WORLD-MODEL POLICY ARBITER FOR GOAL-CONDITIONED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline goal-conditioned reinforcement learning (GCRL) has produced a diverse set of goal-reaching algorithms, yet no single algorithm performs best across environments, goals, and even different phases of the same task. Rather than deploying only the best-performing policy, we ask whether a set of frozen goal-conditioned policies can be used collectively as a *portfolio*, deciding at every state which policy should act. Choosing a policy at each state is not straightforward. The policies' own value functions cannot be compared directly: they may use different scales, and some policies have no value function. We need to judge each policy by the states it is likely to reach, even though we can execute only one policy at a time. We also need to avoid switching so often that control becomes unstable. To address these challenges, we introduce World-Model Policy Arbiter (WMPA), a test-time framework that, given a set of frozen policies as input, rolls out each frozen policy in a learned state-space world model, evaluates the imagined futures with a shared goal-conditioned value function, and executes the highest-scoring policy for a short commitment interval before the next round of arbitration (policy selection). WMPA assumes access to a bank of frozen goal-conditioned policies and requires neither policy retraining nor privileged task-specific knowledge. Under the official OGBench evaluation protocol on 18 state-based datasets spanning maze navigation as well as cube, scene, and puzzle manipulation, WMPA improves the macro-average success rate from the 44% achieved by the best policy selected per dataset to 58%, with statistically significant gains on 12 datasets. These gains include +33 percentage points on `cube-double-play` and +36 percentage points on `scene-play`.

1 INTRODUCTION

Goal-conditioned reinforcement learning (GCRL) seeks policies that reach user-specified goals, many of which require a sequence of qualitatively different phases: to stack two cubes at a target location, a robot arm must bring its free gripper to the first cube, grasp and place it, then repeat for the second cube. GCRL methods learn such policies through behavior cloning, temporal-difference learning, contrastive learning, quasimetric learning, or hierarchical control (Eysenbach et al., 2022; Wang et al., 2023; Park et al., 2023), and because their assumptions differ, no single method dominates across environments, goals, and task horizons (Park et al., 2025). Nor does one method dominate across the phases of a single task: on OGBench's two-cube tasks (`cube-double-play`), GCIQL, the strongest of six standard GCRL methods, succeeds in 36% of episodes and HIQL in only 5%, yet our method, selecting among the six policies within each episode, succeeds in 69% (Table 1). In the stacking episode of Figure 1, for instance, control alternates between HIQL, which moves the free gripper to the next cube, and GCIQL, which grasps and places it. Independently trained policies can thus act as pieces of a larger solution, and we study how to string them together at test time without retraining any of them.

Most prior work aims to train a single stronger policy, and the standard workflow for practitioners is to train several candidates, keep the one with the highest average success, and discard the rest. In the two-cube example, this keeps GCIQL and discards HIQL, the policy that moves the gripper between cubes in Figure 1. Keeping all of them instead raises a question: *given a set of frozen goal-conditioned policies, which policy should act from the current state for the current goal?*

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.

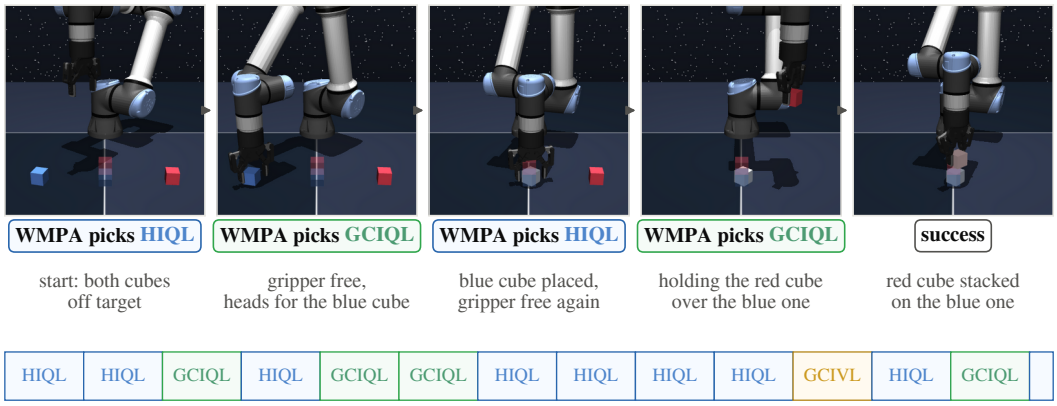


Figure 1: **WMPA composes policies within one episode.** A `cube-double-play` evaluation episode. Top: the real state at four arbitration steps and at success, with the policy the world model selects there; the label under each frame describes the phase of the task. Bottom: the policy executed over the whole episode, one block per arbitration. The arbiter alternates between HIQL, which moves the free gripper to the next cube, and GCIQL, which grasps and places it.

This selection problem is challenging for three reasons. First, the policies cannot be ranked by their own value functions. The most direct approach would pair each policy π_i with its value function V_i and execute π_{i^*} with $i^* = \arg\max_i V_i(s, g)$, a policy-level form of generalized policy improvement (Barreto et al., 2017). This rule presumes that every V_i measures the same quantity on the same scale, which does not hold across GCRL algorithms: V_i may be a temporal-difference value, a quasimetric distance, or a contrastive score, and GCBC learns none. Second, a policy should be judged by where its closed-loop behavior leads, yet at test time only the selected one is executed, so the consequences of the alternatives are never observed. Third, frequent switching between policies can disrupt control.

Our key idea is to compare policies through the consequences they are predicted to produce. Our method, World-Model Policy Arbiter (WMPA), rolls out each frozen policy in a learned world model while conditioning all policies on the same final goal. It scores the resulting imagined futures with a shared goal-conditioned value function, executes the highest-scoring policy for a fixed commitment interval, and repeats this procedure. Unlike action-level model-predictive control, WMPA does not optimize new action sequences. Instead, it evaluates the behaviors already represented in the policy bank, which requires executing each policy just once at each imagined step, rather than a search over possible trajectories. All base policies remain frozen, and the selector requires no hand-designed progress signal, no rollouts in the real environment, and no privileged task knowledge.

Our paper makes the following contributions:

- **Problem: Policy arbitration for offline GCRL.** We study test-time control over a bank of independently trained goal-conditioned policies as a policy-selection problem, which is distinct from selecting a single global winner or retargeting the subgoal of a single controller. We argue that offline GCRL is the natural setting for this problem: the policy bank incurs no additional data collection cost, because its members are trained entirely from the offline dataset without online interaction; moreover, goal-conditioning multiplies the situations in which different members excel.
- **Method: World Model Policy Arbitration.** WMPA compares frozen policies by scoring their closed-loop imagined futures with a shared goal-conditioned value function, without policy retraining or privileged signals. We also derive a conditional guarantee over the best fixed bank member: arbitration improves success when locally useful alternatives are available, true-dynamics scores separate inferior choices, and rollout-score errors rarely overturn that separation (Appendix I).
- **Finding: Empirical gains from choosing frozen policies.** We evaluate WMPA on 18 state-based OGBench datasets under the official protocol. We compare against every policy in the bank, a random-switching baseline with a matched commitment interval, and additional model-free and model-based baselines. We further analyze when arbitration is beneficial, using a per-episode hindsight oracle and policy-bank statistics.

2 Background On Goal-Conditioned Reinforcement Learning

Following Park et al. (2025), we consider goal-conditioned reinforcement learning over a controlled Markov process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{G}, \mu, p)$. Here, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{G}$ is the goal space, $\mu \in \Delta(\mathcal{S})$ is the initial-state distribution, and $p(s' \mid s, a) \colon \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the transition dynamics, where $\Delta(\cdot)$ denotes the space of probability distributions over its argument. A goal-conditioned policy $\pi(a \mid s, g) \colon \mathcal{S} \times \mathcal{G} \to \Delta(\mathcal{A})$ maps the current state and goal to a distribution over actions.

Let $\nu \in \Delta(\mathcal{S} \times \mathcal{G})$ denote the evaluation distribution over initial-state and goal pairs. Given $(s_0, g) \sim \nu$, a policy induces a trajectory $\tau = (s_0, a_0, s_1, \ldots, a_{T-1}, s_T)$ according to the transition dynamics. Its performance is measured by expected goal-reaching success:

$$J(\pi) = \mathbb{E}_{(s_0,g)\sim\nu,\, \tau\sim p^\pi(\tau|s_0,g)} \left[\mathrm{Succ}(\tau, g)\right],$$

where $\mathrm{Succ}(\tau, g) \in \{0, 1\}$ indicates whether τ reaches g. This success criterion is used only for evaluation and is not available during training. Instead, goal-reaching supervision is constructed through hindsight goal relabeling, using states from logged trajectories as goals.

We assume access to a logged trajectory dataset $\mathcal{D} = \{\tau^{(n)}\}_{n=1}^{N}$, where each trajectory $\tau^{(n)} = (s_0^{(n)}, a_0^{(n)}, s_1^{(n)}, \ldots, s_{T_n}^{(n)})$ is collected as part of an offline dataset. We use $\mathcal{D}$ to train the auxiliary world model and value function. In our experiments, $\mathcal{D}$ is the OGBench training split, and all policies are trained offline using the same split. The general formulation, however, does not require the policies to be trained offline or from a shared dataset.

Rather than learning a single goal-conditioned policy, we are given a portfolio of M frozen policies,

$$\mathbf{\Pi} = \{\pi_i(a \mid s, g)\}_{i=1}^{M},$$

whose members may differ in the GCRL algorithm or random seed used for training. All members share a common state, action, and goal interface, and the arbiter requires only black-box access to their action distributions. Internal value estimates and representations are neither assumed to be accessible nor required to be comparable across policies.

At arbitration time t, the arbiter selects a policy index $z_t \in \{1, \ldots, M\}$. The selected policy then controls the agent for a commitment interval of length c:

$$a_{t+j} \sim \pi_{z_t}(\cdot \mid s_{t+j}, g), \qquad j = 0, \ldots, c-1.$$

The resulting controller is therefore a switching policy over the frozen portfolio members. The policy-portfolio objective is to maximize the expected goal-reaching success of this induced policy without modifying any member of $\mathbf{\Pi}$. The central problem is thus to determine, from the current state and goal, which portfolio member is expected to produce the most favorable future trajectory.

3 Method: World-Model Policy Arbiter

At each arbitration point, WMPA imagines k steps of every frozen policy in a learned world model (Section 3.1), scores the imagined states with a single goal-conditioned value function (Section 3.2), and executes the winner for $c = k$ environment steps (Section 3.3); Figure 2 illustrates the loop. Each step answers one challenge from Section 1: imagination supplies the unobserved consequences of the policies that are not selected, one shared value function puts all candidates on a common scale, and both commitment and value-based selection limit excessive switching between policies.

3.1 Learned state-space world model

For each environment, we train an ensemble of E independently parameterized MLP dynamics models (Kurutach et al., 2018) directly in the environment state space. Throughout this section, all states are raw environment states; the fixed normalizer $\nu(s) = (s - \mu_s)/\sigma_s$, whose statistics are computed from the logged trajectories, is applied only at the network input. Each member predicts a normalized one-step delta and returns the next state,

$$\hat{\delta}_t^{(e)} = f_{\theta_e}\big(\nu(s_t), a_t\big), \tag{1}$$

$$\hat{s}_{t+1}^{(e)} = \hat{f}_{\theta_e}(s_t, a_t) := s_t + \mu_\Delta + \sigma_\Delta \hat{\delta}_t^{(e)}, \tag{2}$$

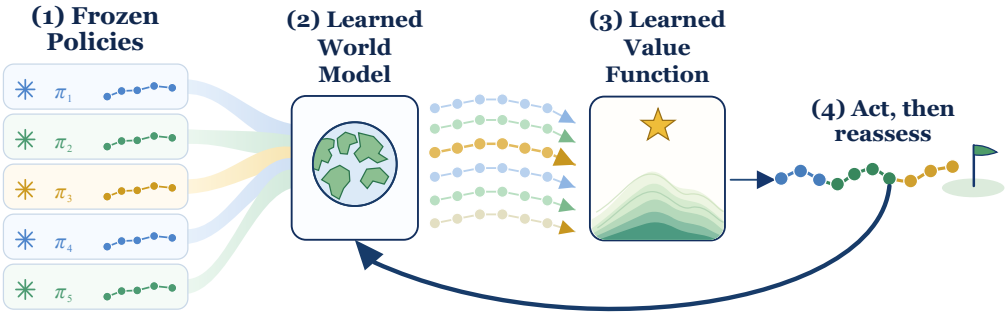


Figure 2: **WMPA overview.** (1) We begin with a frozen portfolio of heterogeneous GCRL policies. (2) WMPA imagines the future induced by each policy through a shared learned world model. (3) A shared goal-conditioned value function scores the predicted futures on a common scale. (4) The highest-ranked policy is executed for c steps, after which the process repeats.

where μ_Δ and σ_Δ are fixed statistics computed from valid logged transitions. Multi-step predictions are obtained by applying the same $\hat{f}_{\theta_e}$ to its own outputs, with re-normalization at every input. The outputs are trained to match non-normalized environment states and are consumed by the frozen policies exactly like real observations. The members are trained on H-step windows by unrolling their own predictions and penalizing the normalized error at every step,

$$\mathcal{L}_{\text{dyn}} = \frac{1}{E}\sum_{e=1}^{E}\mathbb{E}_{\tau\sim\mathcal{D}}\left[\sum_{h=1}^{H} w_h \left\|\nu\big(\hat{s}^{(e)}_{t+h}\big) - \nu\big(s_{t+h}\big)\right\|_2^2\right], \tag{3}$$

with a curriculum that increases the effective horizon from 1 to H over the first 10^5 updates (w_h uniform over the active steps). Predicting in state space isolates arbitration from representation learning and guarantees that imagined states are valid inputs to every frozen policy. Notably, the model is decoupled from the policies: learning it can be done independently of making changes to the policy bank.

3.2 Goal-conditioned value function

The arbiter scores every candidate's imagined states with a single goal-conditioned value function $V(s, g)$, so policies trained with different objectives are compared on a common scale. Only the ordering matters: futures reached by different candidates from the same decision state should be ranked in the same order as their realized outcomes. Nothing in the arbiter depends on how V is parameterized, and we use two parameterizations whose inductive biases match different kinds of task progress; we choose between them per task family (Section 4, Appendix A). We also swap the two value heads to show the performance difference and discuss value-head choice in Appendix C.

Metric value (maze, cube, scene). Where progress is geometric, like moving through a maze, displacing objects toward target poses, we use a latent distance with separate state and goal encoders for each ensemble member,

$$V_{\phi_e}(s, g) = -\left\|\varphi_S^{(e)}(s) - \varphi_G^{(e)}(\psi^{(e)}(g))\right\|_2, \tag{4}$$

where $\psi^{(e)}$ is a low-dimensional goal representation, following Kang et al. (2026). The distance orders imagined state–goal pairs even where closed-loop rollouts leave the support of the logged data, and the distinct encoders keep reachability directional ($V_{\phi_e}(s, g) \neq V_{\phi_e}(g, s)$ in general). Each value member is trained jointly with its corresponding dynamics member by an expectile temporal-difference objective on relabeled goals with a sparse reward (0 at attainment and -1 otherwise),

$$\mathcal{L}_{\text{value}} = \frac{1}{E}\sum_{e=1}^{E}\mathbb{E}\Big[\big|\kappa - \mathbf{1}\{\bar{\delta}_e < 0\}\big|\,\big(y_e - V_{\phi_e}(s, g)\big)^2\Big], \quad y_e = r(s, g) + \gamma\, m(s, g)\, \bar{V}_{\bar{\phi}_e}(s', g), \tag{5}$$

where $\bar{V}$ is a target network, $m(s, g) = 0$ at attainment, $\bar{\delta}_e = y_e - \bar{V}_{\bar{\phi}_e}(s, g)$, and the expectile satisfies $\kappa > 0.5$ (settings in Appendix F).

Direct value (puzzle). On the puzzle datasets progress is combinatorial: a state is a pattern of buttons, a press toggles a button together with its neighbours, and the number of presses to the goal pattern is not monotone in embedding distance. Thus, we use a value with no geometric structure,

$$V_\theta(s, g) = \mathrm{MLP}_\theta\big([\, s;\, g\,]\big), \tag{6}$$

trained by implicit Q-learning (Kostrikov et al., 2022) on the same dataset. Two action-value heads $Q_{\omega,i}(s, a, g)$ are regressed toward the one-step target $r(s, g) + \gamma\, m(s, g)\, V_\theta(s', g)$, and the state value is the κ-expectile of the target value function $\bar{Q} = \min_i \bar{Q}_{\omega,i}$ under the dataset's actions,

$$\mathcal{L}_V = \mathbb{E}\Big[\, \big|\kappa - \mathbf{1}\{\bar{Q}(s, a, g) - V_\theta(s, g) < 0\}\big|\, \big(\bar{Q}(s, a, g) - V_\theta(s, g)\big)^2 \Big], \tag{7}$$

so that V_θ approximates the value of the best dataset-supported action at each relabeled state–goal pair. It is the state-value network of the bank's GCIQL member, trained with the reference implementation's settings and reused unchanged as the scorer. Being one function applied to every candidate's imagined states, it keeps scores comparable across the bank exactly as the metric value does.

3.3 Model-Based Policy Rollout and Ranking

At each arbitration point, every frozen policy is rolled out in closed loop for k model steps, acting deterministically on its own imagined states. All policies in the bank are Gaussian actors whose deterministic mode is the behavior deployed and evaluated by the benchmark, so the imagined branch reproduces the behavior that would be executed; we do not evaluate stochastic rollouts obtained by sampling actions from the policy distributions. For policy i and ensemble member e,

$$\hat{s}_0^{i,e} = s_t, \tag{8}$$

$$\hat{a}_h^{i,e} = \pi_i(\hat{s}_h^{i,e}, g), \tag{9}$$

$$\hat{s}_{h+1}^{i,e} = \hat{f}_{\theta_e}(\hat{s}_h^{i,e}, \hat{a}_h^{i,e}), \qquad h = 0, \ldots, k-1, \tag{10}$$

where $\hat{f}_{\theta_e}$ is the e-th dynamics-model ensemble member of Section 3.1. We score each candidate by the highest value imagined state along the rollout,

$$S_i(s_t, g) = \max_{1 \le h \le k} \frac{1}{E} \sum_{e=1}^{E} V_{\phi_e}(\hat{s}_h^{i,e}, g). \tag{11}$$

The max operator allows a policy to receive credit for reaching the goal before the end of the planning horizon. Table 11 ablates both reductions (*mean* and *final-state* alternatives over the horizon; *min* and *mean−std* over ensemble members). WMPA selects the policy index with the highest score $i^* = \arg\max_i S_i(s_t, g)$ and executes π_{i^*} in the real environment for c steps, and then repeats the arbitration (Algorithm 1, Appendix A).

Rollout length and commitment. Each arbitration costs MEk dynamics and MEk value evaluations (M=6 policies, E=3 ensemble members), or MEk/c per environment step. Setting k=c fixes this at ME, which adds about 3–4 ms per step on 8 CPU threads (Appendix H); the horizon is thus a single hyperparameter, set per task family (Table 2), trading off reactivity against foresight at constant amortized cost. The large gains are robust to k and c, even when $k \neq c$ (Table 3).

4 Experiments

We evaluate WMPA on 18 state-based OGBench datasets (Park et al., 2025): two maze-navigation datasets (`pointmaze-medium` and `antmaze-large`) as well as six cube (`single` to `triple`), two scene, and eight puzzle (`3x3` to `4x6`) manipulation datasets, which together cover the `play` and `noisy` data-collection variants (Appendix A). OGBench exposes the same observation, action, goal, and evaluation interfaces to policies trained with different objectives, and the use of state observations isolates arbitration from visual representation learning. The portfolio contains the six

Table 1: **Success rate (%) on OGBench under the official protocol** (5 goals $\times$ 50 episodes, 3 bank seeds). We report every bank policy, the best policy for each dataset which is the bank policy with the highest three-seed mean success on the reported episodes, and WMPA, which arbitrates over the same frozen policies with the (k, c) setting of its task family (Appendix A). Each entry is the mean success rate $\pm$ the half-width of its 95% hierarchical-bootstrap interval (resampling bank seeds, then episodes), rounded to the nearest integer. Bold indicates values within 95% of the row maximum.

Family	Dataset	GCBC	GCIVL	GCIQL	QRL	CRL	HIQL	BEST-POLICY	WMPA
Maze	pointmaze-medium-navigate	6 ± 4	74 ± 3	54 ± 16	$\mathbf{76 \pm 17}$	35 ± 6	73 ± 8	$\mathbf{76 \pm 17}$	$\mathbf{79 \pm 3}$
	antmaze-large-navigate	23 ± 3	19 ± 5	37 ± 8	78 ± 5	$\mathbf{86 \pm 3}$	$\mathbf{90 \pm 3}$	$\mathbf{90 \pm 3}$	$\mathbf{91 \pm 3}$
Cube	cube-single-play	6 ± 3	49 ± 4	71 ± 3	8 ± 4	16 ± 3	12 ± 6	71 ± 3	$\mathbf{85 \pm 3}$
	cube-single-noisy	8 ± 3	71 ± 8	$\mathbf{99 \pm 1}$	33 ± 10	40 ± 5	40 ± 10	$\mathbf{99 \pm 1}$	$\mathbf{100 \pm 0}$
	cube-double-play	2 ± 1	35 ± 6	36 ± 7	1 ± 1	9 ± 2	5 ± 2	36 ± 7	$\mathbf{69 \pm 3}$
	cube-double-noisy	1 ± 1	14 ± 3	26 ± 9	3 ± 2	3 ± 2	2 ± 1	26 ± 9	$\mathbf{60 \pm 7}$
	cube-triple-play	1 ± 1	1 ± 1	2 ± 1	0 ± 0	$\mathbf{5 \pm 3}$	2 ± 1	$\mathbf{5 \pm 3}$	1 ± 1
	cube-triple-noisy	2 ± 2	9 ± 3	2 ± 2	2 ± 1	1 ± 1	1 ± 1	9 ± 3	$\mathbf{18 \pm 3}$
Scene	scene-play	4 ± 2	42 ± 5	52 ± 5	6 ± 2	22 ± 4	43 ± 8	52 ± 5	$\mathbf{87 \pm 4}$
	scene-noisy	2 ± 1	30 ± 3	27 ± 6	6 ± 3	0 ± 1	26 ± 4	30 ± 3	$\mathbf{70 \pm 5}$
Puzzle	puzzle-3x3-play	2 ± 1	3 ± 1	91 ± 4	0 ± 1	4 ± 2	12 ± 3	91 ± 4	$\mathbf{100 \pm 0}$
	puzzle-3x3-noisy	1 ± 1	38 ± 26	$\mathbf{91 \pm 3}$	0 ± 0	31 ± 5	57 ± 7	$\mathbf{91 \pm 3}$	$\mathbf{93 \pm 2}$
	puzzle-4x4-play	0 ± 0	12 ± 5	24 ± 5	0 ± 0	1 ± 1	7 ± 2	24 ± 5	$\mathbf{51 \pm 8}$
	puzzle-4x4-noisy	0 ± 0	19 ± 3	34 ± 5	0 ± 0	0 ± 0	12 ± 4	34 ± 5	$\mathbf{58 \pm 11}$
	puzzle-4x5-play	0 ± 0	6 ± 2	14 ± 3	0 ± 0	2 ± 1	4 ± 2	14 ± 3	$\mathbf{18 \pm 3}$
	puzzle-4x5-noisy	0 ± 0	$\mathbf{20 \pm 3}$	$\mathbf{20 \pm 3}$	0 ± 0	5 ± 4	3 ± 2	$\mathbf{20 \pm 3}$	$\mathbf{20 \pm 3}$
	puzzle-4x6-play	0 ± 0	11 ± 2	11 ± 2	0 ± 0	3 ± 1	3 ± 2	11 ± 2	$\mathbf{17 \pm 3}$
	puzzle-4x6-noisy	0 ± 0	18 ± 3	$\mathbf{19 \pm 3}$	0 ± 0	5 ± 2	2 ± 1	$\mathbf{19 \pm 3}$	$\mathbf{20 \pm 3}$
Average (18 datasets)		3	26	39	12	15	22	44	**58**

OGBench reference learners (GCBC, GCIVL, GCIQL, QRL, CRL, and HIQL), each trained with the unmodified reference implementation and per-family hyperparameters for 10^6 steps. The policy bank holds one checkpoint per algorithm, and the study is replicated over three bank seeds. All policies remain frozen; the scores of the individual members are reported in Table 1. All methods are evaluated on the same fixed episode seeds, so every comparison is paired per episode; reported intervals are 95% hierarchical bootstraps over bank seeds and then episodes, and a gain is called significant when the interval of the paired difference excludes zero (Appendix G).

4.1 POLICY ARBITER IMPROVES GOAL REACHING

Table 1 reports the success rate of every policy in the bank and of WMPA. Across the 18 datasets, WMPA achieves a mean success rate of 58%, compared with 44% for the best policy and 39% for random switching with the same interval and policy bank (Section 4.2). WMPA attains the highest mean on 17 datasets, and its gain over the best policy is significantly positive on 12 datasets, not significant on 5, and significantly negative on 1. The largest improvements occur on multi-object manipulation tasks, including +41 percentage points on `scene-noisy` and +35 on `cube-double-noisy`; the per-dataset confidence intervals are reported in Appendix G.

We compare WMPA to action-level model-predictive control (MPC) applied to the best policy with the same k and value function, re-planned at every environment step, in Table 6. MPC, given k times the model budget of WMPA ($18k$ versus 18 imagined steps per environment step), averages 53% against 58% for WMPA; it is significantly worse than arbitration on 7 of 18 datasets and significantly better on 1, with its largest deficits on the datasets where arbitration gains most (`cube-single-play`, `cube-double-play`, `scene-play`), which supports the benefit of arbitrating among different policies beyond locally refining the actions of a single best policy.

4.2 DOES THE WORLD MODEL IMPROVE PERFORMANCE? USUALLY.

We study a RANDOM-SWITCH ablation which removes the entire learned model: it uses neither the world-model rollouts nor the value head, and draws a policy uniformly from the bank at every arbitration point. Everything else is shared with WMPA, so the comparison isolates the learned scoring of imagined futures. Under per-episode pairing, WMPA outperforms RANDOM-SWITCH by 19 points on average, with significant gains on 13 datasets. Random switching alone is significantly

below the best policy on 10 datasets and significantly above it on 4. WMPA's major gains therefore come neither from the commitment schedule nor from cycling through policies, but from model-based candidate ranking. Full results of this ablation are presented in Table 4.

To check whether the main gains of policy switching come from avoiding states that the current policy cannot escape, we design an event-triggered model-free baseline. It retains the selected policy and switches to a randomly chosen other policy only when the agent stalls. Stalling is measured as the net displacement of the full observation vector over the last m steps, $\|\nu(s_t) - \nu(s_{t-m})\|_2$, in the standardized space of Section 3.1, so the threshold $\varepsilon = 0.05$ is dimensionless and is fixed across all datasets without tuning. Appendix B evaluates this rule for $m \in \{5, 10, 25, 50\}$: its success varies little with m, and it does not recover the manipulation gains, so a displacement trigger is a poor proxy for the situations in which switching is beneficial (Table 5).

We also isolate the effect of dynamics-model accuracy on the performance of the method. To this end, Table 7 replaces the learned dynamics with true-simulator branches on every dataset. This substitution raises the success rate by +2 points on average, with significant gains on 2 of 18 datasets, and never significantly lowers it, so model error costs some ranking quality but is not the main factor that limits the method.

A Q-select ablation tests whether world-model rollouts are necessary at all: on the same bank and episodes (Table 8), Q-select ranks the action proposed by each member at the current state with the twin-Q value function of the GCIQL member, without imagined states. Re-selecting at every step (c=1), Q-select is a strong selector: it significantly exceeds the best policy on 10 of 18 datasets, and on the puzzle family, where WMPA scores with the V of the same member, its gap to WMPA is +2 points on average. This is expected, since $Q(s, a, g) = r + \gamma V(s', g)$ already summarizes the future. Rollouts nevertheless matter for two reasons. First, Q-select requires an action-value function, whereas the metric value of Section 3.2 can be used only through imagined states and is the better scorer on the maze, cube, and scene families; averaged over all datasets, WMPA exceeds Q-select at c=1 by +12 points. Second, scoring only the first action says nothing about the $c-1$ steps that follow, so at the matched interval c=k, Q-select degrades while the k-step rollout does not: direct-value WMPA exceeds Q-select by +7 points on average, with significant gains on 11 datasets and no significant losses. Rollouts thus enable the better value head and make arbitration safe to amortize over c steps, which keeps the budget at ME model calls per step.

Scoring each rollout with the policy's own value function instead fails even in the setting most favorable to it: restricted to GCIQL, GCIVL, and HIQL, whose value functions estimate the same quantity, the selection collapses onto the most optimistic value function and is significantly worse than shared scoring on 14 of 18 datasets (Table 9).

4.3 Arbitration benefits from complementary behaviors within episodes

WMPA can only select behaviors that already exist in the bank. Here, we investigate whether the gains of arbitration arise solely from selecting the best policy for each episode or whether they also rely on within-episode composition. We construct the per-episode oracle by evaluating every policy in the bank on each episode under a fixed seed and defining the oracle performance on that episode as the maximum performance across all policies.

In Figure 3, we plot the gain of WMPA against the gain of the per-episode oracle, where both gains are measured relative to the best policy in the bank. First, WMPA significantly outperforms the per-episode oracle on 7 datasets, with large margins on `cube-double-play` (69% versus 57%) and `scene-play` (87% versus 74%). Because the oracle already selects a successful policy for each evaluation episode whenever one exists, per-episode selection alone cannot account for these gains; instead, they indicate a benefit of composing policies *within* an episode (Figure 1 shows one such episode). Second, neither a large gap between the best and the second-best policy nor limited headroom between the best policy and the oracle precludes gains from switching. On `puzzle-3x3-play`, WMPA improves over the best policy by +9 percentage points, even though the success rate of the second-best policy is 79 percentage points lower than that of the best policy and the headroom to the per-episode oracle is only 1%. Similarly, on `cube-single-play`, the second-best policy trails the best by 22 percentage points and the oracle provides only 10 points of headroom, yet the gain is +14 percentage points. Both gains exceed the corresponding static headroom, which shows that within-episode composition can improve performance even when a single member is substantially

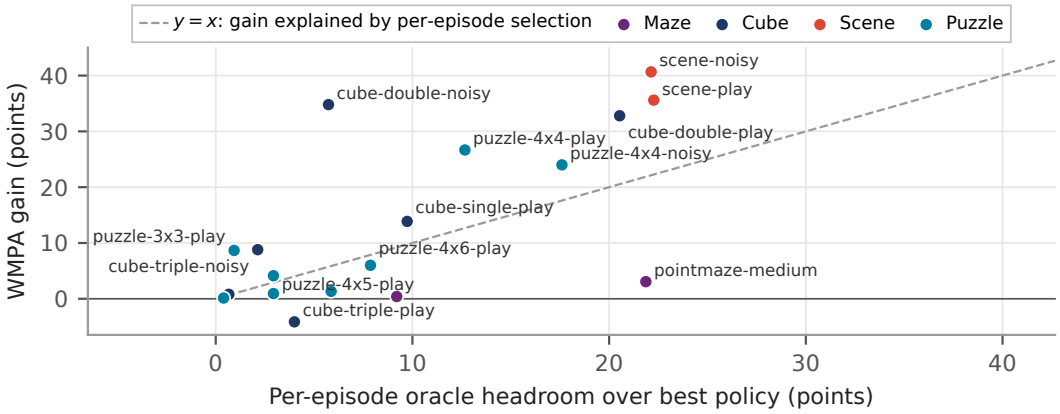


Figure 3: **What makes arbitration work?** Each point is a dataset. Horizontal axis: success added over the best policy by the per-episode oracle, which picks a successful bank policy per episode but never switches within it, so it bounds per-episode selection, not WMPA. Vertical axis: success added by WMPA. Above the dashed $y = x$ line, the extra success must come from switching within an episode; WMPA is significantly above the oracle on 7 datasets and below it on 5 (Appendix G).

stronger than the others overall. Moreover, the bank need not be drawn from a single dataset: pooling the policies trained on the `play` and `noisy` data of an environment into one twelve-member bank never degrades the performance of WMPA and improves it whenever the added members contribute new competencies (Appendix E). A complementary analysis in Appendix I expresses the gain over the best fixed bank member as accumulated one-chunk advantages and bounds the loss from selecting inferior members using score separation and rollout-score error.

5 Related Work

Goal-conditioned reinforcement learning. GCRL learns policies conditioned on a desired goal, using supervised, value-based, contrastive, quasimetric, and hierarchical objectives (Eysenbach et al., 2022; Wang et al., 2023; Park et al., 2023); OGBench provides common datasets and reference implementations, and it shows that the strengths of these methods vary across navigation, stitching, stochasticity, and manipulation (Park et al., 2025). Model-based GCRL uses a model to improve the policy being trained: planning with a learned model to fine-tune offline policies (GOPlan; Wang et al., 2024), planning exploration goals (PEG; Hu et al., 2023), modeling transitions between arbitrary buffer states for goal navigation (Duan et al., 2024), synthesizing goal-reaching data backward (Höftmann et al., 2023), or constructing self-curricula (Gong et al., 2024); see Liu et al. (2022) for a survey. Stronger recent learners (Dong et al., 2026) and abstraction-based state–goal representations (Wibault et al., 2026) are complementary to WMPA: they could serve as additional bank members and can supply the value head. Goal-conditioned hypernetworks even generate policies on demand (Faccio et al., 2023). All of these approaches modify or create the policy; we leave every policy frozen and use the model only to choose among them.

Test-time planning with frozen policies and world models. Existing test-time methods devote computation to *producing* behavior. Goal-conditioned policies serve as low-level controllers for planning through intermediate goals (Nasiriany et al., 2019); Test-Time Graph Search builds a graph over dataset states and feeds subgoals to a single frozen policy (Opryshko et al., 2026); CompPlan composes subgoal-conditioned instances of a single policy family through policy-conditioned jumpy models (Farebrother et al., 2026); and ALPS plans over Laplacian subgoal clusters with a one-step model (Shehmar et al., 2026). All these methods ask *which subgoal should one controller pursue?*, whereas WMPA keeps the final goal fixed and asks *which controller should act now?* CompPlan is closest in spirit, since it also sequences pre-trained policies with a learned model; but its policies form one family parameterized by a subgoal or latent z, its jumpy models condition on z, and it searches over which z to pursue next, whereas WMPA uses a policy-agnostic dynamics model to arbitrate among heterogeneous policies trained by different algorithms, never changing the goal.

Hand-offs between a task policy and a recovery policy in safety-critical offline GCRL (Cao et al., 2024) are policy switching with two controllers and a hand-set cost threshold as the trigger; WMPA switches over an arbitrary bank, with the model-based score as the trigger. World models, in turn, are used for action-level planning or training an amortized policy (Schrittwieser et al., 2020; Hafner et al., 2025); sampling-based MPC with learned models (Williams et al., 2017; Hansen et al., 2022) motivates the action-level baseline of Appendix C which perturbs the first action, and hierarchical or reset-free world models (Schiewer et al., 2024; Yang et al., 2025) likewise generate behavior rather than evaluate it. WMPA uses the model only to evaluate closed-loop behaviors already in the bank.

Policy switching, options, and learned hierarchies. Selecting among fixed policies by simulating each is a long-standing idea: policy switching and parallel rollout execute, at every state, the base policy with the highest simulated return, and both provably improve upon every base policy (Chang et al., 2004; Bertsekas & Castañon, 1999); generalized policy improvement (GPI) achieves the same result with per-policy action-values, provided they are comparable (Barreto et al., 2017). In the options framework, each frozen member of $\mathbf{\Pi}$ is an option with a fixed-duration termination condition, and re-arbitrating every c steps is a scheduled form of option interruption, which is guaranteed not to degrade performance when the switching value is exact (Sutton et al., 1999). WMPA is a descendant of this lineage with a learned model, a truncated horizon, and goal conditioning: the model supplies the rollouts that the classical methods obtain from a simulator, and the shared value function avoids the need for the comparable per-policy action-value estimates required by GPI. Model-based hierarchical RL is a distinct lineage. Director selects latent goals for a worker trained in imagination (Hafner et al., 2022); SkiMo plans with a skill-dynamics model learned jointly with a skill VAE (Shi et al., 2022); SPlaTES shapes skills to be predictable by an abstract world model and plans over them with iCEM (Gürtler & Martius, 2025); Puppeteer trains a separate high-level world model for each task on top of a frozen tracking controller (Hansen et al., 2025); and a learned gate over a frozen planner decides how much search to perform (Muppidi et al., 2026). All of these methods learn the high level, most also co-adapt the low level, and their models generate or search over the executed behavior. WMPA learns neither level: the candidates are complete goal-conditioned policies trained independently with different algorithms, the dynamics model is trained once on the dataset without reference to any policy, and the model only arbitrates among them.

6 Limitations and Future Work

This study uses state observations and per-environment transition models. Pixel observations would require either a latent world model with a shared visual encoder across all policies or a world model that produces realistic images. We leave visual environments to future work. Additionally, WMPA selects among existing behaviors and cannot create new ones: when every member is at the task floor, one member is already saturated, or the members fail on the same subtasks, arbitration typically does no better than the best policy. Finally, arbitration adds a few milliseconds of computation per step (Appendix H), a modest overhead; future work could remove it by using the world model offline, either to distill the arbiter's choices into a single policy or to generate corrected training data in the style of DAgger.

7 Conclusion

We introduced World-Model Policy Arbiter (WMPA), which selects among frozen goal-conditioned policies at test time by rolling each forward in a learned world model, scoring the imagined futures with a shared value function, and executing the highest-scoring policy for a short commitment interval before selecting again. Without retraining any policy, it improves on the best policy in its bank by 13 points on average and by 33–41 points on several multi-object manipulation tasks. Notably, it significantly exceeds a per-episode hindsight oracle on 7 datasets, indicating a benefit of composing policies within an episode. Taken together, these results suggest viewing world models as test-time evaluators of existing behaviors, not only as generators of new actions.

Reproducibility Statement

The supplementary material contains an anonymized copy of our code. It includes the training entry points for the policy bank, which wrap the unmodified OGBench reference implementations, and for the world model with its value head; the evaluation script that runs WMPA and every control and ablation reported in this paper; and the scripts that aggregate the per-episode logs into success rates and paired hierarchical-bootstrap intervals. A per-dataset registry records the bank, the checkpoints, and the reported interval for all 18 datasets, and the software versions are listed. Reset seeds are a deterministic function of the dataset, the goal, and the episode index, so any method can be re-evaluated on exactly the initial states used here. All datasets are the public OGBench releases; trained checkpoints and raw evaluation logs are omitted because of their size.

Ethics Statement

This work studies simulated control benchmarks and does not involve human participants or sensitive personal data.

AI Use Statement

We used generative AI assistants to help implement and edit portions of the experimental, statistical-analysis, and infrastructure code, and to improve the clarity and readability of the manuscript. We also used them to help develop and check the theoretical analysis and the consistency of its proof, and to identify related work that we may have missed. All AI-assisted code was reviewed, tested, and validated by one of the authors, and all AI-assisted text was reviewed and edited by the authors, who also verified all AI-suggested corrections and references. The authors take full responsibility for the final content of this work, including all code, results, claims, and text.

References

André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, Hado van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

Dimitri P. Bertsekas and David A. Castañon. Rollout algorithms for stochastic scheduling problems. *Journal of Heuristics*, 5(1):89–108, 1999.

Chenyang Cao, Zichen Yan, Renhao Lu, Junbo Tan, and Xueqian Wang. Offline goal-conditioned reinforcement learning for safety-critical tasks with recovery policy. In *IEEE International Conference on Robotics and Automation*, 2024.

Hyeong Soo Chang, Robert Givan, and Edwin K. P. Chong. Parallel rollout for online solution of partially observable Markov decision processes. *Discrete Event Dynamic Systems*, 14(3):309–341, 2004.

Zhiqiang Dong, Teng Pang, Rongjian Xu, and Guoqiang Wu. Efficient hierarchical implicit flow Q-learning for offline goal-conditioned reinforcement learning. *arXiv preprint arXiv:2604.08960*, 2026.

Yuanlin Duan, Wensen Mao, and He Zhu. Learning world models for unconstrained goal navigation. In *Advances in Neural Information Processing Systems*, 2024.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

Francesco Faccio, Vincent Herrmann, Aditya Ramesh, Louis Kirsch, and Jürgen Schmidhuber. Goal-conditioned generators of deep policies. In *AAAI Conference on Artificial Intelligence*, 2023.

Jesse Farebrother, Matteo Pirotta, Andrea Tirinzoni, Marc G. Bellemare, Alessandro Lazaric, and Ahmed Touati. Compositional planning with jumpy world models. *arXiv preprint arXiv:2602.19634*, 2026.

Xudong Gong, Dawei Feng, Kele Xu, Bo Ding, and Huaimin Wang. Goal-conditioned on-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, 2024.

Nico Gürtler and Georg Martius. Long-horizon planning with predictable skills. *Reinforcement Learning Journal*, 6:1247–1272, 2025.

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. In *Advances in Neural Information Processing Systems*, 2022.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *Nature*, 640:647–653, 2025.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, 2022.

Nicklas Hansen, Jyothir S V, Vlad Sobal, Yann LeCun, Xiaolong Wang, and Hao Su. Hierarchical world models as visual whole-body humanoid controllers. In *International Conference on Learning Representations*, 2025.

Marc Höftmann, Jan Robine, and Stefan Harmeling. Backward learning for goal-conditioned policies. *arXiv preprint arXiv:2312.05044*, 2023.

Edward S. Hu, Richard Chang, Oleh Rybkin, and Dinesh Jayaraman. Planning goals for exploration. In *International Conference on Learning Representations*, 2023.

Hyungkyu Kang, Byeongchan Kim, and Min-hwan Oh. Latent representation alignment for offline goal-conditioned reinforcement learning. In *International Conference on Machine Learning*, 2026.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. In *International Joint Conference on Artificial Intelligence*, 2022.

Aneesh Muppidi, Firas Darwish, Dylan Cope, João F. Henriques, and Jakob N. Foerster. Finding the time to think: Learning planning budgets in real-time RL. *arXiv preprint arXiv:2606.26463*, 2026.

Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. In *Advances in Neural Information Processing Systems*, 2019.

Evgenii Opryshko, Junwei Quan, Claas Voelcker, Yilun Du, and Igor Gilitschenski. Test-time graph search for goal-conditioned reinforcement learning. In *International Conference on Machine Learning*, 2026.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: Offline goal-conditioned RL with latent states as actions. In *Advances in Neural Information Processing Systems*, 2023.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations*, 2025.

Robin Schiewer, Anand Subramoney, and Laurenz Wiskott. Exploring the limits of hierarchical world models in reinforcement learning. *Scientific Reports*, 14:26856, 2024.

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: 604–609, 2020.

Dikshant Shehmar, Matthew Schlegel, Matthew E. Taylor, and Marlos C. Machado. Laplacian representations for decision-time planning. In *International Conference on Machine Learning*, 2026.

Lucy Xiaoyang Shi, Joseph J. Lim, and Youngwoon Lee. Skill-based model-based reinforcement learning. In *Conference on Robot Learning*, 2022.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999.

Mianchu Wang, Rui Yang, Xi Chen, Hao Sun, Meng Fang, and Giovanni Montana. GOPlan: Goal-conditioned offline reinforcement learning by planning with learned models. *Transactions on Machine Learning Research*, 2024.

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning*, 2023.

Clarisse Wibault, Alexander Goldie, Antonio Villares, Maike Osborne, and Jakob Foerster. Abstraction for offline goal-conditioned reinforcement learning. *arXiv preprint arXiv:2605.22711*, 2026.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information theoretic MPC for model-based reinforcement learning. In *IEEE International Conference on Robotics and Automation*, 2017.

Zhao Yang, Thomas M. Moerland, Mike Preuss, Aske Plaat, and Edward S. Hu. Reset-free reinforcement learning with world models. *Transactions on Machine Learning Research*, 2025.

A Per-Dataset Planning Settings

Algorithm 1 summarizes the arbitration loop of Section 3.

Algorithm 1 World-Model Policy Arbiter

Require: frozen policies $\mathbf{\Pi}$, dynamics ensemble $\{\hat{f}_{\theta_e}\}_{e=1}^{E}$, value ensemble $\{V_{\phi_e}\}_{e=1}^{E}$, goal g, planning horizon k, commitment c

while episode not terminated **do**
  for $i = 1, \ldots, M$ **do**
    Roll out π_i for k imagined steps through every ensemble member
    Compute S_i with Equation (11)
  $i^* \leftarrow \arg\max_i S_i$
  Execute frozen policy π_{i^*} in the real environment for up to c steps

Table 2 lists the settings used throughout WMPA main results (Table 1).

Table 3 varies k and c on three datasets with large gains. The interval is not tuned extensively for each dataset; instead, it is kept the same within each dataset family. Across all 18 cells, every cell retains a gain of at least 26 points over the best policy. This table shows that the large gains on these manipulation tasks are robust to the hyperparameter setting and are attributable to the WMPA algorithm itself.

B No-Model Switching Controls

Table 4 reports Random-Switch alongside WMPA on every dataset, with both methods using the same commitment interval c, so the two columns differ only in how the next policy is chosen: a uniform draw from the bank, or the learned score. Uninformed re-drawing on the same schedule

Table 2: **Settings per task family**. The value head is the parameterization of the scorer applied to every candidate's imagined states (metric: latent-distance value with separate encoders; direct: a single network on the state–goal pair, Section 3.2); the single interval $k{=}c$ is the imagination horizon and commitment of WMPA and, identically, the re-draw interval of RANDOM-SWITCH; the remaining columns (H, γ, κ, λ_{smooth}) are the training settings of the world model and its value head.

Family	datasets	value head	$k{=}c$	WM H	γ	κ	λ_{smooth}
Maze	2	metric	1	10	0.999	0.9	10
Cube	6	metric	5	5	0.99	0.7	0
Scene	2	metric	10	5	0.998	0.7	10
Puzzle	8	direct	10	5	0.99	0.9	–

Table 3: **Sensitivity to the imagination horizon k and the commitment c** on three datasets with large gains ($k \geq c$). A row fixes k, a column fixes c, and the diagonal is $k = c$; the cell reported in Table 1 is underlined. Success is the mean $\pm$ the half-width of the 95% interval, switches are policy changes per episode, and latency is the mean wall-clock time per environment step on the CPU evaluation nodes. A cell costs $18k/c$ model calls per step, so only the diagonal keeps the budget of Table 1.

Dataset	BEST-POLICY	k	success rate (%) $c{=}1$	$c{=}5$	$c{=}10$	switches per episode $c{=}1$	$c{=}5$	$c{=}10$	latency (ms per step) $c{=}1$	$c{=}5$	$c{=}10$
cube-double-play	36 ± 7	1	63 ± 4			147			3.9		
		5	65 ± 4	$\underline{69 \pm 3}$		107	38		15.5	3.0	
		10	64 ± 3	75 ± 3	73 ± 4	90	30	19	27.6	5.9	2.9
scene-play	52 ± 5	1	93 ± 3			135			4.1		
		5	93 ± 2	92 ± 2		94	35		15.5	3.0	
		10	89 ± 3	93 ± 3	$\underline{87 \pm 4}$	85	29	21	31.8	6.6	3.0
puzzle-4x4-play	24 ± 5	1	50 ± 6			231			16.8		
		5	52 ± 11	53 ± 6		167	56		23.0	5.9	
		10	55 ± 8	54 ± 10	$\underline{51 \pm 8}$	131	49	29	41.5	6.7	4.5

averages 39% against 58% for WMPA; paired per episode, WMPA is significantly better on 13 datasets (Section 4.2).

Table 5 reports results for the stall-restart controller (Section 4.2), which switches in response to *evidence of being stuck* rather than on a fixed schedule. The controller draws its first policy uniformly at random from the bank and keeps it until the displacement over the last m steps falls below ε; it then restarts with one of the *other* members, drawn uniformly at random, and waits at least m steps before the trigger can fire again. Because the controller uses neither a world model nor a value function, it serves as a test of whether the gains from switching come from escaping stalls rather than from choosing the right policy. Across $m \in \{5, 10, 25, 50\}$ (shorter windows trigger restarts more readily), the controller exceeds WMPA on only 2 of the 18 datasets: `cube-triple-play`, where all methods have near-zero success, and `pointmaze-medium-navigate`, where it reaches 94–99% success, compared with 79% for WMPA. RANDOM-SWITCH also reaches this range there (Table 4). A deterministic policy that stalls keeps stalling, and a restart with a different member can escape the stall, whereas WMPA may keep re-selecting the stalled policy in such states. A stall trigger is therefore a natural addition to WMPA on tasks where policies stall. On the manipulation datasets, however, the stall-restart controller does not recover the gains of WMPA, so these gains do not come from escaping stalls alone.

C MODEL-BASED CONTROLS AND SCORER COMPARISONS

Table 6 is discussed in Section 4.1 and Table 7 is discussed in Section 4.2. Table 6 exchanges the portfolio for every-step action-level search on the best policy with the same horizon k and value; Table 7 replaces the learned dynamics with true-simulator branches on every dataset.

Table 4: **RANDOM-SWITCH versus WMPA at a matched commitment interval** (the per-family value of c in Table 2). Best is the best bank policy on each dataset (Table 1). In each row, entries that reach at least 95% of the maximum over the three columns are in bold.

Dataset	c	Best	RANDOM-SWITCH	WMPA
pointmaze-medium-navigate	1	76 ± 17	$\mathbf{89 \pm 13}$	79 ± 3
antmaze-large-navigate	1	$\mathbf{90 \pm 3}$	$\mathbf{92 \pm 2}$	$\mathbf{91 \pm 3}$
cube-single-play	5	71 ± 3	41 ± 4	$\mathbf{85 \pm 3}$
cube-single-noisy	5	$\mathbf{99 \pm 1}$	$\mathbf{96 \pm 2}$	$\mathbf{100 \pm 0}$
cube-double-play	5	36 ± 7	24 ± 4	$\mathbf{69 \pm 3}$
cube-double-noisy	5	26 ± 9	27 ± 4	$\mathbf{60 \pm 7}$
cube-triple-play	5	5 ± 3	$\mathbf{5 \pm 3}$	1 ± 1
cube-triple-noisy	5	9 ± 3	$\mathbf{25 \pm 3}$	18 ± 3
scene-play	10	52 ± 5	42 ± 4	$\mathbf{87 \pm 4}$
scene-noisy	10	30 ± 3	47 ± 5	$\mathbf{70 \pm 5}$
puzzle-3x3-play	10	91 ± 4	21 ± 4	$\mathbf{100 \pm 0}$
puzzle-3x3-noisy	10	91 ± 3	$\mathbf{97 \pm 2}$	$\mathbf{93 \pm 2}$
puzzle-4x4-play	10	24 ± 5	6 ± 3	$\mathbf{51 \pm 8}$
puzzle-4x4-noisy	10	34 ± 5	49 ± 6	$\mathbf{58 \pm 11}$
puzzle-4x5-play	10	14 ± 3	4 ± 2	$\mathbf{18 \pm 3}$
puzzle-4x5-noisy	10	$\mathbf{20 \pm 3}$	13 ± 3	$\mathbf{20 \pm 3}$
puzzle-4x6-play	10	11 ± 2	5 ± 2	$\mathbf{17 \pm 3}$
puzzle-4x6-noisy	10	$\mathbf{19 \pm 3}$	15 ± 3	$\mathbf{20 \pm 3}$

Table 5: **Stall-restart control**: a no-model controller that keeps the incumbent policy and switches to a random *other* policy only when the normalized displacement over the last m steps falls below ε; same bank, episodes, and protocol as Table 1. Numbers within 95% of the row maximum are in bold.

Dataset	Best	stall-restart, window m m=5	10	25	50	WMPA
pointmaze-medium-navigate	76 ± 17	$\mathbf{99 \pm 1}$	$\mathbf{95 \pm 2}$	$\mathbf{94 \pm 2}$	$\mathbf{96 \pm 2}$	79 ± 3
antmaze-large-navigate	$\mathbf{90 \pm 3}$	51 ± 4	51 ± 4	51 ± 4	51 ± 4	$\mathbf{91 \pm 3}$
cube-single-play	71 ± 3	24 ± 5	24 ± 5	24 ± 5	24 ± 5	$\mathbf{85 \pm 3}$
cube-single-noisy	$\mathbf{99 \pm 1}$	53 ± 5	52 ± 5	52 ± 5	52 ± 5	$\mathbf{100 \pm 0}$
cube-double-play	36 ± 7	16 ± 4	15 ± 4	15 ± 4	15 ± 4	$\mathbf{69 \pm 3}$
cube-double-noisy	26 ± 9	11 ± 2	12 ± 3	11 ± 3	11 ± 3	$\mathbf{60 \pm 7}$
cube-triple-play	$\mathbf{5 \pm 3}$	2 ± 1	3 ± 2	2 ± 1	2 ± 1	1 ± 1
cube-triple-noisy	9 ± 3	4 ± 2	5 ± 2	4 ± 1	5 ± 2	$\mathbf{18 \pm 3}$
scene-play	52 ± 5	32 ± 4	31 ± 5	30 ± 4	30 ± 4	$\mathbf{87 \pm 4}$
scene-noisy	30 ± 3	19 ± 3	20 ± 3	18 ± 3	20 ± 4	$\mathbf{70 \pm 5}$
puzzle-3x3-play	91 ± 4	20 ± 3	20 ± 3	20 ± 3	20 ± 3	$\mathbf{100 \pm 0}$
puzzle-3x3-noisy	$\mathbf{91 \pm 3}$	44 ± 6	45 ± 5	43 ± 6	44 ± 6	$\mathbf{93 \pm 2}$
puzzle-4x4-play	24 ± 5	8 ± 2	8 ± 2	8 ± 2	8 ± 2	$\mathbf{51 \pm 8}$
puzzle-4x4-noisy	34 ± 5	12 ± 3	12 ± 3	11 ± 2	11 ± 2	$\mathbf{58 \pm 11}$
puzzle-4x5-play	14 ± 3	4 ± 2	4 ± 1	4 ± 1	4 ± 2	$\mathbf{18 \pm 3}$
puzzle-4x5-noisy	$\mathbf{20 \pm 3}$	7 ± 2	8 ± 2	8 ± 2	8 ± 2	$\mathbf{20 \pm 3}$
puzzle-4x6-play	11 ± 2	5 ± 2	5 ± 2	5 ± 2	5 ± 2	$\mathbf{17 \pm 3}$
puzzle-4x6-noisy	$\mathbf{19 \pm 3}$	10 ± 2	11 ± 3	9 ± 2	10 ± 2	$\mathbf{20 \pm 3}$

Table 8 varies only the value head or the selector, holding everything else fixed, on all eighteen datasets. Q-select, the model-free selector built from the twin-Q value function, is significantly above the best policy on 10 of 18 datasets at $c{=}1$ and on 5 at $c{=}k$; compared with direct-value WMPA, which scores imagined states with the state-value function V of the same agent, it trails by +7 points on average at a matched commitment interval, significantly so on 11 of 18 datasets. Replacing the default value head with the alternative reduces mean success on most datasets and lowers macro-average success by approximately 20 percentage points. The effect is nevertheless task dependent: on cube-triple-play, the direct head raises mean success from 1% to 9%.

Table 9 tests whether the value functions of the policies themselves can replace the shared value function in the setting most favorable to them. In this setting, the bank is restricted to GCIQL, GCIVL, and HIQL, whose value functions are trained to estimate the same quantity (all three learn IQL-style value functions with the same reward and discount factor). Each value function scores the imagined rollout of its own policy (own-value scoring), while the world model and the evaluation episodes remain unchanged. Own-value scoring retains none of the gain obtained with the shared value function. Averaged over the 18 datasets, the success rate of own-value scoring is 4 percentage points below that of the best of the three members, whereas scoring with the shared value function improves upon this member by 11 percentage points. The shared value function also yields significant gains over this member and over own-value scoring on 12 and 14 datasets, respectively. Under own-value scoring, the selection collapses onto the member with the most optimistic value function: on most datasets, the same member is selected at the first arbitration point in more than 90% of the episodes and is rarely replaced thereafter. For instance, on `antmaze-large-navigate`, the arbiter persistently selects GCIVL, which succeeds in 19% of the episodes on its own, rather than HIQL, which succeeds in 90%. These results indicate that scoring every rollout with a single value function is essential for a meaningful comparison among the policies.

Table 10 presents a post hoc diagnostic analysis of value-head ranking quality; this analysis did not influence the value-head choices reported in Table 1. We assess each head by measuring how closely its ranking of states agrees with the observed time to reach a shared goal in offline trajectories. We draw a state s_a and a later state g from one trajectory, and a state s_b from a second trajectory that also reaches g under the success predicate of the environment; a head is counted as concordant when it assigns the higher value $V(\cdot, g)$ to the state with the shorter observed time to reach g. Pairs whose arrival times differ by fewer than 20 steps are discarded, and confidence intervals are obtained by bootstrapping over trajectories. The concordance criterion selects the metric head on `cube-double-play` and `scene-play` and the direct head on `puzzle-4x4-play`, in agreement with the heads used in Table 1; on `cube-triple-play`, it favors the direct head, which is indeed the better of the two in online evaluation (Table 8). Taken together, these results suggest that this criterion can guide the selection of the value head on new datasets for which a goal-attainment predicate is available.

D Score Ablations

Table 11 perturbs the score of Equation (11) at the reported cell on every dataset: horizon aggregation (max over $h \leq k \rightarrow$ final state / mean) and ensemble reduction (mean over members $\rightarrow \min$ / mean$-$std). Across datasets the reductions move the macro-average by at most a point (final state +1, mean -1, $\min$ +0, mean$-$std +0), so neither the optimism of the max nor the ensemble mean carries the gains; pessimistic reductions do not eliminate the low-success failure cases.

E Bank Composition

Every bank in the main study is trained on a single dataset. Table 12 tests the next step: for an environment with two datasets, the six policies trained on its `play` data and the six trained on its `noisy` data are pooled into a single twelve-member bank, and WMPA arbitrates over all twelve members with the same world model, value function, interval, and episodes as the six-member run on that dataset (nothing is retrained; the extra members are simply loaded from the checkpoints of the other dataset). This is the simplest case of a heterogeneous portfolio: same environment and interface, different training data. It isolates how arbitration handles members whose failure modes differ because their training data differ. For each base dataset, the table reports the best policy

Table 6: **WM action search vs. arbitration** with the same horizon k and the same value function on every dataset. The action search applies sampling-based MPC on top of the best policy. At every environment step, the candidates are the deterministic action $\pi(s_t, g)$ of the policy and $N{-}1{=}5$ Gaussian perturbations of this action ($\sigma{=}0.2$). Each candidate is imagined for k steps: only the first action differs across candidates, and the remaining $k{-}1$ steps follow the deterministic output of the policy in closed loop on the imagined states, exactly as in the rollout of a candidate policy in WMPA. The candidates are scored as in WMPA, the first action of the best candidate is executed, and the search is repeated at the next step ($c{=}1$). The action search uses $NEk = 18k$ model steps per environment step, compared with $ME = 18$ for WMPA. The two budgets are equal on the mazes, whereas the budget of the action search is $5\times$ larger on the cube family and $10\times$ larger on the scene and puzzle families. On the puzzle family, both methods use the direct value. Selecting the best policy per dataset as a base for MPC serves as oracle. Numbers that are at least 95% of the row maximum are shown in bold.

Dataset	(k, c) of WMPA	Best	every-step WM action search on Best	WMPA
pointmaze-medium-navigate	$(1, 1)$	76 ± 17	$\mathbf{86 \pm 9}$	79 ± 3
antmaze-large-navigate	$(1, 1)$	$\mathbf{90 \pm 3}$	$\mathbf{92 \pm 3}$	$\mathbf{91 \pm 3}$
cube-single-play	$(5, 5)$	71 ± 3	52 ± 4	$\mathbf{85 \pm 3}$
cube-single-noisy	$(5, 5)$	$\mathbf{99 \pm 1}$	$\mathbf{100 \pm 0}$	$\mathbf{100 \pm 0}$
cube-double-play	$(5, 5)$	36 ± 7	39 ± 4	$\mathbf{69 \pm 3}$
cube-double-noisy	$(5, 5)$	26 ± 9	49 ± 6	$\mathbf{60 \pm 7}$
cube-triple-play	$(5, 5)$	$\mathbf{5 \pm 3}$	0 ± 1	1 ± 1
cube-triple-noisy	$(5, 5)$	9 ± 3	$\mathbf{21 \pm 3}$	18 ± 3
scene-play	$(10, 10)$	52 ± 5	69 ± 4	$\mathbf{87 \pm 4}$
scene-noisy	$(10, 10)$	30 ± 3	65 ± 4	$\mathbf{70 \pm 5}$
puzzle-3x3-play	$(10, 10)$	91 ± 4	$\mathbf{97 \pm 2}$	$\mathbf{100 \pm 0}$
puzzle-3x3-noisy	$(10, 10)$	$\mathbf{91 \pm 3}$	$\mathbf{92 \pm 2}$	$\mathbf{93 \pm 2}$
puzzle-4x4-play	$(10, 10)$	24 ± 5	$\mathbf{54 \pm 9}$	51 ± 8
puzzle-4x4-noisy	$(10, 10)$	34 ± 5	52 ± 11	$\mathbf{58 \pm 11}$
puzzle-4x5-play	$(10, 10)$	14 ± 3	$\mathbf{20 \pm 3}$	18 ± 3
puzzle-4x5-noisy	$(10, 10)$	$\mathbf{20 \pm 3}$	$\mathbf{20 \pm 3}$	$\mathbf{20 \pm 3}$
puzzle-4x6-play	$(10, 10)$	11 ± 2	$\mathbf{20 \pm 3}$	17 ± 3
puzzle-4x6-noisy	$(10, 10)$	$\mathbf{19 \pm 3}$	$\mathbf{20 \pm 3}$	$\mathbf{20 \pm 3}$

Table 7: **True-simulator branches** inside WMPA at the selected cell on every dataset (same value head and commitment; simulator steps replace model calls). Δ is the paired contrast simulator $-$ WMPA; on the puzzle family the branches are scored with the direct value at the reported cell. Numbers within 95% of the row maximum are shown in bold (*: interval excludes zero).

Dataset	(k, c)	Best	WMPA	true-sim. branches	Δ (sim $-$ WMPA)
pointmaze-medium-navigate	$(1, 1)$	$\mathbf{76 \pm 17}$	$\mathbf{79 \pm 3}$	$\mathbf{80 \pm 3}$	+0 [-0, +1]
antmaze-large-navigate	$(1, 1)$	90 ± 3	$\mathbf{91 \pm 3}$	$\mathbf{95 \pm 2}$	+4 [-0, +9]
cube-single-play	$(5, 5)$	71 ± 3	$\mathbf{85 \pm 3}$	$\mathbf{86 \pm 3}$	+2 [-3, +7]
cube-single-noisy	$(5, 5)$	$\mathbf{99 \pm 1}$	$\mathbf{100 \pm 0}$	$\mathbf{100 \pm 0}$	+0 [+0, +1]
cube-double-play	$(5, 5)$	36 ± 7	$\mathbf{69 \pm 3}$	$\mathbf{72 \pm 4}$	+3 [-1, +7]
cube-double-noisy	$(5, 5)$	26 ± 9	$\mathbf{60 \pm 7}$	$\mathbf{61 \pm 4}$	+0 [-10, +9]
cube-triple-play	$(5, 5)$	$\mathbf{5 \pm 3}$	1 ± 1	0 ± 1	-0 [-1, +0]
cube-triple-noisy	$(5, 5)$	9 ± 3	$\mathbf{18 \pm 3}$	$\mathbf{18 \pm 4}$	+0 [-3, +4]
scene-play	$(10, 10)$	52 ± 5	87 ± 4	$\mathbf{95 \pm 3}$	+7 [+5, +10]*
scene-noisy	$(10, 10)$	30 ± 3	70 ± 5	$\mathbf{77 \pm 6}$	+7 [+4, +11]*
puzzle-3x3-play	$(10, 10)$	91 ± 4	$\mathbf{100 \pm 0}$	$\mathbf{100 \pm 0}$	-0 [-1, +0]
puzzle-3x3-noisy	$(10, 10)$	$\mathbf{91 \pm 3}$	$\mathbf{93 \pm 2}$	$\mathbf{93 \pm 3}$	+0 [-2, +3]
puzzle-4x4-play	$(10, 10)$	24 ± 5	$\mathbf{51 \pm 8}$	$\mathbf{53 \pm 6}$	+2 [-4, +8]
puzzle-4x4-noisy	$(10, 10)$	34 ± 5	58 ± 11	$\mathbf{62 \pm 11}$	+4 [-0, +7]
puzzle-4x5-play	$(10, 10)$	14 ± 3	$\mathbf{18 \pm 3}$	$\mathbf{18 \pm 3}$	+0 [-1, +1]
puzzle-4x5-noisy	$(10, 10)$	$\mathbf{20 \pm 3}$	$\mathbf{20 \pm 3}$	$\mathbf{20 \pm 3}$	+0 [+0, +0]
puzzle-4x6-play	$(10, 10)$	11 ± 2	$\mathbf{17 \pm 3}$	$\mathbf{18 \pm 3}$	+1 [-1, +3]
puzzle-4x6-noisy	$(10, 10)$	$\mathbf{19 \pm 3}$	$\mathbf{20 \pm 3}$	$\mathbf{20 \pm 3}$	-0 [-1, +0]

Table 8: **Comparison of value heads and selectors** at a fixed cell on all the datasets. The world model, the bank, and the evaluation episodes are held fixed; only the scorer or the selector varies. WMPA scores imagined states with either the metric value or the direct value; † marks the head reported in Table 1. Q-select uses the same twin-Q value function to rank the action proposed by each policy at the current state, $i^* = \arg\max_i \min_j Q_{\omega,j}(s, \pi_i(s,g), g)$, without any imagined states, re-selecting either at every step (c=1) or every c=k steps. The last column is the paired contrast between the two selectors that share the same underlying agent, direct-value WMPA minus Q-select at c=k, with its 95% interval: it isolates the contribution of scoring imagined successor states with the value function. Entries are the mean success rate ± the half-width of the 95% hierarchical-bootstrap interval, and values within 95% of the row maximum are shown in bold (*: interval excludes zero).

			WMPA (imagined states)		Q-select (no rollout)		
Dataset	(k, c)	Best	metric value	direct value	c=1	c=k	Δ (direct − Q-select, c=k)
pointmaze-medium-navigate	(1, 1)	**76 ± 17**	**79 ± 3**†	56 ± 8	53 ± 9	53 ± 9	+3 [+1, +4]*
antmaze-large-navigate	(1, 1)	**90 ± 3**	**91 ± 3**†	28 ± 4	26 ± 6	26 ± 6	+2 [-2, +6]
cube-single-play	(5, 5)	71 ± 3	**85 ± 3**†	76 ± 4	67 ± 20	58 ± 23	+18 [+2, +44]*
cube-single-noisy	(5, 5)	**99 ± 1**	**100 ± 0**†	**100 ± 0**	**100 ± 0**	**100 ± 0**	+0 [-1, +1]
cube-double-play	(5, 5)	36 ± 7	**69 ± 3**†	62 ± 5	51 ± 5	62 ± 7	-0 [-5, +5]
cube-double-noisy	(5, 5)	26 ± 9	**60 ± 7**†	45 ± 6	38 ± 3	38 ± 6	+7 [+3, +12]*
cube-triple-play	(5, 5)	5 ± 3	1 ± 1†	**9 ± 2**	3 ± 2	7 ± 2	+2 [-1, +4]
cube-triple-noisy	(5, 5)	9 ± 3	**18 ± 3**†	10 ± 4	7 ± 3	8 ± 3	+2 [-1, +5]
scene-play	(10, 10)	52 ± 5	**87 ± 4**†	68 ± 4	64 ± 5	61 ± 5	+8 [+1, +13]*
scene-noisy	(10, 10)	30 ± 3	**70 ± 5**†	57 ± 4	41 ± 5	48 ± 5	+9 [+5, +13]*
puzzle-3x3-play	(10, 10)	91 ± 4	13 ± 4	**100 ± 0**†	**96 ± 3**	76 ± 4	+24 [+20, +28]*
puzzle-3x3-noisy	(10, 10)	**91 ± 3**	41 ± 7	**93 ± 2**†	88 ± 3	**93 ± 2**	-0 [-3, +2]
puzzle-4x4-play	(10, 10)	24 ± 5	17 ± 3	**51 ± 8**†	47 ± 9	26 ± 5	+25 [+15, +37]*
puzzle-4x4-noisy	(10, 10)	34 ± 5	43 ± 5	**58 ± 11**†	53 ± 7	55 ± 10	+3 [-1, +7]
puzzle-4x5-play	(10, 10)	14 ± 3	10 ± 3	18 ± 3†	**20 ± 3**	10 ± 2	+8 [+5, +11]*
puzzle-4x5-noisy	(10, 10)	**20 ± 3**	**20 ± 3**	**20 ± 3**†	**20 ± 3**	18 ± 3	+2 [+1, +3]*
puzzle-4x6-play	(10, 10)	11 ± 2	13 ± 3	17 ± 3†	**20 ± 3**	7 ± 2	+10 [+7, +13]*
puzzle-4x6-noisy	(10, 10)	**19 ± 3**	13 ± 3	**20 ± 3**†	**19 ± 3**	18 ± 3	+2 [+1, +4]*

Table 9: **Own-value function versus a shared value function** on the three bank members whose value networks estimate the same quantity (GCIQL, GCIVL, HIQL; same episodes and protocol as Table 1, reported cell of each family). WMPA-own scores member i's imagined rollout with member i's own value network; WMPA-shared scores every rollout with the head of Table 1. Bold marks values within 95% of the row maximum.

Dataset	(k, c)	Best member	WMPA-own	WMPA-shared
pointmaze-medium-navigate	(1, 1)	**74 ± 3**	54 ± 16	**76 ± 6**
antmaze-large-navigate	(1, 1)	**90 ± 3**	19 ± 4	**90 ± 3**
cube-single-play	(5, 5)	71 ± 3	71 ± 3	**81 ± 4**
cube-single-noisy	(5, 5)	**99 ± 1**	94 ± 4	**100 ± 0**
cube-double-play	(5, 5)	36 ± 7	37 ± 7	**76 ± 5**
cube-double-noisy	(5, 5)	26 ± 9	25 ± 5	**53 ± 4**
cube-triple-play	(5, 5)	**2 ± 1**	**2 ± 2**	**2 ± 1**
cube-triple-noisy	(5, 5)	9 ± 3	3 ± 2	**15 ± 4**
scene-play	(10, 10)	52 ± 5	52 ± 5	**82 ± 6**
scene-noisy	(10, 10)	30 ± 4	36 ± 4	**57 ± 5**
puzzle-3x3-play	(10, 10)	91 ± 4	94 ± 3	**99 ± 1**
puzzle-3x3-noisy	(10, 10)	**91 ± 3**	**92 ± 3**	**93 ± 2**
puzzle-4x4-play	(10, 10)	24 ± 5	26 ± 5	**47 ± 6**
puzzle-4x4-noisy	(10, 10)	34 ± 5	41 ± 8	**55 ± 9**
puzzle-4x5-play	(10, 10)	14 ± 3	14 ± 3	**17 ± 3**
puzzle-4x5-noisy	(10, 10)	**20 ± 3**	**20 ± 3**	**20 ± 3**
puzzle-4x6-play	(10, 10)	11 ± 2	13 ± 2	**17 ± 4**
puzzle-4x6-noisy	(10, 10)	**19 ± 3**	**20 ± 3**	**20 ± 3**

of the pooled bank and WMPA over the pooled bank, and two paired contrasts: WMPA versus the best policy of the pooled bank, and pooled-bank WMPA versus six-member WMPA on the same episodes. On puzzle-4x4-play, the best policy of the pooled bank is a foreign member,

Table 10: **Selecting the value head for datasets.** We report the concordance of each head with the ranking labels described in Appendix C, that is, the fraction of state pairs on which the head assigns the higher value to the state that reaches the shared goal sooner. The last column indicates the head used in Table 1, which was chosen per task family before any evaluation episode was run. An asterisk (*) indicates that the paired confidence interval excludes zero; in such cases, the head selected by the criterion is shown in bold.

Dataset	metric	direct	direct − metric [95% CI]	head in Table 1
`cube-double-play`	**0.75**	0.63	−0.12 [−0.13, −0.12]*	metric
`scene-play`	**0.75**	0.67	−0.08 [−0.08, −0.08]*	metric
`puzzle-4x4-play`	0.70	**0.82**	+0.12 [+0.11, +0.12]*	direct
`cube-triple-play`	0.64	**0.66**	+0.02 [+0.01, +0.03]*	metric

Table 11: **Score ablations at the reported cell**: horizon aggregation and ensemble reduction of Equation (11); same protocol and episodes as Table 1; on the puzzle family the ablations are of the direct value at the reported cell. Entries are mean success ± the half-width of the 95% hierarchical-bootstrap interval (bank seeds, then episodes), rounded to integers; numbers within 95% of the row maximum are in bold.

				horizon agg.		ensemble agg.	
Dataset	(k, c)	Best	WMPA	last	mean	min	mean-std
`pointmaze-medium-navigate`	$(1, 1)$	$\mathbf{76 \pm 17}$	$\mathbf{79 \pm 3}$	$\mathbf{79 \pm 3}$	$\mathbf{79 \pm 3}$	$\mathbf{80 \pm 4}$	$\mathbf{80 \pm 4}$
`antmaze-large-navigate`	$(1, 1)$	$\mathbf{90 \pm 3}$	$\mathbf{91 \pm 3}$	$\mathbf{91 \pm 3}$	$\mathbf{91 \pm 3}$	$\mathbf{92 \pm 3}$	$\mathbf{91 \pm 3}$
`cube-single-play`	$(5, 5)$	71 ± 3	$\mathbf{85 \pm 3}$	$\mathbf{87 \pm 3}$	$\mathbf{84 \pm 4}$	$\mathbf{88 \pm 3}$	$\mathbf{87 \pm 3}$
`cube-single-noisy`	$(5, 5)$	$\mathbf{99 \pm 1}$	$\mathbf{100 \pm 0}$	$\mathbf{100 \pm 0}$	$\mathbf{100 \pm 0}$	$\mathbf{100 \pm 0}$	$\mathbf{99 \pm 1}$
`cube-double-play`	$(5, 5)$	36 ± 7	$\mathbf{69 \pm 3}$	$\mathbf{71 \pm 4}$	66 ± 4	$\mathbf{71 \pm 5}$	$\mathbf{70 \pm 4}$
`cube-double-noisy`	$(5, 5)$	26 ± 9	$\mathbf{60 \pm 7}$	56 ± 5	54 ± 4	51 ± 4	56 ± 4
`cube-triple-play`	$(5, 5)$	$\mathbf{5 \pm 3}$	1 ± 1	1 ± 1	0 ± 1	1 ± 1	0 ± 0
`cube-triple-noisy`	$(5, 5)$	9 ± 3	18 ± 3	17 ± 3	15 ± 3	$\mathbf{19 \pm 3}$	$\mathbf{19 \pm 4}$
`scene-play`	$(10, 10)$	52 ± 5	87 ± 4	$\mathbf{91 \pm 3}$	$\mathbf{92 \pm 4}$	$\mathbf{89 \pm 3}$	$\mathbf{90 \pm 3}$
`scene-noisy`	$(10, 10)$	30 ± 3	70 ± 5	72 ± 5	70 ± 5	$\mathbf{75 \pm 5}$	$\mathbf{78 \pm 5}$
`puzzle-3x3-play`	$(10, 10)$	91 ± 4	$\mathbf{100 \pm 0}$	$\mathbf{99 \pm 1}$	$\mathbf{100 \pm 0}$	$\mathbf{99 \pm 1}$	$\mathbf{100 \pm 1}$
`puzzle-3x3-noisy`	$(10, 10)$	$\mathbf{91 \pm 3}$	$\mathbf{93 \pm 2}$	$\mathbf{92 \pm 3}$	$\mathbf{91 \pm 2}$	$\mathbf{94 \pm 2}$	$\mathbf{93 \pm 2}$
`puzzle-4x4-play`	$(10, 10)$	24 ± 5	51 ± 8	$\mathbf{59 \pm 12}$	54 ± 8	53 ± 7	53 ± 5
`puzzle-4x4-noisy`	$(10, 10)$	34 ± 5	$\mathbf{58 \pm 11}$	$\mathbf{58 \pm 9}$	54 ± 11	$\mathbf{56 \pm 12}$	$\mathbf{55 \pm 12}$
`puzzle-4x5-play`	$(10, 10)$	14 ± 3	$\mathbf{18 \pm 3}$	$\mathbf{19 \pm 3}$	$\mathbf{18 \pm 3}$	17 ± 3	17 ± 3
`puzzle-4x5-noisy`	$(10, 10)$	$\mathbf{20 \pm 3}$	$\mathbf{20 \pm 3}$	$\mathbf{20 \pm 3}$	$\mathbf{20 \pm 3}$	$\mathbf{20 \pm 3}$	$\mathbf{20 \pm 3}$
`puzzle-4x6-play`	$(10, 10)$	11 ± 2	17 ± 3	$\mathbf{19 \pm 3}$	$\mathbf{18 \pm 3}$	$\mathbf{18 \pm 3}$	$\mathbf{18 \pm 3}$
`puzzle-4x6-noisy`	$(10, 10)$	$\mathbf{19 \pm 3}$	$\mathbf{20 \pm 3}$	$\mathbf{20 \pm 3}$	$\mathbf{20 \pm 3}$	$\mathbf{19 \pm 3}$	$\mathbf{19 \pm 3}$

namely the GCIQL policy trained on the `noisy` data, which succeeds on 34% of the `play` episodes, compared with 24% for the native one.

Where the added members contribute new competencies, the pooled bank outperforms the original one (`cube-double-play`: +10 [+6, +15]*); where they largely duplicate competencies already present, the pooled bank matches the original (`scene-play`: +1 [-3, +5]; `puzzle-4x4-play`: +2 [-4, +8]), which indicates that the scorer is not misled by redundant or out-of-distribution candidates; moreover, on `puzzle-4x4-play`, WMPA still gains +18 [+11, +26]* points over a best policy that is itself foreign. We interpret this as evidence that WMPA can be applied without re-tuning to banks assembled from policies trained on different data, provided that they share the same observation, action, and goal interfaces.

F World-Model Architecture and Training Details

Data. Each auxiliary model is trained on the OGBench training split of its dataset, with the benchmark's validation split used only for monitoring. Observation and one-step-delta statistics $(\mu_s, \sigma_s, \mu_\Delta, \sigma_\Delta)$ are computed once from valid training transitions and stored in the checkpoint.

Table 12: **Pooled play+noisy banks**: the six policies of the base dataset plus the six trained on its sibling dataset, arbitrated with the base dataset's world model, value, and interval on the base dataset's episodes. P: bank size; best policy of that bank and its success; WMPA over that bank; paired contrasts of WMPA against the bank's best policy and of pooled-bank WMPA against the six-member WMPA of the row above. Numbers within 95% of the row maximum are in bold. (*: interval excludes zero).

Dataset	Bank	P	Best member	WMPA	Δ vs best member	Δ vs six-member WMPA
cube-double-play	Full bank	6	GCIQL (36 ± 7)	$\mathbf{69 \pm 3}$		
	Pooled play+noisy	12	GCIQL (36 ± 7)	$\mathbf{79 \pm 3}$	+43 [+35, +50]*	+10 [+6, +15]*
scene-play	Full bank	6	GCIQL (52 ± 5)	$\mathbf{87 \pm 4}$		
	Pooled play+noisy	12	GCIQL (52 ± 5)	$\mathbf{88 \pm 3}$	+37 [+30, +43]*	+1 [-3, +5]
puzzle-4x4-play	Full bank	6	GCIQL (24 ± 5)	$\mathbf{51 \pm 8}$		
	Pooled play+noisy	12	GCIQL (noisy) (34 ± 4)	$\mathbf{53 \pm 6}$	+18 [+11, +26]*	+2 [-4, +8]

Dynamics ensemble. $E = 3$ members with disjoint parameters; each is an MLP with three hidden layers of width 512, GELU activations, and layer normalization, mapping the normalized state and action to a normalized delta of the full observation vector (observation dimension 28–115 depending on the dataset). Training unrolls each member for H steps ($H = 5$ for cube, scene, and puzzle; $H = 10$ for the two mazes) with a curriculum that increases the unrolled horizon from 1 to H over the first 10^5 steps. Members share the training batches and are diversified only by independent parameter initialization (and by unrolling their own predictions); at arbitration time value head e scores the rollout of dynamics member e, and the ensemble is reduced by the mean over member pairs.

Metric value head. For each ensemble member, a goal representation ψ (an MLP $512 \times 3 \to 10$, length-normalized to $\sqrt{10}$) and state and goal encoders φ_S, φ_G (each an MLP $512 \times 3 \to 64$) define $V = -\|\varphi_S(s) - \varphi_G(\psi(g))\|_2$, following LAVL (Kang et al., 2026); target networks are Polyak-averaged with $\tau = 0.005$, and goals are relabeled with the OGBench value-goal mixture (0.2 current, 0.5 geometric future, 0.3 random). The per-family γ, κ, and λ_{smooth} are listed in Table 2. Where $\lambda_{\text{smooth}} > 0$ (mazes and scene), the local-smoothness regularizer softly bounds the one-step change of the value toward a random goal $\tilde{g}$,

$$\mathcal{L}_{\text{smooth}} = \mathbb{E}\Big[0.1\,\text{softplus}\Big(10\big[\big(V_{\phi_e}(s',\tilde{g}) - V_{\phi_e}(s,\tilde{g})\big)^2 - \tau_{\bar{V}}^2\big]\Big)\Big], \qquad \tau_{\bar{V}} = 1-(1-\gamma)\,\mathbb{E}[\bar{V}], \quad (12)$$

where $\mathbb{E}[\bar{V}]$ is an exponential moving average of the mean value; this term is added to Equation (5) with weight λ_{smooth}. The expectile target is the implicit-maximization heuristic of IQL-style value learning (Kostrikov et al., 2022; Park et al., 2023). Under stochastic transitions, it is also optimistic with respect to next-state noise, biasing the scorer; pessimistic ensemble reductions that counteract this bias change macro-average success by at most one point (Appendix D).

Direct value head (puzzle). The direct value V_θ follows the design of GCIQL, namely an MLP with three hidden layers of width 512 and layer normalization, and is trained with the puzzle recipe of the reference implementation: the same relabeling mixture, $\gamma = 0.99$, $\tau = 0.005$, $\kappa = 0.9$, and no value or target clipping. The two action-value heads $Q_{\omega,1}, Q_{\omega,2}$ share this architecture and the training batches; $\bar{Q}$ in Equation (7) is the minimum of their target copies, and the same $\min_j Q_{\omega,j}$ is the scorer of the model-free Q-select baseline (Table 8).

Optimization. The dynamics and value losses are summed with unit weights and optimized with Adam (learning rate 3×10^{-4}) at batch size 1024 for 10^6 steps on a single H100 MIG slice. The checkpoint at 10^6 steps is used throughout. The same code path serves both training and test-time imagination, so that predicted states can be consumed directly by every frozen policy. The joint optimization described here applies to the dynamics and metric-value networks. The direct scorer is loaded from the pretrained GCIQL checkpoint and remains frozen during model training.

G Evaluation Protocol, Statistics, and the Static Per-Episode Oracle

Evaluation protocol and statistics. Every number follows the official OGBench evaluation: five goals per dataset, 50 episodes per goal, deterministic actions (250 episodes per bank seed, 750 per method). Reset seeds are a deterministic function of the dataset, the goal, and the episode index, so every method in this paper, each bank policy, WMPA, and every model-free and model-based control, is evaluated on the same episodes, and every contrast is paired per episode. Uncertainty is a hierarchical bootstrap over bank seeds and then episodes (10^4 resamples, 95% percentile intervals): intervals on success rates resample a method's episode outcomes, and intervals on contrasts resample the per-episode differences; a difference is significant when its interval excludes zero.

Table 13 reports, for each dataset, the bank statistics introduced in Section 4.3. Here, $B_{\max}$ denotes the success rate of the best policy, and D_{gap} denotes the performance gap between the best and second-best policies. Both quantities are computed from the scores in Table 1. Because every bank member is evaluated on the same episodes as WMPA and RANDOM-SWITCH, namely 750 episodes per dataset (5 goals $\times$ 50 episodes $\times$ 3 bank seeds), the static per-episode oracle requires no additional runs. When we state that WMPA exceeds the oracle, we count only the datasets on which it does so by at least one percentage point (10 datasets). As a pre-specified criterion, we also report a paired hierarchical bootstrap confidence interval for the per-episode difference WMPA$-U$ on the same episodes: the interval lies entirely above zero on 7 datasets.

Table 13: **Bank statistics and the static per-episode oracle (%).** Here, U is the success rate of a hindsight selector that picks a successful policy for each episode whenever one exists, and $U - B_{\max}$ is the headroom available to any per-episode selector that does not switch policies within an episode. The WMPA column reports the scores from Table 1, and ΔWMPA is the difference between WMPA and $B_{\max}$. The WMPA$-U$ column reports the paired per-episode difference between WMPA and the oracle on the same episodes, together with a hierarchical bootstrap confidence interval computed over bank seeds and episodes. Among $B_{\max}$, U, and WMPA, entries at or above 95% of the row maximum are shown in bold; note that U is a hindsight bound rather than a deployable method. The symbol * indicates that the confidence interval excludes zero.

Dataset	$B_{\max}$	D_{gap}	U (oracle)	$U - B_{\max}$	WMPA	ΔWMPA [95% CI]	WMPA$-U$ [95% CI]
pointmaze-medium-navigate	76 ± 17	2	$\mathbf{98 \pm 1}$	22	79 ± 3	+3 [-17, +17]	-19 [-22, -16]*
antmaze-large-navigate	90 ± 3	4	$\mathbf{99 \pm 1}$	9	91 ± 3	+0 [-3, +4]	-9 [-12, -6]*
cube-single-play	71 ± 3	22	$\mathbf{81 \pm 3}$	10	$\mathbf{85 \pm 3}$	+14 [+10, +18]*	+4 [-1, +9]
cube-single-noisy	$\mathbf{99 \pm 1}$	28	$\mathbf{100 \pm 0}$	1	$\mathbf{100 \pm 0}$	+1 [+0, +2]*	+0 [-0, +1]
cube-double-play	36 ± 7	1	57 ± 6	21	$\mathbf{69 \pm 3}$	+33 [+25, +40]*	+12 [+6, +18]*
cube-double-noisy	26 ± 9	11	31 ± 6	6	$\mathbf{60 \pm 7}$	+35 [+30, +40]*	+29 [+24, +34]*
cube-triple-play	5 ± 3	2	$\mathbf{9 \pm 3}$	4	1 ± 1	-4 [-8, -1]*	-8 [-11, -5]*
cube-triple-noisy	9 ± 3	7	11 ± 3	2	$\mathbf{18 \pm 3}$	+9 [+6, +12]*	+7 [+3, +10]*
scene-play	52 ± 5	9	74 ± 6	22	$\mathbf{87 \pm 4}$	+36 [+30, +41]*	+13 [+8, +19]*
scene-noisy	30 ± 3	3	52 ± 4	22	$\mathbf{70 \pm 5}$	+41 [+36, +45]*	+19 [+14, +23]*
puzzle-3x3-play	91 ± 4	79	92 ± 3	1	$\mathbf{100 \pm 0}$	+9 [+5, +13]*	+8 [+5, +11]*
puzzle-3x3-noisy	91 ± 3	34	$\mathbf{97 \pm 2}$	6	$\mathbf{93 \pm 2}$	+1 [-1, +4]	-5 [-7, -2]*
puzzle-4x4-play	24 ± 5	13	37 ± 5	13	$\mathbf{51 \pm 8}$	+27 [+18, +36]*	+14 [+7, +21]*
puzzle-4x4-noisy	34 ± 5	15	52 ± 7	18	$\mathbf{58 \pm 11}$	+24 [+16, +32]*	+6 [-2, +17]
puzzle-4x5-play	14 ± 3	9	17 ± 3	3	$\mathbf{18 \pm 3}$	+4 [+3, +6]*	+1 [-0, +3]
puzzle-4x5-noisy	$\mathbf{20 \pm 3}$	0	$\mathbf{20 \pm 3}$	0	$\mathbf{20 \pm 3}$	+0 [+0, +1]	-0 [-1, +0]
puzzle-4x6-play	11 ± 2	0	$\mathbf{19 \pm 3}$	8	17 ± 3	+6 [+3, +9]*	-2 [-6, +2]
puzzle-4x6-noisy	19 ± 3	1	$\mathbf{22 \pm 3}$	3	20 ± 3	+1 [-0, +2]	-2 [-3, -1]*

H Selection Behavior and Runtime

Training cost. On the same GPU slice (an H100 3g.40gb MIG partition), training the world model together with its value head takes 3.3 hours per dataset on average, compared with 2.7 hours for a single policy and approximately 16 hours for a six-policy bank, a cost that the best-policy baseline also incurs because all candidates must be trained before one is kept. Test-time latency grows linearly with the bank size M, because the arbiter makes ME model calls per environment step: the twelve-member pooled banks of Appendix E run at 5–6 ms per step, roughly twice the per-arbitration latency of a six-member bank.

Arbitration behavior. Table 14 reports how the portfolio is used: episode length, switches per episode for WMPA and RANDOM-SWITCH, arbitration decisions per episode, world-model calls per environment step (dynamics; value calls are identical), the entropy of the policy-usage distribution (bits; $\log_2 6 \approx 2.58$ for uniform usage of a six-member bank), and wall-clock latency per environment step. Latency was measured on the evaluation nodes (AMD EPYC 9135, 8 threads, JAX on CPU, no GPU) and includes batched policy inference, dynamics and value evaluation, and the winner's action. WMPA switches less often than RANDOM-SWITCH at the same interval (it frequently re-selects the incumbent), and its usage entropy is high on the datasets with the largest gains, showing that the arbiter does not collapse to a single bank member. Evidence that within-episode composition improves success is provided separately by the oracle comparison in Appendix G.

Table 14: Selection behavior and runtime of WMPA (selected (k, c)) and RANDOM-SWITCH (same c). ep. len and decisions/ep belong to WMPA.

			switches/ep					ms/step (CPU)		
Dataset	(k, c)	ep. len	WMPA	Rand.	decisions/ep	WM calls/step	usage H (bits)	WMPA	Rand.	fixed
pointmaze-medium-navigate	$(1, 1)$	295	24.8	246.5	294.8	18.0	1.30	3.76	0.21	0.18
antmaze-large-navigate	$(1, 1)$	475	335.9	425.8	475.2	18.0	2.57	3.88	0.19	0.15
cube-single-play	$(5, 5)$	90	13.0	27.3	18.3	18.3	2.54	3.29	0.35	0.23
cube-single-noisy	$(5, 5)$	46	3.7	13.0	9.5	18.8	1.61	3.24	0.24	0.30
cube-double-play	$(5, 5)$	264	37.8	73.8	53.1	18.1	2.55	2.97	0.23	0.22
cube-double-noisy	$(5, 5)$	310	41.4	69.9	62.3	18.1	2.45	2.89	0.23	0.34
cube-triple-play	$(5, 5)$	996	127.1	161.0	199.1	18.0	2.51	2.77	0.25	0.21
cube-triple-noisy	$(5, 5)$	870	103.9	139.9	174.1	18.0	2.54	2.96	0.26	0.26
scene-play	$(10, 10)$	277	20.7	48.7	28.1	18.3	2.48	2.98	0.27	0.24
scene-noisy	$(10, 10)$	355	20.7	44.4	35.8	18.2	2.40	2.60	0.23	0.19
puzzle-3x3-play	$(10, 10)$	195	15.8	36.9	20.0	18.4	2.50	3.85	0.25	0.23
puzzle-3x3-noisy	$(10, 10)$	102	3.3	18.0	10.7	18.7	1.37	4.26	0.26	0.27
puzzle-4x4-play	$(10, 10)$	361	29.1	40.0	36.3	18.1	2.52	4.46	0.26	0.24
puzzle-4x4-noisy	$(10, 10)$	281	16.1	32.0	28.3	18.2	2.09	3.50	0.40	0.20
puzzle-4x5-play	$(10, 10)$	840	61.3	80.5	84.1	18.0	2.34	4.19	0.27	0.29
puzzle-4x5-noisy	$(10, 10)$	811	41.2	73.7	81.2	18.0	2.09	4.13	0.25	0.21
puzzle-4x6-play	$(10, 10)$	858	66.1	80.2	85.9	18.0	2.44	3.82	0.45	0.34
puzzle-4x6-noisy	$(10, 10)$	819	43.9	72.8	82.0	18.0	2.08	4.38	0.27	0.41

I WHEN DO IMAGINED ROLLOUTS SUPPORT USEFUL ARBITRATION?

Fix a commitment interval c, and let π_b be the fixed bank member with the highest success probability under the evaluation distribution. Write $J(\pi)$ for the probability that policy π reaches the goal by the deadline T. A decision state x includes the environment state, goal, remaining time, and success or termination status. Success and termination are absorbing, and the final commitment interval is shortened to the remaining episode length. There are $N = \lceil T/c \rceil$ decision slots, with absorbing trajectories padded to the deadline.

The *chunk advantage* of member i relative to π_b is

$$A_i(x) = \Pr(\text{success} \mid x,\ \pi_i \text{ for one chunk, then } \pi_b) - \Pr(\text{success} \mid x,\ \pi_b \text{ throughout}). \tag{13}$$

This quantity measures the benefit of using member i for the next chunk while holding the subsequent controller fixed. Since $A_b(x) = 0$, the largest available advantage,

$$G(x) = \max_i A_i(x) \geq 0, \tag{14}$$

measures the opportunity to improve on the best fixed member at that decision point.

Let $S_i(x)$ be the imagined-rollout score used by WMPA, and let $S_i^\star(x)$ be the expected value of the same complete scoring rule under the true dynamics, with the learned value head unchanged. In particular, the expectation is taken after applying the rule's aggregation over ensemble members and maximum over rollout steps. Any sampling error in the computed score $S_i(x)$ is included in the discrepancy below. Define

$$e(x) = \max_i |S_i(x) - S_i^\star(x)|. \tag{15}$$

Thus, $e(x)$ measures the largest discrepancy between the computed and true-dynamics scores across the bank. After success or termination, set all advantages and scores to zero.

Assumption I.1 (Separation of inferior choices)**.** For some tolerance $0 \le \delta < 1$ and margin $\Delta > 0$, at every active decision state visited by WMPA,

$$A_i(x) < G(x) - \delta \quad \Longrightarrow \quad S_i^\star(x) \le \max_j S_j^\star(x) - \Delta. \tag{16}$$

The assumption requires the value head, when applied to true-dynamics rollouts, to separate any member whose advantage is more than δ below the best by a score margin of at least Δ. It does not require correct ordering among members whose advantages are within δ of the best. The tolerance is measured in success-probability units, whereas the margin is measured in the scorer's own units.

Theorem I.2 (Improvement through robust selection)**.** *Under Assumption I.1, let X_n be the decision state at slot n under* WMPA*, and let $\mathcal{A}_n$ be the event that X_n is active. Then*

$$\begin{aligned} J(\pi_{\mathrm{WMPA}}) - J(\pi_b) \ge \sum_{n=0}^{N-1} \Bigg(& \mathbb{E}_{\mathrm{WMPA}}[G(X_n)] - \delta \Pr_{\mathrm{WMPA}}(\mathcal{A}_n) \\ & - (1-\delta) \Pr_{\mathrm{WMPA}}\left(\mathcal{A}_n,\, e(X_n) \ge \frac{\Delta}{2}\right)\Bigg). \end{aligned} \tag{17}$$

In particular, a positive right-hand side guarantees improvement over every fixed bank member.

The bound compares the available improvement with two sources of selection loss at active decision slots: the tolerated difference δ from the best chunk advantage, and decisions at which score error is large enough to overturn the separation margin. Errors smaller than half the margin cannot cause a selection more than δ below the best advantage.

Proof. At an active decision state x, let $\hat{i}$ maximize the computed score and let j maximize the true-dynamics score. Since $S_{\hat{i}}(x) \ge S_j(x)$,

$$S_j^\star(x) - S_{\hat{i}}^\star(x) \le 2e(x). \tag{18}$$

If $e(x) < \Delta/2$, Assumption I.1 implies $A_{\hat{i}}(x) \ge G(x) - \delta$. Otherwise, $G(x) - A_{\hat{i}}(x) \le 1$, since the common reference term cancels and the remaining difference is between two success probabilities. At an absorbing decision state, all advantages are zero. Consequently, for every decision state x,

$$A_{\hat{i}}(x) \ge G(x) - \delta \mathbf{1}\{x \text{ is active}\} - (1-\delta)\mathbf{1}\left\{x \text{ is active},\ e(x) \ge \frac{\Delta}{2}\right\}. \tag{19}$$

Let $V_b(x)$ denote the probability of success if π_b is followed from x onward. By definition, the expected change in V_b across a chunk executed by member i is $A_i(x)$. Summing these changes along WMPA's trajectory cancels the intermediate reference values. At the deadline, V_b equals the success indicator; its expectation at the initial state equals $J(\pi_b)$. Therefore,

$$J(\pi_{\mathrm{WMPA}}) - J(\pi_b) = \sum_{n=0}^{N-1} \mathbb{E}_{\mathrm{WMPA}}[A_{\hat{i}_n}(X_n)], \tag{20}$$

where $\hat{i}_n$ is the member selected at slot n. Taking expectations in (19) and summing proves the result. □

The result identifies three requirements for useful arbitration: the bank must offer locally advantageous alternatives, the value head must separate substantially inferior choices under true dynamics, and score errors must be small relative to that separation. Accurate dynamics alone are insufficient if the value head favors unhelpful outcomes. Conversely, imperfect imagined rollouts can support reliable selection when their score errors preserve the relevant margin.