Q-MAP FUSION: RECOVERING SHARED DYNAMICS FROM SEPARATELY LEARNED VALUES

Anonymous authors
Paper under double-blind review

ABSTRACT

Value functions in reinforcement learning are tied to particular rewards, yet tasks in the same environment share the dynamics that give rise to those values. We study how knowledge acquired separately across tasks can be combined after training to support new decisions. Building on Bellman inversion, Q-Map Fusion jointly fits a shared transition model to the constraints supplied by multiple Q-functions, then plans under new rewards and termination rules. Recovery uses the source values and known task definitions, requiring no training trajectories, no updates to the Q-functions, and no additional environment interaction. The usefulness of an additional task depends on the ambiguities its values resolve and the accuracy with which they are learned. For finite-state models, our analysis characterizes when the joint constraints control specified one-step predictions even when the full transition model remains unidentified. Experiments in FourRooms, MountainCar, Pendulum, and Reacher demonstrate effective model recovery and planning for held-out tasks across the evaluated tabular and continuous-control settings. Separate matched-budget and perturbation studies show that the benefit of additional values depends on both complementary task information and robustness to value errors. We hope this perspective encourages the reuse of learned values as shared environment knowledge for decisions beyond their original rewards.

1 INTRODUCTION

Reusing task-specific values. In reinforcement learning, a Q-function summarizes the long-term return of an action for a particular task Harmon & Harmon (1996); Tan et al. (2017); Chebotar et al. (2023); Letcher et al. (2026). Agents trained separately for different goals in the same environment learn different views of a common transition law Chevalier-Boisvert et al. (2023); Richens et al. (2025). A new goal or termination rule, however, calls for decisions that none of these agents was trained to make. We call a collection of task-specific Q-functions a *Q bank*. Our aim is to reuse already learned banks so that the information accumulated across tasks can support planning beyond their original objectives.

What learned values leave unresolved. Accurate source values need not determine the dynamics relevant to a new task Saxe et al. (2013); Candadai & Izquierdo (2020); Rawles et al. (2025); Hafner et al. (2025). Two transition models can agree on the returns specified by a bank yet predict different consequences under another reward. Recovering environment information from decision making is a central concern of generalized inverse reinforcement learning Yu et al. (2019); Rolland et al. (2022). For known rewards and numerical values, Bellman inversion provides a direct route to fitting dynamics Adamczyk (2025); Letcher et al. (2026). Building on this formulation, we study how separately acquired, imperfect banks change the information available for new-task planning.

What additional task values contribute. If two candidate successors have identical Bellman measurements under an existing *anchor* bank, redistributing transition probability between them preserves those measurements Bellemare et al. (2017); Xie et al. (2021); Meyn (2024). A supplemental task that assigns different measurements to these successors can distinguish that redistribution. Repeated estimates of an existing task can improve measurement accuracy; different tasks can add measurement directions. However, learned Q-functions introduce errors on both sides of the Bellman equation: in the current values being fitted and in the successor values defining the measurements

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.

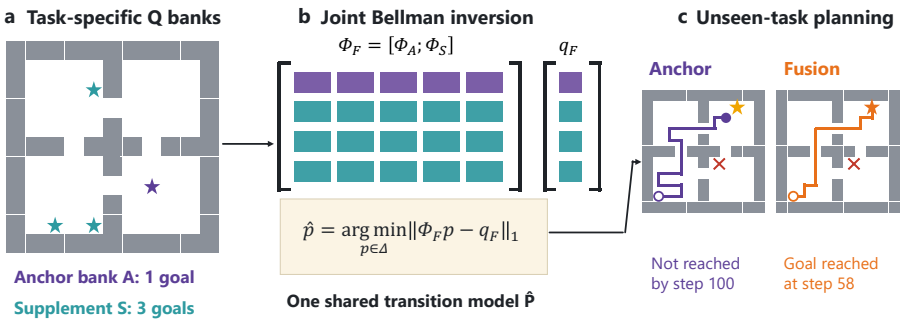


Figure 1: **Q-Map Fusion reuses task-specific values for new-task planning.** (a) An anchor bank and a supplemental bank contain Q-functions learned for different goals in the same environment. (b) Their Bellman measurements are stacked and jointly fitted to a common transition row; solving across state–action queries yields one shared dynamics model. The simplex constraint illustrates the finite-state stochastic case. (c) Planning for a goal outside the source banks is illustrated using the anchor and fused models. The trajectory annotations describe this example; aggregate performance is reported in the experiments.

Chebotar et al. (2023). The usefulness of an added bank therefore depends on both the ambiguities it resolves and the errors it introduces.

Recovering shared dynamics for new tasks. *Separately learned task values can jointly constrain shared dynamics for decisions beyond their original rewards.* We instantiate this view with *Q-Map Fusion*. Given aligned state and action spaces and known reward and termination functions, we jointly fit a dynamics model to the banks' Bellman measurements. The finite-state estimator minimizes summed absolute residuals; a parametric formulation fits a deterministic successor model using the same measurements. Given the query inputs, both fitters use frozen Q-functions without retrieving their training trajectories or using observed successor labels as fitting targets. The recovered model then supports planning under new rewards and termination rules. Figure 1 illustrates this progression from task-specific banks to a shared model and new-task decisions.

A model can support a specified prediction even when its transitions are not fully identified. For finite-state recovery with correctly specified supports and model classes, we analyze feasible, near-optimal fits. We approximately represent a function of the next state through the learned Bellman measurements and a constant, and bound the error in its expectation. The bound accounts for effective Q errors, their amplification by the representation coefficients, the remainder of the chosen representation, and the optimization gap. This connects the information supplied by the banks to one-step predictions used in new-task planning. Additional tasks can improve that representation while introducing estimation error, so their number alone does not determine predictive accuracy.

Evidence across tasks and environments. We evaluate Q-Map Fusion in FourRooms Fu et al. (2020), MountainCar Brockman et al. (2016), Pendulum, and Reacher Towers et al. (2026), covering finite-state and continuous-control settings. Across these environments, the recovered models support planning beyond the source objectives. In Windy FourRooms, acquiring new goals improves average recovery and held-out return more than relearning existing goals at matched sampling and Q-update budgets. At fixed record count, controlled Q-value perturbations show that distinct task measurements preserve recovery and planning more reliably than nearly repeated measurements.

We summarize our contributions as follows:

- We develop Q-Map Fusion for reusing separately learned, frozen Q banks. Building on Bellman inversion, it fits shared dynamics from task measurements and supports planning under new rewards and termination rules.
- We establish a conditional prediction-error bound for finite-state recovery, relating effective Q errors and optimization accuracy to how well the task measurements represent a specified function of the next state. The analysis distinguishes control of specified predictions from full transition identification.
- We evaluate this reuse in finite-state and continuous-control environments, with acquisition-budget comparisons and controlled Q-value perturbations that examine how task information and estimation error affect recovery and planning.

2 Related Work

Recovering dynamics from values. In reinforcement learning, Bellman inversion studies what environment information can be recovered from learned values Malik et al. (2018); Zhao (2022); Hejna & Sadigh (2023); Al-Hafez et al. (2023); Kadurha et al. (2025); Sutton & Barto (2018); Mnih et al. (2013); Schulman et al. (2017). Adamczyk Adamczyk (2025) analyzes successor-state recovery in deterministic environments, including approximate values, and discusses resolving ambiguity by intersecting constraints from multiple cached value functions. Letcher et al. Letcher et al. (2026) develop P-learning, establish identifiability conditions for finite and continuous MDPs, analyze approximate Q-values, and demonstrate planning with extracted models on new tasks. Q-Map Fusion builds directly on this inverse Bellman formulation.

Inverse learning of Markov Decision Process (MDP) components. Generalized inverse reinforcement learning infers uncertain rewards, transitions, and state or action spaces from observed policies that may be suboptimal Ramachandran et al. (2007); Fu et al. (2017); Li et al. (2020); Cao et al. (2021); Dong & Wang (2024). Inverse learning of MDP setting may supplies numerical Q-functions, known rewards and termination rules, and aligned state and action semantics Ng et al. (2000); Dvijotham & Todorov (2010); Herman et al. (2016); Cao et al. (2021). These inputs let us constrain a shared transition model through task-specific Bellman equations and reuse it for subsequent planning.

Reusing values across tasks. The successor representation encodes expected discounted future state occupancy under a policy Barreto et al. (2017); Dayan (1993); Kulkarni et al. (2016); Momennejad et al. (2017); Fujimoto et al. (2021); Borsa et al. (2018). Successor features extend this decomposition to features; combined with generalized policy improvement, they support transfer by evaluating existing policies under a new reward and combining their action values Momennejad et al. (2017); Szepesvári (2022). Universal value function approximators share a value predictor across states and goals Zhong et al. (2013); Schaul et al. (2015). Value composition provides another route: Van Niekerk et al. Van Niekerk et al. (2019); Lambert et al. (2020); Fellows et al. (2024) derive exact composition results for entropy-regularized tasks with deterministic dynamics and rewards differing on a common absorbing set. These approaches reuse structure in value or policy representations.

Models that preserve value predictions. Value-aware model learning measures dynamics errors through their effect on value functions Farahmand et al. (2017); Farahmand (2018); Ayoub et al. (2020); Voelcker et al. (2022). Value equivalence formalizes agreement of Bellman updates on selected policies and functions; for a fixed policy set, the function span determines which models remain indistinguishable Grimm et al. (2020; 2021); Klissarov et al. (2025). Proper value equivalence further shows that preserving the values of all deterministic policies can suffice for optimal planning without uniquely identifying the dynamics Schrittwieser et al. (2020); Grimm et al. (2021); Freed et al. (2024). These results motivate distinguishing full dynamics recovery from useful prediction.

3 Q-Map Fusion

Q-Map Fusion treats source Q-functions as measurements of shared dynamics. We combine the constraints supplied by their tasks, fit one transition model, and reuse that model for new rewards and termination rules. The central question is which additional predictions become available when separately learned task values are brought together.

3.1 Task values as measurements of shared dynamics

Consider tasks with aligned state and action spaces $\mathcal{S}, \mathcal{A}$, shared unknown dynamics $P^\star$, and discount $0 \leq \gamma < 1$. Their banks supply $K \geq 1$ indexed records $\mathcal{B} = \{(\widehat{Q}_i, r_i, d_i)\}_{i=1}^{K}$: frozen estimates of optimal Q-values, known bounded rewards $r_i(s, a, s')$, and termination indicators $d_i(s') \in \{0, 1\}$. Different estimates of the same task remain distinct records. Fusion is performed within each environment; separate environments have separate transition models.

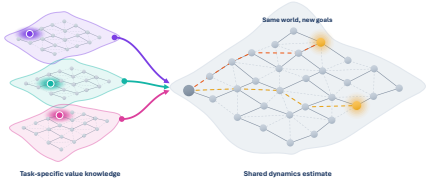


Figure 2: **New-task planning benefit from seen tasks.**

Reward is received on arrival, and termination removes continuation. For $x = (s, a)$ and $\widehat{V}_i(s') = \sup_{a'} \widehat{Q}_i(s', a')$, Bellman inversion Letcher et al. (2026) gives

$$\begin{aligned} \widehat{\phi}_i(x, s') &= r_i(x, s') + \gamma(1 - d_i(s'))\widehat{V}_i(s'), \\ \delta_i(P; x) &= \widehat{Q}_i(x) - \mathbb{E}_{s' \sim P(\cdot|x)}[\widehat{\phi}_i(x, s')]. \end{aligned} \tag{1}$$

All quantities are assumed well-defined and finite. At a fixed query, the current Q-value is a scalar observation; the reward and successor values define the function whose expectation is taken under the unknown transition law. With exact source values, $\delta_i(P^\star; x) = 0$. Different tasks thus measure different expectations of the same dynamics.

Constructing a measurement requires Q-values at the query and the action values needed for continuation at candidate nonterminal successors. At terminal successors only the arrival reward contributes. All queried entries, including terminal-source entries, must follow the stated Bellman convention; Appendix A specifies it and distinguishes optimal-Q inputs from policy-evaluation critics. These queries reuse the frozen functions without retrieving their training trajectories.

3.2 Fitting one model to multiple banks

Supplemental tasks are informative when they distinguish dynamics that the anchor leaves unresolved Adamczyk (2025); Letcher et al. (2026). If two possible successors have identical measurements for every anchor record, moving probability mass between them preserves those measurements. A supplemental task with different measurements at these successors can distinguish that redistribution. The relevant information is variation across possible successors, rather than the bank label attached to a record.

Joint finite-state recovery. For finite, nonempty state and action spaces, supply an ordered, nonempty successor set $\Omega_x = \{s'_1, \ldots, s'_{m_x}\} \subseteq \mathcal{S}$. Set $\widehat{\Phi}_x[i, j] = \widehat{\phi}_i(x, s'_j)$ and $\widehat{q}_x[i] = \widehat{Q}_i(x)$. The prescribed row class $\mathcal{D}_x$ is either the probability simplex Δ_{m_x} or its vertex set $\{e_1, \ldots, e_{m_x}\}$. Supports and model classes remain fixed when records are added.

To isolate information from estimation error, let Φ_x, q_x denote the corresponding exact-value quantities. For record indices I, write $\mathcal{C}_x(I) = \{p \in \mathcal{D}_x : \Phi_x[I, :]p = q_x[I]\}$. Anchor and supplemental indices I_A, I_S satisfy $\mathcal{C}_x(I_A \cup I_S) = \mathcal{C}_x(I_A) \cap \mathcal{C}_x(I_S)$. If the true row is feasible, additional exact constraints preserve it and shrink the compatible set whenever they exclude an anchor-compatible candidate. Repeated exact constraints leave this set unchanged. Unresolved differences h obey $\Phi_x h = 0$ and $\mathbf{1}^\top h = 0$, together with model feasibility. Appendix B distinguishes this ambiguity from the conditions for complete identification.

Learned records may be inconsistent. We therefore fit their joint absolute residuals:

$$L_x(p) = \|\widehat{\Phi}_x p - \widehat{q}_x\|_1, \qquad p_x^{\mathrm{opt}} \in \arg\min_{p \in \mathcal{D}_x} L_x(p). \tag{2}$$

For stochastic rows this is a linear program; for deterministic rows it is column matching. Each column contains the task measurements at one candidate successor: a stochastic row selects a convex combination, while a deterministic row selects one column. After caching successor values, the rows can be solved independently and assembled into $\widehat{P}$, with zero mass outside Ω_x. The objective retains the input reward scales and record multiplicities. Regrouping fixed records changes nothing; repeating some noisy records changes their relative weight. Appendix E gives the solvers and numerical checks.

A parametric realization. A deterministic successor model $f_\theta : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ instead shares parameters across queries. For a prescribed query distribution μ and fixed weights $w_i > 0$, fit

$$\mathcal{L}_\mu(\theta) = \mathbb{E}_{x \sim \mu} \sum_{i=1}^{K} w_i \left| \widehat{Q}_i(x) - \widehat{\phi}_i(x, f_\theta(x)) \right|. \tag{3}$$

The loss must be finite, and predicted successors must lie in the common domain of the rewards, termination functions, and queried values. Fitting changes where the frozen source functions are evaluated; only the successor-model parameters are updated. When available, input derivatives of the

frozen Q-functions may be used to fit the successor model. Query construction, the fitting procedure, and the approximation to the action supremum specify the numerical realization. Parameter sharing couples the queries into a generally nonconvex fit, so the rowwise global-solve properties do not transfer automatically (Appendix E.5).

3.3 From fitted constraints to useful predictions

A model can support a specified prediction without identifying every transition Farahmand et al. (2017); Grimm et al. (2020). For the finite-state estimator, assume $\operatorname{supp} P^\star(\cdot \mid x) \subseteq \Omega_x$ and set $p_x^\star[j] = P^\star(s'_j \mid x)$. The effective error is $\varepsilon_x = \widehat{q}_x - \widehat{\Phi}_x p_x^\star$. Let $Q_i^\star$ denote the exact optimal task values and $V_i^\star(s) = \max_a Q_i^\star(s,a)$. Writing $\Delta Q_i = \widehat{Q}_i - Q_i^\star$ and $\Delta V_i = \widehat{V}_i - V_i^\star$, known and consistent task definitions give

$$\varepsilon_x[i] = \Delta Q_i(x) - \gamma \mathbb{E}_{P^\star(\cdot|x)}[(1 - d_i(S'))\Delta V_i(S')]. \tag{4}$$

Thus errors enter both the fitted current value and the successor measurements. This identity requires neither independence nor zero-mean errors; ε_x is an analysis quantity, not an observed score available to the fitter.

Prediction control. Assume $p_x^\star, \widehat{p}_x \in \mathcal{D}_x \subseteq \Delta_{m_x}$ and $L_x(\widehat{p}_x) \le \min_{p \in \mathcal{D}_x} L_x(p) + \tau_x$, where $\tau_x \ge 0$. For any specified $\psi \in \mathbb{R}^{m_x}$ and representation $\psi^\top = c^\top \widehat{\Phi}_x + \beta \mathbf{1}^\top + \eta_x^\top$,

$$\left|\psi^\top(\widehat{p}_x - p_x^\star)\right| \le \|c\|_\infty \big(2\|\varepsilon_x\|_1 + \tau_x\big) + 2\|\eta_x\|_\infty. \tag{5}$$

Feasibility includes the correct support and model class. Near-optimal fitting controls the discrepancy in the measured directions; probability normalization cancels the constant component. The remaining representation error is bounded using the L1 diameter of the simplex. Appendix C gives the proof.

The bound separates three contributions. The chosen representation leaves a remainder η_x; its coefficients c determine how strongly effective Q errors are amplified; and τ_x accounts for the unnormalized objective gap. A probability-feasibility tolerance is not this objective gap. Different choices of c, β trade representation error against amplification. The bound concerns their combined effect; a small fitted residual alone cannot evaluate the unknown effective error on its right-hand side. When $\eta_x = 0$, exact measurements and an exact solve give identical expectations of ψ, even if the transition row is not unique. With learned values, added records can improve the available representation while also adding error; their number alone does not guarantee a better prediction.

For a specified downstream value V, choose $\psi_{V,x}[j] = r_{\text{new}}(s,a,s'_j) + \gamma(1 - d_{\text{new}}(s'_j))V(s'_j)$. Equation equation 5 then controls its one-step backup under the stated representation and feasibility conditions. Uniform policy-performance bounds require additional assumptions (Appendix D); this rowwise result is not a guarantee for the nonlinear parametric fit.

3.4 Replanning beyond the source tasks

Fixing $\widehat{P}$, a new bounded reward and termination pair $(r_{\text{new}}, d_{\text{new}})$ defines

$$\begin{aligned} b_V^{\widehat{P}}(s,a) &= \mathbb{E}_{s' \sim \widehat{P}(\cdot|s,a)}\big[r_{\text{new}}(s,a,s') + \gamma(1 - d_{\text{new}}(s'))V(s')\big], \\ (\mathcal{T}_{\text{new}}^{\widehat{P}} V)(s) &= \sup_{a \in \mathcal{A}} b_V^{\widehat{P}}(s,a). \end{aligned} \tag{6}$$

The new task enters this backup without requiring a corresponding source Q-function. In finite spaces, exact backups define a γ-contraction. Algorithm 1 accepts the last value update and recomputes action backups at that accepted value before extracting the greedy policy. This convergence concerns the recovered model; performance under $P^\star$ also depends on its prediction error. The complete procedure and stopping analysis are in Appendix E.

For the parametric realization, $\widehat{P}$ places unit mass at $f_{\widehat{\theta}}(s,a)$, reducing the expectation to evaluation at that successor. A continuous-space planner additionally specifies its value representation and action optimization. In either realization, the recovered dynamics remain fixed as the reward or termination rule changes.

Algorithm 1 Finite-state Q-Map Fusion

Require: Finite $\mathcal{S}, \mathcal{A}$; frozen records $\mathcal{B}$; $\gamma \in [0, 1)$; supports $\{\Omega_x\}$; row classes $\{\mathcal{D}_x\}$
Require: New task $(r_{\text{new}}, d_{\text{new}})$; $\epsilon_V > 0$; solver checks and tie rules (Appendix E)
Ensure: Recovered kernel $\widehat{P}$ and policy $\widehat{\pi}$, or failure

1: Form $(\widehat{\Phi}_x, \widehat{q}_x)$ for every $x = (s, a)$ using Equation equation 1.
2: Fit Equation equation 2 rowwise by simplex LP or deterministic column matching.
3: **return** failure if a solver fails or a row fails the declared probability checks.
4: Assemble $\widehat{P}$ from the fitted rows, assigning zero mass outside Ω_x.
5: Set $V_0 \leftarrow 0$.
6: Compute $V_t \leftarrow \mathcal{T}_{\text{new}}^{\widehat{P}} V_{t-1}$, $t = 1, 2, \ldots$, accepting the first V_t with $\|V_t - V_{t-1}\|_\infty \leq \epsilon_V$.
7: Recompute $b_{V_t}^{\widehat{P}}$ and choose $\widehat{\pi}(s) \in \arg\max_a b_{V_t}^{\widehat{P}}(s, a)$ using the fixed tie rule.
8: **return** $(\widehat{P}, \widehat{\pi})$.

4 EXPERIMENTS

We use Q-functions learned for existing tasks to recover an environment model, then use that model to plan for new tasks. The source Q-functions remain frozen during recovery, and the recovered model remains fixed when the test task changes. We evaluate the resulting policies under the true environment dynamics. Thus, our comparisons concern policies planned with different recovered models, rather than the original agents' policies. Source training uses environment transitions; model recovery uses the Q-functions and known task definitions without next-state supervision.

Evaluation. Higher normalized return means better performance on a new task; 1 matches a reference policy planned with the true dynamics. This ratio is not a success rate and can exceed one when the reference planner is approximate. In FourRooms, transition MSE additionally measures error in predicted next-state probabilities. We average the test tasks within each model, then average model trials. Appendix G specifies the protocols and metrics.

4.1 MODEL RECOVERY AND REPLANNING IN FOURROOMS

FourRooms has 68 accessible cells in an 11×11 grid, divided into four rooms connected by narrow doors. The state is the current cell; actions request movement up, right, down, or left, and wall collisions leave the agent in place. We test three dynamics: *Deterministic* executes the requested move, *Windy* can instead move in an adjacent direction, and *Teleporting* jumps to the opposite room on departure from designated cells. The finite state space lets us directly compare the recovered next-state probabilities with the true transition matrix.

Table 1: **Additional source tasks improve stochastic FourRooms models.** Three trials per dynamics variant, with two test tasks per model. Q counts give Anchor/Fusion source records.

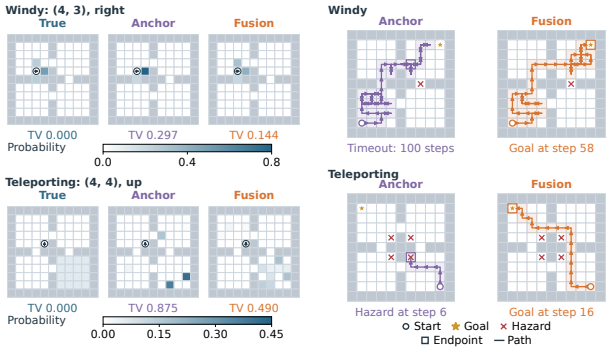
	Q counts	Transition MSE ↓		New-task return ↑	
Dynamics	A/F	Anchor	Fusion	Anchor	Fusion
Deterministic	$1/3$	0	0	1.0000	1.0000
Windy	$1/4$	5.6345	**1.0255**	0.5978	**0.9970**
Teleporting	$5/20$	9.4949	**0.6436**	0.7398	**0.9999**

Source tasks use different goal cells; the deterministic variant also adds fixed background rewards to distinguish its source tasks. We first recover an *Anchor* model from a small set of source Q-functions, then a *Fusion* model that also uses Q-functions for additional goals. The two test tasks require reaching new goals while avoiding unsafe cells: entering an unsafe cell ends the episode without reward. This changes destinations and acceptable routes while leaving the dynamics unchanged. Both recovered models use the same planner. Layout coordinates, source rewards, and test-task definitions are in Appendix H.1.

Table 1 shows improvements in both model accuracy and planning for the stochastic environments. Fusion raises return from 0.598 to 0.997 in Windy and from 0.740 to nearly 1 in Teleporting, while reducing transition MSE by 81.80% and 93.22%, respectively. All six stochastic model pairs improve both metrics. In the deterministic case, the anchor already recovers the dynamics exactly, and fusion

preserves this result. These comparisons add source tasks and their training cost; Section 4.3 separates the effect of different tasks from additional training. Appendix H reports every trial and task.

Recovered transitions and executed plans. Figure 3 connects model recovery to planning. For two fixed state–action queries, panel a shows that Fusion more closely recovers both the local spread of Windy successors and the nonlocal distribution in Teleporting. These selected queries complement the full-kernel errors in Table 1. Panel b executes the resulting policies under the true dynamics. In the selected Windy episode, Fusion reaches the goal at step 58 while Anchor has not reached it by step 100. In Teleporting, Anchor terminates at an unsafe cell at step 6, whereas Fusion reaches the goal at step 16. Appendix H.4 gives the spatial errors, all-start diagnostics, and Q-bank growth analysis for these examples.



(a) Recovered distributions. (b) True environment plans.

Figure 3: **From recovered transitions to new-task behavior. (a)** Successor distributions at one query per environment; TV is the per-query error, not the full-kernel MSE of Table 1. **(b)** Paired rollouts share starts and random streams; trajectories are truncated at 100 of the 200 evaluation steps.

4.2 Planning for new tasks in continuous environments

We test three environments with continuous states and discrete actions. All use custom sparse rewards, with success ending the episode on the first arrival at the target. Each environment has four source goals and three new test tasks. The test tasks change the target location, the state coordinates used by the reward, or a termination constraint.

MountainCar. The state contains the car's position and velocity on a hilly track; three actions push left, apply no thrust, or push right. Source goals specify positions. The new tasks require a target velocity (*Fast car*), arrival at the hilltop without exceeding a speed limit (*Gentle car*), or arrival at either of two positions (*Shortest path*).

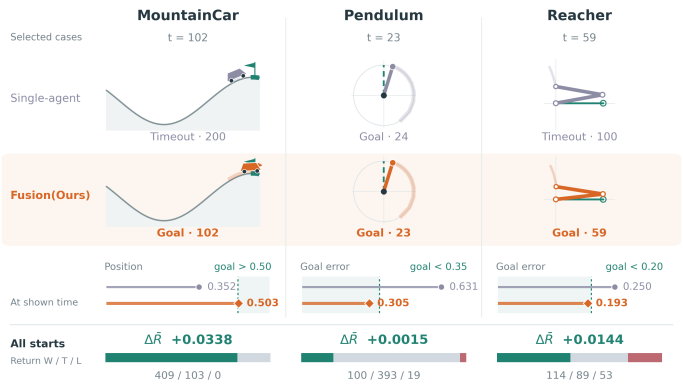


Figure 4: **Model fusion can turn missed goals into arrivals and reach goals earlier.** Cases compare identical starts at the Fusion (Ours) policy's first goal arrival. MountainCar and Reacher show goal arrival versus an eventual single-agent timeout; Pendulum shows a one-step gain through lower angular speed.

Pendulum. The state is angle and angular velocity, controlled by one of three torques. Source goals depend only on angle. The new tasks require arrival near upright with low angular velocity, reaching a target spin velocity, or reaching a specified angle–velocity pair. Success measures first arrival, rather than sustained balance.

Reacher. We use a custom planar two-link simulator with two absolute link angles and two angular velocities. Three torque choices per joint give nine actions, and fingertip position is computed from the angles. Source goals specify fingertip positions; the new tasks target a farther fingertip location, an angle pair, or a velocity pair. Appendix I.2 gives all three environments' task thresholds, planning grids, and evaluation settings.

We compare a *Single-agent model*, recovered from one agent trained on four goals, with a *Two-bank model*, jointly recovered from two specialists trained on two goals each. Both cover the same four

Table 2: **Planning with models recovered from frozen agents.** Mean normalized return over three model fits and all three tasks per environment, conditional on fixed source agents.

Environment	Single-agent model	Two-bank model	Diff.
MountainCar	0.9349	**0.9765**	+0.0416
Pendulum	0.9998	**1.0004**	+0.0006
Reacher	1.0141	**1.0395**	+0.0254



Figure 5: **New tasks benefit from the our model.**

source goals; the two-bank model uses only the specialists. Their source training costs are 20M and 40M environment steps, respectively. Both use the same model architecture, test tasks, planner, and starts. Section 4.3 separately tests equal acquisition budgets.

Fitting and using the model. We fit one deterministic successor model to the frozen source Q-functions, weighting all four source goals equally. For the two-bank model, both specialists constrain this same model. At each predicted successor, we evaluate the source-task reward, termination condition, and maximum Q-value over the discrete actions to form the Bellman residual. Only model parameters are updated; Q-network parameters remain fixed while derivatives through their state inputs are retained. After fitting, we fix the model and run value iteration for each new reward, using the same planning grids and starts in both conditions. Appendix I.1 gives the objective and optimization settings.

Average return increases in all three environments (Table 2). The adjacent task-level comparison in Figure 5 locates the gains: Gentle car and Shortest path improve in MountainCar, and Target angle accounts for most of Reacher's gain. Fast car and Far fingertip have lower mean returns, while Pendulum changes little. These archived comparisons hold source agents fixed and repeat model fitting three times. For MountainCar, only one of three source-training seeds passed the recorded capability criterion. Appendix I gives all details.

How fusion improves task execution. Figure 4 connects the return gains to concrete changes in behavior. We replay one saved model pair per environment in the true simulator, using identical planning protocols and paired starts. The illustrated two-bank MountainCar policy reaches the hilltop at step 102 without violating the speed limit, while the single-agent policy times out at step 200. In Reacher, the two-bank policy reaches the joint-angle target at step 59, whereas the single-agent policy misses it within 100 steps. Pendulum illustrates a smaller timing gain: the two-bank policy enters the upright, low-speed region at step 23 rather than 24. The angular-speed comparison explains why similar-looking pendulum poses can yield different task outcomes at the same time.

The distributions beneath the examples report all initial states. For Gentle car, 409 of 512 starts improve and the remaining 103 are tied. Improvements also outnumber decreases in Pendulum (100 versus 19, with 393 ties) and Reacher (114 versus 53, with 89 ties). Mean discounted return is higher in all three replay cohorts, although most Pendulum starts are tied. The multi-fit evidence remains in Table 2 and Figure 13. Appendix I.5 gives the post-hoc selection rule and complete paired distributions; Appendix I.4 records replay checks and differences from archived returns.

4.3 Benefits of fusing different source tasks

We test whether different source tasks improve fusion at a fixed training budget. Starting from an *Original bank* for two Windy FourRooms goals, g_1 and g_2, *Fuse same tasks* adds an independently trained bank for those goals. *Fuse different tasks* adds a bank for two other goals, g_3 and g_4. Both fusion inputs contain four Q-functions and match sampled transitions and Q updates; the original bank uses half these budgets.

Fusing different tasks reduces mean transition MSE from 2.395×10^{-3} to 1.181×10^{-3} compared with fusing the same tasks, a 50.7% reduction. All six paired trials have lower model error. This improvement also benefits planning: mean normalized return rises from 0.9128 to 0.9950, with gains in five of six trials (Figure 6). The difference cannot be attributed to more Q-functions or a larger training budget in this comparison.

Repeating the comparison across training budgets. *Longer training* trains the original two goals for twice as long, matching the fusion conditions' samples and Q updates while retaining two Q-functions. Across 0.544M, 2.720M, and 10.880M sampled transitions, fusing different tasks has lower mean transition MSE and higher mean new-task return than both fusing the same tasks and longer training. At the largest budget, Longer training reaches 0.8971 return, compared with 0.9950 for different tasks. Appendix Figure 22 shows all three budgets and six trials.

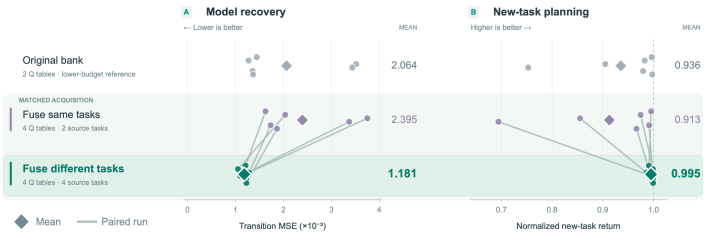


Figure 6: **Different source tasks make fusion more useful.** The fusion conditions match Q counts, sampled transitions, and scalar Q updates; the original bank uses half the samples and updates.

Appendix L reports all checkpoints and acquisition controls. Joint training on all four goals reaches 0.9984 return and lower transition MSE with matching Q updates but half the samples. Fusion's advantage in the main comparison is its ability to use different, separately learned tasks; it does not establish an efficiency advantage over joint acquisition.

4.4 Fusion with inaccurate Q-values

We next test whether the benefit of different tasks persists when source Q-values contain errors. To control task similarity, we use source tasks with a reward assigned to each grid cell, received on arrival, and no terminal states. These replace the goal-reaching source tasks above; the Windy dynamics and two downstream test tasks stay fixed. Each model uses four Q-functions. *Repeated tasks* share one reward map; *Similar tasks* make small changes to a common map; *Different tasks* use four separately generated reward maps.

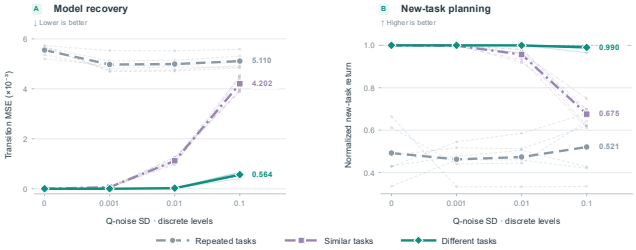


Figure 7: **Different tasks retain useful information under Q-value errors.** Each group uses four Q-functions and the same recovery algorithm. The axis lists entrywise noise standard deviations.

We first compute accurate Q tables and then add Gaussian noise to every entry. Both current Q-values and successor values queried during recovery use the perturbed tables. All three task groups use the same recovery algorithm, reward scale, and noise level. We repeat the comparison over six paired reward/noise realizations at each of four noise levels. Without noise, Similar and Different recover the dynamics to numerical precision and reach return 1. Repeated reaches 0.492: repeating one exact task adds no new constraints. Similar loses accuracy and return more quickly as noise grows (Figure 7). At noise standard deviation 0.1, Different retains return 0.9901 versus 0.6754 for Similar, with 86.6% lower transition MSE. Different tasks therefore support more stable recovery and planning under these perturbations, consistent with the role of task information and error amplification in Section 3.3. Appendix M gives complete results and the corresponding one-step value-prediction diagnostic.

5 Conclusion

Task-specific value functions contain reusable information about shared environment dynamics. Q-Map Fusion combines this information by jointly recovering a dynamics model from frozen Q-functions, then using the model to plan for new tasks. Recovery requires no next-state supervision. Our experiments show that complementary source tasks can improve model recovery and downstream planning, while the continuous-state results demonstrate the reuse of separately trained agents through a single recovered model. These findings support treating learned value functions as resources for model construction beyond their original tasks. Extending this approach to larger environments and more diverse pretrained agents is a natural next step.

AI USE STATEMENT

We used OpenAI Codex to assist with refining the methodological formulation, mathematical arguments, and experimental design; developing evaluation, analysis, and visualization code; reviewing related literature; interpreting results; and drafting, translating, and editing the manuscript. Early exploratory drafts also included AI-generated hypothetical numerical examples. These are distinct from the archived experiment records and simulator replays used in the reported empirical analyses. Code outputs and figures were checked against their underlying records, and replay returns were compared with the original evaluator. The authors take responsibility for verifying all AI-assisted material and for the accuracy, originality, and integrity of the final submission.

REFERENCES

Jacob Adamczyk. Inferring transition dynamics from value functions. *arXiv preprint arXiv:2501.09081*, 2025.

Firas Al-Hafez, Davide Tateo, Oleg Arenz, Guoping Zhao, and Jan Peters. Ls-iq: Implicit reward regularization for inverse reinforcement learning. *arXiv preprint arXiv:2303.00599*, 2023.

Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin Yang. Model-based reinforcement learning with value-targeted regression. In *International Conference on Machine Learning*, pp. 463–474. PMLR, 2020.

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.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pp. 449–458. Pmlr, 2017.

Diana Borsa, André Barreto, John Quan, Daniel Mankowitz, Rémi Munos, Hado van Hasselt, David Silver, and Tom Schaul. Universal Successor Features Approximators, 2018.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Madhavun Candadai and Eduardo J Izquierdo. Sources of predictive information in dynamical neural networks. *Scientific reports*, 10(1):16901, 2020.

Haoyang Cao, Samuel Cohen, and Lukasz Szpruch. Identifiability in inverse reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12362–12373, 2021.

Yevgen Chebotar, Quan Vuong, Karol Hausman, Fei Xia, Yao Lu, Alex Irpan, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Conference on Robot Learning*, pp. 3909–3928. PMLR, 2023.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and J K Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *Advances in neural information processing systems*, 36:73383–73394, 2023.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993.

Chaosheng Dong and Yijia Wang. Towards generalized inverse reinforcement learning. *arXiv preprint arXiv:2402.07246*, 2024.

Krishnamurthy Dvijotham and Emanuel Todorov. Inverse optimal control with linearly-solvable mdps. In *Proceedings of the 27th International conference on machine learning (ICML-10)*, pp. 335–342, 2010.

Amir-massoud Farahmand. Iterative value-aware model learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Amir-massoud Farahmand, Andre Barreto, and Daniel Nikovski. Value-aware loss function for model-based reinforcement learning. In *Artificial Intelligence and Statistics*, pp. 1486–1494. PMLR, 2017.

Mattie Fellows, Brandon Gary Kaplowitz, Christian Schroeder De Witt, and Shimon Whiteson. Bayesian exploration networks. In *Proceedings of the 41st International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2024.

Benjamin Freed, Thomas Wei, Roberto Calandra, Jeff Schneider, and Howie Choset. Unifying model-based and model-free reinforcement learning with equivalent policy sets. In *Reinforcement Learning Conference*, 2024.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto, David Meger, and Doina Precup. A deep reinforcement learning approach to marginalized importance sampling with the successor representation. In *International Conference on Machine Learning*, pp. 3518–3529. PMLR, 2021.

Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. *Advances in neural information processing systems*, 33: 5541–5552, 2020.

Christopher Grimm, André Barreto, Greg Farquhar, David Silver, and Satinder Singh. Proper value equivalence. *Advances in neural information processing systems*, 34:7773–7786, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.

Mance E Harmon and Stephanie S Harmon. Reinforcement learning: A tutorial. *WL/AAFC, WPAFB Ohio*, 45433:237–285, 1996.

Joey Hejna and Dorsa Sadigh. Inverse preference learning: Preference-based rl without a reward function. *Advances in Neural Information Processing Systems*, 36:18806–18827, 2023.

Michael Herman, Tobias Gindele, Jörg Wagner, Felix Schmitt, and Wolfram Burgard. Inverse reinforcement learning with simultaneous estimation of rewards and dynamics. In *Artificial intelligence and statistics*, pp. 102–110. PMLR, 2016.

David Krame Kadurha, Domini Jocema Leko Moutouo, and Yae Ulrich Gaba. Bellman operator convergence enhancements in reinforcement learning algorithms. *arXiv preprint arXiv:2505.14564*, 2025.

Martin Klissarov, Akhil Bagaria, Ziyan Luo, George Konidaris, Doina Precup, and Marlos C Machado. Discovering temporal structure: An overview of hierarchical reinforcement learning. *arXiv preprint arXiv:2506.14045*, 2025.

Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*, 2016.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, Proceedings of Machine Learning Research. PMLR, 2020.

Alistair Letcher, Mattie Fellows, Alexander D Goldie, Jonathan Richens, Jakob N Foerster, and Oliver Richardson. Inverting the bellman equation: From q-values to world models. *arXiv preprint arXiv:2606.21173*, 2026.

Alexander Li, Lerrel Pinto, and Pieter Abbeel. Generalized hindsight for reinforcement learning. *Advances in neural information processing systems*, 33:7754–7767, 2020.

Dhruv Malik, Malayandi Palaniappan, Jaime Fisac, Dylan Hadfield-Menell, Stuart Russell, and Anca Dragan. An efficient, generalized bellman update for cooperative inverse reinforcement learning. In *International Conference on Machine Learning*, pp. 3394–3402. PMLR, 2018.

Sean Meyn. The projected bellman equation in reinforcement learning. *IEEE Transactions on Automatic Control*, 69(12):8323–8337, 2024.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

Ida Momennejad, Evan M Russek, Jin H Cheong, Matthew M Botvinick, Nathaniel Douglass Daw, and Samuel J Gershman. The successor representation in human reinforcement learning. *Nature human behaviour*, 1(9):680–692, 2017.

Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Deepak Ramachandran, Eyal Amir, et al. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pp. 2586–2591, 2007.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, volume 2025, pp. 406–441, 2025.

Jonathan Richens, David Abel, Alexis Bellot, and Tom Everitt. General agents contain world models. *arXiv preprint arXiv:2506.01622*, 2025.

Paul Rolland, Luca Viano, Norman Schürhoff, Boris Nikolov, and Volkan Cevher. Identifiability and generalizability from multiple experts in inverse reinforcement learning. *Advances in Neural Information Processing Systems*, 35:550–564, 2022.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320. PMLR, 2015.

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*, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Richard Sutton and Andrew Barto. *Reinforcement learning: An introduction*. A Bradford Book, 2018.

Csaba Szepesvári. *Algorithms for reinforcement learning*. Springer nature, 2022.

Fuxiao Tan, Pengfei Yan, and Xinping Guan. Deep reinforcement learning: From q-learning to deep q-learning. In *International Conference on Neural Information Processing*, pp. 475–483. Springer, 2017.

Mark Towers, Ariel Kwiatkowski, John Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Kallinteris Andreas, Markus Krimmel, Arjun Kg, Rodrigo Perez-Vicente, et al. Gymnasium: A standard interface for reinforcement learning environments. *Advances in Neural Information Processing Systems*, 38, 2026.

Benjamin Van Niekerk, Steven James, Adam Earle, and Benjamin Rosman. Composing value functions in reinforcement learning. In *International conference on machine learning*, pp. 6401–6409. PMLR, 2019.

Claas Voelcker, Victor Liao, Animesh Garg, and Amir-massoud Farahmand. Value gradient weighted model-based reinforcement learning. *arXiv preprint arXiv:2204.01464*, 2022.

Tengyang Xie, Ching-An Cheng, Nan Jiang, Paul Mineiro, and Alekh Agarwal. Bellman-consistent pessimism for offline reinforcement learning. *Advances in neural information processing systems*, 34:6683–6694, 2021.

Lantao Yu, Jiaming Song, and Stefano Ermon. Multi-agent adversarial inverse reinforcement learning. In *International conference on machine learning*, pp. 7194–7201. PMLR, 2019.

Zhen Zhao. Variants of bellman equation on reinforcement learning problems. In *2nd International Conference on Artificial Intelligence, Automation, and High-Performance Computing (AIAHPC 2022)*, volume 12348, pp. 470–481. SPIE, 2022.

Mingyuan Zhong, Mikala Johnson, Yuval Tassa, Tom Erez, and Emanuel Todorov. Value function approximation and model predictive control. In *2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*, pp. 100–107. IEEE, 2013.

A Problem specification and Bellman construction

This appendix supplies the definitions, proofs, and implementation details supporting Section 3. The Bellman inverse construction and its constrained solvers follow Letcher et al. Letcher et al. (2026). The results below make explicit the assumptions needed for the particular estimator in Equation equation 2. Sections A–B establish the measurement construction and its exact information content. Section C proves the main text's residual and prediction bounds, and Section D connects uniform model accuracy to planning performance. Sections E–F specify the solvers and the recorded experimental interface.

A.1 Tasks, queries, and termination

Let $\mathcal{S}$ and $\mathcal{A}$ be finite, nonempty sets, $P^\star(\cdot \mid s, a)$ a transition distribution, and $0 \le \gamma < 1$. Task i has a known bounded reward $r_i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ and termination indicator $d_i : \mathcal{S} \to \{0, 1\}$. Define $D_i = 1 - d_i$ and the operator

$$(\mathcal{F}_i Q)(s, a) = \sum_{s'} P^\star(s' \mid s, a) \left[r_i(s, a, s') + \gamma D_i(s') \max_{a'} Q(s', a') \right]. \tag{A.1}$$

The reward on arrival is received before termination suppresses continuation. Since the maximum is nonexpansive in the supremum norm, $\|\mathcal{F}_i Q - \mathcal{F}_i W\|_\infty \le \gamma \|Q - W\|_\infty$. Thus $\mathcal{F}_i$ has a unique fixed point $Q_i^\star$. We write $V_i^\star(s) = \max_a Q_i^\star(s, a)$.

Equation equation A.1 defines values at every query state–action pair, including queries whose current state satisfies $d_i(s) = 1$: termination is checked at the subsequent arrival. It does not assign $Q_i^\star(s, a) = 0$ solely from $d_i(s) = 1$. The supplied finite maps $\widehat{Q}_i$ are used as stored; the analysis measures their discrepancy from this fixed point. An episodic implementation may expose different stored entries at states where an episode has already ended. That input convention must be checked when relating a particular implementation to these definitions; no row deletion or zero-value replacement is implicit in the estimator.

There are $K \ge 1$ task records. Task indices identify records, including their reward and termination definitions. All queries use aligned state and action meanings. Recovery requires $\widehat{Q}_i(s, a)$ at the source and the action values needed to compute $\widehat{V}_i$ at candidate nonterminal successors. At a terminal successor, its continuation contribution is zero.

A.2 From Bellman equations to task rows

Fix $x = (s, a)$ and a nonempty ordered candidate set $\Omega_x = \{s'_1, \ldots, s'_{m_x}\}$. Assume for now that $\operatorname{supp} P^\star(\cdot \mid x) \subseteq \Omega_x$, and define $p_x^\star[j] = P^\star(s'_j \mid x)$. For exact values, set

$$\Phi_x[i, j] = r_i(x, s'_j) + \gamma D_i(s'_j) V_i^\star(s'_j), \qquad q_x[i] = Q_i^\star(s, a). \tag{A.2}$$

Expanding Equation equation A.1 over Ω_x gives $\Phi_x p_x^\star = q_x$. Substituting $\widehat{Q}_i$ and $\widehat{V}_i$ gives the measurement matrix and vector in Section 3.2, obtained from Equation equation 1. The two model classes used here are Δ_{m_x} and its vertex set. The assumption $p_x^\star \in \mathcal{D}_x$ includes both support correctness and correctness of the specified model class. Support misspecification is treated in Section C.5.

The recorded finite-state experiments specialize these definitions to arrival-state rewards $r_i(s')$. For either reward convention, each measurement row is linear in the unknown transition probabilities. For a map estimating $Q_i^{\pi_i}$ under a known policy, the corresponding continuation is instead $\widehat{V}_i^{\pi_i}(s') = \sum_{a'} \pi_i(a' \mid s') \widehat{Q}_i(s', a')$. The main method uses the optimality backup; this policy-evaluation version is a separate input convention.

B Exact information and illustrative examples

B.1 Intersecting compatible models

We verify Equation equation **??** using exact source values and a common query, support, and model class. For task-record indices I, define $\mathcal{C}_x(I) = \{p \in \mathcal{D}_x : \Phi_x[I, :]p = q_x[I]\}$. For any two index

collections,

$$\mathcal{C}_x(I_A \cup I_S) = \mathcal{C}_x(I_A) \cap \mathcal{C}_x(I_S). \tag{B.1}$$

Membership on either side means satisfying every row in both collections, which proves the identity. If $p_x^\star \in \mathcal{D}_x$, the exact Bellman equations place it in every such set. Consequently, exact compatible sets can only shrink while retaining the admissible true row. The shrinkage is strict precisely when at least one candidate compatible with I_A violates an additional row. Repeated exact constraints do not shrink the set further.

B.2 Global, local, and deterministic identification

Proposition 1 (Identification on the simplex)**.** *The map $p \mapsto \Phi_x p$ is injective on Δ_{m_x} if and only if*

$$\ker(\Phi_x) \cap \mathbf{1}^\perp = \{0\}, \quad \textit{equivalently} \quad \operatorname{rank} \begin{bmatrix} \Phi_x \\ \mathbf{1}^\top \end{bmatrix} = m_x. \tag{B.2}$$

Proof. The difference of two simplex points has zero coordinate sum. Under the kernel condition, identical measurements therefore imply identical points. Conversely, let nonzero h belong to the displayed intersection. For $u = \mathbf{1}/m_x$ and sufficiently small $t > 0$, $u + th$ and $u - th$ are distinct simplex points with identical measurements. The rank statement expresses the same kernel condition. □

For a particular feasible $p_x^\star$, uniqueness only requires that $h = 0$ be the sole vector satisfying $\Phi_x h = 0$, $\mathbf{1}^\top h = 0$, and $p_x^\star + h \geq 0$. Boundary constraints can therefore identify an individual point even when Equation equation B.2 fails. For example, $\Phi = (0, 1, 2)$ uniquely identifies $p^\star = (1, 0, 0)$ from $q = 0$, since $p_2 + 2p_3 = 0$ forces $p_2 = p_3 = 0$. The map is not injective on the whole simplex: $(0, 1, 0)$ and $(1/2, 0, 1/2)$ have identical measurements.

For the deterministic class $\{e_1, \ldots, e_{m_x}\}$, a particular e_j is uniquely identified exactly when $\Phi_x[:, j]$ differs from every other column. The entire deterministic class is identifiable exactly when the columns are pairwise distinct. These conditions concern a finite set of models and do not require simplex injectivity.

B.3 The new-task decision example

Consider states $\{s, z_1, z_2, z_3\}$ and actions $\{a, b\}$. From s, the transition rows, listed over (z_1, z_2, z_3), are

$$P^\star(\cdot \mid s, a) = (0.2, 0.3, 0.5), \qquad P^\star(\cdot \mid s, b) = (0.6, 0.4, 0). \tag{B.3}$$

Each z_j self-loops under both actions. The inverse problem in this example is supplied with $\Omega_{(s,a)} = \Omega_{(s,b)} = \{z_1, z_2, z_3\}$: all admissible models leave s in one step. All tasks terminate upon arrival at any z_j. Task A rewards only z_1, task B only z_2, and new task C only z_3, with unit reward in each case and zero elsewhere. The exact source-state values are:

Task	$Q(s, a)$	$Q(s, b)$	Optimal action at s
A	0.2	0.6	b
B	0.3	0.4	b
C	0.5	0	a

The two source tasks determine the first two successor probabilities for each action; normalization determines the third. Thus their joint constraints determine the new-task decision.

Each source task alone is insufficient. The alternative rows $(0.2, 0.8, 0)$ and $(0.6, 0, 0.4)$ preserve all task-A values but make b optimal for C. Rows $(0.7, 0.3, 0)$ and $(0, 0.4, 0.6)$ similarly preserve task-B values and reverse C's decision. All other transitions are unchanged, so the source maps also agree at the remaining states. This is an explicitly specified analytical example.

B.4 FIXED ROWS AND BANK LABELS

For fixed numerical rows, feasible sets, and per-row weights, the residual objective is invariant to a repartition of those same records into banks. Its minimizer set is therefore invariant as well. Identical numerical outputs additionally require consistent solver and tie-breaking choices. Repeating the entire collection uniformly multiplies the objective by a positive constant. Repeating only some noisy rows changes their relative weight and may change the fitted model. Distinct estimates of the same task can also change the fit; the exact-set observation above does not rule out benefits from repeated estimation.

C LEARNING ERROR AND USEFUL PREDICTIONS

C.1 HOW VALUE ERRORS ENTER BOTH SIDES

Under support correctness, write $\Delta Q_i = \widehat{Q}_i - Q_i^\star$ and $\Delta V_i = \widehat{V}_i - V_i^\star$. Subtracting the exact Bellman equation from the learned row gives

$$\varepsilon_x[i] = \Delta Q_i(s,a) - \gamma \sum_{s'} P^\star(s' \mid s,a) D_i(s') \Delta V_i(s'). \tag{C.1}$$

If $\|\Delta Q_i\|_\infty \le \alpha_i$, the inequality $|\max_a u_a - \max_a v_a| \le \max_a |u_a - v_a|$ implies $\|\Delta V_i\|_\infty \le \alpha_i$, and hence

$$|\varepsilon_x[i]| \le (1+\gamma)\alpha_i, \qquad \frac{\|\varepsilon_x\|_1}{K} \le \frac{1+\gamma}{K} \sum_{i=1}^{K} \alpha_i. \tag{C.2}$$

No independence or zero-mean assumption is used. Separate random streams do not establish such assumptions for the resulting Bellman errors. Both α_i and ε_x are analysis quantities, not quantities assumed available to recovery.

C.2 RESIDUAL CONTROL WITH OPTIMIZATION ERROR

Proposition 2 (Residual control)**.** *Let $p_x^\star \in \mathcal{D}_x \subseteq \Delta_{m_x}$ and $L_x(p) = \|\widehat{\Phi}_x p - \widehat{q}_x\|_1$. If $\widehat{p}_x \in \mathcal{D}_x$ satisfies $L_x(\widehat{p}_x) \le \inf_{p \in \mathcal{D}_x} L_x(p) + \tau_x$, where $\tau_x \ge 0$, then*

$$\|\widehat{\Phi}_x(\widehat{p}_x - p_x^\star)\|_1 \le 2\|\varepsilon_x\|_1 + \tau_x. \tag{C.3}$$

Proof. Feasibility gives $L_x(\widehat{p}_x) \le L_x(p_x^\star) + \tau_x = \|\varepsilon_x\|_1 + \tau_x$. The triangle inequality then yields

$$\|\widehat{\Phi}_x(\widehat{p}_x - p_x^\star)\|_1 \le L_x(\widehat{p}_x) + \|\widehat{q}_x - \widehat{\Phi}_x p_x^\star\|_1 \le 2\|\varepsilon_x\|_1 + \tau_x.$$

□

The feasible sets used in the method are compact, so their continuous residual objectives attain a minimum. This measured-direction bound is the intermediate step in the main prediction result, Equation equation 5. Setting $\tau_x = 0$ gives exact minimization of the stated objective. A numerical feasibility or solver tolerance is not automatically an objective-gap certificate; feasibility and the actual objective gap are separate conditions in this proposition.

C.3 PROOF OF THE MAIN PREDICTION STATEMENT

Proposition 3 (Prediction control)**.** *Under Proposition 2, let a specified vector $\psi \in \mathbb{R}^{m_x}$ have the representation $\psi^\top = c^\top \widehat{\Phi}_x + \beta \mathbf{1}^\top + \eta_x^\top$, where $c \in \mathbb{R}^K$, $\beta \in \mathbb{R}$, and $\eta_x \in \mathbb{R}^{m_x}$. Then Equation equation 5 holds.*

Proof. Set $h = \widehat{p}_x - p_x^\star$. Feasibility gives $\mathbf{1}^\top h = 0$ and $\|h\|_1 \le 2$. The constant component therefore cancels. Hölder's inequality and Proposition 2 give

$$\begin{aligned} |\psi^\top h| &= |c^\top \widehat{\Phi}_x h + \eta_x^\top h| \\ &\le \|c\|_\infty \|\widehat{\Phi}_x h\|_1 + \|\eta_x\|_\infty \|h\|_1 \\ &\le \|c\|_\infty (2\|\varepsilon_x\|_1 + \tau_x) + 2\|\eta_x\|_\infty. \end{aligned} \tag{C.4}$$

This is the main-text prediction bound. □

Setting $\eta_x = 0$ gives the exact-representation special case of Equation equation 5, while retaining its objective-gap term τ_x. The vector η_x is the residual of the chosen representation, not a uniquely defined or necessarily minimal approximation error. When multiple representations exist, each yields a valid bound; large coefficients can amplify the effect of value errors.

For a fixed downstream value V and query $x = (s, a)$, the test vector is

$$\psi_{V,x}[j] = r_{\text{new}}(s, a, s'_j) + \gamma(1 - d_{\text{new}}(s'_j))V(s'_j).$$

For a representation obtained at another value $\widetilde{V}$,

$$\|\psi_{V,x} - \psi_{\widetilde{V},x}\|_\infty \le \gamma\|V - \widetilde{V}\|_\infty.$$

The expectation-difference bound then acquires an additional $2\gamma\|V - \widetilde{V}\|_\infty$. Neither the representation condition nor the unknown effective errors constitute an automatic certificate for the return of a newly planned policy. Section D supplies the stronger conditions used for a complete planning comparison.

C.4 From measured directions to complete recovery

For $m_x \ge 2$, define

$$\bar{e}_x = \frac{\|\varepsilon_x\|_1}{K}, \qquad \bar{\kappa}_x = \inf_{\substack{h \ne 0 \\ \mathbf{1}^\top h = 0}} \frac{\|\widehat{\Phi}_x h\|_1}{K\|h\|_1}. \tag{C.5}$$

The infimum can be restricted to the compact set $\{h : \mathbf{1}^\top h = 0, \|h\|_1 = 1\}$. Consequently $\bar{\kappa}_x > 0$ exactly when the learned rows, together with probability normalization, are injective on the simplex.

Corollary 4 (Complete recovery under stable constraints)**.** *Under Proposition 2, if $\bar{\kappa}_x > 0$, then*

$$\|\widehat{p}_x - p^\star_x\|_1 \le \min\left\{2,\ \frac{2\bar{e}_x + \tau_x/K}{\bar{\kappa}_x}\right\}. \tag{C.6}$$

Proof. The difference has zero coordinate sum. Apply Equation equation C.5 to that difference, use Equation equation C.3, and divide by $K\bar{\kappa}_x$. The L1 distance between two probability vectors is at most two. □

For $m_x = 1$, the feasible distribution is already determined. Uniformly replicating every row leaves $\bar{e}_x$ and $\bar{\kappa}_x$ unchanged; objective-gap tolerances must also be scaled consistently when comparing optimization accuracy. Additional rows change both the constraints and their errors, so a change in a conditioning statistic alone is not an error improvement guarantee.

With $K = 20$ and $m_x = 68$, the augmented matrix has at most 21 rows, and this uniform simplex condition cannot hold. It does not follow that every individual boundary point is ambiguous, or that every new-task prediction is uncontrolled. The deterministic class has a separate stability criterion. If the true successor is $j^\star$ and

$$g_x = \min_{j \ne j^\star} \|\widehat{\Phi}_x[:, j] - \widehat{\Phi}_x[:, j^\star]\|_1 > 2\|\varepsilon_x\|_1,$$

then exact column matching uniquely returns $j^\star$: its residual is $\|\varepsilon_x\|_1$, while every other residual is at least $g_x - \|\varepsilon_x\|_1$. This is a sufficient column-separation condition.

C.5 Misspecified supports and model classes

Let p_x^{full} be the true distribution on $\mathcal{S}$, and let $\iota_x p$ extend a distribution on Ω_x by zeros. Choose any feasible reference $p_0 \in \mathcal{D}_x \subseteq \Delta_{m_x}$. Assume $\widehat{p}_x \in \mathcal{D}_x$ satisfies the same objective-gap condition with $\tau_x \ge 0$; the true distribution need not belong to this model class. Write $\widehat{\Phi}_x^{\text{full}}$ for the learned matrix constructed over all successors, and set

$$\begin{aligned} \varepsilon_x^{\text{full}} &= \widehat{q}_x - \widehat{\Phi}_x^{\text{full}} p_x^{\text{full}}, \\ b_0 = \widehat{\Phi}_x^{\text{full}} p_x^{\text{full}} - \widehat{\Phi}_x p_0, &\qquad a_0 = \tfrac{1}{2}\|\iota_x p_0 - p_x^{\text{full}}\|_1. \end{aligned} \tag{C.7}$$

The same optimality argument, now using p_0, gives

$$\|\widehat{\Phi}_x(\widehat{p}_x - p_0)\|_1 \le 2\|\varepsilon_x^{\text{full}} + b_0\|_1 + \tau_x. \tag{C.8}$$

For any full-state test function f, write $\operatorname{span}(f) = \max f - \min f$. The difference of two distributions has zero mass, so $|f^\top(\iota_x p_0 - p_x^{\text{full}})| \le a_0 \operatorname{span}(f)$. In particular, $|b_0[i]| \le a_0 \operatorname{span}(\widehat{\Phi}_x^{\text{full}}[i,:])$. If $\bar{\kappa}_x > 0$, the model error is bounded by

$$\|\iota_x \widehat{p}_x - p_x^{\text{full}}\|_1 \le \min\left\{2,\ 2a_0 + \frac{2\|\varepsilon_x^{\text{full}} + b_0\|_1 + \tau_x}{K\bar{\kappa}_x}\right\}. \tag{C.9}$$

This follows by inserting $\iota_x p_0$, using the triangle inequality, and applying Equation equation C.8 to the zero-sum vector $\widehat{p}_x - p_0$.

For simplex recovery, let $\rho_x = P^\star(\mathcal{S} \setminus \Omega_x \mid x)$. If $\rho_x < 1$, the conditional distribution on Ω_x is a feasible reference with $a_0 = \rho_x$. If $\rho_x = 1$, any simplex reference has $a_0 = 1$. Every support-restricted model has full-state L1 error at least $2\rho_x$: the omitted mass contributes ρ_x, and the excess total mass inside the support contributes at least another ρ_x. This minimum is attained by the conditional reference when $\rho_x < 1$, and by every simplex reference when $\rho_x = 1$. For more restrictive model classes, use their actual feasible reference and a_0.

Prediction bounds also acquire a direct model-class bias. Let $\psi \in \mathbb{R}^{|\mathcal{S}|}$ be a full-state test vector. If its restriction satisfies $\psi_{\Omega_x}^\top = c^\top \widehat{\Phi}_x + \beta \mathbf{1}^\top + \eta_x^\top$, with $\eta_x \in \mathbb{R}^{m_x}$, then

$$\begin{aligned} |\psi^\top(\iota_x \widehat{p}_x - p_x^{\text{full}})| \le \|c\|_\infty(2\|\varepsilon_x^{\text{full}} + b_0\|_1 + \tau_x) \\ + 2\|\eta_x\|_\infty + a_0 \operatorname{span}(\psi). \end{aligned} \tag{C.10}$$

Insert the reference distribution, apply Equation equation C.8, and bound the remaining expectation difference by total variation.

C.6 A BIASED SUPPLEMENTAL BANK CAN WORSEN RECOVERY

Consider states $\{s, z_1, z_2\}$ and one action a. At query (s, a), supply $\Omega_{(s,a)} = \{z_1, z_2\}$. From s, these two outcomes have true probabilities $(1, 0)$; both z_1 and z_2 self-loop and terminate every task on arrival. Set all rewards at s to zero. At each outcome, the exact query value under Equation equation A.1 equals its arrival reward; take the supplied value there to be exact. The tasks have the following arrival rewards and prescribed source-value estimates:

Task	$(r(z_1), r(z_2))$	$Q^\star(s,a)$	$\widehat{Q}(s,a)$
Anchor	$(1, 0)$	1	1
Supplement 1	$(2, 1)$	2	1
Supplement 2	$(3, 2)$	3	2

Both outcomes terminate every task, so the Bellman rows equal the reward vectors regardless of the successor Q entries. For $p(t) = (t, 1-t)$, $0 \le t \le 1$, the anchor loss is $1 - t$. The joint loss is

$$|t - 1| + |(1 + t) - 1| + |(2 + t) - 2| = 1 + t.$$

The anchor uniquely selects the true $t = 1$, while joint fitting uniquely selects $t = 0$, with L1 error two. The same reversal holds on the deterministic vertex class. This is a constructed noisy-value example, not a measured run. Its additional rewards supply the same simplex direction with biased observations; it illustrates why constraint count alone cannot imply an improvement.

D FROM MODEL ERROR TO PLANNING PERFORMANCE

This section records standard perturbation arguments that connect transition accuracy to the performance of a policy planned in the recovered model. The bounds require uniform control of the transition error. They complement the main text's guarantees for specified one-step predictions, which alone do not supply such control for every value function encountered in planning.

D.1 Common task and uniform transition error

Fix a new task on the finite state and action spaces of the main text. The true kernel $P^\star$ and recovered kernel $\widehat{P}$ are both stochastic and use the same known reward $r(s,a,s')$, termination indicator $d(s')$, and discount $0 \le \gamma < 1$. Write $D(s') = 1 - d(s')$ and assume $|r(s,a,s')| \le R$. Arrival-state rewards $r(s')$ are a special case. Rewards are received on arrival, before termination suppresses subsequent rewards. Thus, for a stationary policy π and kernel M, the policy and optimality operators are

$$\begin{aligned}(T_M^\pi v)(s) &= \sum_a \pi(a \mid s) \sum_{s'} M(s' \mid s,a)[r(s,a,s') + \gamma D(s') v(s')],\\ (T_M v)(s) &= \max_a \sum_{s'} M(s' \mid s,a)[r(s,a,s') + \gamma D(s') v(s')].\end{aligned} \tag{D.1}$$

Both operators are γ-contractions in the sup norm. Their fixed points are denoted by V_M^π and $V_M^\star$, respectively, and have norm at most $R/(1-\gamma)$. Finite discounted MDPs admit a stationary optimal policy under these conventions.

Define the uniform row error

$$\delta_P = \sup_{s,a} \|P^\star(\cdot \mid s,a) - \widehat{P}(\cdot \mid s,a)\|_1. \tag{D.2}$$

The recovery analysis can upper-bound this quantity when its feasibility and stability conditions hold uniformly over all state–action pairs. An average transition MSE is a different quantity and cannot be substituted for δ_P without an additional conversion bound. All comparisons below are deterministic statements for the realized recovered kernel. That kernel and the output policy may depend on the learned Q banks; no independence between them is required. The true transition error remains an analysis quantity, rather than an additional input to recovery or planning. In particular, these statements do not turn a small fitted Bellman residual into an observable performance certificate without the further recovery conditions.

D.2 Infinite-horizon value and policy comparisons

Proposition 5 (Uniform model comparison)**.** *Under the preceding assumptions, every stationary policy π satisfies*

$$\|V_{P^\star}^\pi - V_{\widehat{P}}^\pi\|_\infty \le B, \qquad \|V_{P^\star}^\star - V_{\widehat{P}}^\star\|_\infty \le B, \qquad B = \frac{R\delta_P}{(1-\gamma)^2}. \tag{D.3}$$

If a stationary output policy $\widehat{\pi}$ has model-side planning error $\eta_{\mathrm{plan}} = \|V_{\widehat{P}}^\star - V_{\widehat{P}}^{\widehat{\pi}}\|_\infty$, then

$$\|V_{P^\star}^\star - V_{P^\star}^{\widehat{\pi}}\|_\infty \le \frac{2R\delta_P}{(1-\gamma)^2} + \eta_{\mathrm{plan}}. \tag{D.4}$$

Proof. For any v with $\|v\|_\infty \le R/(1-\gamma)$, the successor test function $f_{v,s,a}(s') = r(s,a,s') + \gamma D(s')v(s')$ has $\|f_{v,s,a}\|_\infty \le R/(1-\gamma)$ for every (s,a). Consequently,

$$\left| \sum_{s'} [P^\star(s' \mid s,a) - \widehat{P}(s' \mid s,a)] f_{v,s,a}(s') \right| \le \frac{R\delta_P}{1-\gamma}. \tag{D.5}$$

Averaging over a fixed policy preserves this bound; so does taking the difference of two action maxima, since $|\max_a u_a - \max_a w_a| \le \max_a |u_a - w_a|$. Insert $v = V_{\widehat{P}}^\pi$ into the fixed-policy comparison. The fixed-point identities and contraction give

$$\|V_{P^\star}^\pi - V_{\widehat{P}}^\pi\|_\infty \le \gamma \|V_{P^\star}^\pi - V_{\widehat{P}}^\pi\|_\infty + \frac{R\delta_P}{1-\gamma}.$$

Rearranging proves the first bound. The same argument with the optimality operators proves the second. Finally, decompose the true-model policy gap through $V_{\widehat{P}}^\star$ and $V_{\widehat{P}}^{\widehat{\pi}}$. The two model-comparison terms are each bounded by B, and the remaining model-side gap by η_{plan}. True optimality makes the policy gap pointwise nonnegative, establishing the stated sup-norm regret bound. □

There is no additional leading factor of γ in Equation equation D.5: changing the successor distribution changes the expected arrival reward as well as the continuation value. Bounds for known rewards depending only on the current state and action need not have the same constant.

An implementable stopping criterion can control η_{plan}. Let v be a finite value estimate, let $\widehat{\pi}$ exactly maximize the action values defining $T_{\widehat{P}}v$, and write $\rho = \|T_{\widehat{P}}v - v\|_\infty$. Then

$$\eta_{\text{plan}} \leq \frac{2\gamma\rho}{1-\gamma}. \tag{D.6}$$

Indeed, contraction bounds both $\|V^\star_{\widehat{P}} - v\|_\infty$ and $\|V^{\widehat{\pi}}_{\widehat{P}} - v\|_\infty$ by $\rho/(1-\gamma)$, because $T^{\widehat{\pi}}_{\widehat{P}}v = T_{\widehat{P}}v$. Applying the two operators to their fixed points and subtracting this common backup gives Equation equation D.6. The criterion concerns the actual Bellman residual and exact action maximization; a nominal iteration count alone does not establish it.

D.3 Finite-horizon evaluation

Let $V^\pi_{M,h}$ denote the return from h remaining transitions, with $V^\pi_{M,0} = 0$. Finite-horizon policies may depend on the remaining time. Their recursion uses the corresponding decision rule at each stage in Equation equation D.1. The finite-horizon optimal value $V^\star_{M,h}$ uses the maximizing recursion with the zero-horizon initialization $V^\star_{M,0} = 0$ and the same arrival-based termination rule. Set

$$S_h = \sum_{t=0}^{h-1} \gamma^t, \qquad C_h = \sum_{t=0}^{h-1} (t+1)\gamma^t, \qquad S_0 = C_0 = 0. \tag{D.7}$$

Proposition 6 (Finite-horizon comparison)**.** *For any horizon $H \geq 1$, both the value of any fixed, possibly stage-dependent Markov policy and the optimal value differ between the two models by at most $R\delta_P C_H$ in sup norm. For any such output policy $\widehat{\pi}$, including a stationary policy used at every stage, define $\zeta_H = \|V^\star_{\widehat{P},H} - V^{\widehat{\pi}}_{\widehat{P},H}\|_\infty$. Then*

$$\|V^\star_{P^\star,H} - V^{\widehat{\pi}}_{P^\star,H}\|_\infty \leq 2R\delta_P C_H + \zeta_H. \tag{D.8}$$

Proof. All h-step values have norm at most RS_h. A backup with $(h-1)$-step continuation therefore has successor test-function norm at most $R + \gamma RS_{h-1} = RS_h$. Let E_h upper-bound the sup-norm difference between the two models' h-step values, uniformly over fixed stage-dependent policies. The same bound can also cover their optimal values, by the inequality for action maxima used above. Splitting a backup difference into its continuation-value and transition-kernel terms yields

$$E_0 = 0, \qquad E_h \leq \gamma E_{h-1} + R\delta_P S_h. \tag{D.9}$$

Since $C_h = \gamma C_{h-1} + S_h$, induction gives $E_h \leq R\delta_P C_h$. For a fixed policy this induction applies to its common sequence of decision rules in both models; for optimal values it applies to the maximizing recursions. Decomposing the policy gap through the recovered model's optimal and output-policy values then gives two such comparison terms plus ζ_H, proving Equation equation D.8. □

This finite-horizon statement preserves the distinction between planning and evaluation. A stationary policy obtained by infinite-horizon value iteration need not be optimal for the H-step objective, so ζ_H cannot be discarded merely because infinite-horizon value iteration has converged. Keeping ζ_H does not prescribe a different planner. It accounts for the objective actually used to assess the output policy. Within the recovered model, this quantity is defined by comparing finite-horizon optimal dynamic programming with evaluation of that output policy under the same horizon and termination rule; it is distinct from the infinite-horizon planning error η_{plan}.

D.4 Initial-state averages and normalized returns

The sup-norm regret bound also bounds the return difference averaged over any common initial-state distribution μ. Write $J^\pi_{M,H} = \mathbb{E}_{s\sim\mu}[V^\pi_{M,H}(s)]$ and define $J^\star_{M,H}$ analogously. If the normalization uses the true H-step optimal reference and $J^\star_{P^\star,H} > 0$, then

$$1 - \frac{J^{\widehat{\pi}}_{P^\star,H}}{J^\star_{P^\star,H}} \leq \frac{2R\delta_P C_H + \zeta_H}{J^\star_{P^\star,H}}. \tag{D.10}$$

Thus a normalized statement additionally depends on its denominator. If evaluation instead uses a different reference policy, its value, horizon, and initial-state distribution must be specified; a ratio of one does not by itself establish finite-horizon optimality. No uniform numerical guarantee for normalized return follows here without control of a positive reference denominator.

E SOLVERS AND IMPLEMENTATION

This section specifies the general finite-state recovery procedure and the parametric implementation interface. Task values, rewards, termination indicators, state–action indexing, candidate supports, and model class are fixed inputs. Each task retains its own reward and termination definition when its row is added to the joint system. Recovery queries Q entries at source states and candidate nonterminal successors; it does not update the Q tables. Recorded experimental settings are collected in Section F.

E.1 COMPLETE FINITE-STATE PROCEDURE

Algorithm 2 expands the main-text summary. It uses the same row objective, candidate model classes, accepted value iterate, and greedy policy; caching, numerical checks, and failure handling are made explicit here. The summary's V_t is the value accepted after the final update. It therefore corresponds to V after the assignment $V \leftarrow U$ in the complete procedure.

Algorithm 2 Finite-state Q-Map Fusion: complete recovery and planning procedure

Require: Records $\mathcal{B}$ under the stated Bellman convention; finite $\mathcal{S}, \mathcal{A}$; $\gamma \in [0, 1)$
Require: Ordered supports $\{\Omega_x\}$; row classes $\{\mathcal{D}_x\}$; LP solver configuration; numerical feasibility tolerances; discrete tie rules
Require: New task $(r_{\mathrm{new}}, d_{\mathrm{new}})$; planning tolerance $\epsilon_V > 0$
Ensure: Recovered kernel $\widehat{P}$ and policy $\widehat{\pi}$, or solver failure

Cache $C_i(z) \leftarrow 0$ if $d_i(z) = 1$, else $\max_{a'} \widehat{Q}_i(z, a')$, for every i, z.
Initialize $\widehat{P}(s' \mid s, a) \leftarrow 0$ for all s, a, s'.
for each $x = (s, a) \in \mathcal{S} \times \mathcal{A}$ **do**
    Form $\widehat{q}_x[i] \leftarrow \widehat{Q}_i(s, a)$ and $\widehat{\Phi}_x[i, j] \leftarrow r_i(s, a, s'_j) + \gamma C_i(s'_j)$ for every i, j.
    if $\mathcal{D}_x = \{e_1, \ldots, e_{m_x}\}$ **then**
        $j_x \leftarrow \arg\min_{1 \le j \le m_x} \sum_{i=1}^{K} |\widehat{\Phi}_x[i, j] - \widehat{q}_x[i]|$; $\widehat{p}_x \leftarrow e_{j_x}$, using the fixed tie rule.
    else ▷ $\mathcal{D}_x = \Delta_{m_x}$
        Solve $\min_{p,u} \mathbf{1}^\top u$ subject to $-u \le \widehat{\Phi}_x p - \widehat{q}_x \le u$, $p \ge 0$, $\mathbf{1}^\top p = 1$, $u \ge 0$.
        return failure if the solver reports failure.
        Take its probability vector as $\widehat{p}_x$; **return** failure if it violates the declared numerical probability checks.
    end if
    $\widehat{P}(s'_j \mid s, a) \leftarrow \widehat{p}_x[j]$ for $j = 1, \ldots, m_x$.
end for
$V \leftarrow 0$; $U \leftarrow \mathcal{T}^{\widehat{P}}_{\mathrm{new}} V$.
while $\|U - V\|_\infty > \epsilon_V$ **do**
    $V \leftarrow U$; $U \leftarrow \mathcal{T}^{\widehat{P}}_{\mathrm{new}} V$.
end while
$V \leftarrow U$. ▷ Accept the last value update.
$\widehat{\pi}(s) \leftarrow \arg\max_{a \in \mathcal{A}} b^{\widehat{P}}_V(s, a)$ for every s, using the fixed tie rule.
return $(\widehat{P}, \widehat{\pi})$.

E.2 Stochastic recovery and deterministic matching

For each $x = (s, a)$, the stochastic instance of Equation equation 2 has the following linear-program form, with $p \in \mathbb{R}^{m_x}$ and $u \in \mathbb{R}^K$:

$$\begin{aligned} \min_{p,u} \quad & \mathbf{1}^\top u \\ \text{subject to} \quad & \widehat{\Phi}_x p - \widehat{q}_x \le u, \\ & \widehat{q}_x - \widehat{\Phi}_x p \le u, \\ & \mathbf{1}^\top p = 1, \qquad p \ge 0, \quad u \ge 0. \end{aligned} \tag{E.1}$$

For any fixed feasible p, the smallest feasible u is $|\widehat{\Phi}_x p - \widehat{q}_x|$ coordinatewise, proving equivalence to the stochastic residual objective. For dynamics specified as deterministic, recovery instead enumerates all candidate columns:

$$\widehat{j} \in \arg\min_{1 \le j \le m_x} \sum_{i=1}^{K} \left| \widehat{\Phi}_x[i, j] - \widehat{q}_x[i] \right|, \qquad \widehat{p}_x = e_{\widehat{j}}. \tag{E.2}$$

The stochastic formulation is a linear program, and exhaustive column matching gives a global deterministic minimizer. For the stochastic LP, a dual vector $y \in [-1, 1]^K$ gives

$$\ell_x = \min_j (\widehat{\Phi}_x^\top y)_j - \widehat{q}_x^\top y.$$

Indeed, for every $p \in \Delta_{m_x}$,

$$L_x(p) \ge y^\top (\widehat{\Phi}_x p - \widehat{q}_x) \ge \ell_x.$$

Thus primal and dual feasibility together with $L_x(\widehat{p}_x) - \ell_x \le \tau_x$ imply the objective-gap condition used in the main text. This is an optional certification route, not a mandatory step of the recovery algorithm. Ordinary floating-point solver status and feasibility tolerances alone are not exact certificates. The prediction theorem assumes genuinely feasible probability rows; numerical deviations from nonnegativity or unit mass are separate errors, not part of τ_x.

Numerical outputs and exact minimizers need not be unique. A reproducible reference implementation should fix candidate ordering and record tie handling, solver status, objective value, and probability feasibility for every row. These checks are recommendations, not additional settings attributed to the recorded experiments. Probabilities outside Ω_x are assigned zero.

E.3 Candidate supports and computational cost

Candidate supports and row model classes are supplied before fitting. Their roles are distinct: support specifies permitted successors, while model class determines which probability rows on that support are allowed. Correctness of both is needed for the feasible-truth assumption in the recovery analysis. Section F records the supports used in the FourRooms studies.

Caching $\widehat{V}_i(s) = \max_a \widehat{Q}_i(s, a)$ costs $O(K|\mathcal{S}||\mathcal{A}|)$ operations and $O(K|\mathcal{S}|)$ storage. Constructing a local measurement matrix then costs $O(Km_x)$ operations and storage; deterministic matching has the same operation count. Each stochastic LP has $m_x + K$ variables, $2K$ residual inequalities, one mass equality, and nonnegativity bounds. Its solution cost depends on the solver and cannot be identified with matrix-construction cost. The $|\mathcal{S}||\mathcal{A}|$ recovery problems are independent and may be processed sequentially or in parallel. A dense recovered kernel requires $O(|\mathcal{S}|^2|\mathcal{A}|)$ storage.

E.4 Task scales and planning

The objective sums raw absolute residuals over tasks, without bankwise averaging, learned reliability weights, or regularization. Consequently, both bank size and reward scale affect a bank's influence. Multiplying a task's reward and Q values by a positive constant multiplies its residual by that constant. A common scaling of the entire objective leaves its minimizers unchanged; separate row normalization generally does not. Such weighting would define a different estimator and is not an implicit preprocessing step.

For each new bounded reward and termination pair, discounted value iteration applies Equation equation 6 with the recovered kernel fixed, followed by greedy policy extraction. Its output is a stationary policy. A separate finite evaluation horizon does not specify the number of value-iteration updates or change the planning objective.

Algorithm 2 initializes $V = 0$ and maintains $U = \mathcal{T}_{\mathrm{new}}^{\widehat{P}} V$. When $\|U - V\|_\infty \le \epsilon_V$, it accepts U and recomputes the action backups at that accepted value before choosing the policy. Writing $T = \mathcal{T}_{\mathrm{new}}^{\widehat{P}}$, contraction gives

$$\|TU - U\|_\infty = \|TU - TV\|_\infty \le \gamma \|U - V\|_\infty \le \gamma \epsilon_V .$$

The fixed-point bound is therefore $\|U - V_{\widehat{P}}^\star\|_\infty \le \gamma \epsilon_V / (1 - \gamma)$. These statements assume a stochastic kernel and exact backups. Implementation-specific iteration caps, comparison tolerances, and post-update residual checks are separate numerical settings.

E.5 Parametric realization and implementation interface

Equation equation 3 replaces independent finite transition rows with a shared deterministic successor model. Its query distribution μ and positive task weights are fixed, and the expected loss must be finite. The predicted state must lie in the common state domain, where the corresponding rewards, termination indicators, and nonterminal successor Q-values are well-defined. Frozen Q parameters do not by themselves specify whether derivatives through Q inputs are available to the model fitter.

A numerical realization specifies the query construction and data access, model parameterization and state constraints, weighting, optimization procedure, and numerical evaluation of the action supremum. Planning additionally specifies its value representation and action optimization. These choices are separate from the finite-state solver in Algorithm 2; the rowwise prediction theorem is not a guarantee for this nonlinear fit. The same source-Q convention must hold throughout: policy-evaluation critics require their matching continuation, as described in Appendix A.

F Input preparation and evaluation interface

This section records the original FourRooms bank-expansion interface and distinguishes it from the later Windy control protocols. These experimental settings specialize the generic procedure in Section E; unresolved fields remain explicit.

F.1 Original FourRooms environment and recovery inputs

FourRooms contains 68 open states and four cardinal actions in an 11×11 coordinate frame. A cross wall occupies row 5 and column 5, with doors at $(5, 3)$, $(5, 7)$, $(3, 5)$, and $(7, 5)$; the center $(5, 5)$ is a wall. Walls and doors determine legal neighbors. Deterministic moves follow the selected direction and remain in place upon hitting a wall. Windy moves select the intended, left-rotated, and right-rotated directions with probabilities 0.5, 0.25, and 0.25, respectively, retaining wall collisions. The teleport cells are $\mathcal{Z}_{\mathrm{tele}} = \{(4, 4), (4, 6), (6, 4), (6, 6)\}$. At each of these cells, every action moves uniformly to the 16 interior cells of the diagonally opposite room; other cells follow deterministic dynamics.

Table 3 specifies the support and model-class inputs. For windy dynamics, geometric adjacency supplies a conservative support: the current state and every legal cardinal neighbor. It does not supply transition probabilities. A deterministic model is allowed to select any open state, whereas teleporting recovery retains the full stochastic simplex. These choices precede recovery and are not selected using downstream returns.

The original record specifies SciPy's HiGHS interface for stochastic recovery; the numerical fields it leaves unspecified are listed in Table 6.

F.2 Task banks and Q learning

For trial seed z, a permutation generated by `np.random.default_rng(12000 + z)` orders eligible training goals, excluding designated held-out target and unsafe positions. The anchor uses the

Table 3: Original FourRooms bank-expansion inputs. Goal counts are anchor and fused totals; supplemental counts are their differences. The later Windy control studies use separate protocols.

Dynamics	Candidate successors	Model class	Goals: anchor/fused
Deterministic	All 68 open states	One-hot	$1/3$
Windy	Current state and legal cardinal neighbors; at most 5	Simplex	$1/4$
Teleporting	All 68 open states	Simplex	$5/20$

Table 4: Information used at each stage.

Stage	Permitted inputs	Output
Q learning	Generative successor samples; task rewards, termination, and discount	Frozen task Q tables.
Recovery and planning	Q queries, task definitions, supplied support, model class, and new reward	Shared kernel and new-task policy.
Validation and evaluation	True transition kernel, fixed evaluation tasks, start distribution, and recovered policy	Recovery diagnostics and returns.

first K_A goals; the supplemental bank uses the next $K_F - K_A$ goals. Thus the goal sets are disjoint. Windy and teleporting tasks use goal-indicator rewards. Deterministic tasks assign $+1$ to the goal and fixed negative $-\mathrm{Uniform}(0,1)$ perturbations elsewhere, generated with seed $30000 + z$. Goal arrival terminates the task.

Each bank starts from zero Q values and runs 20,000 synchronous generative sweeps, drawing one successor per state–action pair shared across its goals. The anchor and supplement use independent streams seeded by $40000 + 10z$ and $40000 + 10z + 1$, respectively.

A sweep uses the target

$$\begin{aligned} y_i^{(t)}(s,a) &= r_i(s') + \gamma\big(1 - d_i(s')\big) \max_{a'} Q_i^{(t)}(s',a'), \\ Q_i^{(t+1)}(s,a) &= (1-\alpha_t) Q_i^{(t)}(s,a) + \alpha_t y_i^{(t)}(s,a), \end{aligned} \tag{F.1}$$

where all targets in a sweep use the preceding Q tables and learning rates decrease linearly from 0.2 to 0.01. Arrival-based masking ensures that a terminal successor supplies its immediate reward without continuation. The distinct issue of currently terminal source entries remains listed in Table 6. The preliminary 1,000-sweep smoke run is excluded from the canonical protocol.

F.3 Access permissions and resource accounting

Table 4 separates input acquisition, fitting and planning, and post-fit validation and evaluation. Transition labels do not enter the fitting objective or determine the recovered probabilities. Q learning accesses sampled successors, and candidate supports may encode known geometry. The later Windy resource study also uses the true kernel for a post-fit objective check inside its recovery routine; this check does not supply fitted probabilities. Independent training streams do not imply unbiased Q errors or independent Bellman residuals.

For T complete sweeps and B banks, the recorded sampling schedule implies $TB|\mathcal{S}||\mathcal{A}|$ successor draws and $T|\mathcal{S}||\mathcal{A}| \sum_b K_b$ scalar Q updates. The distinction matters because goals share samples within a bank. For $T = 20{,}000$, one bank entails 5,440,000 draws; these counts are derived from the schedule rather than measured runtime statistics. Map counts, successor draws, Q updates, and inversion time should be reported separately. The original anchor/fusion comparison does not match all these budgets.

Table 5: Held-out target and unsafe coordinates. Each dynamics variant uses the common task and its corresponding second task. $\mathcal{Z}_{\text{tele}}$ denotes the four teleport cells.

Phase	Applicable dynamics	Target	Unsafe states
Development	All (common task)	$(9, 1)$	$(5, 3)$
	Deterministic / windy	$(9, 9)$	$(3, 5), (5, 3)$
	Teleporting	$(9, 9)$	$\mathcal{Z}_{\text{tele}}$
Confirmation	All (common task)	$(1, 9)$	$(5, 7)$
	Deterministic / windy	$(1, 1)$	$(5, 7), (7, 5)$
	Teleporting	$(1, 1)$	$\mathcal{Z}_{\text{tele}}$

F.4 Held-out tasks and evaluation quantities

Development uses seeds 0–2; confirmation uses seeds 3–5 with new target positions and a frozen protocol. Each trial has two held-out planning tasks, combining a $+1$ terminal target with zero-reward terminal unsafe states. Confirmation changes neither algorithm nor thresholds, and no failed seed or task is replaced. Table 5 gives the recorded coordinates. Each phase comprises nine variant–seed trials; both planning tasks are evaluated within every trial.

Transition recovery is evaluated over the full kernel, including entries outside candidate supports:

$$\mathrm{MSE}(\widehat{P}, P^\star) = \frac{\sum_{s,a,s'} \big(\widehat{P}(s' \mid s, a) - P^\star(s' \mid s, a)\big)^2}{|\mathcal{S}|^2 |\mathcal{A}|}. \tag{F.2}$$

F.5 Value diagnostics and policy evaluation

For each new reward and termination pair, discounted value iteration applies Equation equation 6 with the recovered kernel fixed, followed by greedy policy extraction. The recorded discount is $\gamma = 0.99$. The output is a stationary policy; the 200-step evaluation horizon does not specify the number of value-iteration updates or a finite-horizon planning objective.

Q Bellman RMSE is a true-kernel training diagnostic. Its aggregation domain is not recorded; it is not used for model selection.

Returns are evaluated by dynamic programming under $P^\star$ for $H = 200$ discounted steps. For a fixed policy, set $v_0 = 0$ and use

$$v_{h+1}(s) = \sum_a \pi(a \mid s) \sum_{s'} P^\star(s' \mid s, a) \big[r(s') + \gamma(1 - d(s'))v_h(s')\big]. \tag{F.3}$$

The initial distribution μ is uniform over nonterminal open states, giving $J_H(\pi) = \mu^\top v_H$. The reported normalized return divides $J_H(\widehat{\pi})$ by the same evaluation applied to the true-model reference policy. The record calls this policy optimal but does not establish whether it is optimized for finite H or for the infinite discounted objective. Consequently, the denominator is not identified here with the finite-horizon optimum without checking its construction. True-model access serves evaluation and reference construction; it does not select the recovered kernel or its policy.

F.6 Unresolved original implementation fields

The original bank-expansion record does not specify the fields in Table 6. They must be verified against the implementation or saved configuration before exact numerical reproduction is claimed. In particular, no solver defaults, postprocessing rule, or planning tolerance is inferred here.

F.7 Separately recorded Windy control protocols

The later Windy resource-control study uses the same unweighted simplex estimator with a separately fixed protocol: two anchor records and two supplemental records, six training seeds, and checkpoints at 1,000, 5,000, and 20,000 sweeps. Its longer-training controls additionally use doubled checkpoints.

Table 6: Implementation fields not specified in the original bank-expansion record; later control protocols are separate.

Component	Fields requiring verification
LP solver	SciPy/HiGHS versions, tolerances, failure handling, and selection among multiple optima.
Returned probabilities	Clipping, renormalization, or other numerical cleanup.
Discrete choices	Column-matching and greedy-action tie rules.
Value iteration	Initialization, stopping criterion, and iteration cap.
Terminal-source entries	Q convention and row inclusion when the current state is terminal for a particular task.
Evaluation	Reference-policy horizon convention and the index set used to aggregate Q Bellman RMSE.

All source-state rows, including currently terminal queries, are updated and retained. These settings do not retroactively specify the original experiment.

The resource study uses HiGHS dual simplex, with primal and dual feasibility tolerances of 10^{-9}. It checks solver success, objective consistency, and numerical probability feasibility, and cross-checks three prescribed rows with HiGHS interior point. Returned probabilities are stored without clipping or renormalization. It does not compute the optional per-row dual certificate in Section E. Its planner initializes at zero, has a 20,000-iteration cap, accepts the last update when the successive-value difference is below 10^{-12}, recomputes its action backups, and checks the accepted value's residual against 1.01×10^{-12}. Action ties use the first action index. The reference is a true-model infinite-horizon stationary policy evaluated over the same 200 discounted transitions.

A separate controlled-perturbation study records numerical primal/dual diagnostics and cross-solver comparisons. Those floating-point diagnostics are useful checks of that study's solves; they are not exact certificates or a universal acceptance rule for all experiments. Its row centering subtracts the same task-specific constant from the measurement row and right-hand side, preserving the L1 objective on the simplex. Solver diagnostics, centering, and any planning refinements belong to the respective protocol.

G Experimental design and evaluation units

The main text follows four experiments. Section 4.1 adds source goals in FourRooms; Section 4.2 uses specialist agents to recover a model for new continuous-state tasks; Section 4.3 compares learning new goals with relearning old ones at matched cost; and Section 4.4 varies source-task similarity and Q-value error. This appendix supplies their implementation details and complete results in the same order.

The experiments use one recovered dynamics model for each environment and source-bank configuration. New rewards and termination functions enter planning after recovery; no new-task Q-function is supplied to the inverse fit. Finite-state studies compare a smaller anchor with expanded source-task coverage. The continuous-state comparison uses a single agent trained on four goals and two specialist agents trained on disjoint halves of those same goals. It examines reuse of different learned value functions with a fixed goal union, rather than an increase in the number of distinct source goals.

The experimental units remain separate. FourRooms confirmation varies source learning in three trials per dynamics variant. The acquisition control varies goals and sampling streams over six paired seeds in one Windy kernel. The perturbation control varies reward/noise realizations over six paired seeds in that kernel. The continuous analysis fixes the pretrained agents and varies three successor-model initializations. Tasks and evaluation starts are nested within these model trials. All specified tasks are included in each reported environment mean.

Access to dynamics. Source-agent training uses environment transitions. The inverse fit uses frozen Q-functions, known rewards and termination, and prescribed query inputs; observed next states are not its fitting targets. Ground-truth dynamics are used subsequently for evaluation and

reference planning. In the controlled-Q study, they additionally construct exact source values before perturbation. These are distinct information accesses, not a claim of learning without simulator access at every stage.

H FINITE-STATE RECOVERY AND CONFIRMATION

H.1 ENVIRONMENT, SOURCE RECORDS, AND ACQUISITION

Layout and dynamics. FourRooms has 68 open cells in an 11×11 layout. Interior coordinates run from 1 to 9; row 5 and column 5 are walls except for doors at $(5,3), (5,7), (3,5), (7,5)$. Coordinates are ordered by row, then column. Actions are up, right, down, and left; wall collisions leave the state unchanged. Deterministic dynamics execute the selected action. Windy dynamics execute it with probability 0.5 and either adjacent direction with probability 0.25. Teleporting dynamics are deterministic except when departing from $(4,4), (4,6), (6,4), (6,6)$: every action then selects uniformly among the 16 interior cells of the diagonally opposite room. A task that terminates on entry to a teleport cell prevents a later teleportation step.

Source tasks and Q learning. For each trial seed z, we exclude that phase's test goals and unsafe cells from the training-goal pool. NumPy's `default_rng` with seed $12000 + z$ permutes the remaining row-major state indices. The first K_F entries define the fused goals; the first K_A belong to the anchor and the rest to its supplement. In stochastic variants, source reward is $+1$ upon arrival at its goal and zero elsewhere; the goal terminates that source task. Deterministic source tasks instead assign fixed independent $-U[0,1)$ background rewards, generated jointly with seed $30000 + z$, then overwrite each goal reward with $+1$. This distinguishing reward perturbation is part of the deterministic experiment.

All Q tables start at zero in float64 and use $\gamma = 0.99$. Each of 20,000 synchronous sweeps draws one successor for every state–action pair from the true generative kernel. Tasks within a bank share these draws, while their rewards, termination masks and Q arrays remain separate. The update uses the preceding sweep's values and learning rate $\alpha_t = 0.2 - 0.19t/19999$, $t = 0, \ldots, 19999$. The anchor and supplement use RNG seeds $40000 + 10z$ and $40001 + 10z$. Current-terminal rows are updated like other source rows; termination masks continuation only at the sampled successor. Fusion retains the original anchor arrays and appends the supplement. True successors supervise Q learning, but are not fitting targets for transition recovery.

Table 7: **Historical bank-expansion budgets.** A/F denote anchor and fused. Counts are logical acquisition operations, not measured runtime. One bank uses $20000 \times 68 \times 4 = 5.44$M successor draws, shared across its goal records.

Dynamics	Records A/F	Draws A/F (M)	Updates A/F (M)
Deterministic	1/3	5.44/10.88	5.44/16.32
Windy	1/4	5.44/10.88	5.44/21.76
Teleporting	5/20	5.44/10.88	27.20/108.80

Model class and historical numerical implementation. Deterministic recovery compares all 68 candidate successors and selects the first column minimizing the summed absolute Bellman residual. Windy recovery uses the current cell and all legal cardinal neighbors as a common candidate support for every action. Teleporting recovery uses all 68 states. Both stochastic variants minimize the same unweighted L1 objective on the simplex as Equation equation 2; rewards and record multiplicities are unchanged. The historical implementation used SciPy `linprog(method="highs")`, checked solver success, clipped negative returned probabilities to zero, and renormalized their sum. For planning it iterated action values from zero until the maximum iterate change was below 10^{-12}, with a 10,000-iteration cap, and took the first maximizing action from the final Q iterate. These historical numerical conventions differ from the explicit post-solve validation and accepted-value backup recomputation used in Algorithm 1 and the later control studies; the archived results were not retroactively recomputed under those newer conventions.

H.2 FROZEN CONFIRMATION AND RETURN EVALUATION

Development used seeds 0–2. Confirmation used seeds 3–5 and new held-out goal locations. The confirmation executor checked the frozen development engine and protocol hashes, unchanged recovery method, bank sizes, acquisition budget, discount, horizon and gates, and disjoint development/confirmation goals and seeds. No trial or task was replaced. The layout and each variant's kernel were unchanged; the confirmation therefore tests new Q-learning realizations and new reward/termination tasks, not new map topologies.

For confirmation, task U has goal $(1, 9)$ and unsafe cell $(5, 7)$. Task M has goal $(1, 1)$; its unsafe cells are $(5, 7), (7, 5)$ for Deterministic/Windy and the four teleport cells for Teleporting. Arrival at a goal yields $+1$ and terminates. Arrival at an unsafe cell yields zero and terminates; all other rewards are zero. The uniform starting distribution excludes goal and unsafe states, leaving 66 starts for U, 65 for local M, and 63 for teleport M. Development used goal $(9, 1)$ and unsafe $(5, 3)$ for U; its second goal was $(9, 9)$ with unsafe $(3, 5), (5, 3)$ for local variants or the same four teleport cells.

Figure 8 summarizes these changes. The two goals and the hazard symbols are superimposed to describe the task family; they do not define a single combined task.

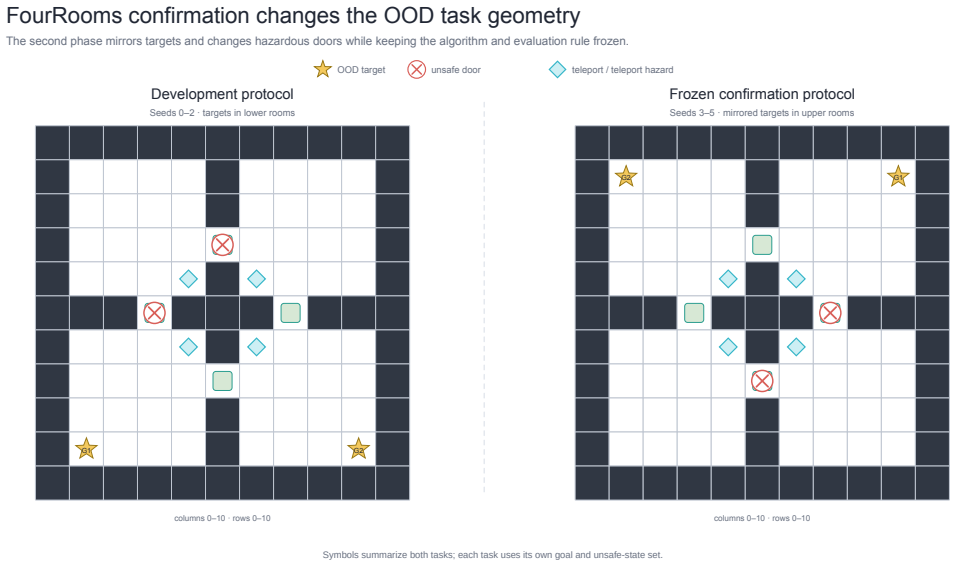


Figure 8: **Development and frozen confirmation task geometry.** Seeds 0–2 use lower-room goals, whereas seeds 3–5 use the mirrored upper-room goals. G1 denotes task U and G2 denotes task M. Task U uses one unsafe door in each phase; local task M uses both marked unsafe doors, while Teleporting task M uses the four cyan cells. Only the corresponding goal and unsafe set are active in each task. The layout and transition kernel are unchanged across phases.

For a fixed policy, let $W_0 = 0$ and compute in the true kernel

$$W_{h+1}(s) = \sum_{s'} P^\star(s' \mid s, \pi(s)) \left[r(s') + 0.99(1 - d(s'))W_h(s') \right]. \tag{H.1}$$

The mean of W_{200} over nonterminal starts is $J_{200}(\pi)$. We report $R = J_{200}(\widehat{\pi})/J_{200}(\pi_{\text{ref}})$, where π_{ref} is the stationary policy planned with the true kernel. This is exact expectation recursion, not a Monte Carlo episode average or a success rate. The denominator does not assert finite-horizon optimality. For tasks U/M respectively, the reference returns are $0.905685859/0.707044247$ in Deterministic, $0.764032000/0.634879448$ in Windy, and $0.938261698/0.924447956$ in Teleporting. We compute transition MSE over all $68 \times 4 \times 68$ tensor entries, including zero entries outside each row's support.

Statistical unit and frozen gates. Each trial fits an anchor/fused model pair and evaluates both tasks. Reported environment means first average tasks within a model and then average the three trial seeds equally. Shared-model tasks are not independent training replications. The frozen protocol required all nine confirmation trials to be valid; fusion's mean normalized return to reach 0.98 for Deterministic/Windy and 0.95 for Teleporting; and its mean return and transition MSE to be no worse than the anchor, with tolerances 10^{-12} and 10^{-15} respectively. All gates passed. These gates concern variant means and do not require every task to improve or establish statistical significance.

H.3 All confirmation trials and task outcomes

Table 8 retains all nine paired model trials; Table 9 retains all 18 task cells. The six stochastic model pairs improve both mean return and transition MSE. At task level there are 11 stochastic improvements and one tie; deterministic tasks all tie at their reference return. Across development and confirmation together, stochastic tasks have 23 improvements and one tie, while all 12 deterministic task cells tie. These are descriptive paired counts.

Table 8: **All nine confirmation model pairs.** P-MSE is multiplied by 10^3; returns average both tasks within one fitted model. A/F denote anchor/fused.

Dynamics	Seed	P-MSE A	P-MSE F	Return A	Return F
Deterministic	3	0.000000	0.000000	1.000000	1.000000
Deterministic	4	0.000000	0.000000	1.000000	1.000000
Deterministic	5	0.000000	0.000000	1.000000	1.000000
Windy	3	5.486784	1.057961	0.628197	0.996637
Windy	4	5.704928	1.054456	0.523622	0.997304
Windy	5	5.711757	0.964104	0.641441	0.997036
Teleporting	3	7.927170	0.213734	0.680974	0.999919
Teleporting	4	13.327192	0.788955	0.816617	0.999919
Teleporting	5	7.230360	0.928200	0.721681	0.999919

Table 9: **All 18 confirmation task cells.** Each row contains the two held-out tasks for one model pair. U and M are defined above; values are normalized discounted returns. The Teleporting seed-5 U tie is retained.

Dynamics	Seed	U: A	U: F	M: A	M: F
Deterministic	3	1.000000	1.000000	1.000000	1.000000
Deterministic	4	1.000000	1.000000	1.000000	1.000000
Deterministic	5	1.000000	1.000000	1.000000	1.000000
Windy	3	0.648856	0.994828	0.607539	0.998446
Windy	4	0.430891	0.998759	0.616352	0.995849
Windy	5	0.674066	0.998043	0.608816	0.996030
Teleporting	3	0.856468	0.999839	0.505480	1.000000
Teleporting	4	0.986576	0.999839	0.646657	1.000000
Teleporting	5	0.999839	0.999839	0.443523	1.000000

Figure 9 places the complete stochastic model pairs from both phases side by side. Each return point averages two tasks within one model; the three confirmation points per dynamics are the observations summarized in the main FourRooms table. The phase distinction is preserved when reporting paired counts.

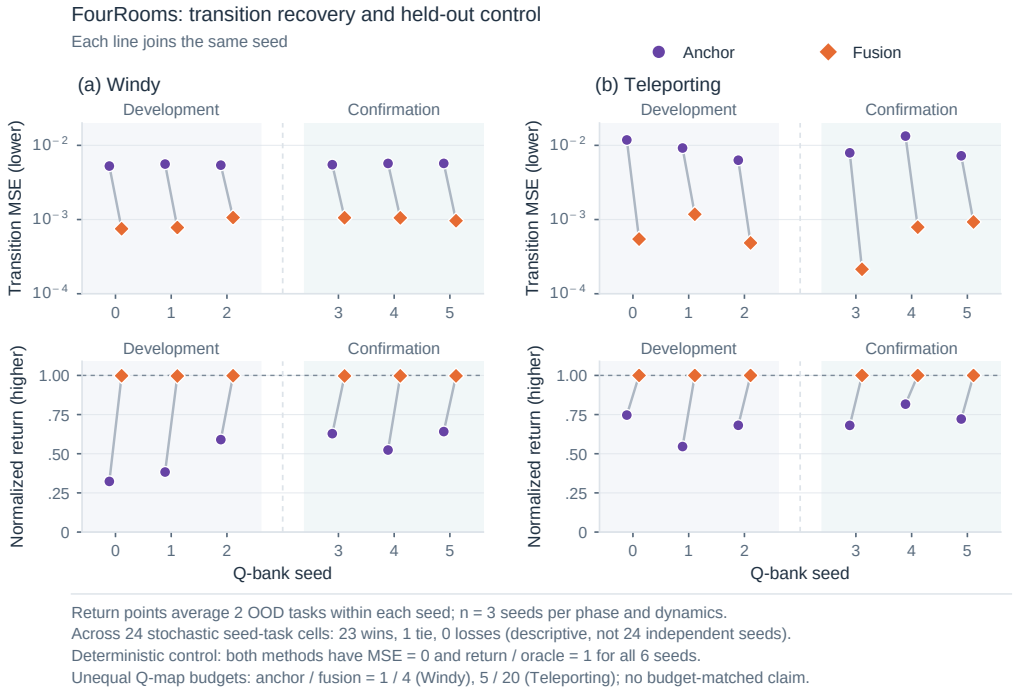


Figure 9: **Development and confirmation results for every stochastic trial.** Lines join Anchor and Fusion for the same source-learning seed. The upper panels show full-kernel MSE on a logarithmic axis; the lower panels show normalized return averaged over the two tasks. Development uses seeds 0–2 and confirmation uses seeds 3–5. The 23 improvements and one tie count 24 task cells across both phases; confirmation alone has 11 improvements and one tie across 12 stochastic task cells. These task cells are nested within model trials. Deterministic trials all have zero MSE and normalized return one, as reported in the tables. Bank sizes and acquisition budgets differ between Anchor and Fusion.

Recorded sources. The accompanying source archive contains the development and confirmation result files with their frozen protocols. Results were generated by source commits `399c483` and `804536a`, respectively. The confirmation JSON contains every reported model-level and per-task statistic, source-goal coordinates, and validity fields. The comparison adds both source-task coverage and acquisition cost; the separate resource study addresses matched acquisition budgets.

H.4 Spatial diagnostics of archived models

Instances and reconstruction. The spatial plots examine Windy confirmation seed 4 on task U and Teleporting confirmation seed 5 on task M, the same selected instances used in Figures 3a and 3b. These are post-confirmation mechanism illustrations, not new independent trials or randomly sampled representative cases. The archived engine reconstructs their source Q tables, simplex-L1 recovery and planned policies under the original numerical conventions. Endpoint MSE and task return were checked against the archived confirmation records within the declared floating-point tolerances. The solver and policy conventions are not replaced by those of the later acquisition/noise controls.

State-wise transition error. For a recovered model $\widehat{P}$, define

$$e_{\mathrm{TV}}(s) = \frac{1}{4}\sum_{a\in\mathcal{A}}\frac{1}{2}\sum_{s'\in\mathcal{S}}\left|\widehat{P}(s' \mid s,a) - P^{\star}(s' \mid s,a)\right|. \tag{H.2}$$

Figure 10 colors every open state by this action-averaged total variation. Its spatial mean falls from 0.480 to 0.162 for Windy and from 0.379 to 0.082 for Teleporting. These maps locate the remaining discrepancies beyond the single state–action rows in Figure 3a. They measure TV, whereas the primary table reports full-kernel MSE.



Figure 10: **Where transition errors remain in the two selected confirmation instances.** Cell colors show the state-wise mean TV in Equation equation H.2; white indicates zero error. Anchor and Fusion share a color scale within each row: $[0, 0.75]$ for Windy and $[0, 1]$ for Teleporting. Purple/orange dots mark source goals; the star and red crosses mark the selected new task. The cyan markers in the Windy row are layout landmarks only; Windy has no teleportation. The displayed spatial means summarize these two instances, not an average over confirmation seeds.

Outcomes from all valid starts. Figure 11 executes each recovered policy once from every nonterminal state in the true environment, using the same start-specific uniform random stream for the paired policies. For row-major state index s, the stream seed is $20260904 + 7919s$; each episode stops on goal arrival, unsafe arrival, or after 100 steps. Fusion reaches the goal from all 66 Windy starts and all 63 Teleporting starts under these streams, compared with 25 and 27 starts for Anchor. This diagnostic shows the spatial extent of the behavior illustrated in Figure 3b. Each cell contains one rollout, so these counts are not repeated-trial success estimates. The primary returns instead use the 200-step exact expectation recursion specified above.

Adding constraints from a fixed learned bank. Figure 12 recovers models from successive prefixes of the same learned Q bank: $K = 1, 2, 3, 4$ for Windy and $K = 5, 10, 15, 20$ for Teleporting. All stages retain the historical recovery objective, support and planning rule. Arrows are the greedy new-task policy, and outlined states choose an action different from the final-prefix policy. Returns use the 200-step exact evaluation, averaged over all valid starts and normalized by the same true-model reference. These are constraint-count diagnostics: the horizontal sequence is neither training time nor an acquisition-budget match.



Figure 11: **Paired execution outcomes over all nonterminal start states.** Rows use Windy seed 4/task U and Teleporting seed 5/task M. Green, red and yellow indicate goal arrival, unsafe termination and 100-step timeout, respectively. Colors reproduce every outcome cell in the archived map; stopping-step labels are omitted for legibility and remain available in the original full-resolution figure. Anchor/Fusion share the initial state and random stream at each cell, with one rollout per policy. Goal and unsafe cells are excluded from the start set. These maps describe two selected models; the confirmation tables supply the complete model-trial results.

The Windy sequence improves from return 0.431 to 0.999 as mean TV decreases from 0.480 to 0.162. Teleporting illustrates why the control outcome is reported alongside model error: from five to ten Q maps, mean TV falls from 0.379 to 0.106, while return decreases from 0.444 to 0.364. The 15- and 20-map models both reach normalized return 1.000 on this task. All intermediate stages are retained. The final bank is useful in both instances, but average model error alone does not order their intermediate policies. Table 10 reports all eight stages at readable numerical precision.

Table 10: **Every stage of the fixed-bank prefix diagnostic.** Mean TV is averaged over all states and actions; return uses the selected new task and the common 200-step reference evaluation. Values are computed from the archived prefix transition arrays and recorded normalized returns, rounded to four decimal places.

Dynamics	Q records	Mean TV	Normalized return
Windy	1	0.4798	0.4309
	2	0.2068	0.9857
	3	0.1893	0.9982
	4	0.1621	0.9988
Teleporting	5	0.3793	0.4435
	10	0.1061	0.3643
	15	0.0807	1.0000
	20	0.0825	1.0000



Figure 12: **Model and policy changes as more learned Q constraints are included.** Top: all four Windy prefixes, using confirmation seed 4/task U. Bottom: all four Teleporting prefixes, using seed 5/task M. Each panel displays action-averaged transition TV, the resulting policy and its normalized return; Table 10 gives all stage metrics. Colors are scaled consistently within each dynamics row. Yellow outlines mark policy disagreement with the final prefix, not an additional error metric. The source Q arrays remain fixed; only the active prefix changes. In particular, the nonmonotone Teleporting return at ten maps is preserved.

I CONTINUOUS-STATE REALIZATION AND COMPLETE RESULTS

Section 4.2 gives the comparison and model-fitting overview. Here we specify the parametric realization of Eq. equation 3 and report all task cells from the three archived cohorts. The reported returns use ordinary single-model value iteration. The source evaluation files also contain calibrated models and a five-model robust planner; their outputs are not substituted for the two-bank model's returns. The present comparison is a retrospective analysis of the archived single-model evaluations, not a newly executed or prospectively frozen confirmation study.

I.1 SOURCE BANKS, MODEL FITTING, AND INFORMATION ACCESS

Source-task inventory. The specialist banks introduced in Section 4.2 are denoted L and R. They are trained separately on interleaved, disjoint pairs of the four goals listed in Table 11; the single-agent reference is trained on their union. The two-bank model therefore uses the specialists' records, without appending records from the reference agent. Both conditions cover the same goal inventory, unlike the smaller-bank anchor comparison in FourRooms.

Each Q agent has a nominal training budget of 20 million environment steps and is frozen before model fitting. Consequently the two-bank model uses 40 million source-interaction steps, whereas the single-agent reference uses 20 million. This comparison evaluates reuse of separately acquired banks; it is not a matched-interaction comparison. The corresponding controlled acquisition comparison is reported separately for FourRooms. The archived agents use a goal-conditioned Q network with four 1024-unit hidden layers in MountainCar and Pendulum, and six 2048-unit hidden layers in Reacher, layer normalization, and sigmoid outputs. The training configuration uses discount 0.99, 256 parallel environments, 64 rollout steps per update, minibatches of 256, one optimization epoch, and a linearly decaying learning rate starting at 10^{-4}. The same frozen constructor is reused across world-model seeds within an environment.

A single model fitted to two banks. At each world-model update, sample a batch of states, independent uniform actions, and one independent uniform goal per bank and state–action pair. If $\ell_b(\theta)$ denotes the sample mean of absolute Bellman residuals for bank b, the fitted objective is $\frac{1}{2}\ell_L(\theta) + \frac{1}{2}\ell_R(\theta)$. Since each bank contains two goals, this is a stochastic estimate of Eq. equation 3 with four equal record weights $w_i = 1/4$. The reference uses its four goals with equal weights. Rewards are evaluated at the predicted successor, and the continuation term is multiplied by one minus the predicted successor's source-task termination indicator. The continuation value is the exact maximum of the frozen network's outputs over the finite action set: three actions in MountainCar and Pendulum, nine in Reacher. There is no additional mask based on whether the queried current state lies in a source goal. Both the current-state Q-value and the successor-state action values follow this arrival-reward convention.

Only the successor-model parameters are optimized. Q parameters and normalization statistics remain fixed, with the Q networks evaluated in inference mode. Derivatives through the Q network's predicted-state input are retained. Neither sampled successor labels nor downstream task returns are used to fit or select the two-bank model. The final optimization iterate is saved after the prescribed updates; there is no selection of a checkpoint by downstream return.

The model is a residual network $f_\theta(s, a) = s + h_\theta(s, a)$ with two 256-unit tanh layers, a one-hot action input, and output-layer initialization scale 0.01. Queries are uniform over the configured state box. In Reacher, four-dimensional angle–velocity states are drawn first and lifted to six-dimensional observations by forward kinematics; independent sampling of redundant fingertip coordinates is not used. The Reacher model takes and predicts four effective coordinates, then recomputes fingertip position for reward and Q queries. Predicted angles are wrapped for Pendulum and Reacher. No additional projection of predicted positions or velocities onto the query box is applied; planning uses the boundary interpolation rule described below.

All models use Adam for $20{,}000$ updates, starting at learning rate 10^{-4}. MountainCar and Pendulum use batches of 4096 and cosine learning-rate decay; Reacher uses batches of 1024 and linear decay to 10^{-20}. These settings were checked against the configurations stored in the recovered checkpoints, including Reacher's override from reset-state sampling to uniform valid-state sampling. This is a

Environment	Goal coordinate	Bank L	Bank R
MountainCar	Position	$-1.2,\ 0$	$-0.6,\ 0.6$
Pendulum	Angle	$-2.4,\ 0.8$	$-0.8,\ 2.4$
Reacher	Fingertip (x, y)	$(1, 0),\ (-1, 0)$	$(0, 1),\ (0, -1)$

Table 11: Source goals for the continuous-state realization. The single-agent reference is trained on the union of both columns. Rewards are sparse, with goal radii 0.1, 0.25, and 0.2, respectively; only the indicated coordinates enter the source reward.

Setting	MountainCar	Pendulum	Reacher
State / effective dimension	2	2	4
Discrete actions	3	3	9
VI grid points	200^2	101^2	17^4
Maximum VI iterations	10,000	2000	1000
VI stopping threshold	10^{-10}	10^{-7}	10^{-6}
Evaluation starts	512	512	256
Episode horizon	200	200	100
World-model seeds	10–12	10–12	13–15
Q constructor seed evaluated	25	25	25

Table 12: Fixed numerical settings for the archived single-model evaluations. All tasks use discount 0.99 and reset-start seed 0. World-model seeds share the same frozen Q constructor within each environment.

nonconvex parametric fit; the global rowwise optimization and finite-state guarantees do not apply to it automatically.

I.2 Tasks, planning, and evaluation

All three environments have continuous states and discrete actions. They use custom sparse, first-arrival tasks, rather than the default benchmark rewards. MountainCar has its usual position–velocity physics and three push actions. Pendulum uses the torque set $\{-2, 0, 2\}$. Reacher is a custom two-link simulator with two absolute angles and two angular velocities: each joint takes a torque from $\{-1, 0, 1\}$, angular acceleration is integrated independently, and fingertip position follows from two unit-length links. It is not the default MuJoCo Reacher environment.

The downstream tasks are fixed within each archived cohort. MountainCar's *Fast car* requires $|v - 0.06| < 0.01$; *Gentle car* requires $|x - 0.6| < 0.1$ while terminating without reward whenever $|v| > 0.05$; and *Shortest path* reaches either $|x + 1| < 0.1$ or $|x - 0.4| < 0.1$. Pendulum requires, respectively, a Euclidean distance below 0.35 from $(\theta, \omega) = (0, 0)$, $|\omega - 4| < 0.5$, or a distance below 0.4 from $(1.6, -1.5)$. Reacher requires fingertip distance below 0.2 from $(\sqrt{2}, \sqrt{2})$, angle-coordinate distance below 0.2 from $(0, \pi)$, or angular-velocity distance below 0.2 from $(2, 2)$. The stored wrapped angles are compared directly in these predicates; angular distances are not replaced by shortest distances on the circle. Every task terminates on its first success. Reaching a target state therefore does not measure sustained stabilization.

The planning grids span $[-1.2, 0.6] \times [-0.07, 0.07]$ for MountainCar, $[-\pi, \pi] \times [-8, 8]$ for Pendulum, and $[-\pi, \pi]^2 \times [-2, 2]^2$ for Reacher. Reacher's velocity box specifies the training and planning range; it is not a physical velocity clip. For every model and task, the same ordinary value-iteration routine is applied to its predicted successors, the new reward, and the new termination rule. Continuation values use multilinear interpolation, with coordinates clipped to the grid edges. At evaluation time, the policy maximizes interpolated action values and executes the chosen action in the true simulator. The single-agent reference and two-bank model use identical grids, stopping rules, starts, and task predicates. Neither uses a worst-case model backup.

For a model m and task t, the reported score is

$$S_{m,t} = \frac{R_{m,t}}{R_{\text{grid},t}}, \qquad R_{m,t} = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=0}^{H-1} 0.99^k r_t(s_{n,k+1}).$$

The denominator evaluates, on the same simulator starts, a policy computed by value iteration with the true dynamics on the same finite grid. It is a numerical reference policy, not a proven optimal policy, so scores can exceed one. These scores are neither success probabilities nor clipped to $[0, 1]$. Table 13 provides the reference returns, allowing recovery of absolute returns from normalized scores.

I.3 Complete task-level results and cohort scope

Table 13 averages the three world-model seeds for each of the nine tasks; Table 14 retains every paired cell. Two-bank fusion has higher family-mean return in all three cohorts, but this is not a statement that every task or seed improves. MountainCar Fast car regresses in all three pairs. Reacher's mean gain is concentrated in Target angle, while its Far fingertip mean is slightly lower. Pendulum's gain is small because both models already perform close to the numerical reference.

Figure 13 displays the paired differences underlying these task means. It retains all nine tasks and each of the three model fits, including negative differences.

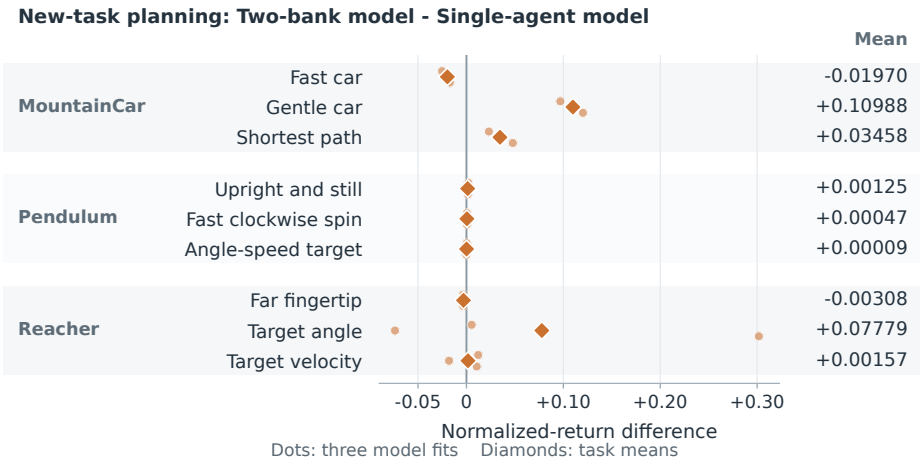


Figure 13: **Task-level changes under the two-bank model.** Return differences are Two-bank minus Single-agent, paired across three model fits with fixed source agents. Dots show individual fits; diamonds show task means. Positive values favor the two-bank model. All nine tasks enter Table 2, and all 27 paired cells appear in Table 14. Model-fit seeds share one source Q constructor per environment.

Environment	Task	R_{grid}	Single agent	Two banks	Difference
MountainCar	Fast car	0.436341	0.959305	0.939604	-0.019701
	Gentle car	0.376531	0.883842	0.993720	+0.109878
	Shortest path	0.625265	0.961514	0.996098	+0.034584
Pendulum	Upright	0.789348	0.999776	1.001024	+0.001247
	Spin	0.887545	0.999504	0.999970	+0.000466
	Diagonal	0.807783	1.000118	1.000212	+0.000095
Reacher	Fingertip	0.690616	1.002588	0.999507	-0.003081
	Angle	0.455076	1.095689	1.173484	+0.077794
	Velocity	0.621345	0.944036	0.945604	+0.001568

Table 13: All nine downstream task means, averaged over the three listed world-model seeds conditional on the frozen Q constructor. Single agent denotes the four-goal reference; two banks denotes the fixed equal-weight joint fit. Difference is two banks minus single agent. R_{grid} is an absolute discounted return; the two model columns are normalized discounted returns.

Table 14: All 27 paired task cells from the archived continuous-state cohorts. Q constructor seed is 25 throughout. No row is removed based on its return; negative differences are retained.

Environment	Task	WM seed	Single agent	Two banks	Difference
MountainCar	Fast car	10	0.982341	0.957154	-0.025187
	Gentle car	10	0.901301	0.998190	+0.096889
	Shortest path	10	0.976485	0.999609	+0.023124
	Fast car	11	0.960454	0.943685	-0.016769
	Gentle car	11	0.886967	0.999493	+0.112526
	Shortest path	11	0.967403	1.000178	+0.032775
	Fast car	12	0.935120	0.917974	-0.017146
	Gentle car	12	0.863258	0.983476	+0.120218
	Shortest path	12	0.940654	0.988508	+0.047854
Pendulum	Upright	10	0.999410	1.001379	+0.001969
	Spin	10	0.999450	0.999959	+0.000509
	Diagonal	10	0.999978	1.000253	+0.000276
	Upright	11	1.000177	1.000530	+0.000354
	Spin	11	0.999514	0.999956	+0.000442
	Diagonal	11	1.000215	1.000310	+0.000095
	Upright	12	0.999742	1.001161	+0.001420
	Spin	12	0.999548	0.999994	+0.000446
	Diagonal	12	1.000160	1.000074	-0.000086
Reacher	Fingertip	13	1.002245	0.998672	-0.003573
	Angle	13	1.022101	1.027439	+0.005339
	Velocity	13	0.927875	0.940020	+0.012145
	Fingertip	14	1.002886	1.000574	-0.002312
	Angle	14	1.275563	1.201782	-0.073782
	Velocity	14	0.968590	0.950622	-0.017969
	Fingertip	15	1.002633	0.999274	-0.003358
	Angle	15	0.989403	1.291229	+0.301826
	Velocity	15	0.935642	0.946169	+0.010527

Replication unit and eligibility. Each environment's three fitted models share one frozen Q constructor, seed 25, and the same reset cohort. A model seed is therefore not an independent Q-training replication; the three task cells evaluated with that model are also dependent. We report the complete paired observations and descriptive means without treating the 27 cells as 27 independent training runs or pooling them into a cross-environment significance test.

The MountainCar campaign originally prescribed Q seeds 24–26 crossed with world-model seeds 10–12. Only Q25 passed its source-Q capability criterion: three of nine prescribed world-model trials, producing nine of the 27 planned task cells, have valid downstream evaluations. The six Q24/Q26 trials are recorded as invalid by that fixed criterion, not replaced, assigned zero returns, or counted as successful runs. The full campaign's primary gate consequently failed; the table is conditional evidence for the recovered Q25 constructor, not a successful full-inventory confirmation. Pendulum has three valid world-model trials conditional on Q25. Reacher uses the three independent world-model seeds 13–15 after a grid-resolution diagnostic fixed the 17^4 grid, also conditional on Q25. This does not establish generalization to arbitrary Q-training seeds.

Model identity and interpretation. The same uncalibrated two-bank model is reported in every environment and every task; it is not selected separately from the archived candidate models by their test returns. The archived single-agent reference is stronger in source-goal coverage than a two-goal specialist, and both compared models cover the same four source goals. Combined with their unequal acquisition budgets, this makes the continuous-state comparison evidence that frozen banks can be jointly realized as one usable model, rather than an isolated test of new-goal information or equal-cost superiority. The latter question is answered by the separate finite-state acquisition study. The archive's robust-planning pass/fail fields concern a different planner and are not repurposed as pass/fail tests for this two-bank comparison.

I.4 Paired simulator replays of the recovered single models

The physical scenes in Figure 14 show policies computed separately from the same two types of recovered model used in the continuous-state tables. We loaded the archived single-agent and uncalibrated two-bank checkpoints, recomputed ordinary single-model value iteration, and executed each policy in the actual simulator. This supplementary replay records states and actions that the original aggregate evaluation did not save. No Q agent or world model was retrained, and no model was selected by its replay return. The two-bank policy uses its one recovered successor model; it does not use the five-model robust backup from the historical visualizations.

Replay configuration and reset states. We use Q constructor seed 25 throughout, world-model seed 10 for MountainCar and Pendulum, and seed 13 for Reacher. The illustrated tasks are, respectively, Gentle car, Upright and still, and Target angle. The source banks and task predicates are those specified in Appendices I.1 and I.2. Both policies receive the same complete reset cohort generated with seed 0: 512 states in MountainCar, 512 in Pendulum, and 256 in Reacher. We retain the respective 200^2, 101^2, and 17^4 planning grids, the original interpolation and stopping rules, discount 0.99, and episode limits of 200, 200, and 100 steps. We run the replay on CPU with the saved model parameters. These executions reuse the existing agents and model fits; they add no independent training replications.

Return and termination checks. The recorder saves each state, chosen action, reward, first goal or unsafe event, terminal step, terminal category, and discounted return. For every initial state, its return is checked against the original single-model rollout evaluator using the same reconstructed policy, with absolute and relative tolerances of 10^{-6}. The recorded per-start return difference from the canonical evaluator is exactly zero for all six policy evaluations. After the first terminal event, saved scene positions are held for display and actions are marked inactive. Such padding contributes no reward or additional successful time steps. The diagnostic curves below end at each policy's own terminal step.

Table 15 separately compares CPU replay means with the archived means for exactly the same task and model seed. The archive-comparison tolerance is $2 \times 10^{-4} + 2 \times 10^{-3}|R_{\mathrm{archive}}|$. The following replay means fall outside this tolerance: MountainCar single-agent, Reacher two-bank. We retain the measured differences; the CPU replays do not replace the archived multi-seed results in Tables 13 and 14. All entries in this comparison are absolute discounted returns, so the discrepancy is not caused by choosing a new normalization denominator. The retained replay records also report normalized returns using the original archived finite-grid reference return.

Median-return examples. The rule for Figure 14 was fixed before these replays. For each initial state n, let Δ_n be the two-bank discounted return minus the single-agent discounted return. We select

Environment	Model	Archive R	CPU replay R	Difference
MountainCar	Single-agent	0.339367	0.342030	+0.002663
	Two-bank	0.375849	0.375812	-0.000037
Pendulum	Single-agent	0.788882	0.788937	+0.000055
	Two-bank	0.790436	0.790396	-0.000040
Reacher	Single-agent	0.465133	0.465151	+0.000018
	Two-bank	0.467563	0.479549	+0.011986

Table 15: **CPU replay and archived single-model returns.** Each mean uses all fixed initial states for one task and model seed: MountainCar Gentle car (WM10), Pendulum Upright and still (WM10), and Reacher Target angle (WM13), conditional on Q25. Difference is CPU minus archive. These are absolute discounted returns, not success rates. The archive columns retain the original recorded values.

the state minimizing $|\Delta_n - \mathrm{median}_j\, \Delta_j|$, breaking ties by the smallest zero-based state index. This rule uses the entire fixed cohort and does not select the largest gain or the largest visual separation. It gives an illustrative pair close to the cohort's median paired return difference; it is not a claim that its poses are representative of the full state distribution.

Within a selected pair, both methods are displayed at the earliest successful terminal step reached by either method; if neither succeeds, the shared time is the episode horizon. The two scenes use the same initial state, physical geometry, and display scale. Any earlier terminal pose is explicitly marked as held. The median-example figure therefore compares the same time rather than different completion times. MountainCar selects start 1: the single-agent policy reaches the goal at step 105, and the two-bank policy reaches the goal at step 104. The shared display time is $t = 104$, and the paired discounted-return difference is $+0.003552$.

Pendulum selects start 0: the single-agent policy reaches the goal at step 29, and the two-bank policy reaches the goal at step 29. The shared display time is $t = 29$, and the paired discounted-return difference is 0.000000.

Reacher selects start 0: the single-agent policy times out at step 100, and the two-bank policy times out at step 100. The shared display time is $t = 100$, and the paired discounted-return difference is 0.000000.

Outcomes over all retained starts. Table 16 reports every initial state's first terminal category or timeout. Each state contributes exactly one outcome. The two-bank policy reaches the goal from 512 rather than 507 MountainCar starts and from 208 rather than 203 Reacher starts. Both Pendulum policies succeed from all 512 starts. These counts describe executions of one fixed policy per condition and task; they are not counts of independently trained agents. Success denotes first entry into the configured goal, not sustained balance or stability. In MountainCar, exceeding the speed bound takes precedence over a simultaneous goal entry. In Reacher, the angle-coordinate predicate is evaluated directly on the stored wrapped angles, as in the original task.

Environment	Model	Goal	Unsafe	Timeout
MountainCar	Single-agent	507	1	4
	Two-bank	512	0	0
Pendulum	Single-agent	512	0	0
	Two-bank	512	0	0
Reacher	Single-agent	203	0	53
	Two-bank	208	0	48

Table 16: **Complete reset-cohort outcomes in the new CPU replays.** The tasks and model seeds match Table 15. Counts sum to 512 per method in MountainCar and Pendulum and to 256 in Reacher. Every reset state is retained, including unsuccessful executions.

Task processes and fixed initial states. Figures 15–17 show the selected pairs on a common time axis, together with the state quantities used by the task. The positions are actual saved simulator states,



Figure 14: **Task execution for the original median-return examples.** Single-agent and Two-bank policies execute from paired simulator starts at one shared time and physical scale within each column. The selection rule was fixed before replay: choose the paired return difference nearest the full-cohort median, with ties resolved by start index. The car reaches its goal at steps 105 and 104; both pendulum policies succeed at step 29; both Reacher policies time out at step 100. The display time is the pair's first target arrival or the horizon if neither succeeds. These original cases and their process and fixed-start views are retained alongside the post hoc benefit examples in Figure 4.

with no fitted trajectories or interpolation of physical states. To expose cases beyond the selected pair, Figures 18–20 also display the first four prescribed starts, indices 0–3, without replacing any start based on its outcome. In these common-time views, goal and timeout labels identify completed episodes; a running label denotes an episode that has not terminated at the displayed time.

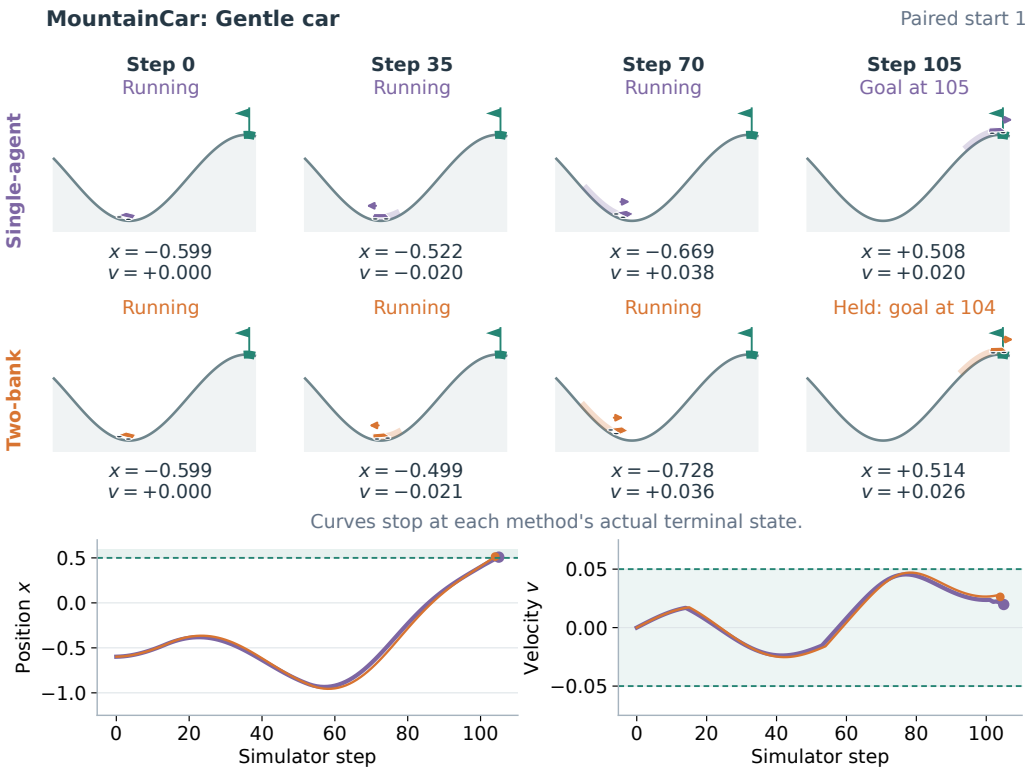


Figure 15: **Executing the speed-constrained MountainCar task.** The selected pair in Figure 14 is shown at shared time points, with position and velocity diagnostics. Gentle car requires $|x-0.6| < 0.1$ and terminates unsuccessfully whenever $|v| > 0.05$. The final pose of a terminated episode may be held in the scene; its diagnostic curve stops at the terminal event.

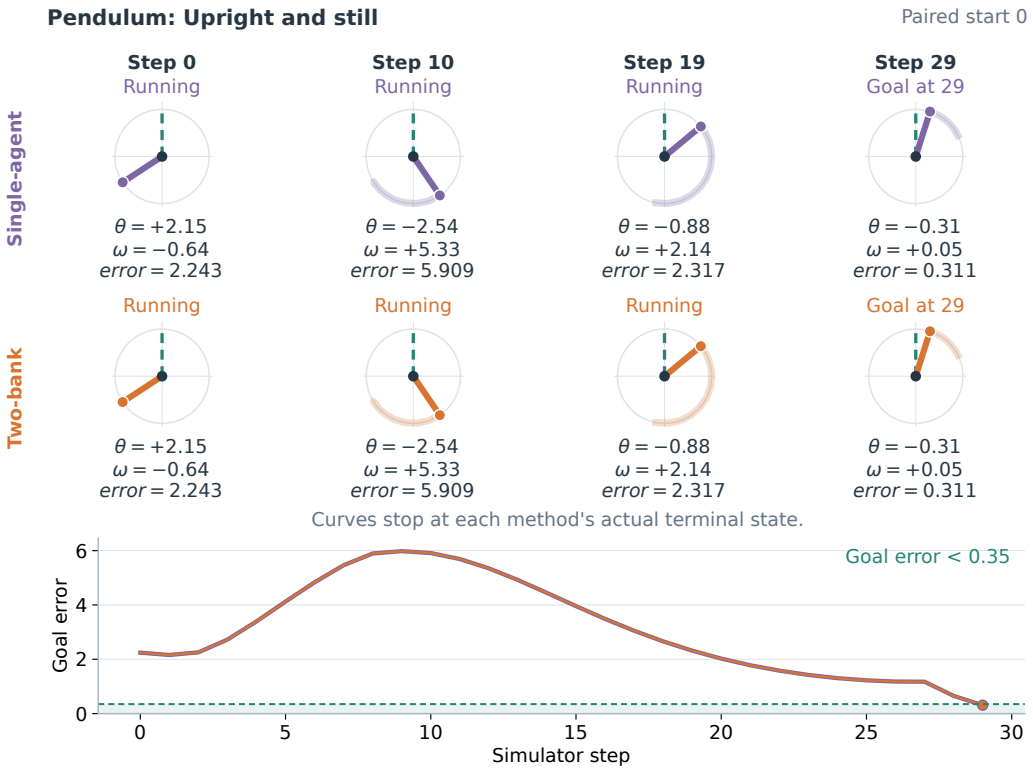


Figure 16: **Entering the upright, low-speed Pendulum goal.** Common-time scenes and angle–velocity diagnostics show the selected pair. Success requires $\sqrt{\theta^2 + \omega^2} < 0.35$ under the original coordinate convention. A vertical-looking pose alone does not establish success. Curves end at first termination; a held terminal scene does not indicate continued balancing.

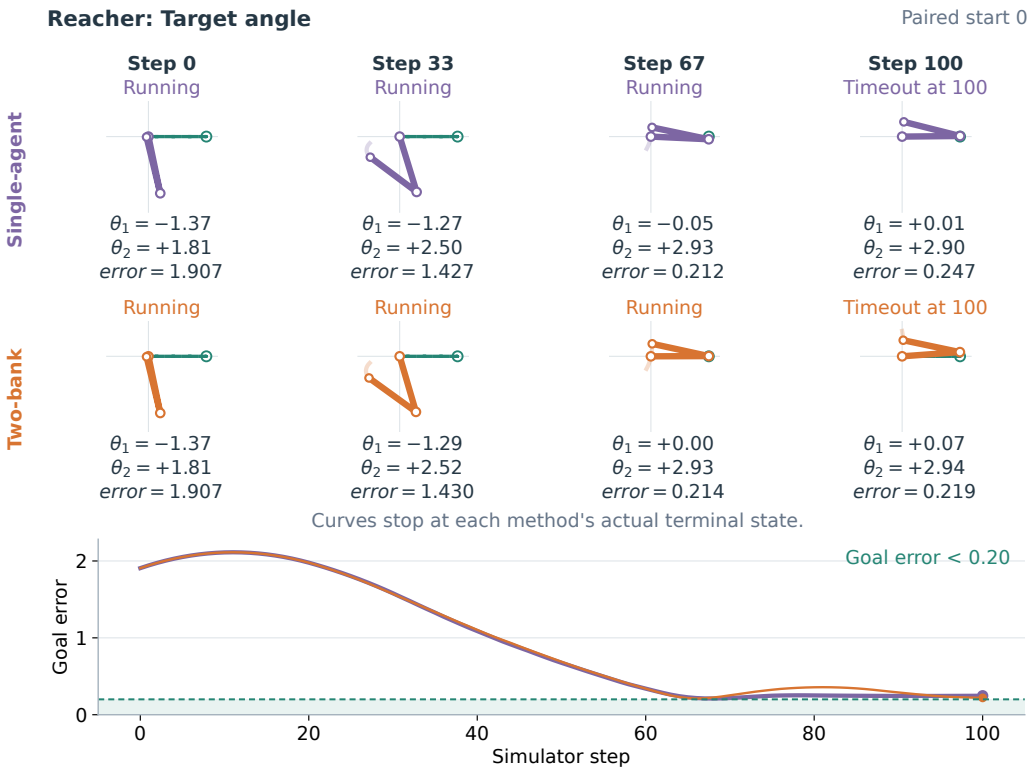


Figure 17: **Executing the Reacher joint-angle task.** Common-time scenes and the original angle-coordinate error show the selected Reacher pair. The target is $(\theta_1, \theta_2) = (0, \pi)$, with direct coordinate error below 0.20. Dashed geometry indicates the target configuration. Fingertip proximity and wrapped shortest angular distance are not substituted for this predicate. Both selected episodes time out; this view shows their approach to the goal boundary.

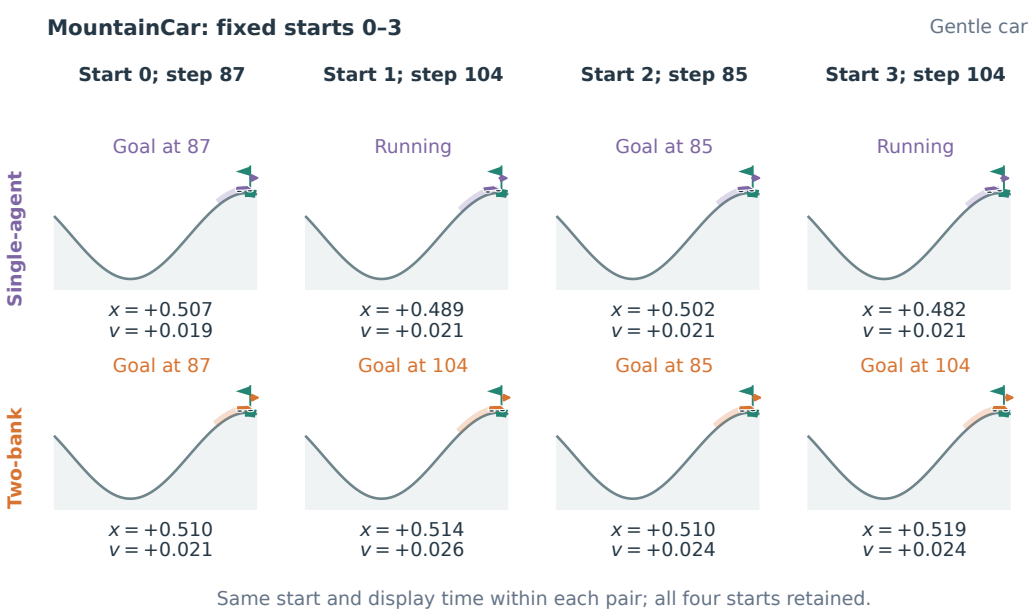


Figure 18: **MountainCar from the first four prescribed starts.** Both fixed models are shown for reset indices 0–3 under Gentle car. Each column compares the same start and time, using the synchronized display rule above. Outcome and terminal-step annotations retain the status at that time, with all four prescribed starts retained.

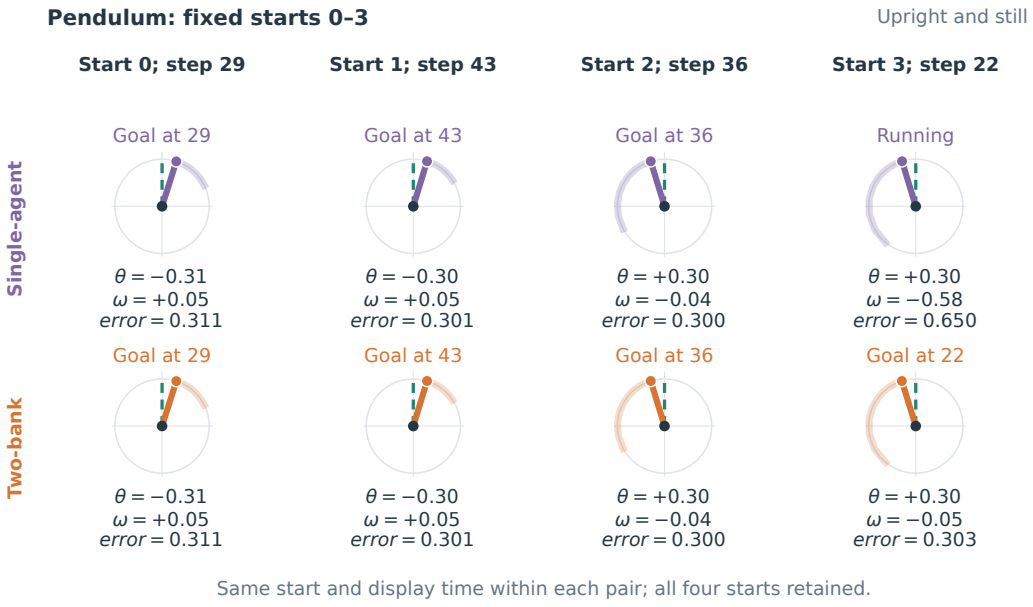


Figure 19: **Pendulum from the first four prescribed starts.** Both fixed models are shown for reset indices 0–3 under Upright and still, without selecting starts by outcome or arrival-time difference. Each pair uses one common display time and preserves first-arrival termination. A held successful pose does not measure sustained control.

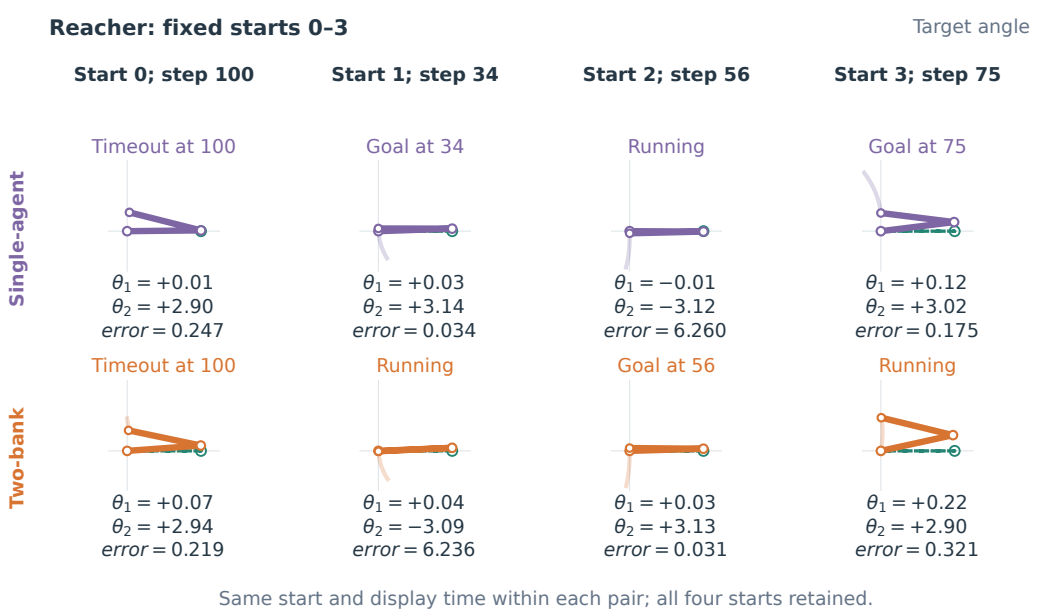


Figure 20: **Reacher from the first four prescribed starts.** Both fixed models are shown for reset indices 0–3 under Target angle, with the same target configuration and direct angle-coordinate threshold for every pair. Terminal categories and steps are retained; no unsuccessful start is replaced to improve the visual comparison. Physically similar poses can have large coordinate errors across the $-\pi/\pi$ boundary; the original task predicate is retained.

I.5 Illustrative improvements and complete paired comparisons

Figure 4 illustrates two ways in which the recovered two-bank model can improve task execution: reaching a goal that the single-agent policy misses within the horizon, and reaching a goal earlier when both policies succeed. These examples are drawn from the same saved simulator replays described in Appendix I.4; no policy, world model, initial state, task threshold, or simulator observation was changed for this figure. The original full-cohort median examples remain in Figure 14.

Post hoc example selection. After inspecting the completed replays, we chose a benefit class for each task. For MountainCar and Reacher, the class contains every start at which the two-bank policy reaches the goal and the single-agent policy does not. For Pendulum, where both policies succeed from every start, the class contains starts where the two-bank policy reaches the goal earlier. The latter comparison requires both policies to succeed, so it does not treat a timeout as an observed arrival time.

Within the selected class $\mathcal{C}$, we choose the start minimizing $|\Delta_n - \mathrm{median}_{j\in\mathcal{C}}\,\Delta_j|$, where $\Delta_n = R_{\mathrm{two},n} - R_{\mathrm{single},n}$ is the absolute discounted-return difference. Ties use the smallest zero-based start index. This is a post hoc illustration rule; it is distinct from the rule fixed before replay for Figure 14. The selected cases describe benefits that occurred in the saved cohort, rather than a typical episode or the frequency of improvement. We therefore report each class's size and the complete paired-return distribution below.

The three illustrated cases. In MountainCar, start 57 shows the two-bank policy reaching the hilltop at step 102 within the speed limit, while the single-agent policy times out at step 200. The benefit class contains 5 of 512 starts: four single-agent timeouts and one unsafe termination become goal arrivals under the two-bank policy.

In Pendulum, start 116 shows first arrival at step 23 for the two-bank policy and step 24 for the single-agent policy. This earlier-arrival class contains 100 of 512 starts. Both policies succeed from all 512 starts; the two-bank policy arrives at the same step in 393 pairs and later in 19 pairs.

In Reacher, start 80 shows the two-bank policy entering the angle target at step 59, while the single-agent policy times out at step 100. Their direct angle-coordinate errors at the displayed time are 0.193 and 0.250, respectively, against the unchanged threshold of 0.20. The illustrated benefit class contains 16 of 256 starts. The reverse outcome also occurs: on 11 starts, only the single-agent policy reaches the goal. These paired counts account for the net change from 203 to 208 Reacher goal arrivals in Table 16.

Each main-figure pair uses the same initial state, physical scale, and simulator time. The common display time is the two-bank policy's first goal arrival in the selected pair. The annotations distinguish the state at that time from the single-agent policy's later recorded outcome. All angles, velocities, rewards, and termination predicates are unchanged. In particular, Pendulum success is first arrival in the low-speed upright region, and Reacher success uses the original direct angle-coordinate error rather than fingertip distance.

All-start paired returns. Table 17 includes every fixed reset state. We classify a two-bank win when $\Delta_n > 10^{-6}$, a tie when $|\Delta_n| \leq 10^{-6}$, and a loss when $\Delta_n < -10^{-6}$. Returns use the original 0.99 discount and first-terminal reward convention. These are within-task comparisons of the same two fixed policies, not independent model-fitting trials or a new multi-seed performance table.

Environment	Starts	Wins	Ties	Losses	Mean Δ	Median Δ
MountainCar	512	409	103	0	+0.033782	0.003552
Pendulum	512	100	393	19	+0.001459	0.000000
Reacher	256	114	89	53	+0.014397	0.000000

Table 17: **All paired outcomes behind the benefit illustrations.** Wins, ties, and losses compare Two-bank with Single-agent absolute discounted returns, using tolerance 10^{-6}. Tasks and model seeds match the unchanged CPU replay tables: MountainCar Gentle car (WM10), Pendulum Upright and still (WM10), and Reacher Target angle (WM13), all conditional on Q25. Mean and median use all starts, including zero-return and negative-difference pairs.

Figure 21 places each selected case within its full paired-return distribution. This makes the difference between a benefit illustration and cohort performance explicit. The original outcome counts, CPU–archive discrepancies, median examples, and fixed starts 0–3 in Appendix I.4 remain part of the reported evidence.

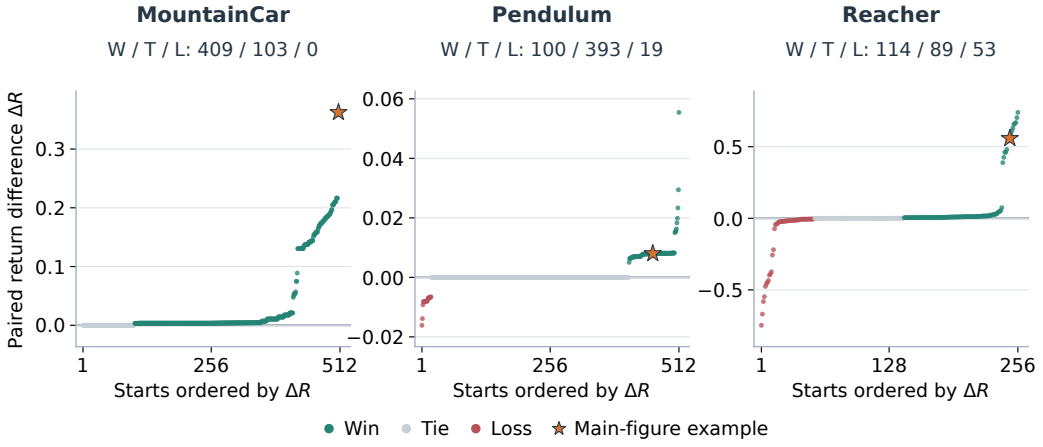


Figure 21: **Where the selected improvements lie among all starts.** Each panel plots every fixed start's absolute discounted-return difference, Two-bank minus Single-agent, in ascending order. The horizontal reference marks zero; a star identifies the corresponding post hoc example in Figure 4. Counts use the 10^{-6} tolerance in Table 17. Each point is an actual saved pair, with no smoothing or fitted trajectory. Starts share the same fitted model pair within each environment and are not independent training replications.

J CONTROLLED-STUDY CONSTRUCTION AND NUMERICS

Learned-Q acquisition study. Each seed selects four distinct source goals from the sorted eligible state set using `default_rng(12000+seed)`; the first two form the anchor. The excluded cells are the four corners $(1,1),(1,9),(9,1),(9,9)$ and the four doors $(3,5),(5,3),(5,7),(7,5)$. Source rewards are goal-indicator arrival rewards, with termination at the corresponding goal. All Q tables start at zero. A synchronous sweep draws one successor per state–action pair and shares that draw across tasks within a bank. The update uses the preceding Q table and the learning rate

$$\alpha_t = \max\{0.01,\, 0.2 - 0.19t/19999\}, \qquad t = 0, 1, \ldots.$$

Thus continuation beyond 20,000 sweeps retains learning rate 0.01. All source rows, including currently terminal states, are updated; only successor termination masks the continuation term. Anchor, supplement, and pooled streams use seeds $40000+10z$, $40000+10z+1$, and $40000+10z+2$. The same-goal and new-goal supplements share sampling randomness, while remaining independent of the anchor stream.

Controlled-Q study. The second control computes optimal Q maps for four prescribed, continuing source tasks and perturbs their complete arrays. Its source values are numerical fixed-point solutions rather than learned checkpoints. It isolates task measurements and Q errors; its six realizations are not additional acquisition seeds. Appendix M gives the construction and accuracy checks.

J.1 RECOVERY AND PLANNING CONFIGURATIONS

Both controls use one reconstructed Windy kernel, $\gamma = 0.99$, and fixed candidate successors consisting of the current state and all legal cardinal neighbors. Every action at a state uses that same candidate set. Rows are fitted on the stochastic simplex with the unweighted sum of absolute Bellman residuals. No residual normalization, regularization, or data-dependent bank weighting is added. After fitting, the rows form a single transition kernel.

The acquisition study uses SciPy's HiGHS dual-simplex solver with primal and dual feasibility tolerances 10^{-9} and one solver thread. It checks success, returned mass and nonnegativity, and agreement between the original residual and the reported objective within 10^{-7}. Three prespecified rows are cross-checked by interior point. A post-fit check compares the objective with that at the true row; this uses the known kernel for numerical validation and does not provide fitted probabilities. Returned rows are not clipped or renormalized. This validation is not an exact objective-gap certificate.

Planning initializes the value at zero, iterates the recovered-model Bellman operator, accepts the last update at successive-value difference below 10^{-12}, and recomputes action backups before choosing the greedy policy. Its iteration cap is 20,000 and the final Bellman residual is checked against 1.01×10^{-12}. Ties take the first action in the specified action order. The controlled-Q study applies its separately recorded numerical polishing and checks in Appendix M.

K METRICS, AGGREGATION, AND STATISTICAL UNITS

For the completed Windy controls, transition MSE averages over the entire $68 \times 4 \times 68$ tensor, including entries outside the candidate support:

$$E_P = \frac{1}{|\mathcal{S}|^2|\mathcal{A}|} \sum_{s,a,s'} \big(\widehat{P}(s' \mid s,a) - P^\star(s' \mid s,a)\big)^2. \tag{K.1}$$

A second test task does not provide another independent measurement of this same model error. Continuous-coordinate successor MSE has different units and is not pooled with this probability-kernel MSE.

Return and its reference. The two completed controls evaluate the same two held-out tasks: goal $(1,9)$ with unsafe $(5,7)$, and goal $(1,1)$ with unsafe $(5,7),(7,5)$. Goal arrival pays 1; unsafe arrival pays 0; both terminate. For a stationary policy π, exact true-kernel recursion starts from $V_0^\pi = 0$ and runs for $H = 200$ transitions:

$$V_{h+1}^\pi(s) = \sum_{s'} P^\star(s' \mid s, \pi(s)) \left[r(s') + \gamma(1 - d(s'))V_h^\pi(s')\right]. \tag{K.2}$$

The score $J_H(\pi)$ is the uniform mean of V_H^π over the task's nonterminal initial states. The normalized return is $R_j(\widehat{\pi}) = J_{H,j}(\widehat{\pi})/J_{H,j}(\pi_j^{\mathrm{ref}})$, where π_j^{ref} is the true-kernel infinite-horizon optimal stationary policy evaluated for the same 200 steps. The two denominators are approximately 0.7640320003 and 0.6348794480. They are not asserted to be finite-horizon optimal denominators. Ratios are not clipped. Each model's return is $\overline{R} = (R_1 + R_2)/2$; aggregation across realizations occurs after this within-model average.

Fixed-reference prediction. For task j, the true-model infinite-horizon optimal value fixes

$$\psi_j(s') = r_j(s') + \gamma(1 - d_j(s'))V_j^\star(s').$$

The mean absolute backup discrepancy is

$$E_{\mathrm{backup}} = \frac{1}{2|\mathcal{S}||\mathcal{A}|} \sum_{j=1}^{2} \sum_{s,a} \left| \sum_{s'} \big(\widehat{P} - P^\star\big)(s' \mid s, a)\psi_j(s') \right|. \tag{K.3}$$

The same reference functions are used across conditions. They enter offline evaluation, not recovery. This average diagnostic examines specified prediction errors relevant to the Method analysis; it is not the uniform error required by a policy-performance theorem.

Repeated observations. Acquisition seeds 701–706 vary source goals and sampling streams within one layout and one Windy kernel. Controlled-Q seeds 601–606 vary reward/noise realizations within that same fixed kernel. Both held-out tasks and all state–action rows stay nested within their fitted model. Neither is treated as an independent replicate. Tables report arithmetic means across six realizations; paired comparisons retain matching seed indices. The main plots show the individual observations and means without significance marks. The three model-fitting seeds per continuous environment are conditional on its fixed pretrained agents, as detailed in Appendix I. No confidence intervals or significance tests are inferred from nested task observations.

L ACQUISITION-BUDGET CONTROLS

The acquisition study isolates the contribution of new source tasks from repeated estimation at matched cost. For each of six seeds, the anchor has two goals. Same-goal addition relearns those two goals in another bank; new-goal addition learns the other two selected goals. Longer training doubles the anchor sweeps without adding records. Pooled conditions train all four goals together, sharing each successor sample across their Q updates. Recovery and evaluation are otherwise common across conditions.

Table 18: Acquisition accounting at base checkpoint $T = 20{,}000$. Counts follow the executed synchronous schedule. Draws and scalar updates are different resources; neither is a wall-clock measurement. Draw and update counts scale linearly with the base checkpoint.

Condition	Records	Distinct goals	Draws (M)	Updates (M)
Anchor	2	2	5.44	10.88
Same goals	4	2	10.88	21.76
New goals	4	4	10.88	21.76
Longer training	2	2	10.88	21.76
Pooled (updates)	4	4	5.44	21.76
Pooled (draws)	4	4	10.88	43.52

At base checkpoint T, one bank uses $272T$ successor draws. A bank with K tasks uses $272KT$ scalar Q updates, since the same draw updates every task. Same and New each train two two-goal banks for T sweeps. Longer training uses one two-goal bank for $2T$ sweeps; Pooled (updates) uses one four-goal bank for T, and Pooled (draws) uses that bank for $2T$. Thus the pooled conditions match different budgets. Recovery costs, query counts, and planning costs should not be silently equated with these acquisition counts.

Section 4.3 compares Same goals, New goals, and Longer training across the three base checkpoints; Figure 22 plots all six paired seed trajectories and their means at these recorded budgets. Table 19

retains all six conditions at every checkpoint, giving 108 fitted-model records. Each record contains both task returns, full-kernel MSE, backup errors, and solver diagnostics. Within each condition and acquisition seed, the three checkpoints share a training run; they are not additional independent replicates. The six paired observations for the final-checkpoint Same/New contrast appear in Table 20.

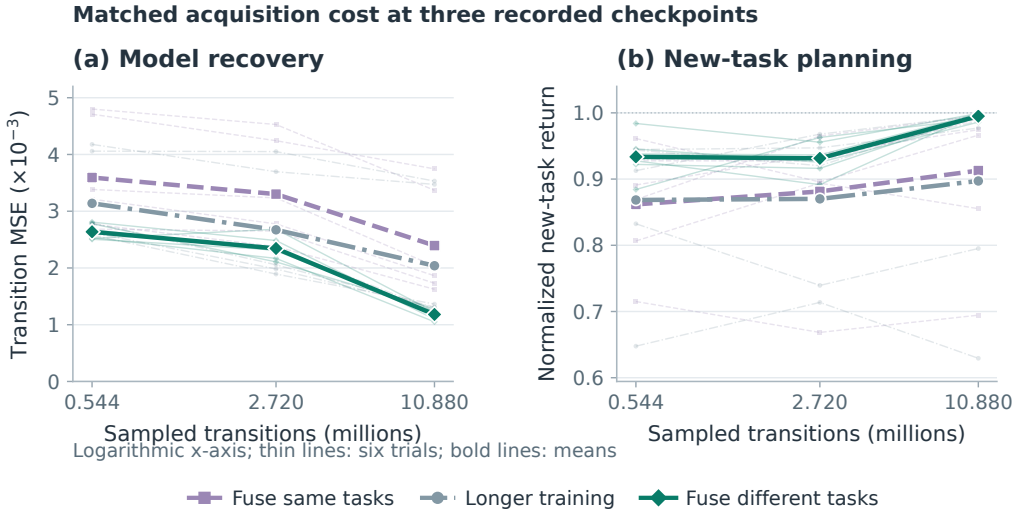


Figure 22: **Task information across three recorded acquisition budgets.** Thin lines show all six acquisition seeds and thick lines show means for Same goals, Longer training and New goals. The horizontal axis shows total sampled transitions on a logarithmic scale, at 0.544, 2.720 and 10.880 million. At each point all three conditions match samples and scalar Q updates. Same and New have four Q records, whereas Longer retains two. Left: full-kernel transition MSE. Right: normalized return, averaging the two new tasks within each fitted model. Connected observations from the same seed share a training run; they are not independent replications. The lines join recorded checkpoints without smoothing.

Table 19: All acquisition-study checkpoints, averaged over six seeds. MSE and backup error are multiplied by 10^3 for display. Return first averages the two held-out tasks within each model.

Base T	Condition	Return	MSE $\times 10^3$	Backup $\times 10^3$
1,000	Anchor	0.824775	3.198448	8.561584
	Same goals	0.861763	3.593602	9.013937
	New goals	0.933568	2.637245	8.567013
	Longer training	0.868436	3.138780	8.782121
	Pooled (updates)	0.917019	2.161337	7.814414
	Pooled (draws)	0.929258	2.322582	8.092994
5,000	Anchor	0.884298	3.002479	8.946848
	Same goals	0.881342	3.301501	8.402557
	New goals	0.931344	2.343661	7.347871
	Longer training	0.870107	2.672266	8.185746
	Pooled (updates)	0.949214	2.009442	7.431868
	Pooled (draws)	0.969421	1.767260	6.011544
20,000	Anchor	0.935549	2.063943	4.363012
	Same goals	0.912824	2.394571	5.162437
	New goals	0.994997	1.181014	2.740769
	Longer training	0.897126	2.039367	4.237114
	Pooled (updates)	0.998408	0.885467	2.233612
	Pooled (draws)	0.997563	0.878494	1.887948

Table 20: The primary matched-budget contrast at $T = 20{,}000$. Every acquisition seed is shown, including the return decrease for seed 701. S and N denote Same goals and New goals, respectively.

Seed	Return		MSE $\times 10^3$		Backup $\times 10^3$	
	S	N	S	N	S	N
701	0.995217	0.990933	1.624778	1.207799	4.027108	3.819978
702	0.974527	0.998788	2.034094	1.051001	4.715599	2.403820
703	0.855299	0.985967	3.746690	1.279946	8.517030	2.919396
704	0.694457	0.996326	3.368497	1.099143	7.644715	2.539006
705	0.991115	0.998872	1.730305	1.222967	3.410910	2.300648
706	0.966331	0.999095	1.863060	1.225231	2.659264	2.461767

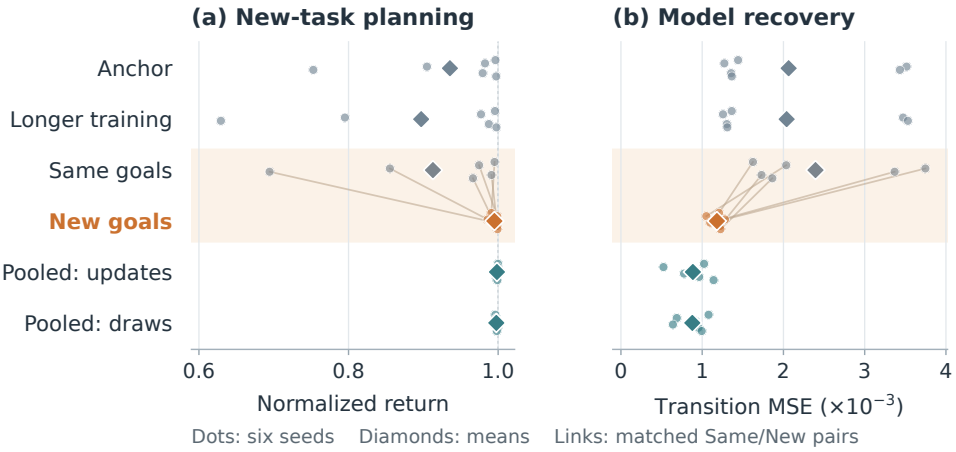


Figure 23: **All acquisition controls at the final checkpoint.** This expands the focused comparison in Figure 6. Same goals and New goals are Fuse same tasks and Fuse different tasks in the main text; Anchor is the Original bank. Longer training keeps the original goals, while pooled conditions train all four goals together. Dots show all six acquisition trials; diamonds show means. Pooled conditions match either updates or samples, as specified in Table 18.

At the final checkpoint, New improves return in five of six pairs, with mean gain 0.082173, and reduces transition MSE in all six. Its mean MSE is 1.181014×10^{-3} versus 2.394571×10^{-3} for Same. The pooled controls achieve still lower MSE in every paired seed; update-matched pooling reaches mean return 0.998408 with half the draws used by New. These comparisons support the contribution of distinct task information under this protocol. They do not establish that acquiring separate banks is better than training the task union together. Intermediate checkpoints retain nonmonotone outcomes rather than imposing a monotone learning curve.

M Controlled Q-value perturbations

The controlled study fixes the Windy kernel, four source records, reward sup-norm 1, $\gamma = 0.99$, and candidate supports. All source termination indicators are zero. For each realization, four independent uniform $[-1, 1]$ reward fields are individually normalized to sup-norm 1. Let the first be u and the remaining three be h_i. The source rewards are

$$\begin{aligned} &\text{Repeated:} && u,\ u,\ u,\ u, \\ &\text{Near:} && u,\ (u + 0.05h_i)/\|u + 0.05h_i\|_\infty, \quad i = 1, 2, 3, \\ &\text{Distinct:} && u,\ h_1,\ h_2,\ h_3. \end{aligned} \tag{M.1}$$

Near and Distinct correspond to Similar tasks and Different tasks in the main text. The last family is stored as `independent` in the data. Reward generation initializes its seed sequence with $[z, 11]$, where z is the study seed. These continuing reward fields differ from the sparse goal tasks of the acquisition study.

Perturbing both sides of the inverse problem. Optimal Q maps are computed before adding the perturbation

$$\widehat{Q}_i(s,a) = Q_i^\star(s,a) + 100\sigma\,\Xi_i(s,a), \qquad \sigma \in \{0, 10^{-5}, 10^{-4}, 10^{-3}\}, \tag{M.2}$$

where $100 = 1/(1-\gamma)$ and Ξ has independent standard-normal entries over the complete $4 \times 68 \times 4$ array. The actual entrywise noise standard deviations are therefore 0, 0.001, 0.01, and 0.1. The same noise array, generated by `SeedSequence([seed,29])`, is reused across families and scales. Current values and successor maxima are both recomputed from the perturbed Q maps. There is no Q clipping, common bias, or additional correlated-noise term. The common anchor is cached bitwise across families, and all exact repeated records copy the same cached Q map. Consequently, the perturbation affects both the response and the measurement matrix in the Method's effective-error identity.

Six realizations, three reward families, and four noise levels give 72 fits. Each fit solves all 272 transition rows and evaluates the same two held-out tasks. Table 21 reports all 12 condition means without excluding a realization.

Table 21: Controlled-Q results, means over six reward/noise realizations. The second column is the actual entrywise Q-noise standard deviation 100σ. MSE and backup errors are unscaled. Zero-noise numerical residuals are retained instead of rounded to exact identification.

Family	Q-noise SD	Transition MSE	Backup error	Return
Repeated	0	0.005551	0.011627	0.491989
Repeated	0.001	0.004973	0.011050	0.462318
Repeated	0.01	0.004993	0.011102	0.473310
Repeated	0.1	0.005110	0.011533	0.520631
Near	0	3.271×10^{-27}	3.627×10^{-15}	1.000000
Near	0.001	6.394×10^{-5}	0.000608	0.998446
Near	0.01	0.001120	0.003825	0.956659
Near	0.1	0.004202	0.009611	0.675355
Distinct	0	1.418×10^{-29}	1.252×10^{-15}	1.000000
Distinct	0.001	1.888×10^{-6}	2.761×10^{-5}	1.000000
Distinct	0.01	2.287×10^{-5}	0.000269	0.999760
Distinct	0.1	0.000564	0.002366	0.990092

Near and Distinct both recover the local distributions to numerical precision at zero perturbation. Under the largest perturbation, their mean returns are 0.675355 and 0.990092, respectively. Repeated remains poorly identified, and its return varies nonmonotonically with noise. These patterns support the role of measurement geometry under the specified perturbation, not a universal model of errors produced by Q learning.

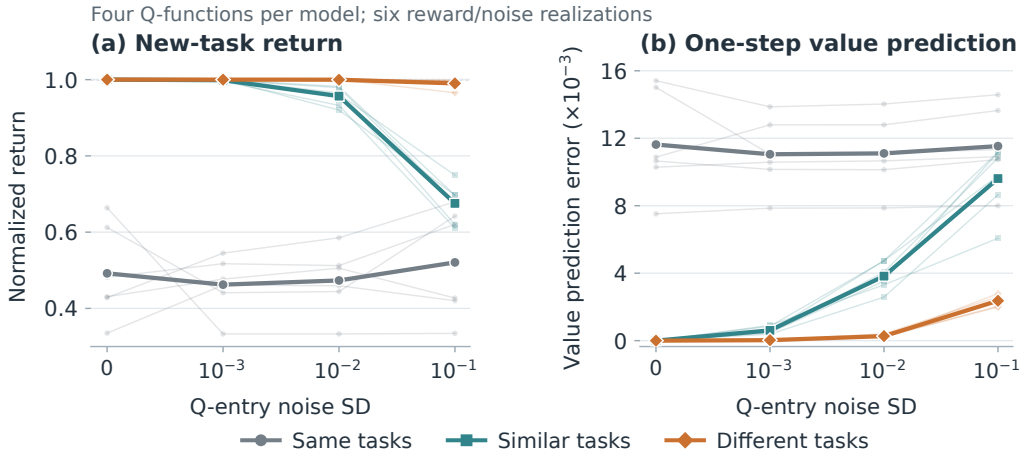


Figure 24: **Planning and fixed-reference value prediction under Q errors.** The conditions and observations match Figure 7. Same tasks denotes Repeated tasks in the main text. The right panel reports the mean absolute one-step backup discrepancy defined in Equation equation K.3, using common true-model reference values. Thin lines show all six reward/noise realizations and bold lines show means. This is an average prediction diagnostic, not the uniform error required for a policy-performance bound.

M.1 Geometry and numerical checks

For a support of size m, let $U \in \mathbb{R}^{m\times(m-1)}$ have orthonormal columns spanning $\mathbf{1}^{\perp}$. The study retains the complete singular spectrum of $\widehat{\Phi}U/\sqrt{4}$, including all null directions. Rank uses absolute tolerance 10^{-10} and relative tolerance 10^{-9}; the sensitivity diagnostic is zero when rank deficient. It is an L2 diagnostic, not the coefficient factor or the L1 statement in Equation equation 5. Independent perturbations can make the repeated family's noisy matrices numerically full rank while its recovered predictions remain poor. Noisy rank alone therefore does not establish useful new task information.

Source dynamic programming uses fixed-point tolerance 10^{-11} and a 20,000-iteration limit, followed by linear policy evaluation to polish the greedy policy and reconstruct Q. The optimal Bellman residual is checked against 2×10^{-11}. The LP uses HiGHS dual simplex with 10^{-9} primal and dual tolerances. For numerical stability, each task row subtracts its first candidate coefficient from both the measurement row and its response; on the simplex this preserves every residual exactly. Final checks use the original, uncentered coefficients. Every 17th row is independently compared with an interior-point solve, giving 19,584 primary solves and 1,152 cross-checks across the 72 fits.

The acceptance checks require mass error below 10^{-8} and minimum probability at least -10^{-10}. Epigraph and dual violations must be below 10^{-7}; the objective gap and the cross-solver difference must each be below 10^{-7} times one plus the objective. All final conditions passed. Worst recorded mass error was 2.19×10^{-11} and minimum probability was -8.73×10^{-12}. The largest original epigraph violation, primal–dual gap, and cross-solver difference were 1.49×10^{-9}, 2.98×10^{-9}, and 2.18×10^{-10}, respectively. Returned probabilities were retained without clipping or renormalization. Earlier rejected numerical attempts remain archived; equivalent centering and dynamic-programming polishing were applied to the complete rerun, without changing tasks, noise, seeds, or acceptance thresholds.

The resulting floating-point checks establish numerical consistency for this study. They are not exact certificates of the Method's objective-gap parameter and do not extend its finite-state bound to a nonlinear successor model. The experiment holds support correctness fixed and does not test misspecified supports or new environment layouts.

Reproduction records. The accompanying data include the nine FourRooms confirmation trials, the continuous single-model evaluation records, all 108 acquisition fits across three checkpoints, and all 72 perturbation fits. Plotting and table construction retain task pairing and use the recorded

unnormalized returns to verify normalization. Statistical figures are vector renderings of these records; spatial diagnostics retain the archived full-resolution maps and their stated instance-selection rules. No missing outcomes are interpolated.