ROBUST PLANNING IN STOCHASTIC ENVIRONMENTS VIA SELECTIVE ADVERSARIAL INTERVENTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Stochastic MuZero and Stochastic AlphaZero have achieved the state-of-the-art for planning in stochastic environments by modeling stochasticity using afterstates and chance events. Their expectation-based training focuses on maximizing average outcomes, at the expense of robustness to rare but consequential chance events. To address this, we introduce *Robust Stochastic Zero* (RSZ), the first method to extend Stochastic AlphaZero using robust adversarial reinforcement learning (RARL) with selective adversarial interventions. Unlike typical RARL methods that constantly perturb states, RSZ intervenes only at the most critical afterstates, which rarely occur but cause significant value degradation. Furthermore, it leaves the otherwise dynamics untouched, minimizing the interventions to avoid distorting the inherent stochastic dynamics. This approach improves the planning robustness without obstructing the ability to plan in the natural environment. On benchmark stochastic environments, 2048, Tetris Block Puzzle, and Server Allocation, RSZ achieves improved performance over the baseline with intervening in only 0.01% of events, and remains comparable when no interventions are applied. Our findings demonstrate that the right rather than constant interventions are a direction for robust planning in stochastic environments.

1 INTRODUCTION

Search-based planning methods, specifically AlphaZero (Silver et al., 2018) and MuZero (Schrittwieser et al., 2020), have recently achieved remarkable success. By combining Monte Carlo tree search (Kocsis & Szepesvári, 2006) with deep neural network predictions for lookahead search, they mastered classic games such as Go, Chess, and Atari. Furthermore, they have been extended to complex domains beyond games, including robotics control (Hubert et al., 2021), stochastic environments (Antonoglou et al., 2022), matrix multiplication (Fawzi et al., 2022), and algorithm discovery (Mankowitz et al., 2023). Among these extensions, *Stochastic MuZero* and *Stochastic AlphaZero* (Antonoglou et al., 2022) model the environmental stochasticity as *afterstates* (Sutton & Barto, 1998) and *chance events*, where an afterstate is a post-action, pre-chance intermediate from which a chance event occurs, transitioning to the next state. By considering the environment as an additional player that samples chance events based on expected outcomes during lookahead search, they achieve the current state-of-the-art for planning in stochastic environments.

Reliable planning requires attention to robustness and safety. Given the ongoing focus, recent works have increasingly enhanced AlphaZero and MuZero in this area. For example, RobustZero (Li et al., 2025) enhances the robustness of MuZero against worst-case and random-case state perturbations in environments like physical or energy control systems. ConstrainedZero (Moss et al., 2024) extends AlphaZero with learned failure probabilities to guide safe action selection in partially observable environments. RiskZero (Yassin & Wen, 2026) learns distributional quantities to estimate risk and avoid rare but severe outcomes in deterministic dynamics. Nevertheless, the robustness of Stochastic MuZero and Stochastic AlphaZero remains underexplored. Their expectation-based training optimizes average performance, but it may fail to expose decision weaknesses that appear only when rare but consequential chance events occur. This issue is especially critical in long-horizon stochastic environments, where a small number of rare adversarial events can derail an otherwise successful trajectory comprising thousands of steps. On the other hand, robust adversarial reinforcement learning (RARL) (Pinto et al., 2017) is a common approach that introduces an adversary to adversarially perturb states and expose such weaknesses. However, in stochastic environments,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

inherent stochasticity becomes a challenge: constant adversarial perturbations can violate the original dynamics, fundamentally change the task being learned, and ultimately sacrifice overall performance.

To address this gap, we introduce *Robust Stochastic Zero* (RSZ), the first method to extend Stochastic AlphaZero using RARL with selective adversarial interventions, enhancing robustness against extreme scenarios while maintaining competitive overall performance. For the selective interventions, we establish a concept of *criticality* – a joint metric combining the severity of value degradation with the rarity of occurrence. Furthermore, we enhance Monte Carlo tree search to capture critical afterstates, constraining the adversary to intervene only in highly critical afterstates while leaving stochastic dynamics otherwise untouched, thereby preventing environmental distortion and avoiding conservative agent behavior. These contributions make RSZ the first search-based RARL planning method to use criticality-based interventions to enhance worst-case robustness without sacrificing average-case performance. On three benchmark stochastic environments, 2048, Tetris Block Puzzle, and Server Allocation, we demonstrate that RSZ achieves improved robustness under intervening in only 0.01% of events, and it maintains comparable performance without interventions. Overall, our work demonstrates that making the right few interventions – not many – can achieve robust search-based planning in stochastic environments.

2 Related works

Our work extends Stochastic AlphaZero with a focus on robust search-based planning in stochastic environments. Below, we outline the key differences between our work and other related works.

Robust adversarial RL Robust adversarial reinforcement learning (RARL) models uncertainties as external disturbances through an adversary that applies destabilizing forces (Pinto et al., 2017). Using random-case or worst-case perturbations, RARL-based methods help the agent to mitigate vulnerable decisions and have thereby emerged as a research topic of great interest. However, prior works are typically not designed for stochastic environments. Among them, the adversarial perturbations are often noise added to the states (Li et al., 2025; Mandlekar et al., 2017; Pattanaik et al., 2018; Lin et al., 2017), making them not directly applicable in optimization problems where state perturbations are unnatural. In addition, perturbations are constantly added at each time step (Pinto et al., 2017; Lecarpentier & Rachelson, 2019; Pan et al., 2019), making them unnatural in stochastic settings that must reflect inherent stochasticity. On the other hand, some prior works are not designed for search-based planning (Huang et al., 2022; Klima et al., 2019; Zhang et al., 2021; Tang et al., 2025; Gleave et al., 2020; Lin et al., 2017), which also significantly limit the performance when planning in stochastic environments.

Risk-averse RL Risk-averse reinforcement learning optimizes the decision-making to avoid risk, and has long been studied in risk-sensitive domains. These methods typically have a formally defined risk quantity to optimize, which is different from our RARL-based method that consistently changes the optimizing risk (Pinto et al., 2017). On the other hand, risk-averse RL has been widely applied to stochastic environments, but not to the type we focus on – long-horizon environments where stochasticity arises in a way different from actions. Specifically, prior works have mostly focused on stochasticity arising from noise (Chow et al., 2018b; 2015; Eysenbach & Levine, 2022; Li et al., 2025; Mandlekar et al., 2017; Pan et al., 2019; Rigter et al., 2023), and some on those with deterministic transitions but stochastic rewards (Greenberg et al., 2022; Zhou et al., 2025; Ns et al., 2024; Jacobs et al., 2021). Only a few have explored our target stochastic environment type (Chow et al., 2018a; Greenberg et al., 2022), while they were conducted in small-scale settings with shorter episode lengths. Finally, risk-averse RL methods are typically not search-based. While some recent works have begun to investigate risk-averse MCTS (Zhang et al., 2025; Santos et al., 2026), they have done so in standard MCTS settings, distinct from deep RL approaches.

Critical states The critical state is a widely used RL concept, referring to a state that significantly influences the final outcome. Previous works identify a critical state either by return redistribution (Arjona-Medina et al., 2019) or by whether a random action can be taken (Jacq et al., 2022; Liu et al., 2023; Cheng et al., 2023). However, our work applies this concept to identify critical afterstates with the largest decreases in value and the lowest probability of occurring.

3 BACKGROUND

Below, we introduce the foundations of search-based planning in stochastic environments.

Stochastic environments In stochastic environments, an action can lead to multiple possible outcomes, which can be modeled explicitly using *afterstates* and *chance events* (Antonoglou et al., 2022; Sutton & Barto, 1998), as in Equation 1. Starting from the current state s, an action a deterministically produces an afterstate as, which is a hypothesized intermediate that occurs immediately after the agent applies the action, before any stochasticity takes effect. Then, a chance event c is sampled according to the environment's stochastic dynamics, transforming as into the next state s', and obtaining a deterministic reward r. Therefore, the original stochastic transition from a state-action pair s, a to the next state s' is modeled into the afterstate transition probability, i.e., $\Pr(s' \mid s, a) = \Pr(s' \mid as) = \Pr(c \mid as)$.

$$s \xrightarrow{a} as \xrightarrow{c} s' \tag{1}$$

Such an environment can be described as an extended Markov Decision Process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{E}, \mathcal{C}, \mathcal{P}, \mathcal{Q}, \mathcal{R}, \boldsymbol{\rho}, \gamma \rangle$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $\mathcal{E}$ is an afterstate space, $\mathcal{C}$ is a chance event space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{E}$ is the state transition function, $\mathcal{Q} : \mathcal{E} \times \mathcal{C} \to \mathcal{S}$ is the afterstate transition function, $\mathcal{R} : \mathcal{E} \times \mathcal{C} \to \mathbb{R}$ is the reward function, $\boldsymbol{\rho} : \mathcal{E} \to \Delta(\mathcal{C})$ is the inherent environmental stochasticity, and $\gamma \in [0, 1]$ is a discount factor.

Stochastic AlphaZero and Stochastic MuZero *Stochastic AlphaZero* (SAZ) and *Stochastic MuZero* (SMZ) (Antonoglou et al., 2022) extend AlphaZero (Silver et al., 2018) and MuZero (Schrittwieser et al., 2020) to stochastic environments, by introducing the concept of afterstates into Monte Carlo tree search (MCTS) (Kocsis & Szepesvári, 2006; Coulom, 2007; Browne et al., 2012). The search runs simulations repeatedly for planning, with each consisting of three phases: *selection*, *expansion*, and *backup*.

During selection, nodes are traversed recursively starting from the root node s^0. At each state s^i, the child node as^{i+1} corresponds to the action a^{i+1} that has the highest pUCT (Rosin, 2011; Silver et al., 2017) score is selected:

$$a^{i+1} = \arg\max_a \Big[q_a + p_a \frac{\sqrt{\sum_b n_b}}{n_a + 1} \eta_{\text{puct}} \Big], \tag{2}$$

where q_a, p_a, and n_a are the mean value, prior probability, and visit count of action a at s^i; $\Sigma_b n_b$ is the visit count of s^i; η_{puct} is a constant. The pUCT selection balances exploitation and exploration: the q_a term favors higher-value actions, while the remaining term encourages less-visited actions. On the other hand, at each afterstate as^i, the child node s^{i+1} and its chance event c^{i+1} with the highest *quasi-random sampling* (QRS) (Ozair et al., 2021; Antonoglou et al., 2022) score is chosen:

$$c^{i+1} = \arg\max_c \Big[\frac{\rho_c}{n_c + 1} \Big], \tag{3}$$

where $\rho_c = \Pr(c \mid as^i)$ and n_c is the visit count of chance event c. The QRS selection simulates stochastic outcomes using a deterministic visiting order, allowing the proportion of visits to each c to converge to ρ_c. The selection algorithm repeatedly applies pUCT and QRS selections until it reaches an unevaluated leaf node, which may be a state s^ℓ or an afterstate as^ℓ.

During expansion, the leaf node is evaluated, and its child nodes are expanded. For a state s^ℓ, the *policy* and *value* are evaluated by $\boldsymbol{p}^\ell, v^\ell \leftarrow f(s^\ell)$ using the *prediction function* f, and the child nodes are expanded with prior probability $p_a \in \boldsymbol{p}^\ell$ assigned to each node. Otherwise, for an afterstate as^ℓ, the *chance distribution* and *afterstate value* are evaluated by $\boldsymbol{\rho}^\ell, \nu^\ell \leftarrow \psi(as^\ell)$ using the *afterstate prediction function* ψ, and the child nodes are expanded with corresponding $\rho_c \in \boldsymbol{\rho}^\ell$ assigned. Finally, during backup, the mean values q^i and visit counts n^i for all selected nodes s^i or as^i are updated by

$$q^i \leftarrow \frac{n^i q^i + G^i}{n^i + 1} \text{ and } n^i \leftarrow n^i + 1, \tag{4}$$

where G^i is the discounted sum of all rewards r^j for $i < j \leq \ell$ and the leaf node value v^ℓ or ν^ℓ.

4 ROBUST STOCHASTIC ZERO

Robust Stochastic Zero (RSZ) consists of a robust agent and a lurking adversary, where the agent continues to maximize performance, and the adversary exposes the agent's decision vulnerability by delivering maximal impact with minimal interventions without distorting dynamics.

4.1 OVERALL CONCEPT

RSZ aims to identify *critical afterstates* for use in adversarial training, illustrated in Figure 1 through the game-playing of 2048. In this example, the game begins with puzzle (a), and the player slides the tiles to merge smaller ones into larger ones. After each move, a new tile is randomly added. As the game progresses, the player encounters puzzle (b), deciding to slide right and forming a Γ-shape empty space. Then, a 2-tile appears as in (c), and the game continues through (d) and (e), eventually ending at (f). However, the decision to slide right at (b) is risky: if a 4-tile is added with only a 2% chance at the specific location in (g), the player is forced to slide left. This shifts the largest tiles away from the border, allowing a subsequent unfortunate event to add a 2-tile like (h), which breaks the alignment and can end the game within four moves, as shown in (i).

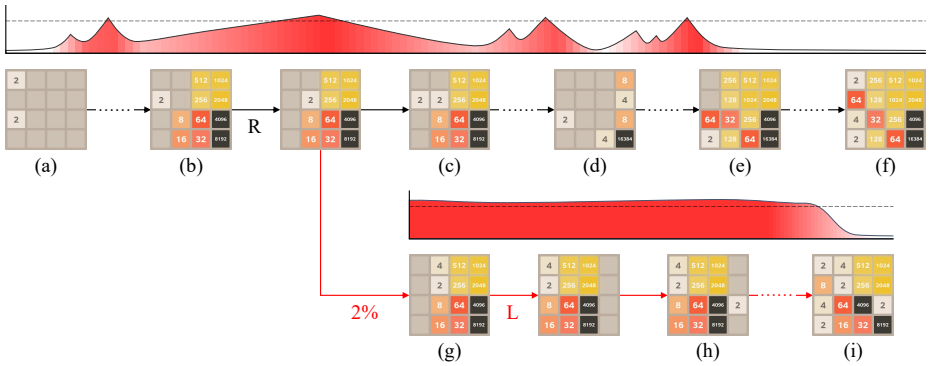


Figure 1: The critical afterstates in 2048. Puzzles (a)-(f) show a normal game; (g)-(i) demonstrate the vulnerability of the risky decision at (b). The chart above the puzzles measures the criticality. Labeled and unlabeled puzzles are states and afterstates, respectively.

RSZ keeps interventions rare to avoid distorting stochasticity and to concentrate them precisely on highly critical afterstates. At such afterstates, the low probability makes them likely to be disregarded during training, leaving the player unaware of the risk, such as sliding right at (b). Nevertheless, the adversary can strategically attack at (g) to expose the vulnerability. In contrast, RSZ avoids intervening in non-critical afterstates. For those without catastrophes like (a), intervention is unnecessary. For those with unavoidable catastrophes like (e), the relevant chance event occurs with sufficiently high probability and is expected to appear naturally without the need for intervention. Therefore, the adversary focuses on rare but highly malicious attacks, ultimately teaching the player to robustly avoid catastrophes in an undistorted environment.

4.2 PROBLEM FORMULATION

Overall, RSZ aims to optimize an adversarial training objective of

$$\max_{\boldsymbol{\pi}} \min_{\boldsymbol{\mu} \in \mathcal{U}} \mathbb{E}_{\boldsymbol{\pi},\boldsymbol{\mu}}\Big[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(\mathcal{P}(s_t, a_{t+1}), c_{t+1})\Big] \text{ s.t. } \mathcal{U} = \big\{\boldsymbol{\mu} \mid \boldsymbol{\mu}(\cdot|as) = \boldsymbol{\rho}(\cdot|as)\ \forall\ as \notin \mathcal{E}_\star\big\} \quad (5)$$

with an agent optimizing $\boldsymbol{\pi}$, and an adversary optimizing $\boldsymbol{\mu}$ but constrained by a set of critical afterstates $\mathcal{E}_\star$. An agent is to find a policy $\boldsymbol{\pi} : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected value. An adversary is to find a policy $\boldsymbol{\mu} : \mathcal{E} \to \Delta(\mathcal{C})$ that minimizes the value by intervening only at $as \in \mathcal{E}_\star$ and maintaining the overall inherent stochasticity ρ otherwise. To derive $\mathcal{E}_\star$, a quantifiable surrogate measure $\mathscr{T} : \mathcal{E} \to \mathbb{R}$ is introduced as

$$\mathcal{E}_\star = \{as \in \mathcal{E} \mid \mathscr{T}(as) \geq \tau_{\text{th}}\}, \quad (6)$$

where $\mathscr{T}(as)$ is the *criticality* of as, and τ_{th} is the *criticality threshold* of critical afterstates.

Criticality The *criticality* of an afterstate as is measured by factoring in *severity* and *rarity* by

$$\mathscr{T}(as) = \underbrace{\left(1 - \frac{q_\star}{\nu}\right)}_{\text{severity}} \underbrace{(1 - \rho_\star)}_{\text{rarity}}, \tag{7}$$

where $q_\star$ and $\rho_\star$ are the mean value and the occurring probability of the *worst-case event* $c_\star$, defined as the one yielding the lowest mean value among all possible events originating from as. The severity $(1 - \frac{q_\star}{\nu})$ is the ratio of the drop from as's value ν to $q_\star$,[1] reflecting how severe the worst-case event $c_\star$ is. The rarity $(1 - \rho_\star)$ is the complement probability of $c_\star$, reflecting how uncommon it can occur. By prioritizing higher criticality, we target afterstates that not only have a significant negative outcome in the worst case, but also have the worst case more likely to be overlooked during training because it occurs with low probability. Further details are discussed in Appendix A.

Criticality threshold The *criticality threshold* τ_{th} targets the afterstates with the top λ proportion of the criticality, where λ is a predefined intervention rate. τ_{th} is essentially the $(1 - \lambda)$-quantile of the criticality measures for all afterstates resulting from the agent's decisions, namely

$$\tau_{\text{th}} = \inf\{\tau \in \mathbb{R} \mid \Pr(\mathscr{T}(as) \geq \tau) \leq \lambda\}. \tag{8}$$

However, no universal fixed τ_{th} exactly captures the desired $(1 - \lambda)$-quantile for all agents. Since agents with different policies have $\mathscr{T}(as)$ with diverse distributions, especially during training, when the policy evolves over time. To address this, we introduce an online quantile estimator, modified from the *Fast Algorithm for Median Estimation* (FAME) (Feldman & Shavitt, 2007), to dynamically determine τ_{th} from decision records. Specifically, it estimates τ_{th} as the $(1-\lambda)$-quantile of the $\mathscr{T}(as)$ observed at recent time steps, such that $\Pr(\mathscr{T}(as) \geq \tau_{\text{th}} \mid \boldsymbol{\pi}) \approx \lambda$. The details and pseudocode are provided in Appendix A and Appendix B.

4.3 Robust Stochastic Monte Carlo Tree Search

RSZ uses *robust stochastic Monte Carlo tree search* (RS-MCTS) to focus on critical afterstates during lookahead search. It allows not only the agent to anticipate and avoid potential future threats, but also the adversary to initiate catastrophes by intervening in a series of events. RS-MCTS cycles through three planning phases – *selection*, *expansion*, and *backup* – followed by the *decision* phase at the end. The pseudocode is available in Appendix B, and an overview is shown in Figure 2.

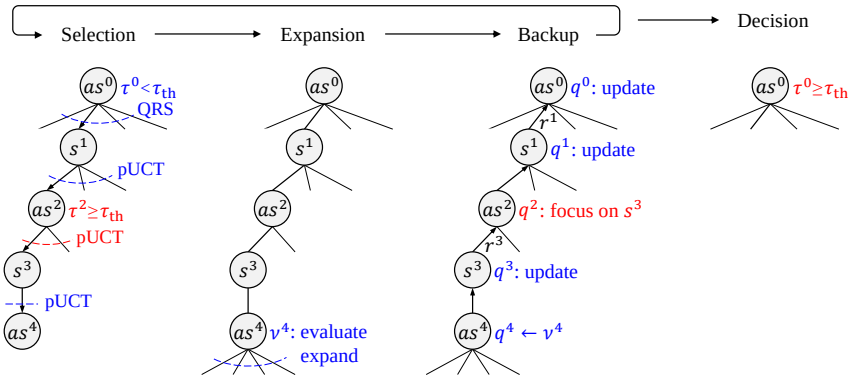


Figure 2: Robust stochastic Monte Carlo tree search for planning in stochastic environments. In selection, as^2 has a criticality $\tau^2 \geq \tau_{\text{th}}$ and selects s^3 using pUCT. In expansion, as^4 is evaluated and expanded. In backup, as^2 is updated to focus on s^3's mean value, and s^1 and as^0 are also updated to reflect the focus change. Finally, in decision, as^0 is identified as critical after the search.

Selection As both agent and adversary take turns planning, the root node can be either a state s^0 or an afterstate as^0. During the recursive selection, at each state s^i, the next afterstate as^{i+1} is selected by pUCT shown in Equation 2 since the agent always aims to maximize its value. However, at each

[1] Assume a value range of $0 \leq q_\star \leq \nu < \infty$ here; other scenarios are covered in Appendix A.

afterstate as^i, the lurking adversary may keep the chance or intervene, depending on the criticality $\tau^i = \mathscr{T}(as^i)$. For a non-critical as^i with $\tau^i < \tau_{\text{th}}$, the adversary keeps the stochasticity $\boldsymbol{\rho}^i$ by using QRS as in Equation 3. For a critical as^i with $\tau^i \geq \tau_{\text{th}}$, the adversary intervenes by using pUCT to select the next state s^{i+1} with an event c^{i+1} that aims to minimize the agent's value:

$$c^{i+1} = \arg\max_c \left[-q_c + \delta_c \frac{\sqrt{\sum_b n_b}}{n_c + 1} \eta_{\text{puct}}\right], \tag{9}$$

which is essentially the same as Equation 2, but uses $-q_c$ for minimizing and δ_c for the adversary prior. Finally, the selection arrives at the leaf node, s^ℓ or as^ℓ, then moves to the next phase.

Expansion This phase follows the expansion in Section 3, where leaf node s^ℓ or as^ℓ is evaluated and expanded. When the leaf node is as^ℓ, we additionally obtain the *adversarial policy* $\boldsymbol{\delta}^\ell$ by $\boldsymbol{\delta}^\ell, \boldsymbol{\rho}^\ell, \nu^\ell \leftarrow \psi(as^\ell)$ using the modified afterstate prediction function. When expanding child nodes, $\delta_c \in \boldsymbol{\delta}^\ell$ is also assigned to the child node c, serving as the prior in pUCT selection.

Backup As interventions are rare and may occur deep in the search, simply following the standard backup routine in Equation 4 can cause them to be overshadowed by the large number of normal samples. Therefore, we strengthen the rare outcomes to enable more efficient propagation of q^i by using different rules depending on whether the adversary intervenes at the node:

$$q^i \leftarrow \begin{cases} \dfrac{\sum_a n_a q_a + v^i}{n^i + 1} & \text{if } s^i. \qquad (10) \\ \dfrac{\sum_c n_c (r_c + \gamma q_c) + \nu^i}{n^i + 1} & \text{if } as^i \text{ is not intervened.} \qquad (11) \\ r^{i+1} + \gamma q^{i+1} & \text{if } as^i \text{ is intervened.} \qquad (12) \end{cases}$$

To elaborate on this, we take the backup in Figure 2 for example. First, we set $q^4 \leftarrow \nu^4$ by Equation 11 and update q^3 by Equation 10. At as^2, as there is an intervention, we set $q^2 \leftarrow r^3 + \gamma q^3$ directly by Equation 12. Then, we recalculate q^1 by Equation 10, and finally q^0 by Equation 11. In this process, the strengthening part occurs at as^2. Without this specific design, the intervention outcome q^3 would only contribute a fraction of $1/(n^2+1)$ to q^2, diluted by other chance events.

Decision For the agent's turn, an action is determined based on MCTS statistics by sampling from the softmax distribution over the root action visit counts. However, for the environment's turn, the adversary may be lurking or intervene. If the root node as^0 does not satisfy $\tau^0 \geq \tau_{\text{th}}$, namely not critical enough, the adversary samples a chance event based on the environment probabilities ρ^0. Otherwise, an intervention proceeds by using MCTS statistics, as in the agent's turn.

4.4 Model training

Training an RSZ model costs roughly twice as much as its baseline because the adversarial environment also requires a search.[2] In this work, RSZ is built on SAZ for demonstration, as SAZ is more efficient, whereas SMZ requires learning the transition rules. Therefore, the neural network $\boldsymbol{\theta}$ learns $\boldsymbol{p}$, v, ν, and $\boldsymbol{\delta}$, with learning objectives $\boldsymbol{\pi}$, z, z, and $\boldsymbol{\omega}$, respectively. The training for $\boldsymbol{p}$, v, and ν follows the same approach used in SAZ. For the adversarial policy $\boldsymbol{\delta}$, the objective $\boldsymbol{\omega}$ is a one-hot distribution for the intervention event $c_\star$, or a uniform distribution when there is no intervention. The total loss $\mathcal{L}$ combines the original SAZ loss and the adversarial policy loss with a scaling factor η_{adv}:

$$\mathcal{L} = -\boldsymbol{\pi}^\mathsf{T} \log \boldsymbol{p} + (z - v)^2 + (z - \nu)^2 - \eta_{\text{adv}}\, \boldsymbol{\omega}^\mathsf{T} \log \boldsymbol{\delta} + \eta_{\text{norm}} \|\boldsymbol{\theta}\|^2. \tag{13}$$

5 Experiments

To demonstrate RSZ's improvements in planning robustness, we compare it with SAZ, the current state-of-the-art on which it is based, using three stochastic environments: *2048* (Cirulli, 2014), a puzzle game where the player slides the puzzle to merge randomly appearing tiles into larger tiles;

[2] After training, deployment costs are comparable because the environment does not need to make decisions.

Tetris Block Puzzle (The Tetris Company, Inc, 2023), a puzzle game where the player places randomly appearing tetrominoes on the empty locations of the puzzle; *Server Allocation* (Greenberg et al., 2022), a resource management task where the agent adjusts the number of active servers to handle randomly arriving requests. The detailed specifications are provided in Appendix C.

For training and evaluation, one challenge in stochastic environments is their high variance. To address this while considering computing limitations, we drop performance outliers (the best and worst models) to focus on the median models after training. Additionally, under computational constraints, the high variance makes it difficult to achieve statistical significance. To improve stability, we instead increase the number of evaluation episodes to obtain more reliable mean scores for comparison. Full experiment setups are available in Appendix D.

5.1 Performance improvement

Our primary objective is to enhance robustness without sacrificing average return. For this, we train five RSZ models with an intervention rate of $\lambda = 0.01\%$ alongside five SAZ models; training statistics are available in Appendix E. We evaluate them under two settings, denoted as E_N and $E_{0.01\%}$: a natural evaluation with the original stochasticity, and an adversarial evaluation with 0.01% additional interventions. For both methods, we retain three models based on their E_N evaluations. The E_N evaluation takes 200 episodes per model, and the $E_{0.01\%}$ evaluation takes 1,800 episodes using every retained adversary.

Table 1 presents the comparison, including overall and tail performance; the detailed return distributions are provided in Appendix F. For the overall performance under E_N, RSZ shows mixed results, achieving relative scores of 99.42%, 103.50%, and 149.65%, respectively.[3] Under the $E_{0.01\%}$ intervention, performance drops for both methods, while RSZ achieves relative scores of 107.14%, 104.20%, and 149.55% compared with SAZ. Namely, RSZ demonstrates better robustness, maintaining not only higher performance under adversarial interventions, but also comparable performance under natural dynamics. For tail performance, an improvement indicates a lower worst-case vulnerability. Here, we analyze $\text{CVaR}_{10\%}$, i.e., the mean of the worst 10% of episode returns. RSZ shows significantly better mean CVaR in both 2048 and Server Allocation, under both E_N and $E_{0.01\%}$. In Tetris Block Puzzle, the significant gap between overall and tail performance under E_N (from 3,691 to 294) highlights numerous naturally occurring unfortunate events. This indicates that increasing the intervention rate from $\lambda = 0.01\%$ could lead to better performance, as explored further in Appendix G.

Table 1: Performance comparison on 2048, Tetris Block Puzzle, and Server Allocation. Reported statistics are mean return $\pm$ standard deviation; boldface indicates the higher mean.

Method	2048		Tetris Block Puzzle		Server Allocation	
	E_N	$E_{0.01\%}$	E_N	$E_{0.01\%}$	E_N	$E_{0.01\%}$
Overall Performance						
SAZ	**160,305 ± 1,051**	141,117 ± 1,453	3,691 ± 206	3,045 ± 210	-14.80 ± 5.56	-14.82 ± 5.57
RSZ	159,370 ± 4,589	**151,191 ± 2,328**	**3,820 ± 25**	**3,173 ± 174**	**-9.89 ± 1.63**	**-9.91 ± 1.72**
Tail Performance ($CVaR_{10\%}$)						
SAZ	60,765 ± 283	34,505 ± 4,992	**294 ± 92**	27 ± 2	-14.96 ± 5.60	-14.99 ± 5.59
RSZ	**72,078 ± 3,763**	**57,697 ± 67**	291 ± 50	**32 ± 10**	**-11.01 ± 3.16**	**-11.24 ± 3.47**

Nevertheless, RSZ takes roughly twice as much computing as SAZ to achieve the same number of training iterations. To examine this trade-off, we compare them under approximately matched training computation, detailed in Appendix H. The comparisons across the three environments show mixed results. In 2048 and Tetris Block Puzzle, SAZ tends to learn faster at the beginning, while RSZ can gradually reduce the gap as training proceeds. In Server Allocation, RSZ already shows an advantage in the early stages. Furthermore, to achieve near convergence, we also take the smaller 3×3 2048 board for the analysis. At the end of training, RSZ attains 99.98% of SAZ's mean under E_N and 116.18% under $E_{0.01\%}$. These results suggest that the robustness gain can persist under approximately matched training computation.

[3] For Server Allocation, we take $\frac{\text{SAZ}}{\text{RSZ}}$ as the ratio so that higher values still indicate better RSZ performance.

5.2 ALTERNATIVE APPROACH AND ABLATION STUDY

We investigate adversarial interventions in search-based planning. To better validate the design of RSZ, we compare it with straightforward AlphaZero-based alternatives, including three approaches of intervention, FULL-ATTACK (always intervenes during search), RANDOM-ATTACK (randomly intervenes during search), and NO-ATTACK (never intervenes during search), as well as two ablations, STANDARD-BACKUP (performs standard MCTS backup) and SEVERITY-ONLY (considers only severity in criticality). Specifically, we train models for these methods using an intervention rate of $\lambda = 0.01\%$, and then evaluate the trained agents under the $\text{E}_{0.01\%}$ setting, detailed in Appendix I.

As summarized in Table 2, RSZ outperforms all alternatives in the mean. Across the three intervention approaches, FULL-ATTACK shows a significant decline, implying that fully worst-case interventions fundamentally alter the effective task. In contrast, RANDOM-ATTACK is slightly better than NO-ATTACK. Overall, precise interventions are better than random ones, and better than none at all. For both ablation methods, STANDARD-BACKUP shows that RSZ still functions but is less effective when the rare intervention outcomes are not strengthened. SEVERITY-ONLY indicates that considering only severity, as typical worst-case interventions in adversarial RL methods, is simply not enough for the constrained adversary. For high-severity events whose occurrence probabilities are already high, SEVERITY-ONLY still allocates budget to them. This is essentially a waste, as they are expected to occur naturally. Therefore, RSZ becomes more effective under constrained interventions.

Table 2: Comparison between RSZ and other search alternatives. Reported statistics are mean return $\pm$ standard deviation; boldface indicates the higher mean.

Method	2048	Tetris Block Puzzle	Server Allocation
FULL-ATTACK	52,896 $\pm$ 2,705	175 $\pm$ 40	-13.52 $\pm$ 4.22
RANDOM-ATTACK	147,764 $\pm$ 5,483	3,004 $\pm$ 166	-10.23 $\pm$ 0.49
NO-ATTACK	146,915 $\pm$ 6,023	2,933 $\pm$ 223	-11.86 $\pm$ 1.90
STANDARD-BACKUP	147,314 $\pm$ 3,235	3,065 $\pm$ 257	-10.36 $\pm$ 1.65
SEVERITY-ONLY	140,298 $\pm$ 2,892	3,121 $\pm$ 51	-11.30 $\pm$ 1.60
SAZ	141,117 $\pm$ 1,453	3,045 $\pm$ 210	-14.82 $\pm$ 5.57
RSZ	**151,191 $\pm$ 2,328**	**3,173 $\pm$ 174**	**-9.91 $\pm$ 1.72**

5.3 BEHAVIOR ANALYSIS ON CRITICALITY

Criticality is central to RSZ. First, we check how the criticality $\mathscr{T}(as)$ changes during training. Figure 3 tracks the average $\mathscr{T}(as)$ during training for RSZ models, and Appendix E provides detailed statistics including the distribution changes. Initially, the high criticality reflects agents' vulnerability. With training, they converge to a low value, reflecting that agents' decisions are progressively optimized. Furthermore, the criticality threshold τ_{th} is by definition the $(1-\lambda)$-quantile of all $\mathscr{T}(as)$ values for intervening at a rate of λ. As the distribution of $\mathscr{T}(as)$ shifts toward lower values, τ_{th} must also decrease to maintain the same λ. This shows why relying on a fixed τ_{th} cannot guarantee a consistent intervention rate through the training process.

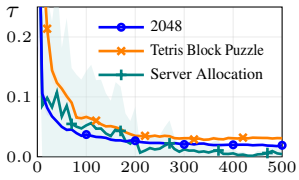


Figure 3: $\mathscr{T}(as)$ in training.

Table 3: Average criticality $\mathscr{T}(as)$ of interventions in evaluation. Lower values indicate that the decisions are less vulnerable.

Method	2048	Tetris Block Puzzle	Server Allocation
SAZ	0.7769 $\pm$ 0.0440	0.7858 $\pm$ 0.0181	0.4275 $\pm$ 0.1340
RSZ	**0.4505 $\pm$ 0.1097**	**0.7777 $\pm$ 0.0146**	**0.2268 $\pm$ 0.3566**

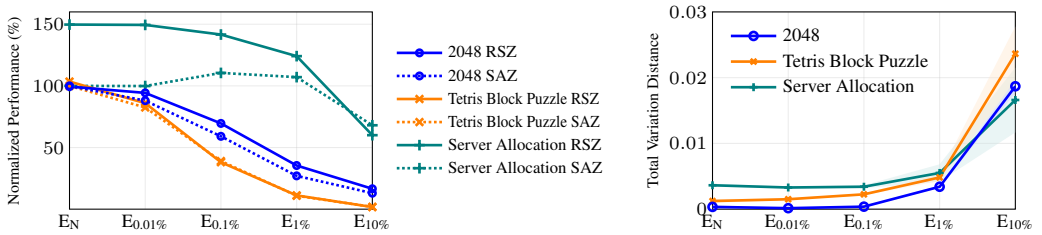
Next, we assess whether RSZ has lower criticality values than SAZ. Specifically, we extract all afterstates in which the adversary intervenes from the $\text{E}_{0.01\%}$ evaluations and average their criticality, as reported in Table 3. As the adversary always intervenes at the top λ critical afterstates, a lower value indicates fewer effective interventions. In summary, RSZ has a lower mean criticality in both

2048 and Server Allocation, reflecting the observed performance boost from SAZ to RSZ. However, the improvement is marginal in Tetris Block Puzzle, likely due to the same factor discussed earlier that $\lambda = 0.01\%$ is too low for this game. We provide more analysis of criticality in Appendix J.

5.4 Impact of more interventions

To demonstrate the negative impact of aggressive interventions, we evaluate RSZ agents trained with $\lambda = 0.01\%$ and SAZ agents under E_N, $E_{0.01\%}$, $E_{0.1\%}$, $E_{1\%}$, and $E_{10\%}$. During evaluation, the RSZ agents are made aware of the increased interventions, allowing RS-MCTS to target a larger proportion of critical afterstates. As shown in Figure 4 (a), the performance of both RSZ and SAZ degrades substantially as the interventions increase across all three environments, although RSZ still slightly outperforms SAZ in most settings. These results show that agents remain highly sensitive to interventions. However, severe interventions are implausible under natural stochasticity, so poor performance under them is not a major concern. Details are available in Appendix K.

Furthermore, to assess whether the stochasticity is preserved, we analyze distributional changes using total variation (TV) distance. In Figure 4 (b), we show the *tile frequency* for 2048 (whether tile types follow the correct distribution); the *block frequency* for Tetris Block Puzzle (whether block types remain uniformly generated); and the *request sequence* for Server Allocation (whether requests follow the arrival distribution). Appendix K details additional environmental characteristics. Surprisingly, even an intervention rate as low as 1% causes noticeable deviations in all environments. Taken together with Figure 4 (a), an overly aggressive adversary does more than challenge the agent; it fundamentally distorts the environment, making the learned policy less effective.



(a) Performance

(b) Variation of environments

Figure 4: The impact of increasing interventions in evaluation. (a) Performance of RSZ and SAZ, normalized using SAZ's E_N score. (b) Variation from the theoretical environment; lower is better.

6 Conclusion and Discussions

We introduce *Robust Stochastic Zero* (RSZ), the first extension of SAZ for robust search-based planning under selective adversarial interventions in stochastic environments. To better anticipate the right intervention moments, we propose *robust stochastic Monte Carlo tree search* (RS-MCTS), which models the environment as a lurking adversary constrained to intervene only in critical afterstates, while leaving otherwise dynamics untouched. Across three evaluated benchmarks, RSZ improves robustness in adversarial settings while remaining comparable in natural settings. Overall, our work establishes a foundation for robust search-based planning in stochastic environments by leveraging adversarial training with sparse yet precise interventions.

As a beginning toward improving the robustness of search-based planning in stochastic environments, we highlight current limitations and propose future directions. First, extend RSZ with Stochastic MuZero. While RS-MCTS applies to it without major changes, challenges arise in learning the chance distributions $\boldsymbol{\rho}$ and the environment simulator. Second, improve training efficiency, e.g., with a hybrid method that uses SAZ/SMZ to learn quickly at the start and gradually switches to RSZ. Third, automatically control the intervention rate λ. As observed in the experiment, a large drop in tail performance may support increasing λ, potentially allowing for adaptive control. Finally, analyze the convergence properties and safety assurance of RS-MCTS. This provides a better understanding of its theoretical guarantee. Exploring these extensions would expand RSZ's application domain to support a broader range of high-value stochastic problems.

AI USE STATEMENT

In this work, we used generative AI tools for refining the presentation, finding related work, and double-checking correctness. This includes fixing typos and grammar issues, improving writing clarity, creating figures and tables, surveying the relevant literature, and identifying potential flaws and suggesting fixes. We always review content generated by AI tools. This includes making decisions on the AI recommendations, verifying the correctness of generated figures and tables, and reviewing the relevant literature found. We have not used AI tools for identifying research topics or designing methods. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The pseudocode of the proposed method is provided in Appendix B. The used benchmark environments are introduced in Appendix C. The training and evaluation specifications are detailed in Appendix D. Upon acceptance, we will release the source code and the trained models for reproducibility.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In *Advances in Neural Information Processing Systems*, volume 34, pp. 29304–29320. Curran Associates, Inc., 2021.

Alan Agresti. *Categorical Data Analysis*. John Wiley & Sons, New Jersey, 3rd edition, 2013.

Ioannis Antonoglou, Julian Schrittwieser, Sherjil Ozair, Thomas K. Hubert, and David Silver. Planning in Stochastic Environments with a Learned Model. In *International Conference on Learning Representations*, 2022.

Jose A. Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. RUDDER: Return Decomposition for Delayed Rewards. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

James Vandiver Bradley. *Distribution-Free Statistical Tests*. United States Air Force, 1960.

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, March 2012.

Zelei Cheng, Xian Wu, Jiahao Yu, Wenhai Sun, Wenbo Guo, and Xinyu Xing. StateMask: Explaining Deep Reinforcement Learning through State Mask. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-Sensitive and Robust Decision-Making: A CVaR Optimization Approach. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-Constrained Reinforcement Learning with Percentile Risk Criteria. *Journal of Machine Learning Research*, 18 (167):1–51, 2018a.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A Lyapunov-based Approach to Safe Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018b.

Gabriele Cirulli. 2048, success and me, 2014. URL https://web.archive.org/web/20140517152011/http://gabrielecirulli.com/articles/2048-success-and-me.

Rémi Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In *Computers and Games*, Lecture Notes in Computer Science, pp. 72–83, Berlin, Heidelberg, 2007. Springer.

Ivo Danihelka, Arthur Guez, Julian Schrittwieser, and David Silver. Policy improvement by planning with Gumbel. In *International Conference on Learning Representations*, April 2022.

Benjamin Eysenbach and Sergey Levine. Maximum Entropy RL (Provably) Solves Some Robust RL Problems. In *International Conference on Learning Representations*, 2022.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, October 2022.

Dima Feldman and Yuval Shavitt. An Optimal Median Calculation Algorithm for Estimating Internet Link Delays from Active Measurements. In *2007 Workshop on End-to-End Monitoring Techniques and Services*, pp. 1–7, May 2007.

Alison L. Gibbs and Francis Edward Su. On Choosing and Bounding Probability Metrics. *International Statistical Review*, 70(3):419–435, 2002. doi: 10.1111/j.1751-5823.2002.tb00178.x.

Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial Policies: Attacking Deep Reinforcement Learning. In *International Conference on Learning Representations*, 2020.

Ido Greenberg, Yinlam Chow, Mohammad Ghavamzadeh, and Shie Mannor. Efficient Risk-Averse Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 32639–32652, December 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Peide Huang, Mengdi Xu, Fei Fang, and Ding Zhao. Robust Reinforcement Learning as a Stackelberg Game via Adaptively-Regularized Adversarial Training. In *Thirty-First International Joint Conference on Artificial Intelligence*, volume 4, pp. 3099–3106, July 2022.

Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatain, Simon Schmitt, and David Silver. Learning and Planning in Complex Action Spaces. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 4476–4486. PMLR, July 2021.

Tobias Jacobs, Francesco Alesiani, and Gulcin Ermis. Reinforcement Learning for Route Optimization with Robustness Guarantees. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 2592–2598, August 2021.

Alexis Jacq, Johan Ferret, Olivier Pietquin, and Matthieu Geist. Lazy-MDPs: Towards Interpretable RL by Learning When to Act. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, pp. 669–677, Richland, SC, May 2022. International Foundation for Autonomous Agents and Multiagent Systems.

Chih-Yu Kao, Hung Guei, Ti-Rong Wu, and I-Chen Wu. Gumbel MuZero for the Game of 2048. In *2022 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pp. 42–47, Tainan, Taiwan, December 2022. IEEE.

Richard Klima, Daan Bloembergen, Michael Kaisers, and Karl Tuyls. Robust Temporal Difference Learning for Critical Domains. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, pp. 350–358, Richland, SC, May 2019. International Foundation for Autonomous Agents and Multiagent Systems.

Levente Kocsis and Csaba Szepesvári. Bandit Based Monte-Carlo Planning. In *Machine Learning: ECML 2006*, volume 4212 of *Lecture Notes in Computer Science*, pp. 282–293, September 2006.

Erwan Lecarpentier and Emmanuel Rachelson. Non-Stationary Markov Decision Processes, a Worst-Case Approach using Model-Based Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Yushuai Li, Hengyu Liu, Torben Bach Pedersen, Yuqiang He, Kim Guldstrand Larsen, Lu Chen, Christian S. Jensen, Jiachen Xu, and Tianyi Li. RobustZero: Enhancing MuZero Reinforcement Learning Robustness to State Perturbations. In *Forty-Second International Conference on Machine Learning*, June 2025.

Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 3756–3762, 2017. doi: 10.24963/ijcai.2017/525.

Haozhe Liu, Mingchen Zhuge, Bing Li, Yuhui Wang, Francesco Faccio, Bernard Ghanem, and Jürgen Schmidhuber. Learning to Identify Critical States for Reinforcement Learning from Videos. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1955–1965, Paris, France, October 2023. IEEE.

Ajay Mandlekar, Yuke Zhu, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Adversarially Robust Policy Learning: Active construction of physically-plausible perturbations. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3932–3939, September 2017.

Daniel J. Mankowitz, Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, Thomas Köppe, Kevin Millikin, Stephen Gaffney, Sophie Elster, Jackson Broshear, Chris Gamble, Kieran Milan, Robert Tung, Minjae Hwang, Taylan Cemgil, Mohammadamin Barekatain, Yujia Li, Amol Mandhane, Thomas Hubert, Julian Schrittwieser, Demis Hassabis, Pushmeet Kohli, Martin Riedmiller, Oriol Vinyals, and David Silver. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618(7964):257–263, June 2023.

Robert J. Moss, Arec Jamgochian, Johannes Fischer, Anthony Corso, and Mykel J. Kochenderfer. ConstrainedZero: Chance-Constrained POMDP Planning Using Learned Probabilistic Failure Surrogates and Adaptive Safety Constraints. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, volume 7, pp. 6752–6760, August 2024.

Karthik Somayaji Ns, Yu Wang, Malachi Schram, Jan Drgona, Mahantesh M. Halappanavar, Frank Liu, and Peng Li. Extreme Risk Mitigation in Reinforcement Learning using Extreme Value Theory. *Transactions on Machine Learning Research*, January 2024.

Sherjil Ozair, Yazhe Li, Ali Razavi, Ioannis Antonoglou, Aaron Van Den Oord, and Oriol Vinyals. Vector Quantized Models for Planning. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 8302–8313. PMLR, July 2021.

Xinlei Pan, Daniel Seita, Yang Gao, and John Canny. Risk Averse Robust Adversarial Reinforcement Learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8522–8528, May 2019.

Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust Deep Reinforcement Learning with Adversarial Attacks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, pp. 2040–2042, Richland, SC, July 2018. International Foundation for Autonomous Agents and Multiagent Systems.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 2817–2826, Sydney, NSW, Australia, August 2017. JMLR.org.

Marc Rigter, Bruno Lacerda, and Nick Hawes. One Risk to Rule Them All: A Risk-Sensitive Perspective on Model-Based Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 77520–77545, December 2023.

Christopher D. Rosin. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61(3):203–230, March 2011.

Pedro P. Santos, Jacopo Silvestrin, Alberto Sardinha, and Francisco S. Melo. Entropic Risk-Aware Monte Carlo Tree Search, February 2026.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588 (7839):604–609, December 2020.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, October 2017.

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

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Xiaohang Tang, Zhuowen Cheng, and Satyabrat Kumar. Robust Adversarial Reinforcement Learning in Stochastic Games via Sequence Modeling. In *NeurIPS 2025 Workshop: Reliable ML from Unreliable Data*, October 2025.

The Tetris Company, Inc. Tetris Block Puzzle, 2023. URL `https://tetris.com/products/video-game/tetris-block-puzzle`.

Chun-Jui Wang, Jian-Ting Guo, Hung Guei, Chung-Chin Shih, Ti-Rong Wu, and I-Chen Wu. Evaluating Game Difficulty in Tetris Block Puzzle. In *The 30th Game Programming Workshop (GPW-25)*, Kanagawa, Japan, November 2025.

Bernard L. Welch. The Generalization of Student's Problem When Several Different Population Variances Are Involved. *Biometrika*, 34(1-2):28–35, 1947. doi: 10.1093/biomet/34.1-2.28.

Ti-Rong Wu, Hung Guei, Pei-Chiun Peng, Po-Wei Huang, Ting Han Wei, Chung-Chin Shih, and Yun-Jui Tsai. MiniZero: Comparative Analysis of AlphaZero and MuZero on Go, Othello, and Atari Games. *IEEE Transactions on Games*, 17(1):125–137, March 2025.

Yousef Yassin and Junfeng Wen. RiskZero: Plan More to Risk Less with a Learned Model. In *Forty-third International Conference on Machine Learning*, 2026.

Huan Zhang, Hongge Chen, Duane S. Boning, and Cho-Jui Hsieh. Robust Reinforcement Learning on State Observations with Learned Optimal Adversary. In *International Conference on Learning Representations*, 2021.

Zuyuan Zhang, Arnob Ghosh, and Tian Lan. Tail-Risk-Safe Monte Carlo Tree Search under PAC-Level Guarantees, August 2025.

Ruiwen Zhou, Minghuan Liu, Kan Ren, Xufang Luo, Weinan Zhang, and Dongsheng Li. Is Risk-Sensitive Reinforcement Learning Properly Resolved?, November 2025.

A CRITICALITY MEASUREMENT

A.1 IMPLEMENTATION

As defined in Equation 7, criticality consists of two factors: severity and rarity,

$$\mathscr{T}(as) = \underbrace{\left(1 - \frac{q_\star}{\nu}\right)}_{\text{severity}} \underbrace{(1 - \rho_\star)}_{\text{rarity}}, \tag{7}$$

The current design represents a simple approach to our core concept: focusing on afterstates with significant negative impacts but low-probability events. While alternative designs are possible, we use this formulation to demonstrate RSZ, with the following implementation details.

Severity measures the value drop caused by the worst-case event $c_\star$, assuming a non-negative value range where $0 \leq q_\star \leq \nu < \infty$. Note that $q_\star$ should not exceed ν, as an afterstate as's value ν is the expected value of all events, i.e., $\sum_{c \in \mathcal{C}(as)} \rho_c q_c$. Specifically, we use a MCTS mean value for $q_\star$ and a network-predicted value for ν. This is designed so that the MCTS mean value $q_\star$ can immediately reflect catastrophes that occur during search, while the network output ν provides a stable estimate that is not affected by fluctuations from a small number of MCTS simulations. To ensure numerical stability, e.g., to handle practically observed noisy values that violate the condition, we apply regularization as $1 - \text{clip}(\frac{\min(q,\nu)}{\nu+\varepsilon}, 0, 1)$, where $\varepsilon = 10^{-8}$ is a constant. On the other hand, for value ranges where 0 is the optimal, $-\infty < q_\star \leq \nu \leq 0$, we maintain an increasing result by simply swapping the variables $q_\star$ and ν in the calculation. Finally, for value ranges spanning both negative and positive numbers, without loss of generality, a normalization can be done by finding the minimum and shifting all values to nonnegative.

Rarity measures the infrequency of the worst-case event $c_\star$. This requires that the probability satisfies $\rho_\star \in (0, 1]$, which is satisfied in SAZ-based RSZ as the perfect probability is available. However, for the SMZ-based RSZ implementation, regularization will be required to make $0 < \rho_\star \leq 1$. Note that a probability $\rho_\star = 0$ results in the maximum rarity in the current definition, but it must be avoided as such an event is impossible to occur in the target discrete stochastic domains.

A.2 EXAMPLES

Figure 1 (b) and (g) discuss a straightforward situation where severity and rarity are both high. Here, we provide examples of calculations.

Example 1 – same severity: Let two afterstates as_1 and as_2 have chance event values $\{1, 3, 3, 4, 4\}$ and $\{1, 3, 5\}$, with occurring probabilities $\{\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}\}$ and $\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$, respectively. In this case, it is clear that as_1 should be considered more critical: they have the same afterstate values $\nu_1 = \nu_2 = 3$, same worst-case outcomes $q_{\star 1} = q_{\star 2} = 1$, but as_1 has a lower probability for the worst case. Now we calculate their severity and rarity. For the severity, as_1 has $1 - \frac{1}{3} = \frac{2}{3}$ and so does as_2. For the rarity, as_1 has $1 - \frac{1}{5} = \frac{4}{5}$, but as_2 has $1 - \frac{1}{3} = \frac{2}{3}$. Finally, $\mathscr{T}(as_1) = \frac{2}{3} \times \frac{4}{5} > \mathscr{T}(as_2) = \frac{2}{3} \times \frac{2}{3}$.

Example 2 – same rarity: Let another two afterstates as_3 and as_4 have chance event values $\{1, 3, 3\}$ and $\{2, 3, 3\}$, both with the probabilities $\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$. For this case, the rarity of their worst-case events is the same, so as_3 is more critical, derived from their worst-case event values $q_{\star 3} = 1 < q_{\star 4} = 2$. For the severity, as_3 has $1 - \frac{1}{(\cdot)} = \frac{4}{7}$, while as_4 has $1 - \frac{2}{(\cdot)} = \frac{2}{8}$. For the rarity, both as_3 and as_4 have $1 - \frac{1}{3} = \frac{2}{3}$. Finally, $\mathscr{T}(as_3) = \frac{4}{7} \times \frac{2}{3} > \mathscr{T}(as_4) = \frac{2}{8} \times \frac{2}{3}$.

A.3 CRITICALITY IN RS-MCTS

Measuring criticality requires $q_\star$, which is by definition the minimum of all chance events' mean $\min\{q_c \mid c \in \mathcal{C}(as)\}$. Finding the minimum is computationally costly during the search, as it requires expanding and evaluating all of the chance nodes. To improve efficiency, we still expand and evaluate only one node at a time, and derive $q_\star = \min\{q_c \mid c \in \mathcal{C}(as) \wedge n_c > 0\}$ solely from the set of previously visited chance events. However, a challenge arises at the *initial visit* of as, since none of its chance events have been visited yet. Note that during the initial visit, it only yields the afterstate value ν, chance event probabilities $\boldsymbol{\rho}$, and the adversarial policy $\boldsymbol{\delta}$ via the afterstate prediction function ψ. As all event values q_c are unavailable, it cannot immediately determine whether as is critical.

To resolve this, when as is selected as a non-leaf node for the first time (i.e., the first visit after its expansion), we bypass the criticality check and apply the pUCT selection (Equation 9). At this moment, the pUCT selection is totally driven by adversarial probability $\delta_c \in \boldsymbol{\delta}$. As noted in Subsection 4.4, $\boldsymbol{\delta}$ is trained to predict the worst-case event $c_\star$ for critical afterstates but defaults to a uniform distribution otherwise. If as is indeed critical, we expect $\boldsymbol{\delta}$ to exactly target the worst-case event, resulting in a value $q_\star$ that triggers $\mathscr{T}(as) \geq \tau_{\text{th}}$, allowing as to keep intervening if it is visited again later. Conversely, if as is not critical, the resulting $q_\star$ will likely not satisfy $\mathscr{T}(as) \geq \tau_{\text{th}}$, allowing the method to switch to QRS for subsequent selections.

RS-MCTS has the ability to discover critical afterstates without the need for the adversarial policy $\boldsymbol{\delta}$. In MCTS, as the number of visits increases, the mean value of each node converges to its optimal, and the visits eventually converge on the node with the optimal value. Since the criticality fundamentally relies only on the value estimate and true probability, even if $\boldsymbol{\delta}$ wrongly predicts at first and causes the selection to fall to QRS, as each child node will still be accordingly visited by QRS, their mean value will still be discovered, allowing RS-MCTS to eventually enter pUCT for the intervention. Namely, RS-MCTS will find the critical afterstates with sufficient simulation budgets.

Even when the afterstate criticality is sufficiently high, it does not guarantee that the worst-case event $c_\star$ will always be selected, because pUCT involves exploration. To prevent such explorations from updating the focus to ordinary events, the strengthened backup rules explicitly exclude them:

$$q^i \leftarrow \begin{cases} \dfrac{\sum_a n_a q_a + v^i}{n^i + 1} & \text{if } s^i. \qquad (10) \\ \dfrac{\sum_c n_c (r_c + \gamma q_c) + \nu^i}{n^i + 1} & \text{if } as^i \text{ is not intervened, i.e., } \tau^i < \tau_{\text{th}} \vee q^{i+1} > q^i_\star. \qquad (11) \\ r^{i+1} + \gamma q^{i+1} & \text{if } as^i \text{ is intervened, i.e., } \tau^i \geq \tau_{\text{th}} \wedge q^{i+1} = q^i_\star. \qquad (12) \end{cases}$$

Here, $q^i_\star = \min\{q_c \mid c \in \mathcal{C}(as^i) \wedge n_c > 0\}$ is the worst-case event mean of as^i. When a pUCT exploration happens, the subsequent selected node s^{i+1} becomes not as^i's worst-case event $c^i_\star$, thereby letting its mean value $q^{i+1} > q^i_\star$. For this case, we let the backup fall back to Equation 11, resulting in the worst-case event $c^i_\star$'s share in q^i being reduced. Such a change in q^i will result in a decrease in τ^i, namely, making as^i seem less critical. Nevertheless, if as^i is indeed critical, it may still satisfy $\tau^i \geq \tau_{\text{th}}$, and when the next pUCT returns to exploiting $c^i_\star$, q^i will immediately focus back. On the other hand, if $\tau^i < \tau_{\text{th}}$, indicating that as^i is not really critical, and it will go back to QRS from pUCT, recovery from a wrongly identified critical afterstates.

A.4 Criticality threshold estimation

As introduced in Subsection 4.2, the criticality threshold τ_{th} is estimated by an online quantile estimator modified from the *Fast Algorithm for Median Estimation* (FAME) (Feldman & Shavitt, 2007). Such an estimator observes the criticality $\tau = \mathscr{T}(as)$ for all afterstates as that happen and updates τ_{th} accordingly:

$$\tau_{\text{th}} \leftarrow \begin{cases} \tau_{\text{th}} \leftarrow \tau_{\text{th}} - \lambda k & \text{if } \tau_{\text{th}} > \tau, \\ \tau_{\text{th}} \leftarrow \tau_{\text{th}} + (1-\lambda) k & \text{if } \tau_{\text{th}} < \tau, \end{cases} \qquad (14)$$

where k is the step size of the update, which gradually decreases during threshold fitting. In practice, as multiple concurrently running games can contribute to a single instance of the estimator, it can quickly estimate τ_{th} for targeting critical afterstates.

However, accurately targeting the threshold can be challenging since the distribution of criticality measures is often highly imbalanced, varies from agent to agent, and can shift during training when the agent's behavior changes rapidly. For example, in an extreme scenario where 99% and 1% of observed τ are 0s and 1s respectively, there is no effective threshold to exactly target the top 0.01% critical afterstates.

To handle these scenarios, we additionally design a *target controller* to further improve the estimation of τ_{th}. Here, we break down the intervention rate in more detail: λ^* is the overall target intervention rate that is fixed during training; λ is a varying target used by the threshold estimator, which can be adjusted by the controller; $\hat{\lambda}$ is the true intervention rate of the adversary observed from recent records. The controller operates periodically, monitoring whether the intervention rate $\hat{\lambda}$ in recent decision

history is significantly off the desired target intervention rate λ^*. If this happens, the estimator's current target λ is adjusted, allowing the estimator to fit a more relaxed target. For example, if we have an observed rate $\hat{\lambda} = 0.005\%$ under a desired rate $\lambda^* = 0.01\%$, we would expect more interventions. The intervention rate error is $\lambda^* - \hat{\lambda} = 0.005\%$, and the adjustment will be $\lambda \leftarrow \lambda + 0.005\%$. A larger λ allows the estimator to find τ_{th} at a lower quantile, effectively triggering more interventions to reduce the error. The pseudocode for the mechanism is provided in Appendix B.

B PSEUDOCODE

In this section, we provide pseudocode for all proposed algorithms. First, an overview of the Robust Stochastic Zero (RSZ) algorithm is provided in Algorithm 1. The robust stochastic Monte Carlo tree search (RS-MCTS) is shown in Algorithm 2, and its planning phases, selection, expansion, backup, and decision, are detailed in Algorithm 3, Algorithm 4, Algorithm 5, and Algorithm 6, respectively. For the criticality threshold τ_{th}, Algorithm 7 and Algorithm 8 show the detailed control mechanisms, including the estimator and the target controller.

Algorithm 1 Robust Stochastic Zero (RSZ)

Parameter: intervention rate λ^*
Output: network model parameters $\boldsymbol{\theta}$

$\boldsymbol{\theta} \leftarrow$ initialize network parameters
$\mathcal{T} \leftarrow$ initialize criticality threshold estimator
while training **do**
  while collecting self-play records for a batch **do**
    start a new episode
    while the episode is not finished **do**
      $o \leftarrow$ obtain the current observation (agent's turn or environment's turn)
      $\tau_{\text{th}} \leftarrow$ obtain latest threshold from $\mathcal{T}$
      make a decision by RS-MCTS using Algorithm 2
      apply the decision (action or chance event) to the environment
      update $\mathcal{T}$ using Algorithm 7 if it is a chance event
    end while
    add the finished episode into the self-play record
  end while
  update $\mathcal{T}$ using Algorithm 8
  optimize network parameters $\boldsymbol{\theta}$
end while
return $\boldsymbol{\theta}$

Algorithm 2 Robust stochastic Monte Carlo tree search (RS-MCTS)

Input: current observation o, criticality threshold τ_{th}
Parameter: search budget each move
Output: decision
▷ The current observation can be either for the agent's turn or the environment's turn

initialize the root node
while search budget available **do**
  run selection phase using Algorithm 3 ▷ Subsection 4.3 Selection
  run expansion phase using Algorithm 4 ▷ Subsection 4.3 Expansion
  run backup phase using Algorithm 5 ▷ Subsection 4.3 Backup
end while
run decision phase using Algorithm 6 ▷ Subsection 4.3 Decision
return decision

Algorithm 3 Selection in RS-MCTS

Input: root node s^0, criticality threshold τ_{th}
Output: selected nodes $s^0, as^1, s^2, \ldots, s^\ell$
▷ Both the root and leaf nodes can be a state or an afterstate; here we write s^0 and s^ℓ for simplicity.

$i \leftarrow 0$
while s^i (or as^i) is not a leaf node **do**
  if s^i **then**
    $a^{i+1} \leftarrow \arg\max_a \left[q_a + p_a \dfrac{\sqrt{\sum_b n_b}}{n_a + 1} \eta_{\text{puct}} \right]$ ▷ Equation 2
    Select the node as^{i+1} corresponding to a^{i+1}
  else if as^i **then**
    $\tau^i \leftarrow \mathscr{T}(as^i)$ ▷ Equation 7
    if $\tau^i \geq \tau_{\text{th}}$ **or** $n^i = 1$ **then**
      $c^{i+1} \leftarrow \arg\max_c \left[-q_c + \delta_c \dfrac{\sqrt{\sum_b n_b}}{n_c + 1} \eta_{\text{puct}} \right]$ ▷ Equation 9
    else
      $c^{i+1} \leftarrow \arg\max_c \left[\dfrac{\rho_c}{n_c + 1} \right]$ ▷ Equation 3
    end if
    Select the node s^{i+1} corresponding to c^{i+1}
  end if
  $i \leftarrow i + 1$
end while
$\ell \leftarrow i$
return all selected nodes $s^0, as^1, s^2, \ldots, s^\ell$

Algorithm 4 Expansion in RS-MCTS

Input: leaf node s^ℓ (or as^ℓ)
Parameter: neural network parameters $\boldsymbol{\theta}$
Output: leaf node value v^ℓ and action reward r^ℓ (or leaf node value ν^ℓ)
▷ In SAZ, the perfect environment simulator provides g, ϕ, and $\boldsymbol{\rho}$ in ψ.

if s^ℓ **then**
  $s^\ell, r^\ell \leftarrow g(as^{\ell-1}, c^\ell)$ ▷ *dynamics* function
  $\boldsymbol{p}^\ell, v^\ell \leftarrow f(s^\ell)$ ▷ *prediction* function
  for $a \in \mathcal{A}(s^\ell)$ **do**
    Expand s^ℓ's child node for a: $q_a \leftarrow 0$, $n_a \leftarrow 0$, and $p_a \in \boldsymbol{p}^\ell$
  end for
  return v^ℓ and r^ℓ
else if as^ℓ **then**
  $as^\ell \leftarrow \phi(s^{\ell-1}, a^\ell)$ ▷ *afterstate dynamics* function
  $\boldsymbol{\delta}^\ell, \boldsymbol{\rho}^\ell, \nu^\ell \leftarrow \psi(as^\ell)$ ▷ *afterstate prediction* function
  for $c \in \mathcal{C}(as^\ell)$ **do**
    Expand as^ℓ's child node for c: $q_c \leftarrow 0$, $n_c \leftarrow 0$, $\delta_c \in \boldsymbol{\delta}^\ell$, and $\rho_c \in \boldsymbol{\rho}^\ell$
  end for
  return ν^ℓ
end if

Algorithm 5 Backup in RS-MCTS

Input: selected nodes $s^0, as^1, \ldots, s^\ell$, leaf node value v^ℓ (or ν^ℓ), reward r^ℓ (if applicable)
▷ Both the root and leaf nodes can be a state or an afterstate; here we write s^0 and s^ℓ for simplicity.

if s^ℓ **then**
  Set value v^ℓ for s^ℓ
  Set reward r^ℓ for the transition from $as^{\ell-1}$ to s^ℓ
else if as^ℓ **then**
  Set afterstate value ν^ℓ for as^ℓ
end if
for s^i (or as^i) **in** $s^\ell, as^{\ell-1}, \ldots, s^0$ **do**
  if s^i **then**
    $q^i \leftarrow \dfrac{\sum_a n_a q_a + v^i}{n^i + 1}$ ▷ Equation 10
  else if as^i **when** $\tau^i < \tau_{\text{th}}$ **or** $q^{i+1} > \min\{q_c \mid c \in \mathcal{C}(as^i) \wedge n_c > 0\}$ **then**
    $q^i \leftarrow \dfrac{\sum_c n_c(r_c + \gamma q_c) + \nu^i}{n^i + 1}$ ▷ Equation 11
  else if as^i **when** $\tau^i \geq \tau_{\text{th}}$ **and** $q^{i+1} = \min\{q_c \mid c \in \mathcal{C}(as^i) \wedge n_c > 0\}$ **then**
    $q^i \leftarrow r^{i+1} + \gamma q^{i+1}$ ▷ Equation 12
  end if
  $n^i \leftarrow n^i + 1$
  Update tree value normalization bound with q^i
end for

Algorithm 6 Decision in RS-MCTS

Input: root node s^0 (or as^0), criticality threshold τ_{th}
Output: an action a^* (or a chance event c^*)

if s^0 **then**
  $\boldsymbol{\pi} \leftarrow \text{softmax}(\{n_a \mid a \in \mathcal{A}(s^0)\})$ ▷ softmax of action visit counts
  return $a^* \sim \boldsymbol{\pi}$
else if as^0 **then**
  $\tau^0 \leftarrow \mathscr{T}(as^0)$ ▷ Equation 7
  if $\tau^0 \geq \tau_{\text{th}}$ **then**
    $\boldsymbol{\mu} \leftarrow \text{softmax}(\{n_c \mid c \in \mathcal{C}(as^0)\})$ ▷ softmax of event visit counts n_c
  else
    $\boldsymbol{\mu} \leftarrow \boldsymbol{\rho}^0$ ▷ chance event probabilities
  end if
  return $c^* \sim \boldsymbol{\mu}$
end if

Algorithm 7 Criticality threshold estimator

Input: observed criticality measure τ
Parameter: criticality threshold τ_{th}, varying intervention rate λ, step size k
Output: updated criticality threshold τ_{th}

if $\tau_{\text{th}} > \tau$ **then**
  $\tau_{\text{th}} \leftarrow \tau_{\text{th}} - \lambda k$
else if $\tau_{\text{th}} < \tau$ **then**
  $\tau_{\text{th}} \leftarrow \tau_{\text{th}} + (1 - \lambda)k$
end if
if $|\tau_{\text{th}} - \tau| < k$ **then**
  $k \leftarrow 0.5k$
end if
return τ_{th}

Algorithm 8 Target controller for criticality threshold estimator

Input: self-play records D of the current iteration
Parameter: target intervention rate λ^*, varying intervention rate λ, step size k
$\triangleright$ λ^* is the overall desired target, λ is a varying target of the estimator, and $\hat{\lambda}$ is the self-play result.

$\hat{\lambda} \leftarrow$ calculate intervention rate in D
$\varepsilon \leftarrow \lambda^* - \hat{\lambda}$
if $|\varepsilon| < 0.05\,\lambda^*$ **then**
  do nothing $\triangleright$ the error is sufficiently small
else
  if k was just reset in the last iteration **then**
    $\lambda \leftarrow \text{clip}\big(\lambda + \varepsilon, 0, 1\big)$ $\triangleright$ adjust intervention rate for the estimator
  end if
  $k \leftarrow 0.2$ $\triangleright$ reset step size for the estimator
end if

C Benchmark environments

C.1 Environmental rules

2048 (Cirulli, 2014) is a single-player puzzle game, with a goal of sliding tiles and merging smaller tiles into larger tiles. 2048 is played on a 4×4 puzzle, where all tiles are numbered in powers of two. At each turn, the player can slide the puzzle left, right, up, or down to move all tiles as far as possible in the chosen direction. When two k-tiles collide during a slide, they are merged into a single $2k$-tile, and the player receives a reward of $2k$. For example, two 2-tiles can be merged into a single 4-tile, receiving 4 points. After each move, the environment uniformly selects an empty location to generate a new 2-tile or 4-tile with probabilities 90% and 10%, respectively. The goal of 2048 is to achieve the 2048-tile and accumulate as many points as possible until no further moves are available.

Tetris Block Puzzle (The Tetris Company, Inc, 2023) is a single-player puzzle game, with the goal of placing tetrominos on the puzzle and filling lines to obtain scores. This game has several variants; we use the one described below. The game is played on an 8×8 board, and starts with two holding blocks. Each turn, the player selects one tetromino from the holding blocks and places it on the board without any rotation. When a line (a row or column) is filled, it is cleared, and the player receives a reward of one point. After each placement, a new holding block will be uniformly sampled from the 19 possible Tetris tetrominoes (7 base shapes, including all valid rotations). The game ends when there are no more spaces to place the tetromino, or when a maximum of 13,500 steps has been reached.

Server Allocation (Greenberg et al., 2022) is a single-agent resource management problem that models the control of computational capacity. In this work, we adapt the rules as follows. The environment evolves every second over a one-hour horizon, and each active server processes one request per second. The agent adjusts active servers (bounded between 3 and 10, initialized at 4) by keeping the current number, adding one, or removing one. User requests arrive randomly from a capped Poisson distribution with a minimum of 0 and a maximum of 19 requests per second, where the Poisson rate is updated using an exponential moving average (EMA). Demand usually remains near baseline but may occasionally spike to a high-load peak, with an expected interval of three days. The agent's goal is to minimize the total cost, including the running cost of hosting servers and the waiting cost of pending requests.

C.2 Characteristics of stochasticity

The three environments share the commonality that their discrete chance events operate under a different mechanism than the agent's actions. However, the detailed characteristics of the stochasticity differ, specifically in how chance events can occur at each specific afterstate. In 2048, only a subset of events can occur at each afterstate. In Tetris Block Puzzle, all events can occur at each afterstate, with a uniform probability. In Server Allocation, all events can occur at each afterstate, while certain events are favored based on afterstate conditions.

D EXPERIMENT DETAILS

D.1 IMPLEMENTATION

Our RSZ and SAZ implementations are built upon MiniZero (Wu et al., 2025), a publicly available AlphaZero/MuZero framework available at `https://github.com/rlglab/minizero`. To improve training efficiency, our RSZ and SAZ algorithms are both integrated with Gumbel AlphaZero (GAZ) (Danihelka et al., 2022), which is provided by the framework. However, we make the following slight modifications. At the agent's turn, Gumbel AlphaZero's selection mechanism is applied at the root node, using sequential halving to allocate the search budget; however, at the adversary's turn, we apply standard RSZ selection without using Gumbel.

For the network architecture, we follow Stochastic AlphaZero and use a residual network (He et al., 2016) with the number of blocks and channels provided in the subsequent subsection. The input features for 2048, Tetris Block Puzzle, and Server Allocation are following Kao et al. (2022), Wang et al. (2025), and Greenberg et al. (2022), respectively:

- **2048**: The state is represented as a 4×4 spatial grid with 16 channels. Each channel acts as a one-hot encoding for the 16 possible tile values (an empty cell or a specific power of two), marking matching spatial positions with 1.0.
- **Tetris Block Puzzle**: The 8×8 board is encoded using two binary channels for empty and filled cells. To represent the player's actions, 19 channels are appended for each holding block slot, with the channel corresponding to the currently held tetromino set to 1.0.
- **Server Allocation**: The state is structured as 1×1 with 9 channels. The first 8 channels provide a one-hot encoding of the number of active servers, while the final channel encodes the pending user requests, scaled down by a factor of 30.0.

D.2 EXPERIMENT SETUP

To reduce the influence of outliers under the computation budget, we first train five models for each setting and evaluate them under the natural evaluation setting $\mathrm{E_N}$. Each evaluation uses 200 episodes to obtain a stable estimate of the average return. Similar to the concept of interquartile mean (IQM) (Agarwal et al., 2021), we remove the best- and worst-performing models and retain the three median-performing models for adversarial evaluation and further analysis. Thus, the reported $\mathrm{E_N}$ performance is the average return over these three retained models, which is essentially the 20% trimmed mean from the original five model evaluation results. For adversarial evaluations E_λ with $\lambda > 0$, we compare each player setting with each adversary setting by pairwise matching the retained player and adversary models, resulting in 3×3 player–adversary evaluations. During these evaluations, each adversary initializes its criticality threshold τ_{th} using the value obtained from its own training process, allowing it to start from a reasonable threshold rather than from scratch.

The experiments are conducted on Ubuntu 20.04 workstations equipped with two E5-2678 v3 CPUs, 256GB of memory, four GTX 1080 Ti GPUs, and at least 10TB of available storage. Training or evaluating a single model only needs one workstation, but we use multiple workstations to train independent models in parallel. The hyperparameters for training are listed in Table 4.

D.3 COMPUTATIONAL BUDGET AND EVALUATION TRADE-OFFS

Search-based planning makes both independent training runs and evaluations costly. Furthermore, single-agent stochastic environments can exhibit high variance. Episode returns vary even for a fixed trained model, and independent training runs add another source of variation. With computational budget, we therefore train five independent models per setting, use their natural evaluation scores to retain the middle three, and evaluate those models against three retained adversaries. This reduces the number of player–adversary pairs to $3 \times 3 = 9$ per setting, while still requiring all five training runs and their natural evaluations. That is, three retained players, each with 200 episodes in natural evaluations and 600 episodes in adversarial evaluation. The resulting average is a 20% trimmed mean across training runs, and the reported $\pm$ values describe variation across retained players. Taken together, trimming focuses on median training results, and increasing the number of tested episodes to 600 (or 1,800) improves the reliability of their mean scores for the comparison.

Table 4: Hyperparameters for training.

Parameter	**2048**	**Tetris Block Puzzle**	**Server Allocation**
Optimizer	Adam	Adam	Adam
Optimizer: learning rate	0.001	0.001	0.001
Optimizer: weight decay	0	0	0
Discount factor	0.999	0.999	0.999
Priority exponent (α)	0.6	0.6	0.6
Priority correction (β)	0.4	0.4	0.4
Bootstrap step (n-step return)	1	1	1
# Self-play actors	512	512	512
MCTS simulation	50	50	50
Softmax temperature	1	1	1
Iteration	500	500	500
Training steps	32,000	32,000	32,000
Batch size	1,024	1,024	1,024
# Blocks	3	3	1
# Channels	256	256	64
Replay buffer size	1,000,000	1,000,000	1,000,000
Gumbel c_{visit}	50	50	50
Gumbel c_{scale}	1	1	1
Gumbel sample actions (m)	3	12	3
Adversarial intervention rate (λ)	0.0001	0.0001	0.0001
Adversarial policy loss scale (η_{adv})	0.5	0.5	0.5

E Training Statistics

E.1 Returns

During training, we track the average self-play return, as shown in Figure 5, and report results averaged across the three retained models for each method. From the curves, it appears that RSZ can consistently perform worse than SAZ, particularly in 2048 and Tetris Block Puzzle. This difference is expected because RSZ performs self-play under adversarial interventions, whereas SAZ is trained in the natural environment. When both methods are compared under the same evaluation settings, RSZ matches or outperforms SAZ, as reported in Table 1 of the main text. In Server Allocation, learning is substantially more unstable, likely because high-load peaks are extremely rare but can produce large negative rewards when they occur. The decreasing variance of RSZ over training further supports this interpretation. Early in training, even infrequent interventions can destabilize learning. As training progresses, these interventions repeatedly expose the agent to such vulnerabilities, allowing it to adapt and leading to more stable returns with narrower deviations.

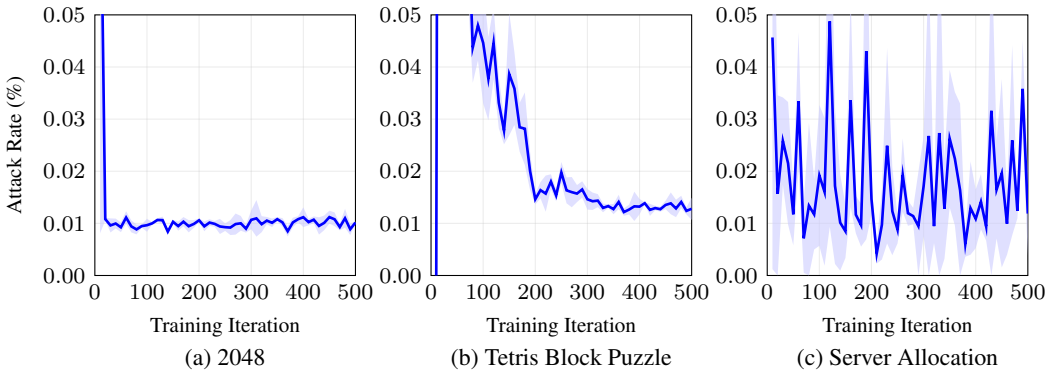


(a) 2048 (b) Tetris Block Puzzle (c) Server Allocation

Figure 5: Average self-play returns during training. Higher is better. Note that SAZ is trained without interventions, while RSZ is trained with a rate of 0.01%.

E.2 INTERVENTION RATES

For the RSZ agents, we also track the average intervention rates during training, as shown in Figure 6. The intervention rate in 2048 remains quite stable during training, reflecting a better distinguishable safe and risky afterstates. On the other hand, the intervention rate in Tetris Block Puzzle mostly remains above the target throughout training, especially at the beginning. Even more, it becomes highly inaccurate in Server Allocation. The potential reason is the situation discussed in Appendix A.4, where the distribution of criticality is highly imbalanced for a threshold to fit.

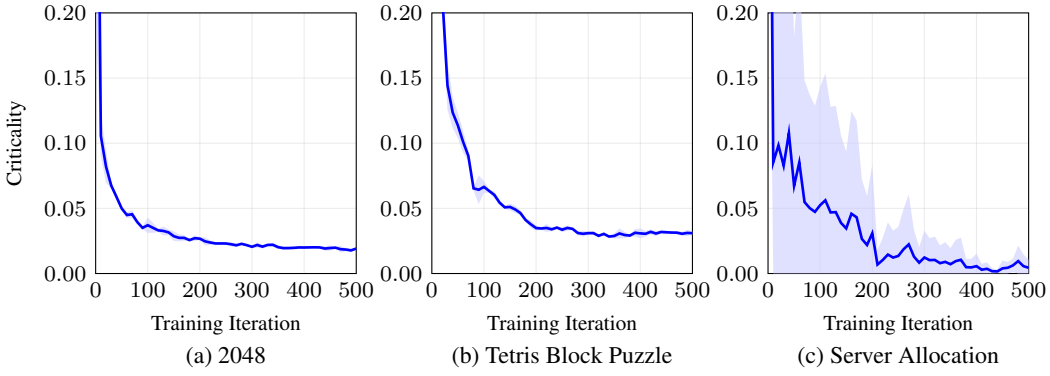


(a) 2048 (b) Tetris Block Puzzle (c) Server Allocation

Figure 6: Average intervention rates observed during training. Target is 0.01%.

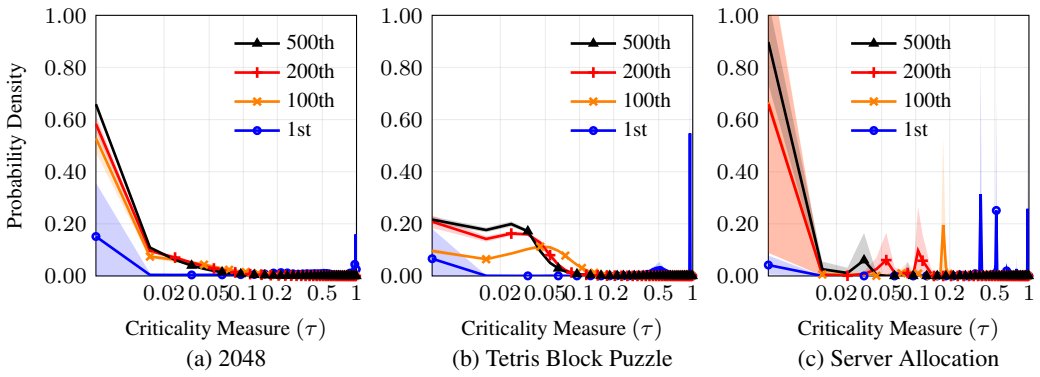
E.3 CRITICALITY

As criticality is key to RSZ, we monitor the average criticality measure, as shown in Figure 7. The decreasing curves across three environments clearly show that, with adversarial training, agents gradually learn to produce afterstates that are less identifiable to the adversary. Furthermore, to gain better insights into how criticality evolves at different stages of training, we visualize the distributions of the criticality measure scores at 1st, 100th, 200th, and 500th iterations, with probability density and cumulative distribution functions plotted in Figure 8 and Figure 9, respectively.



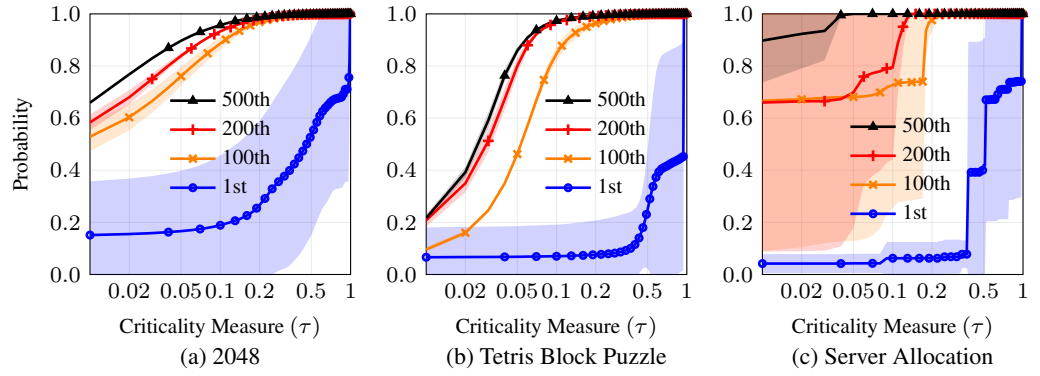
(a) 2048 (b) Tetris Block Puzzle (c) Server Allocation

Figure 7: Average criticality scores observed during training.



(a) 2048 (b) Tetris Block Puzzle (c) Server Allocation

Figure 8: Probability density of the criticality measure during training.



(a) 2048 (b) Tetris Block Puzzle (c) Server Allocation

Figure 9: Cumulative distribution of criticality measure during training.

From the distribution change, we observe that the afterstate criticality gradually shifts from relatively high to low, reflecting improved robustness. However, the distributions evolve substantially differently across environments. Across all environments, the initial stage of training (1st iteration, blue line) exhibits a high probability density at higher criticality values, often spiking near 1. This indicates that an untrained agent perceives most states and actions as highly critical, as it has not yet learned to be robust. In (a) 2048, the shift toward low criticality is smooth. By the 500th iteration, the entire probability density is concentrated near a criticality measure of 0, with a fraction below 0.01 of approximately 65.91%. This suggests that the agent quickly learns a stable policy in which individual moves become less critical. In (b) Tetris Block Puzzle, the distribution also shifts leftwards as training progresses. However, rather than collapsing completely to zero, the density forms a distinct peak that narrows and centers approximately between 0.02 and 0.05. This implies a baseline level of criticality is maintained. In (c) Server Allocation, it exhibits the most instability and the highest variance. The distribution is also the most concentrated, with around 89.75% of values below 0.01 by the 500th iteration. The highly unbalanced distribution reflects why the intervention rate shown in Figure 6 is less accurate than in the other two environments.

E.4 Time

Under our hardware setting (Subsection D.2), training one RSZ model takes approximately 13, 23, and 8 hours for 2048, Tetris Block Puzzle, and Server Allocation, respectively. Training one SAZ model takes roughly half as long as training one RSZ model because RSZ additionally runs tree search for the adversary. For the main experiments in Subsection 5.1, we train one RSZ setting and one SAZ setting. For the comparison experiments in Subsection 5.2, we train four RSZ-based settings (Full-Attack, Random-Attack, Standard-Backup, and Severity-Only) and one SAZ-based setting (No-Attack). Since each setting is trained with five models across three environments, the total training time is approximately 1,320 hours using a single machine:

- Main: $5 \times (13 + 23 + 8) + 5 \times (13 + 23 + 8) \times 0.5 = 330$ hours.
- Comparison: $4 \times 5 \times (13 + 23 + 8) + 5 \times (13 + 23 + 8) \times 0.5 = 990$ hours.

F Evaluation statistics

F.1 Untrimmed statistics

As outliers commonly occur during training in stochastic environments, we typically report the median-performing models in our primary experiments. In Table 5, we present the full E_N evaluation results for both SAZ and RSZ models across three environments for five independent trials. Note that the standard deviation reported for each individual trial represents the variance that arises during evaluation—specifically, across 200 evaluation episodes. Conversely, the deviations for "Mean" and "Trimmed" represent the standard deviation of the mean values across the distinct trials. Furthermore, we reproduce Table 1 with all five models, increasing the required adversarial evaluation player–adversary pairs from 3×3 to 5×5. Table 6 summarizes the full performance results. Compared with results that retain only three, it shows substantially higher variance, while the advantage of RSZ still holds. Overall, applying a 20% trimmed mean reduces inter-trial variance across all environments, while not changing the comparison between SAZ and RSZ much. This allows us to still evaluate the median training results within the computation budget.

Table 5: Performance comparison under E_N including overall and 20% trimmed mean. Underlines mark the highest and lowest trials for each method and environment; these trials are excluded from the trimmed statistics.

	2048		**Tetris Block Puzzle**		**Server Allocation**	
Trial	**SAZ**	**RSZ**	**SAZ**	**RSZ**	**SAZ**	**RSZ**
1	159,094 ± 62,268	147,404 ± 47,573	3,650 ± 2,323	4,119 ± 2,441	-27.70 ± 5.40	-8.81 ± 0.14
2	156,846 ± 60,976	166,059 ± 51,013	3,425 ± 2,328	3,801 ± 2,469	-8.79 ± 0.07	-9.09 ± 0.13
3	174,543 ± 57,666	164,601 ± 53,271	3,915 ± 2,423	3,791 ± 2,457	-19.76 ± 0.21	-13.18 ± 0.06
4	160,846 ± 60,927	157,484 ± 44,355	3,994 ± 2,462	3,848 ± 2,441	-8.77 ± 0.11	-11.77 ± 1.58
5	160,975 ± 63,106	156,023 ± 53,277	3,508 ± 2,444	3,811 ± 2,406	-15.86 ± 0.15	-8.41 ± 0.10
Overall	162,461 ± 6,958	158,315 ± 7,491	3,698 ± 249	3,874 ± 139	-16.18 ± 7.99	-10.25 ± 2.10
Trimmed	160,305 ± 1,051	159,370 ± 4,589	3,691 ± 206	3,820 ± 25	-14.80 ± 5.56	-9.89 ± 1.63

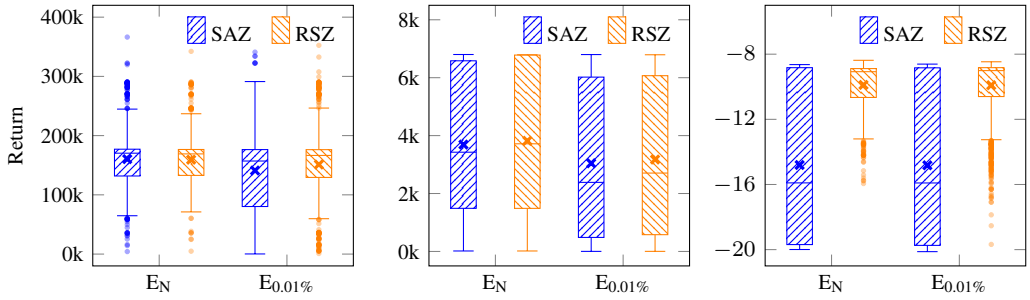
Table 6: Performance comparison under E_N and $E_{0.01\%}$ using all five trained models. Reported statistics are mean return ± standard deviation; boldface indicates the higher mean.

Method	**2048**		**Tetris Block Puzzle**		**Server Allocation**	
	E_N	$E_{0.01\%}$	E_N	$E_{0.01\%}$	E_N	$E_{0.01\%}$
			Overall Performance			
SAZ	**162,461 ± 6,958**	144,302 ± 7,813	3,698 ± 249	2,136 ± 478	-16.18 ± 7.99	-16.15 ± 7.93
RSZ	158,315 ± 7,491	**150,939 ± 2,813**	**3,874 ± 139**	**2,408 ± 410**	**-10.25 ± 2.10**	**-11.56 ± 2.16**
			Tail Performance ($CVaR_{10\%}$)			
SAZ	63,001 ± 5,950	33,630 ± 7,572	266 ± 84	25 ± 2	-18.31 ± 11.81	-18.46 ± 12.08
RSZ	**71,603 ± 4,190**	**54,943 ± 6,476**	**295 ± 69**	**28 ± 5**	**-10.99 ± 2.79**	**-12.95 ± 3.29**

F.2 Return distributions

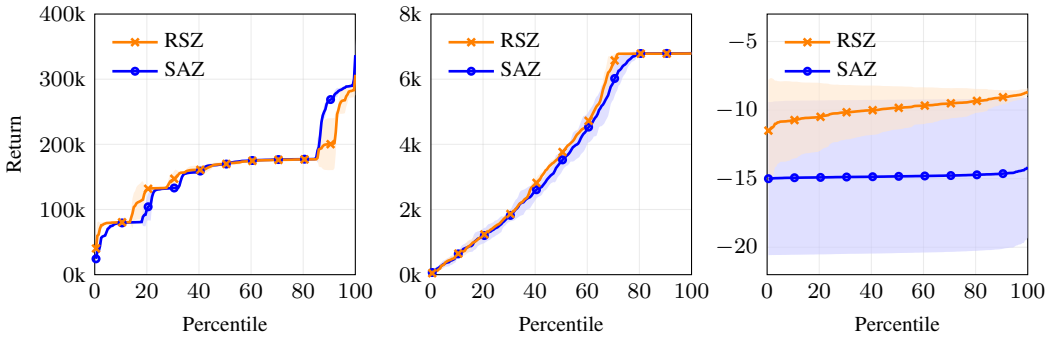
Besides the tail performance discussed in Table 1 of the main text, we further analyze the detailed return distributions using the box plots in Figure 10. In (a) 2048, we observe that while both SAZ and RSZ achieve comparable median returns, RSZ demonstrates a tighter interquartile range under both natural (E_N) and adversarial ($E_{0.01\%}$) evaluation conditions. This tightening indicates a policy less vulnerable to failures during gameplay. In (b) Tetris Block Puzzle, however, the improvements

are less clear and mainly appear in Q2 and the mean. This potentially indicates that the intervention rate of 0.01% may be too low for RSZ to produce improvements in this environment. In (c) Server Allocation, RSZ demonstrates a massive reduction in variance compared to SAZ, lifting distributions across both evaluation settings, confirming its superior stability.



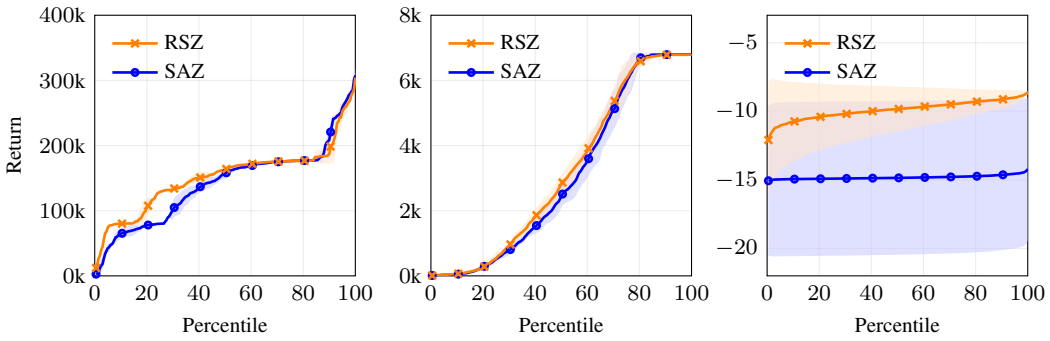
(a) 2048 (b) Tetris Block Puzzle (c) Server Allocation

Figure 10: Box plot for natural evaluation E_N and adversarial evaluation $E_{0.01\%}$.



(a) 2048 (b) Tetris Block Puzzle (c) Server Allocation

Figure 11: Percentile plots for the episode returns in the natural evaluation E_N.



(a) 2048 (b) Tetris Block Puzzle (c) Server Allocation

Figure 12: Percentile plots for the episode returns in the adversarial evaluation $E_{0.01\%}$.

We also investigate the returns of each episode during evaluation and visualize them using percentile plots in Figure 11 and Figure 12. These plots show that RSZ often achieves higher returns than SAZ, indicating improved performance in lower-return episodes.

In (a) 2048, the percentile curves show a stair-shaped pattern, which corresponds to bottlenecks in creating the next large tile. The higher steps achieved by RSZ indicate that it helps agents overcome these bottlenecks more effectively during gameplay. For example, the higher return around the 20th percentile in Figure 11 (a) suggests that RSZ improves the agent's ability to handle this bottleneck. However, RSZ obtains lower returns above the 90th percentile in 2048. Since this region corresponds to another stair-step bottleneck, we suspect that the RSZ agent has not yet learned a robust strategy for overcoming this challenge. In (b) Tetris Block Puzzle, the percentile curves display a smoother, more continuous upward trajectory, reflecting a steady accumulation of lines cleared rather than the distinct, threshold-based bottlenecks seen in 2048. Across both natural (E_N) and adversarial ($E_{0.01\%}$) evaluations, the performance of RSZ and SAZ remains highly competitive. In (c) Server Allocation, RSZ mostly outperforms SAZ across the entire percentile range, from the worst-case to the best-case episodes. The relatively flat nature of these curves indicates a more uniform distribution of returns across episodes compared to the other games. However, the shaded region associated with SAZ indicates high instability. In contrast, RSZ shows significantly tighter shaded regions with higher mean values, suggesting that RSZ provides more reliable performance.

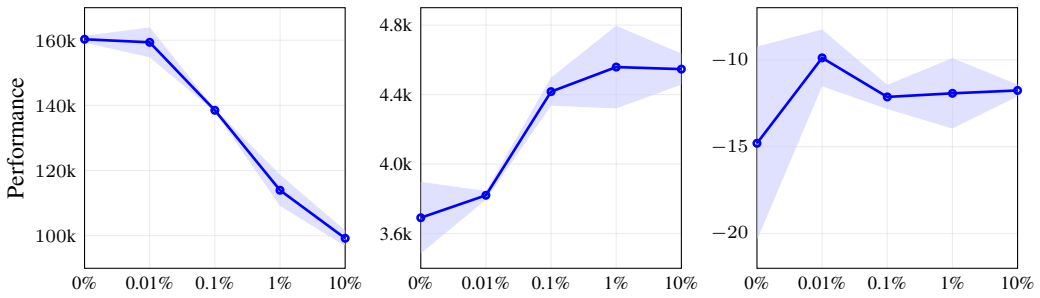
F.3 Time

Under our hardware setting (Subsection D.2), evaluating a model with natural settings takes approximately 10, 4.5, and 0.3 hours for 2048, Tetris Block Puzzle, and Server Allocation, respectively; while with adversarial settings it takes approximately double the time, as it has an adversary doing RS-MCTS. For the main experiments in Subsection 5.1, we evaluate five natural and nine adversarial settings per model. For the comparison experiments in Subsection 5.2, we evaluate nine adversarial settings per model for the five additionally trained methods. As a machine supports eight evaluations at a time, the total evaluation time is approximately 250 hours using a single machine.

- Main: $2 \times 5 \times (10 + 4.5 + 0.3) + 2 \times 9 \times (10 + 4.5 + 0.3) \times 2 \times \frac{1}{8} = 85.1$ hours.
- Comparison: $5 \times 9 \times (10 + 4.5 + 0.3) \times 2 \times \frac{1}{8} = 166.5$ hours.

G More interventions during training

As observed in Subsection 5.1, there is sometimes a significant gap between overall and tail performance under natural evaluation E_N, particularly in Tetris Block Puzzle. This suggests that the optimal adversarial intervention rate λ is not strictly universal and likely differs depending on the environment's specific dynamics. To systematically examine this, we additionally train RSZ agents with higher intervention rates $\lambda = 0.1\%$, 1%, and 10%, then evaluate them under E_N. The results, including the original $\lambda = 0\%$ and 0.01%, are visualized in Figure 13.



(a) 2048 (b) Tetris Block Puzzle (c) Server Allocation

Figure 13: Performance under varying adversarial intervention rates λ for training.

Among the three environments, RSZ performs best at $\lambda = 0.01\%$, 1%, and 0.01%, respectively, confirming that Tetris Block Puzzle is well-suited to more interventions than the other two environments. In (a) 2048, increasing λ causes a drastic degradation, indicating that too many interventions prevent the agent from learning. In (b) Tetris Block Puzzle, conversely, performance improves by 19.32%

when increasing λ from 0.01% to 1%. In (c) Server Allocation, $\lambda = 0.01\%$ remains the best, while additional interventions also outperform the 0% baseline.

Perfectly tuning λ for maximum performance may require specific domain knowledge and empirical trials. However, tuning is not strictly necessary for general deployment, as a conservative, low intervention rate consistently improves robustness and remains more beneficial than training without any interventions.

H TRAINING UNDER SAME COMPUTATIONAL COST

In our primary experiments, both the RSZ and SAZ models are trained for an equal number of 500 iterations. However, because RSZ incorporates the adversary running RS-MCTS for training, its computational cost is roughly twice that of SAZ. To rigorously control for this difference and evaluate them under approximately equal computational budgets, we conduct additional comparisons allocating SAZ double the iterations of RSZ.

H.1 EXTENDED LONG-TERM TRAINING

In this experiment, for each environment, we compare the early training setting of 500 SAZ iterations versus 250 RSZ iterations, and an extended setting of 5,000 SAZ iterations versus 2,500 RSZ iterations. Because extended training requires excessive compute, we train and evaluate *only a single model per setting* for this analysis, rather than the "train five, retain three" median approach used in the main experiments. The results of this equal-computation comparison are detailed in Table 7, revealing several insights into how RSZ and SAZ utilize their compute. Specifically, SAZ can leverage high iteration counts for rapid early gains in certain environments, while RSZ's computational investment in adversarial planning yields a better long-term learning curve. Even with half the iterations, RSZ's robust policy either catches up to or surpasses SAZ as training progresses, especially when evaluated under adversarial conditions.

Table 7: Performance comparison under the same amount of training. Ratios show how RSZ performs relative to SAZ; a value greater than 100% indicates RSZ outperforms SAZ.

Method	**2048**		**Tetris Block Puzzle**		**Server Allocation**	
	E_N	$E_{0.01\%}$	E_N	$E_{0.01\%}$	E_N	$E_{0.01\%}$
500 SAZ iterations / 250 RSZ iterations						
SAZ	160,305	141,117	3,691	3,045	-14.80	-14.82
RSZ	116,827	104,419	2,079	1,809	-10.49	-11.04
	72.88%	73.99%	56.32%	59.39%	141.13%	134.29%
5,000 SAZ iterations / 2,500 RSZ iterations						
SAZ	280,274	267,157	4,924	4,059	-9.33	-9.38
RSZ	250,223	237,612	4,612	4,098	-8.97	-8.98
	89.28%	88.94%	93.65%	100.97%	104.09%	104.46%

Early training (500 SAZ vs. 250 RSZ) Under a constrained computational budget, there is no universal winner; the advantage depends heavily on the environment. In environments that likely benefit heavily from rapid learning from new states, such as 2048 and Tetris Block Puzzle, SAZ's ability to complete twice as many iterations gives it a massive early lead. Conversely, in Server Allocation, RSZ's robust, adversarial interventions already outperform simply adding iterations.

Extended training (5,000 SAZ vs. 2,500 RSZ) As training proceeds and approaches saturation, the most striking observation is that the performance gap narrows, trending in favor of RSZ's robust planning. In 2048, RSZ closes its massive initial losses, pulling up to nearly 90% of SAZ's performance. In Tetris Block Puzzle, the catch-up is even more pronounced, and RSZ actually overtakes SAZ under $E_{0.01\%}$. In Server Allocation, while extended training allows SAZ to finally stabilize and narrow the massive early gap, RSZ still maintains a lead.

H.2 LARGER REPEATED-TRIAL STUDY ON SMALLER 2048 BOARD

The high cost in standard environments prevents us from training the SAZ or RSZ models to full convergence and achieving statistical significance. To complement the above comparisons and answer whether RSZ's extra computation is worthwhile, we conduct a larger repeated-trial study on a 3×3 version of 2048. Using a smaller board greatly reduces computational cost, making it possible for us to *evaluate 15 independently trained trials per method at several stages of learning*. Specifically, we compare RSZ after $k \in \{100, 200, 300, 400, 500\}$ training iterations with SAZ after $2k$ iterations. With more adversaries, each SAZ agent at iteration $2k$ and each RSZ agent at iteration k is evaluated against every RSZ adversary at iteration k.

As shown in Table 8, RSZ performs slightly worse than SAZ on natural evaluations. However, it achieves significantly better results on adversarial evaluations at every matched budget. At 1,000 SAZ iterations versus 500 RSZ iterations, the natural means differ by only 0.02%, while RSZ's adversarial mean is 16.18% higher. These results support RSZ's clear advantage under approximate computing when achieving near convergence, with only a small natural performance difference beyond the earliest stage.

Table 8: Performance comparison under the same amount of training on 3×3 2048. Ratios show how RSZ performs relative to SAZ; a value greater than 100% indicates RSZ outperforms SAZ.

Iterations	E_N		$E_{0.01\%}$	
SAZ / RSZ	SAZ	RSZ	SAZ	RSZ
200 / 100	$4{,}515 \pm 134$	$4{,}336 \pm 140$ (96.04%)	$3{,}117 \pm 127$	$3{,}520 \pm 142$ (112.91%)
400 / 200	$4{,}619 \pm 163$	$4{,}600 \pm 144$ (99.60%)	$3{,}434 \pm 100$	$3{,}809 \pm 87$ (110.92%)
600 / 300	$4{,}627 \pm 166$	$4{,}621 \pm 196$ (99.87%)	$3{,}395 \pm 145$	$3{,}901 \pm 96$ (114.89%)
800 / 400	$4{,}702 \pm 150$	$4{,}620 \pm 185$ (98.25%)	$3{,}657 \pm 163$	$4{,}005 \pm 120$ (109.51%)
1,000 / 500	$4{,}652 \pm 158$	$4{,}651 \pm 152$ (99.98%)	$3{,}334 \pm 100$	$3{,}873 \pm 100$ (116.18%)

I ALTERNATIVE AND ABLATION METHODS

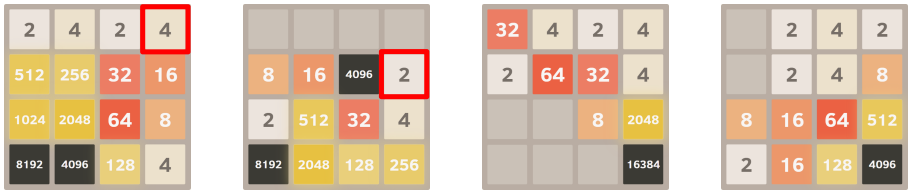
The other search-based planning methods compared in Subsection 5.2 are designed as follows.

- **FULL-ATTACK** always intervenes *during search*. It applies a typical two-player zero-sum competition, where the adversary aims to minimize the agent's outcome. However, it follows the RSZ decision phase to determine whether to intervene *after search*.
- **RANDOM-ATTACK** randomly intervenes *during search*. It works similarly to RSZ but uses randomness instead of criticality for interventions. Also, it decides whether to intervene *after search* randomly; if it intervenes, it selects the worst-case event $c_\star$.
- **NO-ATTACK** never intervenes *during search*. It follows SAZ, with neither the agent nor the adversary explicitly considering interventions. However, it decides whether to intervene based on criticality *after search*; if it intervenes, it selects the worst-case event $c_\star$.
- **STANDARD-BACKUP** performs *standard MCTS backup*. It applies the typical routine described in Section 3, instead of the RSZ backup routine described in Section 4.3. Besides, it follows the RSZ routines for the other functions.
- **SEVERITY-ONLY** considers only *severity in criticality*. It disregards rarity and completely determines whether to intervene based on the value drop, in both the tree search and the decision. Besides, it follows the RSZ routines for the other functions.

J VULNERABLE AFTERSTATES

Besides the criticality analysis in Subsection 5.3, we further examine whether criticality effectively identifies vulnerable situations by manually checking critical afterstates with their corresponding interventions in each specific environment. Specifically, we collect critical and non-critical afterstates from the $E_{0.01\%}$ evaluation records, as shown in Figure 14, Figure 15, and Figure 16. These examples show that RSZ can distinguish vulnerable afterstates from invulnerable ones using criticality.

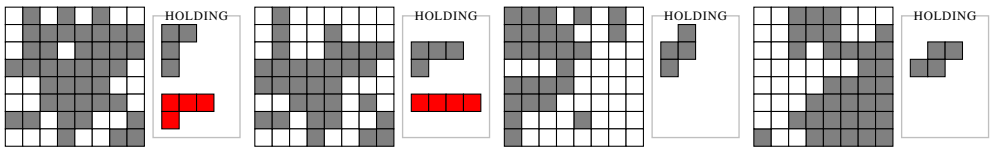
2048 High-criticality situations include (a), where placing a 4-tile causes an immediate game over while placing a 2-tile allows the player to continue, and (b), where the new 2-tile forces the agent to slide up, exposing a similar vulnerability illustrated in Figure 1. Low-criticality situations include (c) and (d), where many empty locations remain or many tiles can still be merged.



(a) $\tau = 0.900000$ (b) $\tau = 0.532352$ (c) $\tau = 0.066187$ (d) $\tau = 0.000000$

Figure 14: Vulnerable and invulnerable afterstates in 2048. (a) (b) are vulnerable afterstates with their corresponding attacking event shown as the highlighted tile; (c) (d) are invulnerable afterstates.

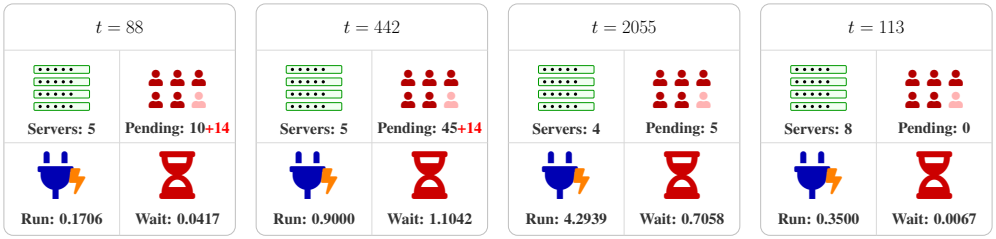
Tetris Block Puzzle High-criticality situations include (a), where neither the existing J-tetromino nor the new L-tetromino can be placed, causing an immediate game over, and (b), where placing either the L-tetromino or the I-tetromino makes it impossible to place the other. Low-criticality situations include (c) and (d), where large empty regions remain available for placing multiple types of tetrominoes.



(a) $\tau = 0.947368$ (b) $\tau = 0.743781$ (c) $\tau = 0.059380$ (d) $\tau = 0.000000$

Figure 15: Vulnerable and invulnerable afterstates in Tetris Block Puzzle. The blocks to the right of each puzzle are holding blocks. (a) (b) are vulnerable afterstates with their corresponding attacking event shown as the highlighted holding block; (c) (d) are invulnerable afterstates.

Server Allocation High-criticality situations include (a) and (b), where many new requests arrive while relatively few servers are active and many requests are already pending. Low-criticality situations include (c) and (d), where there are few pending requests or a relatively sufficient number of active servers.



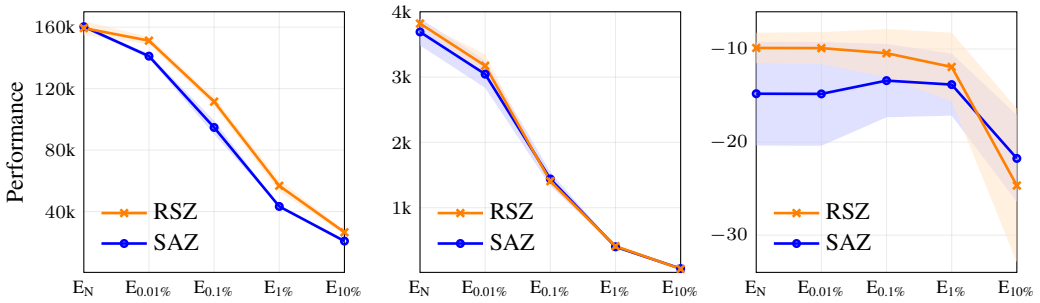
(a) $\tau = 0.883488$ (b) $\tau = 0.140978$ (c) $\tau = 0.028727$ (d) $\tau = 0.000000$

Figure 16: Vulnerable and invulnerable afterstates in Server Allocation. (a) (b) are vulnerable afterstates with their corresponding attacking event "+k"; (c) (d) are invulnerable afterstates.

K More interventions during evaluation

K.1 Impact on performance

Figure 17 presents the unnormalized scores in three environments for Figure 4 (a). In this analysis, as mentioned in the main text, the RSZ agents are informed of the increased intervention rate. Also, the RSZ adversaries perform increased interventions by using a higher intervention rate for attack, though they are trained at $\lambda = 0.01\%$. In 2048, both methods lose performance as interventions increase, with RSZ scores above SAZ at every adversarial rate, although their natural scores are similar. In Tetris Block Puzzle, there is a similar significant decline, but the scores of both methods overlap under high interventions. In Server Allocation, where less negative return is better, RSZ outperforms SAZ from E_N through $E_{1\%}$. At $E_{10\%}$, however, the RSZ mean drops below SAZ. Taken together, the raw scores show substantial sensitivity to stronger interventions, and that the relative advantage of RSZ depends on the environment and intervention rate.



(a) 2048 (b) Tetris Block Puzzle (c) Server Allocation

Figure 17: Performance under varying adversarial evaluations.

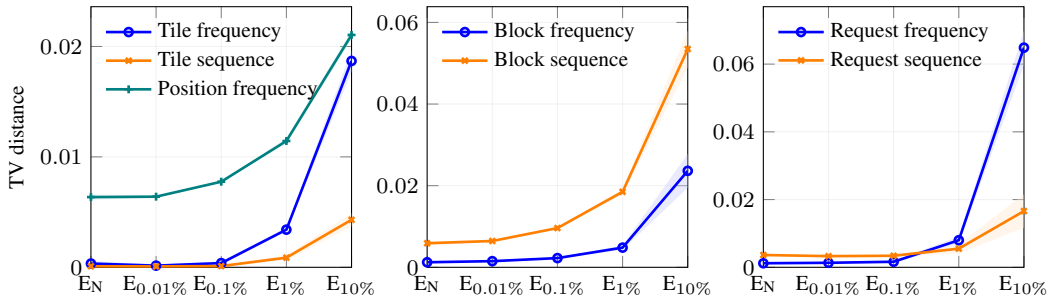
K.2 Distributional changes

To investigate whether the environment may change under interventions, we characterize the stochastic characteristics of each environment:

- **2048 tile frequency:** the proportions of generated 2-tiles and 4-tiles, whose natural probabilities are 0.9 and 0.1.
- **2048 tile sequence:** the ordering of generated tile types, including relationships between consecutive tiles within an episode.
- **2048 position frequency:** the distribution of tile placements among empty cells, conditional on the board's occupancy pattern.
- **Tetris block frequency:** the proportions of the 19 block types, which are uniform under natural generation.
- **Tetris block sequence:** the ordering of generated blocks, including relationships between consecutive block types within an episode.
- **Server request frequency:** the distribution of the number of arriving requests at each stochastic arrival step.
- **Server request sequence:** the temporal ordering of request counts, including relationships between adjacent arrival steps within an episode.

For each characteristic, we use total variation (TV) distance (Gibbs & Su, 2002) to measure the distributional change in adversarial evaluation records. Specifically, E_N contains $3 \times 200 = 600$ episodes, while each of $E_{0.01\%}$, $E_{0.1\%}$, $E_{1\%}$, and $E_{10\%}$ contains $3 \times 3 \times 200 = 1{,}800$ episodes. Figure 18 reports seven targets across all five evaluation conditions, and Figure 4 (b) in the main text shows one representative characteristic of each environment. Generally, all seven characteristics have larger mean TV distances under $E_{10\%}$ than under $E_{1\%}$, although the trends at lower rates are

not uniformly monotonic. This suggests that an adversary that ignores considering stochasticity can fundamentally distort the environment, causing the agent to learn a different task than intended.



(a) 2048 (b) Tetris Block Puzzle (c) Server Allocation

Figure 18: Total variation distances across evaluation conditions.

K.3 Hypothesis tests

As secondary evidence, we assess whether the same evaluation records provide evidence against specified stochastic models, to complement the distances in Figure 18. For the same seven environmental null hypotheses, we apply Holm adjustment across all 35 p-values and reject a null hypothesis when its adjusted p-value is below 0.05. Figure 19 displays these decisions, where filled markers indicate rejection, and labels beside the markers give the Holm-adjusted p-values. Generally, more null hypotheses are rejected at higher intervention rates. Under E_N, no null hypotheses are rejected. As interventions increase, more nulls are rejected, and by $E_{10\%}$, all are rejected.

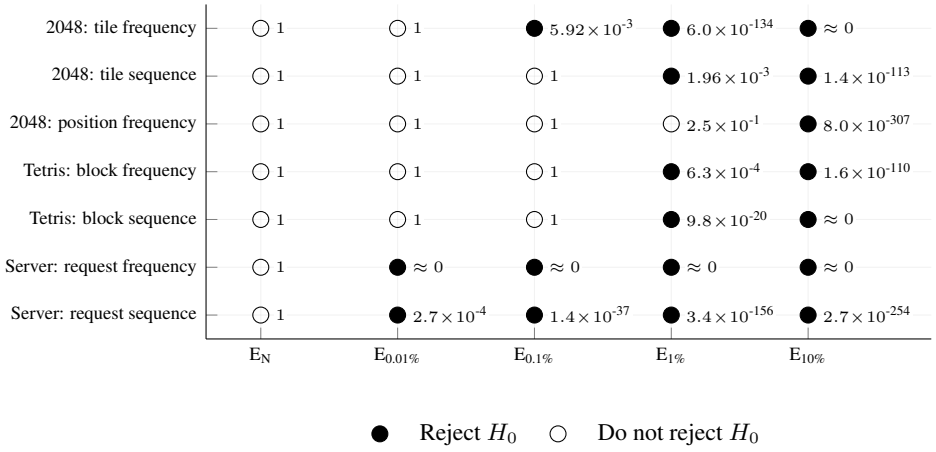


Figure 19: Hypothesis tests across varying adversarial evaluation conditions. Filled markers reject the specified null.

The null model and test of each stochastic characteristic are summarized as follows.

- **2048 tile frequency:** a Pearson goodness-of-fit test (Agresti, 2013) against probabilities 0.9 and 0.1 for 2-tiles and 4-tiles.
- **2048 tile sequence:** a two-sided runs test (Bradley, 1960) within each episode to verify that the generated tile types do not follow a deterministic pattern.
- **2048 position frequency:** a conditional Pearson test (Agresti, 2013) for uniform placement among empty cells, stratified by canonical board occupancy.
- **Tetris block frequency:** a Pearson goodness-of-fit test (Agresti, 2013) against a uniform distribution over the 19 block types.

- **Tetris block sequence:** a Pearson independence test (Agresti, 2013) to verify that the block-generation sequence remains sufficiently random.
- **Server request frequency:** a Pearson goodness-of-fit test (Agresti, 2013) against the arrival distribution, counts of 14–19 requests are combined into one tail category.
- **Server request sequence:** a two-sided Welch test (Welch, 1947) to compute mutual information between adjacent requests within each episode, and compare it from independently simulated episodes.