponent stones
- **Scoring**: Area scoring with 6.5 point komi for White
- **Termination conditions**:
  - Two consecutive passes (traditional Go rule)
  - Maximum of 75 moves
  - Board reaches 90% occupancy
- **Special rules**: Ko prevention implemented through Zobrist hashing

E.1.3 Evaluation Protocol

Tournament evaluation follows these principles:

1. Each algorithm pair plays 30 games, alternating Black and White to ensure fairness
2. Actions selected by highest Q-value at root node after search completion
3. Performance measured by win rate with 95% Wilson score confidence intervals
4. All experiments conducted on identical hardware for fair comparison

Table 4: Hyperparameter configuration for all experimental domains

Parameter	9×9 Go	GSM8K	VirtualHome
Experimental Settings			
Games/Problems per evaluation	30	50	180 total tasks
Random seed	42	42	42
Computational budget	-	30 hours	30 hours/category
MCTS Parameters			
Rollouts/Simulations per move	5	20	100
Maximum depth	15	10	15 (compositional) 10 (others)
Tree policy	PUCT	PUCT	PUCT
PUCT exploration constant c	1.414	2.0	2.0
Discount factor γ	1.0	0.95	0.95
Hybrid Estimator Parameters			
Base hybrid weight β_{base}	$\{0.25, 0.5, 0.75\}$	$\{0.25, 0.5, 0.75\}$	$\{0.25, 0.5, 0.75\}$
Adaptive decay parameter λ	0.01	0.05	0.01
Cross-validation folds K	2	2	2
Policy and Behavior Settings			
Behavior policy	Uniform	Uniform	LLM-guided
World model	-	GPT-4o-mini	GPT-4o-mini
Policy model	-	-	GPT-4o
Few-shot examples	-	-	3-shot
Domain-Specific Parameters			
Board/State space size	9×9	-	-
Maximum steps per episode	75 moves	5 reasoning steps	Task-dependent
Komi (Go scoring)	6.5	-	-
Action space size	≤ 81	6 operations	Primitive actions

E.2 GSM8K Mathematical Reasoning

E.2.1 Experimental Settings

GSM8K Cobbe et al. (2021) provides grade-school mathematics problems requiring multi-step reasoning. We frame each problem as a sequential decision-making task suitable for tree search. Complete hyperparameters are specified in Table 4.

E.2.2 Action Space Design

We define six reasoning operations that capture common problem-solving strategies:

- **identify_key_information**: Extract relevant numbers and relationships from the problem statement
- **set_up_equation**: Formulate mathematical equations based on identified relationships
- **perform_calculation**: Execute arithmetic operations and algebraic manipulations
- **break_down_problem**: Decompose complex problems into manageable sub-problems
- **check_intermediate_result**: Verify the reasonableness of intermediate calculations
- **provide_final_answer**: State the numerical answer to the problem

To encourage meaningful reasoning, *provide_final_answer* becomes available only after at least one reasoning step has been completed.

E.2.3 LLM INTEGRATION

- **World Model (GPT-4o-mini)**: Generates mathematical work for each reasoning action, maintaining context across the solution trajectory
- **Temperature**: Set to 0.1 for deterministic reasoning while allowing minor variations
- **Token tracking**: Input and output tokens monitored for cost analysis

E.2.4 EVALUATION PROTOCOL

- **Dataset**: 50 randomly selected problems from GSM8K test set
- **Success criterion**: Predicted numerical answer matches ground truth (tolerance 10^{-6})
- **Episode termination**: Maximum 5 reasoning steps or when final answer provided
- **Reward structure**:
  - +10.0 for correct final answer
  - -1.0 for incorrect final answer
  - -0.1 step penalty to encourage efficiency
- **Computational budget**: 30 hours total per algorithm
- **Metrics**: Accuracy rate, average reasoning steps, total computation time, cost per problem

E.3 VIRTUALHOME HOUSEHOLD PLANNING

E.3.1 DATA GENERATION AND TASK DESIGN

Following the methodology of Zhao et al. (2023), we create a comprehensive dataset for evaluation:

- **Training data**: 2,000 tasks with randomly initialized scenes and expert trajectories
- **Expert agent**: Oracle agent with full environment knowledge, using regression planning with handcrafted heuristics
- **Total trajectories**: 10,000 expert demonstrations for baseline training
- **Few-shot prompts**: 200 instances randomly sampled for LLM prompting (no fine-tuning)
- **Evaluation set**: 180 tasks across three complexity categories

E.3.2 MODEL CONFIGURATION

Complete hyperparameters are provided in Table 4. Key configuration details include:

- **World Model (GPT-4o-mini)**: Provides commonsense knowledge about household environments, object locations, and likely state transitions
- **Policy Model (GPT-4o)**: Generates action proposals based on current observations and task goals
- **Prompt selection**: 3 most similar examples selected based on instruction embedding similarity
- **Action selection**: Highest Q-value at root after 100 MCTS rollouts

E.3.3 EVALUATION PROTOCOL

- **Success criterion**: Task completion within the specified step limit
- **Step limits**:
  - 10 steps for Novel Simple and Novel Objects categories
  - 15 steps for Novel Compositional category
- **Computational budget**: 30 hours per task category
- **Metrics**: Success rate per category
- **Baselines**: Standard MCTS and IS-MCTS with identical LLM configurations

F Code Availability and License

We commit to releasing our full implementation upon acceptance, which will include:

- Complete implementations of DR-MCTS, IS-MCTS, and baseline MCTS algorithms
- Environment wrappers for Go (9×9), GSM8K, and VirtualHome
- Reproduction scripts for all experiments
- Hyperparameter configurations and random seeds for reproducibility
- Detailed documentation and usage instructions

Our implementation builds upon the following open-source resources:

- **VirtualHome Environment**: Based on Watch-and-Help[1] (CC BY-NC-SA 4.0 license) for training/test data generation
- **LLM-MCTS Integration**: Adapted from Zhao et al.'s codebase[2] (CC BY-NC-SA 4.0 license) for LLM-guided tree search

All code will be released under the MIT license to facilitate broader adoption. We acknowledge that our experiments utilized GPT-4o and GPT-4o-mini APIs; users will need their own API credentials to reproduce VirtualHome results.

F.1 Reproducibility Guidelines

To reproduce our experiments:

- **9×9 Go**: Any modern CPU with at least 1GB RAM
- **GSM8K**: GPU GPU with minimum 16GB memory (A100 recommended for exact timing replication)
- **VirtualHome**: GPU with minimum 16GB memory (A100 recommended for exact timing replication)
- **Software requirements**: Python 3.8+, PyTorch 1.10+, and API access to GPT-4o and GPT-4o-mini
- **Estimated total compute time**:
  - 9×9 Go: up to 3 hour for all experiments
  - GSM8K: Up to 30 hours
  - VirtualHome: Up to 30 hours for each task category

[1] https://github.com/xavierpuigf/virtualhome/tree/master
[2] https://github.com/1989Ryan/llm-mcts