HIERARCHICAL DISTANCE DECOMPOSITION FOR MINIMUM-ACTION-DISTANCE ESTIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Estimating how many actions an agent needs to move between distant states is difficult. The Minimum Action Distance (MAD) formalizes this quantity, but approximations learned from experience can be locally accurate yet unreliable over long horizons. We introduce *hierarchical distance decomposition*, a post-hoc method that improves long-range estimates by composing local predictions through a sparse graph of landmark states. It can be applied to any existing learned distance model without retraining. We present theoretical results that bound errors in the resulting distance estimates and establish order preservation for sufficiently separated state pairs. Empirically, the method improves both long-range distance estimation accuracy and downstream planning performance.

1 INTRODUCTION

For an autonomous agent, knowing the distance between two states can support many aspects of learning and decision-making, including exploration, goal-directed control, planning, reward shaping, and subgoal selection. One particularly direct distance measure is the *Minimum Action Distance* (MAD), defined as the minimum number of actions required for an agent to transition from one state to another. In deterministic environments, the MAD is the shortest-path distance in the state transition graph. In stochastic environments, it is the shortest-path distance in the support of the transition function, and acts as a lower bound on the expected number of actions required to reach a target state.

In practice, computing the MAD directly is generally impractical, so it must be learned from experience. Existing approaches learn a distance function, often defined as a quasimetric, over a learned state representation. A key limitation of existing approaches is that such approximations can produce accurate distance estimates for nearby state pairs but become unreliable over long horizons, primarily due to error accumulation, limited representation capacity, and distributional shifts.

We propose *hierarchical distance decomposition*, a method for estimating long-range distances by composing short-range predictions from an existing learned distance model. The method selects a sparse set of *landmark states* from a dataset of experience and builds a directed graph whose edges connect pairs of landmarks that the model estimates to be close. For a state pair (s, g), it connects s and g to nearby landmarks and estimates their distance using a shortest path through the landmark graph. In this way, the method replaces a potentially unreliable long-range prediction with a sequence of more reliable local predictions. The method is agnostic to the choice of base learned distance model, is applied post hoc, and requires no retraining of the base model or underlying representation.

Our contributions are: (1) a hierarchical estimator that uses only short-range evaluations of an existing learned distance model to answer long-range queries (Section 4); (2) theoretical results that conditionally bound under- and over-estimation and establish an order-preservation result for sufficiently separated state pairs (Section 5); and (3) empirical results showing improvements in both long-range distance estimation and downstream planning performance when applied across multiple base distance models, environments, and experience datasets (Section 6).

2 RELATED WORK

Temporal distance learning. Many reinforcement learning methods use notions of temporal distance that reflect how long it may take an agent to transition from one state to another. Early work

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.

explored the connection between goal-conditioned value functions and the cost of reaching different goals (Kaelbling, 1993), while later work defined temporal distances more explicitly. Hartikainen et al. (2020) learn a policy-dependent temporal distance based on the *expected* number of decision stages required to reach one state from another, while Steccanella & Jonsson (2022) alternatively define the Minimum Action Distance (MAD) as the *minimum* number of actions required to do so.

Several earlier methods embed states into a learned latent space where Euclidean distance captures temporal distance (Steccanella & Jonsson, 2022; Park et al., 2024a;b). However, Euclidean distances are symmetric, whereas temporal distances such as the MAD may not be. To capture this asymmetry, more recent work has used quasimetrics to represent temporal distance functions. Wang et al. (2023) exploit the quasimetric structure of optimal goal-conditioned value functions, while Steccanella et al. (2026) use quasimetric representations to directly approximate the MAD from offline state trajectories. Related quasimetric methods recover optimal goal-conditioned policies from suboptimal and stochastic offline data (Myers et al., 2026) and use multi-step updates to better combine local and global distance information (Zheng et al., 2026).

Graph-based planning. Graph search has been used to compose local information over longer horizons, often to support long-horizon control. Eysenbach et al. (2019) construct a graph over observed states using distances derived from a goal-conditioned value function, and use shortest paths through this graph to select intermediate subgoals. Later work extends this idea in several ways: Baek et al. (2025) cluster states from offline trajectories using a learned representation that captures temporal distance, and search the resulting graph to select subgoals; Opryshko et al. (2026) bias graph search towards paths composed of more reliable local transitions, retaining longer-distance edges only when necessary; and Kobanda et al. (2026) construct a landmark graph using a learned distance function and update it online when observed traversal times disagree with its predictions.

The closest prior methods to ours use graphs to directly improve learned distance or value estimates. Yang et al. (2020) use shortest paths over a graph built using learned local distances to generate long-horizon distance targets, and distill these into a global distance representation. Similarly, Bagatella & Martius (2023) aggregate shorter-horizon estimates along shortest paths through a graph to replace unreliable long-horizon estimates for model-based planning.

Hierarchical reinforcement learning. A related line of work uses hierarchical reinforcement learning to decompose long-horizon tasks, often using learned distances or values to structure their hierarchies. These methods variously learn higher- and lower-level control policies (Park et al., 2023; Canesse et al., 2025; Wibault et al., 2026), temporally abstract value functions (Ahn et al., 2026), or abstract models of the environment (Rodriguez-Sanchez & Konidaris, 2024; Piriyakulkij et al., 2026).

Relation to our work. Hierarchical distance decomposition differs from prior work in two main ways. First, we use graph-based decomposition directly to improve learned distance estimates, rather than to support any particular downstream use. Second, we apply it post-hoc to an existing learned distance model, without retraining the model or its underlying representation. Temporal-distance methods focus on how distance models themselves are learned, while hierarchical methods focus on how to introduce hierarchy to improve policies, value functions, or models to support planning and control. The closest graph-based approaches also differ on these points: Yang et al. (2020) retrain a global representation using graph-derived targets, while Bagatella & Martius (2023) correct goal-conditioned value estimates specifically for planning.

3 Problem Setting

We consider a Markov Decision Process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathrm{P}, \mathrm{R}, \gamma \rangle$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathrm{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition function, $\mathrm{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a reward function, and $\gamma \in [0, 1]$ is a discount factor. At each decision stage, an agent observes a state $s \in \mathcal{S}$, selects an action according to a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, and subsequently transitions to a next-state $s' \in \mathcal{S}$ according to P and receives a reward $r \in \mathbb{R}$ according to R. We assume access to a dataset $\mathcal{D}$ of state-only trajectories collected by an agent following an unknown behavior policy.

Let $d^* : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ denote the target distance we wish to estimate. Although our construction applies more generally, we consider $d^* = d_{\mathrm{MAD}}$ in this paper, where the MAD is the minimum

Algorithm 1 Hierarchical Distance Decomposition

procedure BUILDHIERARCHY($\mathcal{D}, d_\theta, \delta, \kappa$)
  Input: Dataset of state trajectories $\mathcal{D} \subset \mathcal{S}$; base quasimetric d_θ; spacing parameter $\delta > 0$; edge-threshold multiplier $\kappa > 1$
  Output: Landmark set $\mathcal{C}$; all-pairs shortest-path matrix D_h
  $\mathcal{C} \leftarrow \emptyset$
  for each state s in $\mathcal{D}$ **do** ▷ Greedy two-sided δ-net over the dataset
    if $\mathcal{C} = \emptyset$ **or** $\min_{c \in \mathcal{C}} \max\big(d_\theta(s, c),\ d_\theta(c, s)\big) > \delta$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{s\}$ ▷ Retire s only if one landmark is close *both* ways
    end if
  end for
  Construct directed graph $G_\mathcal{C} = (\mathcal{C}, E_\mathcal{C})$ with $E_\mathcal{C} = \big\{(c_i, c_j) : d_\theta(c_i, c_j) \le \kappa\delta\big\}$ and $w(c_i, c_j) = d_\theta(c_i, c_j)$ ▷ Keep only short edges: long-range predictions are never trusted
  $D_h \leftarrow$ FLOYDWARSHALL($G_\mathcal{C}$) ▷ All-pairs shortest paths
  return $\mathcal{C}, D_h$
end procedure

procedure QUERYDISTANCE($s_0, s_g, d_\theta, \mathcal{C}, D_h, \kappa, k$)
  Input: Query pair (s_0, s_g); base quasimetric d_θ; landmark set $\mathcal{C}$; shortest-path matrix D_h; edge-threshold multiplier κ; neighbourhood size k
  Output: Distance estimate $\hat{d}_H(s_0, s_g)$
  $c_s^* \leftarrow \arg\min_{c \in \mathcal{C}} d_\theta(s_0, c)$ ▷ Nearest forward landmark
  $c_g^* \leftarrow \arg\min_{c \in \mathcal{C}} d_\theta(c, s_g)$ ▷ Nearest backward landmark
  if $c_s^* = c_g^*$ **then** ▷ Same-landmark regime: too close to route
    return $d_\theta(s_0, s_g)$
  end if
  $\mathcal{C}_s \leftarrow \{c \in \mathcal{C} : d_\theta(s_0, c) \le \kappa\delta\}$ ▷ Nonempty by Proposition 1; truncated to the k nearest
  $\mathcal{C}_g \leftarrow \{c \in \mathcal{C} : d_\theta(c, s_g) \le \kappa\delta\}$ ▷ Nonempty by Proposition 1; truncated to the k nearest
  return $\min_{c_s \in \mathcal{C}_s,\ c_g \in \mathcal{C}_g} d_\theta(s_0, c_s) + D_h(c_s, c_g) + d_\theta(c_g, s_g)$
end procedure

number of actions required to move from one state to another (Steccanella et al., 2026). We assume that d^* is a quasimetric: it is non-negative and satisfies the triangle inequality for all state pairs, but need not be symmetric. We also assume access to a learned function $d_\theta : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_{\ge 0}$ that approximates d^*. We make no assumptions about how d_θ was trained. Our goal is to use the dataset of offline trajectories $\mathcal{D}$ and existing distance model d_θ to obtain more accurate long-range distances than using d_θ alone, without requiring any retraining of d_θ.

Learned distance models may be reliable locally, but become less accurate over longer horizons (Wang et al., 2023; Park et al., 2024b; Steccanella et al., 2026). Three factors contribute to this degradation. First, *errors may accumulate* as distance information is propagated across many local relations during training. Second, *finite-capacity models* may compress a wide range of high-magnitude distance values into similar outputs. Third, offline data collected through local exploration often *differs in distribution* from the distant state pairs whose distances may be queried at test time. Hierarchical distance decomposition addresses these problems by composing a model's more accurate short-range estimates to obtain improved long-range estimates.

4 HIERARCHICAL DISTANCE DECOMPOSITION

We present pseudocode for hierarchical distance decomposition in Algorithm 1. The method consists of two phases: *construction* (BUILDHIERARCHY) and *querying* (QUERYDISTANCE). During construction, the method selects a set of landmark states from a dataset, connects nearby landmarks in a directed graph, and precomputes all-pairs shortest-path distances. At query time, these precomputed distances are combined with local predictions to estimate distances between arbitrary state pairs.

4.1 CONSTRUCTING THE LANDMARK GRAPH

The algorithm begins by using the state-only trajectory dataset $\mathcal{D}$ and an existing learned distance function d_θ to generate a set of landmarks $\mathcal{C}$. It processes the states in $\mathcal{D}$ greedily, adding a state s as

a new landmark only when no existing landmark $c \in \mathcal{C}$ satisfies both $d_\theta(s, c) \leq \delta$ and $d_\theta(c, s) \leq \delta$. The parameter δ therefore controls the spacing between landmarks. This two-sided requirement is important because d_θ may be asymmetric: a landmark that is close when moving from s to c need not be close when moving from c to s. We say that a landmark c *covers* a state s when both conditions hold. This construction ensures the following covering property.

Proposition 1 (Covering property) *After* BUILDHIERARCHY *terminates, every state $s \in \mathcal{D}$ has a landmark $c \in \mathcal{C}$ with*

$$\max\big(d_\theta(s, c),\ d_\theta(c, s)\big) \ \leq \ \delta.$$

Proof. Algorithm 1 visits every state in $\mathcal{D}$. Any state not covered by the landmarks selected so far is added to $\mathcal{C}$ and therefore covers itself with zero distance in both directions. Since landmarks are never removed, this coverage is preserved. □

Why coverage is two-sided. To answer a distance query for a starting state s_0 and a goal state s_g, the start state must connect to a landmark c_0 through $d_\theta(s_0, c_0)$, while a landmark c_g must connect to the goal through $d_\theta(c_g, s_g)$. Since any state may appear as either the start or goal, the covering condition requires each state to have a landmark that can support both roles.

Consider a cheaper alternative (which we do not use) that constructs two separate one-sided landmark sets: one ensures coverage from each state to a landmark, the other ensures coverage from a landmark to each state, and the method takes their union. However, different landmarks may satisfy these two conditions. The resulting set therefore does not guarantee the directed connectivity required to answer a query; Appendix B gives an example in which this alternative produces an infinite distance estimate even when $d_\theta = d^*$. Our two-sided rule avoids this failure by requiring a single landmark to support both roles, at the cost of selecting more landmarks.

The landmark graph. After obtaining the landmark set $\mathcal{C}$, we construct a directed graph over its elements. We retain a directed edge (c_i, c_j) only when $d_\theta(c_i, c_j) \leq \kappa\delta$, assigning it weight $d_\theta(c_i, c_j)$. Applying Floyd–Warshall to this graph gives the matrix D_h, where $D_h(c_i, c_j)$ is the length of the shortest path from c_i to c_j, or infinity if no such path exists. We compute D_h once offline and reuse it for each query. On a fully connected graph, shortest-path search could select a single long edge that the base model severely underestimates, recreating the problem we aimed to avoid. By retaining only edges with predicted distance at most $\kappa\delta$, every finite distance in D_h is a sum of short-range predictions. The multiplier κ controls the resulting trade-off: lower values favour more local predictions, while higher values raise connectivity at the cost of retaining longer-range edges.

4.2 QUERYING DISTANCES

For a query (s_0, s_g), QUERYDISTANCE considers routes of the form $s_0 \to c_s \to \cdots \to c_g \to s_g$. We distinguish *edges*, which connect pairs of landmarks and have stored weights $d_\theta(c_i, c_j)$ in D_h, from *legs*, which connect a query state to a landmark and whose distances are evaluated at query time. Each edge or leg is restricted to model distances of at most $\kappa\delta$, so the overall estimate consists only of short-range predictions from the base model d_θ. A candidate route therefore consists of an entry leg from s_0 to c_s, a shortest landmark path from c_s to c_g with cost $D_h(c_s, c_g)$, and an exit leg from c_g to s_g. We denote the closest landmarks to the state and goal in the relevant directions by

$$c_s^* = \arg\min_{c \in \mathcal{C}} d_\theta(s_0, c), \qquad c_g^* = \arg\min_{c \in \mathcal{C}} d_\theta(c, s_g)$$

for the landmark closest to the start in the forward direction and the landmark closest to the goal in the backward direction. The resulting hierarchical distance estimate is

$$\hat{d}_H(s_0, s_g) = \begin{cases} d_\theta(s_0, s_g) & \text{if } c_s^* = c_g^*, \\ \min\limits_{c_s \in \mathcal{C}_s,\, c_g \in \mathcal{C}_g} \ d_\theta(s_0, c_s) + D_h(c_s, c_g) + d_\theta(c_g, s_g) & \text{otherwise.} \end{cases} \tag{1}$$

For each query, we consider sets of candidate entry and exit landmarks. Entry candidates are landmarks within $\kappa\delta$ of s_0 in the forward direction, while exit candidates are landmarks within $\kappa\delta$ of

s_g in the backward direction. In practice, we truncate each set to its k nearest landmarks, with $k = 5$. For states in $\mathcal{D}$, Proposition 1 ensures that both sets are nonempty by supplying a landmark within δ in both directions. Considering multiple candidates is important because the nearest entry and exit landmarks do not necessarily yield the cheapest complete route.

The first case of Equation 1 handles state pairs that share the same closest landmark. For such pairs, routing through the graph can introduce an $O(\delta)$ detour that dominates their true separation. So, the estimator simply returns the direct local prediction $d_\theta(s_0, s_g)$ instead.

5 Theoretical Analysis

Under two assumptions—local accuracy of the base model and coverage of relevant paths by the dataset—we recover a two-sided bound on estimator error:

$$\mathrm{err}(s_0, s_g) \;=\; \hat{d}_H(s_0, s_g) \;-\; d^*(s_0, s_g)$$

Theorem 1 shows that

$$d^*(s_0, s_g) - (m+2)\,\varepsilon \;\le\; \hat{d}_H(s_0, s_g) \;\le\; d^*(s_0, s_g) + M\,(2\delta + 3\varepsilon) + 2\delta,$$

where ε bounds the local error of the base model, m is the number of graph edges on the route the estimator returns, and $M = \lceil d^*(s_0, s_g)/\delta \rceil$ is the number of hops of the route used for comparison. The left inequality says the estimate can fall below the target distance only by the local error accumulated along the returned route. The right inequality charges a fixed cost per hop of the route used for comparison: 2δ of detour and 3ε of conversion, plus 2δ for the two legs. Throughout, d^* is a quasimetric: $d^*(x, x) = 0$, $d^* \ge 0$, and $d^*(x, z) \le d^*(x, y) + d^*(y, z)$. We make no triangle-inequality assumption about d_θ.

Units. We express d^* in the same units as d_θ, allowing d^*, d_θ, δ, and ε to use a common scale. This is without loss of generality: rescaling the model by a positive constant rescales both sides of every condition below. In particular, δ is a threshold on the model output. Applying the same numerical δ to models with different output scales produces landmark graphs of different resolution.

5.1 Assumptions

The analysis rests on two assumptions: the model must be accurate on every evaluated local pair (Assumption 1), and landmarks must cover the path used for comparison (Assumption 2). Only the overestimation side uses the second one, as the proof idea below explains.

Assumption 1 (Local accuracy) *There is an $\varepsilon \in [0, \delta]$ such that, for all $(x, y) \in \mathcal{S} \times \mathcal{S}$,*

$$\min\big(d_\theta(x, y),\; d^*(x, y)\big) \;\le\; \kappa\delta \quad\Longrightarrow\quad \big|d_\theta(x, y) - d^*(x, y)\big| \;\le\; \varepsilon.$$

This ensures that when either the model or target distance classifies a pair as local, their values differ by at most ε. Because the condition uses a minimum, the assumption applies in the following ways:

$$d_\theta(x, y) \le \kappa\delta \implies d^*(x, y) \;\le\; d_\theta(x, y) + \varepsilon, \tag{2a}$$

$$d^*(x, y) \le \kappa\delta \implies d_\theta(x, y) \;\le\; d^*(x, y) + \varepsilon. \tag{2b}$$

Equation 2a excludes *false shortcuts*: a retained edge classified as short by the model is also short in the target distance, up to ε. Equation 2b ensures that a genuinely short pair is not removed by pruning. The first property is particularly important, as shortest-path search actively selects underestimated edges. Pairs large under both d_θ and d^* are unconstrained; their predictions are never used in an estimate.

Although the assumption is stated for all $(x, y) \in \mathcal{S} \times \mathcal{S}$, it is only applied to the following pair types: two landmarks, a query state and a landmark, a state of the reference path with its covering landmark. Landmarks come from $\mathcal{D}$, so the only pairs that reach beyond the dataset are the two query legs and the snaps along the reference path. Accuracy elsewhere in $\mathcal{S}$ is irrelevant to the bounds.

The second assumption concerns data coverage rather than model accuracy. It requires landmarks near the reference paths, but says nothing about the rest of $\mathcal{S}$.

Assumption 2 (Coverage) *For every query pair (s_0, s_g), some shortest d^*-path from s_0 to s_g stays within δ of $\mathcal{C}$: every state s on this path has a landmark $c \in \mathcal{C}$ with*

$$\max\big(d_\theta(s,c),\ d_\theta(c,s)\big)\ \le\ \delta.$$

Proposition 1 gives this condition only for states in $\mathcal{D}$, so it discharges the assumption solely on the part of the path that the dataset contains. It says nothing about unseen states: for $s_0, s_g \notin \mathcal{D}$ nothing prevents $\min_{c\in\mathcal{C}} d_\theta(s_0, c) > \delta$ or $\min_{c\in\mathcal{C}} d_\theta(c, s_g) > \delta$. Coverage of the two query states is therefore assumed here rather than inherited from the construction. In practice, the assumption ensures some shortest path through covered states can be traced for each query, and is only required along such paths.

Every evaluation of d_θ entering the estimate, edge weights and legs alike, is on a pair at model distance at most $\kappa\delta$, so Assumption 1 applies to all of them (Lemma 1 in Appendix A).

Proof idea. The two inequalities are proved in opposite ways.

No underestimation. The route the estimator returns is itself a chain $s_0 \to c_0 \to \cdots \to c_m \to s_g$, and $\hat{d}_H(s_0, s_g)$ is the sum of its $m + 2$ terms. Every term is a local evaluation of d_θ (Lemma 1), so Assumption 1 places it within ε of the target distance of the corresponding segment. Those segments form a path from s_0 to s_g, so the triangle inequality for d^* bounds the sum of their target lengths from below by $d^*(s_0, s_g)$, which gives the left inequality. Coverage plays no part here, since the argument uses only the route that is actually returned.

Bounded overestimation. In this direction the returned route is unknown, so we use that the estimate is a minimum over feasible routes: it is enough to exhibit one route and bound its cost, because the estimator's own answer cannot be larger. Assumption 2 supplies such a route. Along a shortest d^*-path we mark waypoints $w_0, \ldots, w_M$ with $w_0 = s_0$ and $w_M = s_g$ at spacing at most δ, and replace each waypoint w_i by a landmark c_i that covers it, so that the route runs through the landmarks rather than through the waypoints (Figure 1, left). Each of its edges is then priced through the two waypoints it spans (Figure 1, right),

$$d^*(c_{i-1}, c_i)\ \le\ \underbrace{d^*(c_{i-1}, w_{i-1})}_{\text{snap}} + \underbrace{d^*(w_{i-1}, w_i)}_{\text{progress}} + \underbrace{d^*(w_i, c_i)}_{\text{snap}}, \tag{3}$$

one step of progress along the reference path plus two snaps of size at most δ. Summed over the M edges, the progress terms telescope to $d^*(s_0, s_g)$, while each hop is charged 2δ for its two snaps, and the two legs add a further 2δ: this is the per-hop cost in the right inequality. The right panel of Figure 1 shows where each ε enters: one per snap when coverage is converted from d_θ into d^*, and a third when the finished edge is converted back into the model's units, which is the value that pruning tests.

Two further facts make this route usable, and both are established below: its edges must survive pruning, which is what the threshold condition $\kappa\delta \ge 3\delta + 3\varepsilon$ gives (Lemma 2), and its two endpoints must be candidates, which is what coverage of s_0 and s_g gives. Equation 3 also shows why a landmark must be near its waypoint in both directions: c_i enters through $d^*(w_i, c_i)$ in the edge that ends at it and through $d^*(c_i, w_i)$ in the next one. A third case, treated in Theorem 1(iii), covers pairs that share their nearest landmark, where routing through the graph would only be a detour and the estimator returns $d_\theta(s_0, s_g)$ directly.

The route just described also survives pruning, and its total cost is bounded: this is Lemma 2 in Appendix A, which supplies the landmarks $c_0, \ldots, c_M$ used by Theorem 1(ii).

The edge threshold. The threshold enters only through Lemma 2. The constructed route may require an edge containing two snaps and one waypoint gap, with model length up to $3\delta + 3\varepsilon$. Dividing by δ gives

$$\kappa\ \ge\ 3 + 3\varepsilon/\delta. \tag{4}$$

Assumption 1 gives $\varepsilon \le \delta$. Thus $\kappa = 6$ always satisfies Equation 4, while $\kappa = 4$ requires $\varepsilon \le \delta/3$. Increasing the threshold is not free—$\kappa\delta$ is also the radius over which local accuracy must hold—so a larger κ admits more edges and requires accuracy over a wider range.

Equation 4 is sufficient, not necessary. It ensures graph connectivity by constructing a conservative landmark route along a shortest d^*-path. Each edge along the constructed route may include a

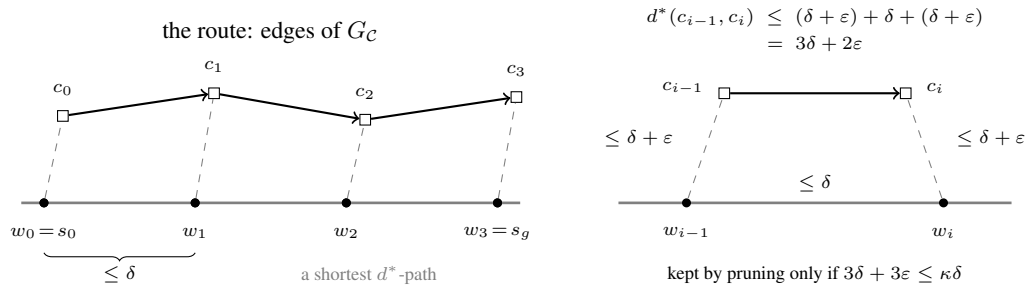


Figure 1: The route exhibited for the overestimation bound. *Left:* waypoints (dots) sit on a true shortest path with spacing δ. Each is snapped to a landmark (squares) beside it through which the route runs, never through waypoints. *Right:* one hop, the source of each ε—contained only by the two snaps. Coverage is stated in d_θ, so converting a snap into d^* costs ε apiece. The gap carries none, as waypoints are spaced with d^* by default. This gives $3\delta + 2\varepsilon$ instead of $3\delta + 3\varepsilon$. The third ε only appears later, converting the finished edge back into the model's units, the price $3\delta + 3\varepsilon$ that pruning tests against $\kappa\delta$.

waypoint gap and two snaps. The graph can contain a shorter landmark chain, for example when landmarks lie densely near the reference path. Such a chain can survive a stricter threshold. The condition is therefore a worst-case guarantee. In practice, smaller values of κ may work.

5.2 Two-Sided Error Bound

Theorem 1 (Two-sided hierarchical bound) *Let Assumptions 1 and 2 hold with κ satisfying Equation 4, and let $\hat{d}_H$ use the untruncated candidate sets $\mathcal{C}_s, \mathcal{C}_g$. Fix a query pair (s_0, s_g) and put $M = \lceil d^*(s_0, s_g)/\delta \rceil$.*

(i) (No underestimation.) *In the multi-hop regime, if the route attaining $\hat{d}_H(s_0, s_g)$ uses m graph edges, then*

$$\hat{d}_H(s_0, s_g) \;\geq\; d^*(s_0, s_g) - (m + 2)\,\varepsilon.$$

(ii) (Bounded overestimation.) *In the multi-hop regime,*

$$\hat{d}_H(s_0, s_g) \;\leq\; d^*(s_0, s_g) + M\,(2\delta + 3\varepsilon) + 2\delta.$$

(iii) (Same-landmark regime.) *If $c_s^* = c_g^*$, then $|\hat{d}_H(s_0, s_g) - d^*(s_0, s_g)| \leq \varepsilon$.*

The proof is given in Appendix A.

The same two bounds also order two queries whose target distances are far enough apart; Corollary 1 in Appendix A states this and proves it.

The condition in Corollary 1 applies to sufficiently separated pairs: their target distances must differ by more than the cost $M(2\delta + 3\varepsilon) + 2\delta$ that the comparison route pays on top of $d^*(s_0, s_g)$, plus the error $(m' + 2)\varepsilon$ accumulated on the other query. It does not guarantee the ordering of arbitrarily close pairs or directly imply a Spearman correlation. Instead, it identifies one type of long-range ordering that the hierarchy preserves even when a base model saturates.

The constants are conservative. The proof charges every local error at its worst value, and prices one particular route rather than the route selected by minimization. The result should therefore be read as an existence guarantee for bounded distortion, not as a prediction of the error the estimator will attain.

Beyond MAD. This analysis is not specific to MAD, and in fact applies to any quasimetric target for which the base model is locally accurate. Stochastic control introduces an additional issue: an edge may be executed by a local policy that fails to reach its intended target. Appendix C discusses an options-style abstraction that models these outcomes explicitly.

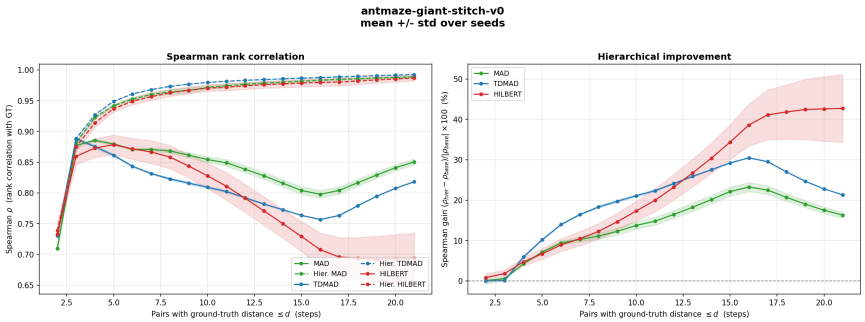


Figure 2: Spearman correlation with grid-maze distance as the evaluated range increases (left) and relative improvement from the hierarchy (right), on the largest setting: AntMaze Giant, *stitch* split. Results show the mean ± standard deviation over five seeds for all three base models.

6 Empirical Analysis

In this section, we present an empirical evaluation of hierarchical distance decomposition. Our analysis is directed by the following questions: Does hierarchical distance decomposition improve the long-horizon MAD estimates produced by existing distance-learning methods? If so, do these distance estimation improvements translate into better downstream planning performance?

Environments and data. We evaluate on the PointMaze and AntMaze environments from the OGBench suite (Park et al., 2025), using the *medium*, *large*, and *giant* environment sizes. For both environments, we evaluate performance using the *navigate* and *stitch* dataset variants. For AntMaze Medium, we also use the *explore* dataset.

Evaluation metrics. To evaluate distance quality, we follow the methodology of Steccanella et al. (2026) and compare learned distances with the grid-geodesic distance through the underlying maze. This is computed by applying the Floyd–Warshall algorithm to the maze topology. We measure the agreement between the predicted and ground-truth grid-geodesic distances using Spearman rank correlation.

While the grid-geodesic distance provides a useful proxy for navigation progress, it is not the exact MAD—particularly in environments like AntMaze where maze topology alone does not capture the agent's embodied dynamics and control effort. We therefore complement this distance-quality evaluation with planning success rate, which measures whether improvements in the learned distance estimates actually translate into better downstream goal-reaching performance.

Base distances. We evaluate hierarchical distance decomposition using three learned base distance models. We use two quasimetric models trained as described in Steccanella et al. (2026): **MadDist** trained using the contrastive and trajectory-distance losses, and **TDMAD**, its temporal-difference variant. We also include the **HILBERT** method of Park et al. (2024b), a symmetric distance embedding trained using hindsight relabelling. We train each model once for each environment–dataset combination and seed. We report all the hyperparameters selected for training these base models in Appendix D. Note that the proposed approach is agnostic to the base model used as long as it does not violate Assumptions 1 and 2.

Hierarchy. We construct the hierarchical decomposition separately for each base distance model. Each decomposition uses the corresponding base model for local distance evaluation, so each flat–hierarchical comparison uses *exactly* the same underlying base distance model.

We construct the landmark net using the offline dataset of trajectories provided for each OGbench environment (Park et al., 2025). We use spacing values δ of 6 and 4, a threshold multiplier $\kappa = 2$, and $k = 5$ entry and exit candidates. Appendix E reports an ablation study over these parameters. We use the max selection rule from Proposition 1 and precompute all-pairs shortest paths once for each environment–dataset setting, base distance model, and seed.

Table 1: Planning success rate on the *giant* mazes, over 50 episodes, reported as mean ± standard deviation over five seeds. *Base* is the learned distance used by the planner and *Data* is its training split. Flat uses the base distance directly; Hierarchical applies Algorithm 1. The complete results, including the medium and large mazes, are given in Appendix G.

Environment	Base / Data	Flat	Hierarchical	Δ
PointMaze Giant	HILBERT / navigate	0.000 ± 0.000	1.000 ± 0.000	$+1.000$
	HILBERT / stitch	0.000 ± 0.000	0.960 ± 0.049	$+0.960$
	MAD / navigate	0.072 ± 0.068	0.980 ± 0.040	$+0.908$
	MAD / stitch	0.040 ± 0.062	1.000 ± 0.000	$+0.960$
	TDMAD / navigate	0.028 ± 0.039	1.000 ± 0.000	$+0.972$
	TDMAD / stitch	0.000 ± 0.000	1.000 ± 0.000	$+1.000$
AntMaze Giant	HILBERT / navigate	0.000 ± 0.000	0.548 ± 0.083	$+0.548$
	HILBERT / stitch	0.000 ± 0.000	0.368 ± 0.102	$+0.368$
	MAD / navigate	0.004 ± 0.008	0.620 ± 0.379	$+0.616$
	MAD / stitch	0.016 ± 0.015	0.988 ± 0.010	$+0.972$
	TDMAD / navigate	0.000 ± 0.000	0.572 ± 0.178	$+0.572$
	TDMAD / stitch	0.000 ± 0.000	0.984 ± 0.023	$+0.984$

Planner. To evaluate downstream planning performance, we use a sampling-based planner that considers candidate short-horizon action sequences and uses the learned distance model to select the one that leads to progress towards a given goal. Within each environment, we keep the planner and its search budget fixed across all distance models, varying only the distance function provided to the planner. Full details can be found in Appendix F.

Results. Across all environments, datasets, and base models, the proposed hierarchical distance decomposition improves Spearman correlation compared to each base distance model on its own. Figure 2 shows learning curves for the largest setting, AntMaze Giant on the *stitch* split, and Appendix H reports the same comparison for all remaining environment–dataset settings. Across environments, the base models typically lose rank agreement at longer distances, while the hierarchical estimates consistently improve over the base models. We also report hierarchical gains showing that the percentage improvement in how well hierarchical predictions correlate with the reference grid-geodesic distances increases as the distance between state pairs increases.

The same pattern is visible in Table 1, which shows planning success rates. Gains are largest on long-horizon tasks where the base models fail entirely. Several flat baselines achieve near-zero success on giant mazes while hierarchical versions succeed reliably. Gains shrink when the base is already accurate. The hierarchy improves HILBERT on PointMaze, but less so on the hardest AntMaze tasks, where its local predictions are too inaccurate to compose effectively over long horizons.

The patterns in these results align well with the motivation for hierarchical distance decomposition: replacing long-range predictions with compositions of short-range predictions increases their accuracy, and these improvements lead to increased performance on downstream planning tasks.

7 Conclusion

We presented a training-free hierarchical decomposition that improves over any base model by decomposing long-distance queries into short-range components.

Empirical evaluation on multiple goal-conditioned tasks demonstrates that the hierarchy improves rank correlation with the ground-truth distance, and more importantly improves planning performance. The improvements are largest for base models that degrade most at long range, while the method provides less room for improvement when the base model is already strong.

One limitation of the approach is that both the theoretical guarantee and the empirical method depend on a base model with sufficient local accuracy, a necessary condition for the hierarchical decomposition to be reliable.

We believe the proposed algorithm can serve as a foundation for future work that explores task division, where local models are trained to be accurate locally and long-horizon reasoning is delegated to the hierarchical distance decomposition.

AI USE STATEMENT

In this work, we used generative AI tools as assistants for writing the LaTeX version of the paper, generating TikZ plots, and developing the code required for the experiments. We carefully reviewed all AI-assisted content and take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work studies distance estimation on standard offline-navigation benchmarks. It involves no human subjects, personally identifiable data, or deployment on physical systems.

REPRODUCIBILITY STATEMENT

Algorithm 1 specifies the method, while Appendix E and Appendix D specify all the hyperparameters used to construct the hierarchy and train the base models. Based on the acceptance of the paper, we plan to release the full code to reproduce all our results.

REFERENCES

Hongjoon Ahn, Heewoong Choi, Jisu Han, and Taesup Moon. Option-aware temporally abstracted value for offline goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 38, 2026.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Seungho Baek, Taegeon Park, Jongchan Park, Seungjun Oh, and Yusung Kim. Graph-assisted stitching for offline hierarchical reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 2391–2408. PMLR, 2025.

Marco Bagatella and Georg Martius. Goal-conditioned offline planning from curious exploration. In *Advances in Neural Information Processing Systems*, volume 36, pp. 15358–15383. Curran Associates, Inc., 2023.

Alexi Canesse, Mathieu Petitbois, Ludovic Denoyer, Sylvain Lamprier, and Rémy Portelas. Navigation with QPHIL: Quantizing planner for hierarchical implicit Q-learning. In *Proceedings of the 2025 International Joint Conference on Neural Networks*. IEEE, 2025.

Ben Eysenbach, Russ R. Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Leslie Pack Kaelbling. Learning to achieve goals. *International Joint Conference on Artificial Intelligence*, 2:1094–1098, 1993.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*. PMLR, 2015.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

Anthony Kobanda, Waris Radji, Odalric-Ambrym Maillard, and Rémy Portelas. Adaptive quasimetric mapping: Principled topological abstraction for robust offline goal-conditioned navigation. In *Proceedings of the 43rd International Conference on Machine Learning*. PMLR, 2026.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline Reinforcement Learning with Implicit Q-Learning. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Vivek Myers, Bill Zheng, Benjamin Eysenbach, and Sergey Levine. Offline goal-conditioned reinforcement learning with quasimetric representations. volume 38, pp. 19654–19679. Curran Associates, Inc., 2026.

Whitney K Newey and James L Powell. Asymmetric Least Squares Estimation and Testing. *Econometrica: Journal of the Econometric Society*, pp. 819–847, 1987.

Evgenii Opryshko, Junwei Quan, Claas A Voelcker, Yilun Du, and Igor Gilitschenski. Test-time graph search for goal-conditioned reinforcement learning. In *Proceedings of the 43rd International Conference on Machine Learning*. PMLR, 2026.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: offline goal-conditioned RL with latent states as actions. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34866–34891. Curran Associates, Inc., 2023.

Seohong Park, Oleh Rybkin, and Sergey Levine. METRA: scalable unsupervised RL with metric-aware abstraction. In *Proceedings of the 12th International Conference on Learning Representations*, 2024a.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with Hilbert representations. pp. 39737–39761, 2024b.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *Proceedings of the 13th International Conference on Learning Representations*, volume 2025, pp. 94937–94982, 2025.

Wasu T. Piriyakulkij, Wolfgang Lehrach, Kevin Ellis, and Kevin Murphy. Joint learning of hierarchical neural options and abstract world model. *arXiv preprint arXiv:2602.02799*, 2026.

Rafael Rodriguez-Sanchez and George Konidaris. Learning abstract world models for value-preserving planning with options. In *Proceedings of the 1st Reinforcement Learning Conference*. RLJ, 2024.

Lorenzo Steccanella and Anders Jonsson. State representation learning for goal-conditioned reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 84–99. Springer, 2022.

Lorenzo Steccanella, Joshua B. Evans, Özgür Şimşek, and Anders Jonsson. Learning the minimum action distance. In *Proceedings of the 43rd International Conference on Machine Learning*. PMLR, 2026.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-Learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 36411–36430. PMLR, 2023.

Clarisse Wibault, Alexander D. Goldie, Antonio L. Villares, Maike A. Osborne, and Jakob N. Foerster. Abstraction for offline goal-conditioned reinforcement learning. In *Workshop on Decision-Making from Offline Datasets to Online Adaptation: Black-Box Optimization to Reinforcement Learning*, 2026.

Ge Yang, Amy Zhang, Ari Morcos, Joelle Pineau, Pieter Abbeel, and Roberto Calandra. Plan2vec: Unsupervised representation learning by latent plans. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, pp. 935–946. PMLR, 2020.

Bill Chunyuan Zheng, Vivek Myers, Benjamin Eysenbach, and Sergey Levine. Multistep quasimetric learning for scalable goal-conditioned reinforcement learning. In *Proceedings of the 14th International Conference on Learning Representations*, 2026.

APPENDIX

A PROOFS

In this appendix we present the proofs deferred from Section 5.

Lemma 1 (Only local evaluations are used) *Every evaluation of d_θ that enters $\hat{d}_H(s_0, s_g)$ in the multi-hop regime—edge weights and legs alike—is performed on a pair (x, y) such that $d_\theta(x, y) \le \kappa\delta$, and hence satisfies $|d_\theta(x, y) - d^*(x, y)| \le \varepsilon$.*

Proof. An edge (c_i, c_j) is retained only if $d_\theta(c_i, c_j) \le \kappa\delta$. Likewise, $\mathcal{C}_s$ contains only landmarks with $d_\theta(s_0, c) \le \kappa\delta$, and $\mathcal{C}_g$ contains only landmarks with $d_\theta(c, s_g) \le \kappa\delta$. Assumption 1 therefore applies through its d_θ condition. □

Lemma 2 (A short reference route survives pruning) *Let Assumptions 1 and 2 hold, suppose the pruning threshold satisfies $\kappa\delta \ge 3\delta + 3\varepsilon$, and let (s_0, s_g) be a query pair with $M = \lceil d^*(s_0, s_g)/\delta \rceil$. Then there are landmarks $c_0, \dots, c_M \in \mathcal{C}$ such that $d_\theta(s_0, c_0) \le \delta$ and $d_\theta(c_M, s_g) \le \delta$, consecutive distinct landmarks are joined by edges of $G_\mathcal{C}$, and*

$$\sum_{i=1}^{M} d_\theta(c_{i-1}, c_i) \;\le\; d^*(s_0, s_g) + M(2\delta + 3\varepsilon).$$

Proof. Assumption 2 gives a shortest d^*-path from s_0 to s_g whose states are all covered. The path is cut into M pieces of length at most δ, with the form $w_0 = s_0, w_1, \dots, w_{M-1}, w_M = s_g$. The pieces are ordered along a shortest path, such that $d^*(w_{i-1}, w_i) \le \delta$ and $\sum_{i=1}^{M} d^*(w_{i-1}, w_i) = d^*(s_0, s_g)$.

Let c_i be a landmark covering w_i, so that $\max\big(d_\theta(w_i, c_i), d_\theta(c_i, w_i)\big) \le \delta$; the cases $i = 0$ and $i = M$ are the two endpoint bounds. Coverage bounds each snap by δ in the model, within the local range $\kappa\delta$, and are converted by Equation 2a into the target distance: $\max\big(d^*(w_i, c_i), d^*(c_i, w_i)\big) \le \delta + \varepsilon$. Equation 3 then prices the edge (c_{i-1}, c_i) as

$$d^*(c_{i-1}, c_i) \;\le\; (\delta + \varepsilon) + d^*(w_{i-1}, w_i) + (\delta + \varepsilon) \;\le\; 3\delta + 2\varepsilon \;\le\; \kappa\delta.$$

Since $d^*(c_{i-1}, c_i) \le \kappa\delta$, Equation 2b applies and converts the edge back into the model's units:

$$\begin{aligned} d_\theta(c_{i-1}, c_i) &\le d^*(c_{i-1}, c_i) + \varepsilon && \text{by Equation 2b} \\ &\le \big[(\delta + \varepsilon) + d^*(w_{i-1}, w_i) + (\delta + \varepsilon)\big] + \varepsilon \\ &= d^*(w_{i-1}, w_i) + 2\delta + 3\varepsilon \\ &\le 3\delta + 3\varepsilon \;\le\; \kappa\delta, && \text{since } d^*(w_{i-1}, w_i) \le \delta \end{aligned} \tag{5}$$

where the last inequality is the hypothesis $\kappa\delta \ge 3\delta + 3\varepsilon$. The final value is the test applied by pruning, so every consecutive distinct pair (c_{i-1}, c_i) is an edge of $G_\mathcal{C}$.

It remains to sum Equation 5 over the M edges:

$$\begin{aligned} \sum_{i=1}^{M} d_\theta(c_{i-1}, c_i) &\le \sum_{i=1}^{M} \big[\, d^*(w_{i-1}, w_i) + 2\delta + 3\varepsilon \,\big] && \text{by Equation 5} \\ &= \sum_{i=1}^{M} d^*(w_{i-1}, w_i) \;+\; M(2\delta + 3\varepsilon) \\ &= d^*(s_0, s_g) + M(2\delta + 3\varepsilon), \end{aligned}$$

where the last line uses $\sum_{i=1}^{M} d^*(w_{i-1}, w_i) = d^*(s_0, s_g)$, established in the first paragraph. This is the claimed bound. □

With the two lemmas in place we restate the theorem and prove it.

Theorem 1 (Two-sided hierarchical bound, restated) *Let Assumptions 1 and 2 hold with κ satisfying Equation 4, and let $\hat{d}_H$ use the untruncated candidate sets $\mathcal{C}_s, \mathcal{C}_g$. Fix a query pair (s_0, s_g) and put $M = \lceil d^*(s_0, s_g)/\delta \rceil$.*

(i) (No underestimation.) *In the multi-hop regime, if the route attaining $\hat{d}_H(s_0, s_g)$ uses m graph edges, then*

$$\hat{d}_H(s_0, s_g) \;\geq\; d^*(s_0, s_g) - (m+2)\,\varepsilon.$$

(ii) (Bounded overestimation.) *In the multi-hop regime,*

$$\hat{d}_H(s_0, s_g) \;\leq\; d^*(s_0, s_g) + M\,(2\delta + 3\varepsilon) + 2\delta.$$

(iii) (Same-landmark regime.) *If $c_s^* = c_g^*$, then $|\hat{d}_H(s_0, s_g) - d^*(s_0, s_g)| \leq \varepsilon$.*

Proof of Theorem 1. *(i)* Let $s_0 \to c_0 \to c_1 \to \cdots \to c_m \to s_g$ attain the minimum in Equation 1. Then $\hat{d}_H(s_0, s_g) = d_\theta(s_0, c_0) + \sum_{i=1}^{m} d_\theta(c_{i-1}, c_i) + d_\theta(c_m, s_g)$. By Lemma 1, every one of these $m+2$ terms is at least the target length of its segment, minus ε. The segments form a path from s_0 to s_g, so by the triangle inequality their target lengths sum to at least $d^*(s_0, s_g)$. If no route exists, $\hat{d}_H(s_0, s_g) = \infty$ and the claim is immediate.

(ii) The estimate is the minimum over feasible routes, so the route $s_0 \to c_0 \to \cdots \to c_M \to s_g$ constructed in Lemma 2 upper-bounds it. That route is feasible: its endpoints satisfy $c_0 \in \mathcal{C}_s$ and $c_M \in \mathcal{C}_g$, and its graph edges bound $D_h(c_0, c_M)$. Its two legs satisfy $d_\theta(s_0, c_0) \leq \delta$ and $d_\theta(c_M, s_g) \leq \delta$, also by Lemma 2. Hence

$$\begin{aligned}
\hat{d}_H(s_0, s_g) \;&\leq\; d_\theta(s_0, c_0) \;+\; \sum_{i=1}^{M} d_\theta(c_{i-1}, c_i) \;+\; d_\theta(c_M, s_g) && \text{minimality over routes} \\
&\leq\; \delta \;+\; \Big[\, d^*(s_0, s_g) + M(2\delta + 3\varepsilon) \,\Big] \;+\; \delta && \text{legs; Lemma 2} \\
&=\; d^*(s_0, s_g) + M(2\delta + 3\varepsilon) + 2\delta,
\end{aligned}$$

which is the claim. Each term is accounted for: $d^*(s_0, s_g)$ is the progress along the reference path, $2M\delta$ are the two snaps of each of the M edges, $3M\varepsilon$ are the three conversions between d_θ and d^* charged per edge, and 2δ are the entry and exit legs.

(iii) Let $c = c_s^* = c_g^*$. The covering landmark c_0 of s_0 satisfies $d_\theta(s_0, c_0) \leq \delta$. Since c_s^* minimizes $d_\theta(s_0, \cdot)$ over $\mathcal{C}$, we have $d_\theta(s_0, c) \leq \delta$. Similarly, $d_\theta(c, s_g) \leq \delta$. Equation 2a gives $d^*(s_0, c) \leq \delta + \varepsilon$ and $d^*(c, s_g) \leq \delta + \varepsilon$. Therefore $d^*(s_0, s_g) \leq 2\delta + 2\varepsilon \leq \kappa\delta$, and Assumption 1 applies to the returned value $d_\theta(s_0, s_g)$ through its d^* condition. □

Corollary 1 (Order preservation) *Under the hypotheses of Theorem 1, let (s_0, s_g) and (s_0', s_g') be two multi-hop queries, with $M = \lceil d^*(s_0, s_g)/\delta \rceil$ and m' the number of graph edges on the route attaining $\hat{d}_H(s_0', s_g')$. Then,*

$$d^*(s_0', s_g') \;>\; d^*(s_0, s_g) + M\,(2\delta + 3\varepsilon) + 2\delta + (m'+2)\,\varepsilon \quad\Longrightarrow\quad \hat{d}_H(s_0', s_g') \;>\; \hat{d}_H(s_0, s_g).$$

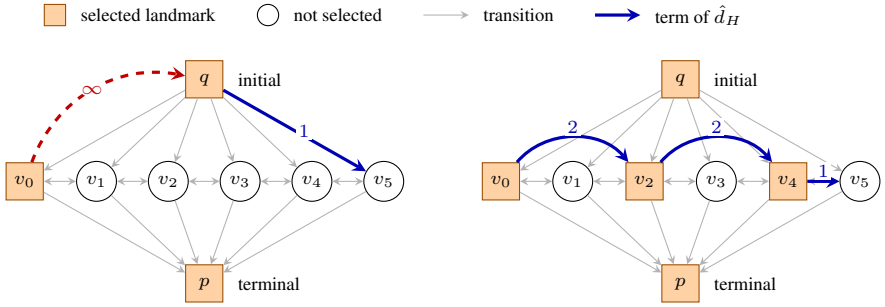
Proof. By (i) applied to (s_0', s_g') and (ii) applied to (s_0, s_g),

$$\begin{aligned}
\hat{d}_H(s_0', s_g') \;&\geq\; d^*(s_0', s_g') - (m'+2)\,\varepsilon && \text{by (i)} \\
&>\; d^*(s_0, s_g) + M\,(2\delta + 3\varepsilon) + 2\delta && \text{by the hypothesis} \\
&\geq\; \hat{d}_H(s_0, s_g) && \text{by (ii).}
\end{aligned}$$

□

B ONE-SIDED LANDMARK NETS

In Section 4.1 a state s is skipped when some landmark c satisfies $\max\big(d_\theta(s,c), d_\theta(c,s)\big) \le \delta$, so that one landmark is close to s in both directions. This appendix explains why that requirement is the principled one and compares it with the cheaper alternative of building two one-sided nets: a *forward* net, which skips s when some landmark satisfies $d_\theta(s,c) \le \delta$; a *backward* net, which skips s when some landmark satisfies $d_\theta(c,s) \le \delta$; and their union. The counter example below shows that the two one-sided nets, and their union, can make the estimate infinite even when the base model is exact.



(a) Union of one-sided nets
$\hat{d}_H(v_0, v_5) = 0 + \infty + 1 = \infty$

(b) Max rule
$\hat{d}_H(v_0, v_5) = 0 + 2 + 2 + 1 = 5$

Figure 3: The example of Appendix B for the query (v_0, v_5), with $d^*(v_0, v_5) = 5$. Gray arrows are unit-cost transitions and colored arrows are the terms of $\hat{d}_H$. (a) The one-sided union keeps v_0 as its only corridor landmark, and nothing reaches q, so $D_h(v_0, q) = \infty$ (dashed). (b) The max rule also selects v_2 and v_4 and is exact.

Example. Consider a deterministic MDP with unit action costs, and three kinds of states (Figure 3): corridor states $v_0, \dots, v_n$ with transitions $v_i \leftrightarrow v_{i+1}$; an initial state q with a transition $q \to v_i$ to every corridor state and no transition into q; and a terminal state p with a transition $v_i \to p$ from every corridor state. Every episode starts at q, moves to a corridor state, and can terminate only at p. For all i, j,

$$d^*(v_i, v_j) = |i - j|, \qquad d^*(q, v_i) = d^*(v_i, p) = 1, \qquad d^*(q, p) = 2,$$

while $d^*(p, y) = \infty$ for every $y \neq p$ and $d^*(y, q) = \infty$ for every $y \neq q$. Let the base model be exact, $d_\theta = d^*$ (so $\varepsilon = 0$), and let $n = 5$, $\delta = 1$, and $\kappa = 4$. Consider the dataset of the two episodes

$$\tau_1 = (q,\ v_0,\ p), \qquad \tau_2 = (q,\ v_0,\ v_1,\ v_2,\ v_3,\ v_4,\ v_5,\ p),$$

which together visit every state. Algorithm 1 visits them in this order, and the states of each episode in time order. We consider the query (v_0, v_5), for which $d^*(v_0, v_5) = 5$, and Figure 3 shows the landmarks that each rule selects.

Backward rule. Both episodes start at q, so q is the first state visited and becomes a landmark. Since $d^*(q, v_i) = 1$ for every i, the backward rule skips every corridor state, and it selects p because $d^*(q, p) = 2$. The backward net is therefore $\{q, p\}$. For two corridor states v_a and v_b, the only entry candidate is p, because $d^*(v_a, q) = \infty$, and the only exit candidate is q, because $d^*(p, v_b) = \infty$. The query is in the multi-hop regime and $D_h(p, q) = \infty$, so $\hat{d}_H(v_a, v_b) = \infty$ for every pair of corridor states, including neighbours with $d^*(v_a, v_b) = 1$.

Forward rule and the union. The forward rule selects q, then v_0 because $d^*(v_0, q) = \infty$, and then p at the end of τ_1 because $d^*(p, y) = \infty$ for every $y \neq p$. From then on $d^*(v_i, p) = 1$ covers every corridor state, so τ_2 adds nothing and the forward net, like the union, is $\{q, v_0, p\}$. Since $d^*(v_0, v_5) = 5 > \kappa\delta$, the only exit candidate for v_5 is q, and $D_h(c, q) = \infty$ for every landmark $c \neq q$. Hence

$$\hat{d}_H(v_0, v_5) = \infty \qquad \text{while} \qquad d^*(v_0, v_5) = 5.$$

Max rule. The max rule also selects q and p, but neither covers a corridor state, because each is at infinite distance from it in one direction. Corridor states are therefore covered only by corridor

landmarks, and v_0, v_2, and v_4 are selected. The candidate sum with entry leg $d^*(v_0, v_0) = 0$, edges $v_0 \to v_2 \to v_4$ of length $2 \le \kappa\delta$ each, and exit leg $d^*(v_4, v_5) = 1$ equals 5.

The analysis needs both directions. The example above exhibits one-way in which the one-sided δ nets fail. The same condition also enters the analysis of Section 5. The upper bound of Theorem 1(ii) needs one candidate sum close to $d^*(s_0, s_g)$. Take waypoints $w_0 = s_0, \ldots, w_M = s_g$ along a shortest path from s_0 to s_g and a landmark c_i for each w_i. The triangle inequality bounds the terms of that sum by

$$d^*(c_{i-1}, c_i) \;\le\; d^*(c_{i-1}, w_{i-1}) + d^*(w_{i-1}, w_i) + d^*(w_i, c_i),$$

so an interior landmark c_i appears in two consecutive terms: the term that ends at c_i is controlled by $d^*(w_i, c_i)$, and the term that starts at c_i by $d^*(c_i, w_i)$. The max rule bounds both quantities at the same landmark. The two one-sided nets bound one each, at possibly different landmarks c and c', and chaining them would add a term $d^*(c, c')$ that neither covering condition controls.

C Beyond Minimum-Action Distance

The analysis in Section 5 applies to any quasimetric target with a locally accurate estimator. In a stochastic control problem, however, a graph edge must also be executable. We use *option* to mean the local policy intended to carry the agent from one landmark to another.

Shortest path in $G_{\mathcal{C}}$ is appropriate when every edge is implemented by a *proper* option: it reaches its target landmark with probability one, and its edge weight represents expected execution cost. The abstract transition is then deterministic and expected costs compose, so Floyd–Warshall gives the optimal cost of a landmark chain. If an option can fail, terminate at another landmark, or fail to terminate, this interpretation breaks down. The landmark graph is then an abstract MDP, and deterministic shortest path is optimistic because it treats every selected transition as certain. Replanning after each subgoal helps with compounding execution failures but does not correct this optimistic edge model.

One remedy is an options-style semi-Markov abstraction. Represent an option o_{ij}, which attempts to reach c_j from c_i, by its expected cost $\bar{c}(o_{ij})$ and termination distribution $P(c' \mid c_i, o_{ij})$, where c' is the landmark closest to the state at which the option stops. Long-range values then solve the abstract stochastic shortest-path problem

$$V_g(c) \;=\; \min_{o \in \mathcal{O}(c)} \Big[\bar{c}(o) + \sum_{c' \in \mathcal{C}} P(c' \mid c, o)\, V_g(c') \Big], \qquad V_g(c_g) = 0, \tag{6}$$

The query then replaces $D_h(c_s, c_g)$ in Equation 1 with $V_g(c_s)$. If every option is proper, then $P(c' \mid c_i, o_{ij}) = \mathbb{1}[c' = c_j]$ and Equation 6 reduces to shortest path in $G_{\mathcal{C}}$. If an option succeeds with probability p and otherwise returns near its start, the recursion accounts for retries and assigns cost on the order of $\bar{c}(o_{ij})/p$. Rare-transition shortcuts are therefore penalized by the model rather than filtered through a hand-tuned threshold. The additional cost is estimating an outcome distribution for each option from rollouts and solving one abstract stochastic shortest-path problem per goal landmark.

D Base Models

In this section, we describe the implementation details of each base model included in our evaluation.

D.1 Computer Resources

We run all experiments on a single NVIDIA RTX 4070 GPU with 8GB of VRAM and an Intel i7-4700-HX with 32GB of RAM.

D.2 MadDist

To train the MadDist distance models, we used the Adam optimizer with a learning rate of 1×10^{-4}, a batch size of 256 for the objective ($\mathcal{L}_o$, $\mathcal{L}_r$), and a separate batch of size 1024 for the constraint loss

($\mathcal{L}_c$). For our main experiment, we used the simple quasimetric function Steccanella et al. (2026) and a latent dimension size of 512. In contrast with the original code of Steccanella et al. (2026) we use $w_r = 1$, $w_c = 0.1$ this enforce the model to be more locally accurate, making the constrain stronger.

The full set of hyperparameter values used to train the MadDist models can be found in Table 2.

Table 2: Hyperparameters used to train the MadDist algorithm.

Hyperparameter	**Value**
Quasimetric Function	d_{simple}
Optimizer	Adam Kingma & Ba (2015)
Learning Rate	1×10^{-4}
Batch Size ($\mathcal{L}_o$, $\mathcal{L}_r$)	256
Batch Size ($\mathcal{L}_c$)	1024
Activation Function (Hidden Layers)	SELU Klambauer et al. (2017)
Neural Network	(512, 512, 256, 128)
w_r	1
w_c	0.1
d_{max}	100, 500
H_c	6

D.3 TDMADDIST

To train the TDMadDist distance models, we used the the Adam optimizer with a learning rate of 1×10^{-4}, a batch size of 256 for the objective ($\mathcal{L}_o$, $\mathcal{L}_r$), and a separate batch of size 1024 for the constraint loss ($\mathcal{L}_c$). For our main experiment, we used the novel simple quasimetric function and a latent dimension size of 512.

For TDMadDist, we remove the hyperparameter $d_{\max}$ from the MadDist algorithm, because it is not included in TDMadDist's objective ($\mathcal{L}_r$). The temporal-difference update used when training the TDMadDist distance models involves the use of a target network, $d_{\theta'}$, which is updated using a Polyak averaging factor $\tau = 0.005$.

The full set of hyperparameter values used to train the TDMadDist models can be found in Table 3.

Table 3: Hyperparameters used to train the TDMadDist algorithm.

Hyperparameter	**Value**
Quasimetric Function	d_{simple}
Optimizer	Adam Kingma & Ba (2015)
Learning Rate	1×10^{-4}
Batch Size ($\mathcal{L}_o$, $\mathcal{L}_r$)	256
Batch Size ($\mathcal{L}_c$)	1024
Activation Function (Hidden Layers)	SELU Klambauer et al. (2017)
Neural Network	(512, 512, 256, 128)
w_r	1
w_c	0.1
H_c	6
τ	0.005

D.4 HILBERT REPRESENTATION

We trained Hilbert representation models following the approach of Park et al. (2024b), using action-free Implicit Q-Learning (IQL) (Park et al., 2023) and Hindsight Experience Replay (HER) (Andrychowicz et al., 2017).

We used a dataset of state–next-state pairs (s, s'), which we relabeled using HER to produce state–next-state–goal tuples (s, s', g). Goals were sampled from a geometric distribution Geom(γ) over future states in the same trajectory with probability 0.625, and uniformly from the entire dataset with probability 0.375.

We trained the Hilbert representation model ϕ to minimize the temporal-difference loss

$$\mathbb{E}[l_\tau(-\mathbf{1}(s \neq g) - \gamma||\phi(s') - \phi(g)|| + ||\phi(s) - \phi(g)||)], \tag{7}$$

where l_τ denotes the expectile loss (Newey & Powell, 1987), an asymmetric loss function that approximates the $\max$ operator in the Bellman backup (Kostrikov et al., 2022). This objective naturally supports the use of target networks (Mnih et al., 2015) and double estimators (Van Hasselt et al., 2016) to improve learning stability. We included both in our implementation, following the original setup used by Park et al. (2024b).

The full set of hyperparameter values used to train the Hilbert models can be found in Table 4.

Table 4: Hyperparameters used to train the Hilbert representation models.

Hyperparameter	**Value**
Latent Dimension	32
Expectile	0.95
Discount Factor	0.99
Learning Rate	0.0003
Target Network Smoothing Factor	0.005
Multi-Layer Perceptron Dimensions	(512, 512, 512) Fully-Connected Layers
Activation Function (Hidden Layers)	GELU (Hendrycks & Gimpel, 2016)
Layer Normalization (Hidden Layers)	True
Activation Function (Final Layer)	Identity
Layer Normalization (Final Layer)	False
Optimizer	Adam (Kingma & Ba, 2015)
Batch Size	1024

E Ablation on choices of Delta and Kappa

The sufficient threshold condition in Equation 4 is conservative. We therefore treat κ and δ as an empirical hyperparameter in the experiments.

We sweep the spacing $\delta \in \{4, 6, 9, 12\}$ against the threshold multiplier $\kappa \in \{2, 3, 4, 6, 8\}$ on the *giant* navigate datasets, for all three base models and averaged over seeds. Figures 4 and 5 report the resulting Spearman correlation over the whole (δ, κ) grid, together with the gain over the base model as the evaluated range widens.

The best cell is always $\delta = 4$ with $\kappa = 2$, and the correlation decreases as either parameter grows. As expected the effect of κ is the stronger of the two, and it is most pronounced for the weakest base model: on AntMaze Giant with HILBERT the correlation falls from 0.980 at $\kappa = 2$ to 0.844 at $\kappa = 8$ with δ fixed at 4, whereas for MAD on PointMaze Giant the same sweep moves it only from 0.982 to 0.974. This is the behaviour the analysis predicts qualitatively: $\kappa\delta$ is the radius over which the base model has to be accurate, so raising it admits more edges the base model can missclassify. Smaller δ also means more landmarks, shown as n beside each row, so the preferred setting is the one that keeps every evaluation short at the cost of a larger graph.

In our experiment we use $\delta = 4$ and $\kappa = 2$ in all experiments, with one exception: the hierarchical TDMAD variant uses $\delta = 6$. At $\delta = 4$ that model's net grows to $13{,}858$ landmarks on AntMaze Giant, and becomes impractical. Raising the spacing to $\delta = 6$ brings the net down to $4{,}363$ landmarks, about a thirtyfold saving in preprocessing, and costs almost nothing in quality.

F Planning algorithm

Table 5 compares the planner horizon H with the average number of decision stages required to reach the goal in each environment. In both AntMaze and PointMaze, H covers only a fraction of the full goal-reaching horizon, so the planner must rely on the learned distance to successfully navigate towards distant goals.

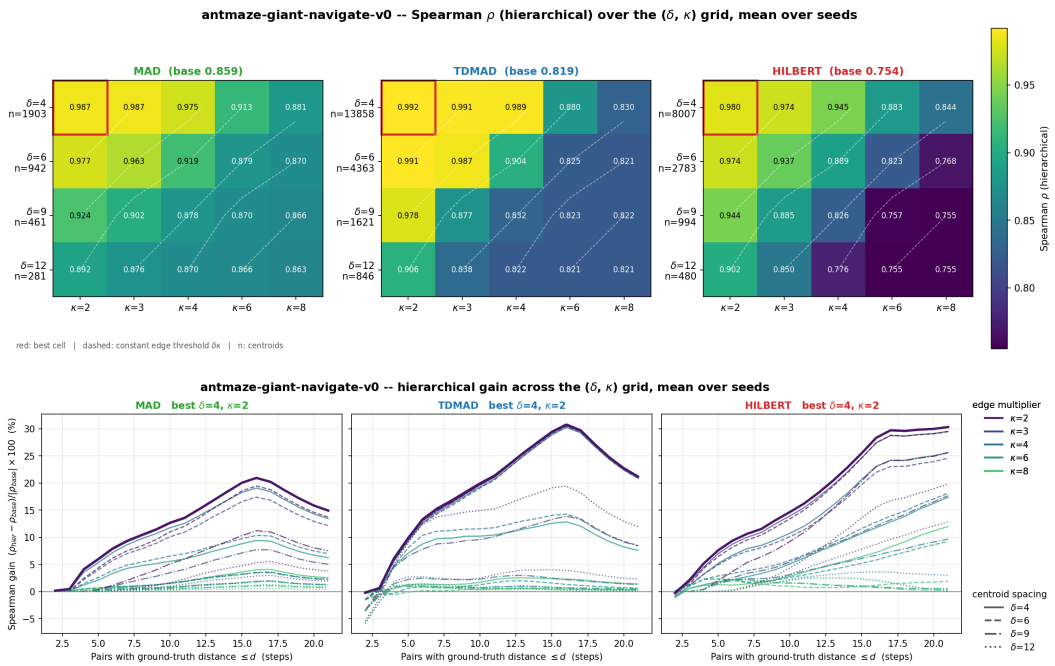


Figure 4: Ablation over the (δ, κ) grid on AntMaze Giant, *navigate* split, for the three base models, averaged over seeds. *Top:* Spearman correlation of the hierarchical estimate with the grid-maze distance; the base model's correlation is given in each panel title, the best cell is outlined in red, dashed lines are contours of constant edge threshold $\kappa\delta$, and n is the number of landmarks selected at that spacing. *Bottom:* relative gain over the base model as the evaluated range widens, with κ shown by colour and δ by line style.

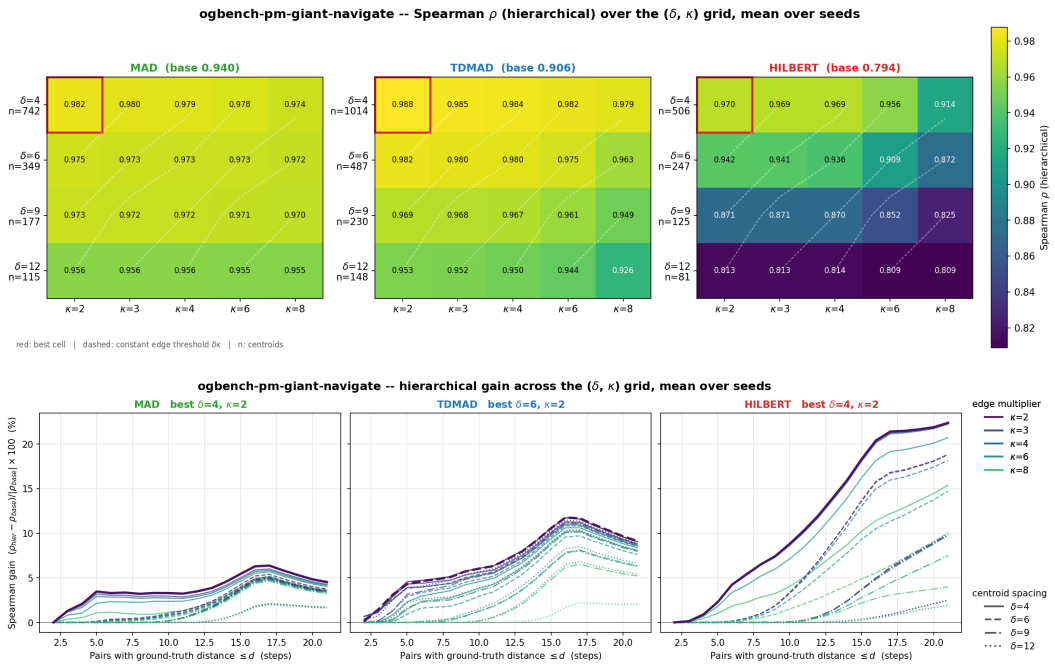


Figure 5: The same ablation on PointMaze Giant, *navigate* split. Axes, colours, and annotations are as in Figure 4.

Table 5: Planning horizon compared to average steps to goal. The goal is never inside the planner's lookahead, so the direction of travel comes from the learned distance rather than from the simulated rollouts.

Domain	Maze	H (steps)	Steps to goal	H / steps to goal
AntMaze	Medium	30	271	11.1%
	Large	30	451	6.7%
	Giant	30	863	3.5%
PointMaze	Medium	5	184	2.7%
	Large	5	316	1.6%
	Giant	5	542	0.9%

Planning in PointMaze. We validate distance quality using simple random shooting. At each replanning step, we sample 50 actions uniformly from a 600-bin grid (24 per dimension), simulate each for $H = 5$ steps, and execute the action whose trajectory minimizes learned distance to the goal. No refinement, elite selection, or smoothing is applied. This deliberately weak planner ensures planning success depends on distance accuracy.

Planning in AntMaze. Naive random sampling fails in AntMaze's 8D torque space, sampling 240 free parameters yields uncoordinated walking. Instead, we sample a central pattern generator (CPG) with 25 parameters: amplitude, phase, and offset per joint, plus a shared frequency. These parameters decode into a 30-step torque sequence:

$$a_j(t) \;=\; \mathrm{offset}_j + \mathrm{amp}_j \sin(2\pi f t + \mathrm{phase}_j). \tag{8}$$

This reduces the search space by an order of magnitude while restricting sampling to locomotion-compatible sequences. We refine with 3 CEM iterations (keeping 10 best candidates), warm-start the gait across replans.

Comparability. Both domains use identical planners and budgets when comparing flat vs. hierarchical distances. Any success rate difference is thus due to distance quality, not planner differences. The extra machinery in AntMaze (CPG, CEM refinement) is necessary for the dynamics, not an advantage for any particular method.

G Additional Planning Results

This appendix reports the planning success rate of Table 1 for every environment, base distance model, and dataset split.

H Additional Distance-Quality Results

This appendix reports the Spearman-correlation comparison of Figure 2 for every remaining environment and dataset split.

Table 6: Planning success rate on every environment, over 50 episodes, reported as mean ± standard deviation over five seeds. Columns are as in Table 1, whose *giant* rows are repeated here so that the comparison across maze sizes can be read in one place.

Environment	Base / Data	Flat	Hierarchical	Δ
PointMaze Medium	HILBERT / navigate	0.408 ± 0.152	1.000 ± 0.000	$+0.592$
	HILBERT / stitch	0.360 ± 0.113	1.000 ± 0.000	$+0.640$
	MAD / navigate	0.552 ± 0.041	1.000 ± 0.000	$+0.448$
	MAD / stitch	0.372 ± 0.094	1.000 ± 0.000	$+0.628$
	TDMAD / navigate	0.388 ± 0.098	1.000 ± 0.000	$+0.612$
	TDMAD / stitch	0.380 ± 0.174	1.000 ± 0.000	$+0.620$
PointMaze Large	HILBERT / navigate	0.056 ± 0.050	1.000 ± 0.000	$+0.944$
	HILBERT / stitch	0.040 ± 0.070	1.000 ± 0.000	$+0.960$
	MAD / navigate	0.324 ± 0.145	1.000 ± 0.000	$+0.676$
	MAD / stitch	0.476 ± 0.084	1.000 ± 0.000	$+0.524$
	TDMAD / navigate	0.308 ± 0.082	1.000 ± 0.000	$+0.692$
	TDMAD / stitch	0.272 ± 0.081	1.000 ± 0.000	$+0.728$
PointMaze Giant	HILBERT / navigate	0.000 ± 0.000	1.000 ± 0.000	$+1.000$
	HILBERT / stitch	0.000 ± 0.000	0.960 ± 0.049	$+0.960$
	MAD / navigate	0.072 ± 0.068	0.980 ± 0.040	$+0.908$
	MAD / stitch	0.040 ± 0.062	1.000 ± 0.000	$+0.960$
	TDMAD / navigate	0.028 ± 0.039	1.000 ± 0.000	$+0.972$
	TDMAD / stitch	0.000 ± 0.000	1.000 ± 0.000	$+1.000$
AntMaze Medium	HILBERT / explore	0.104 ± 0.079	0.976 ± 0.015	$+0.872$
	HILBERT / navigate	0.224 ± 0.069	0.936 ± 0.056	$+0.712$
	HILBERT / stitch	0.320 ± 0.090	0.804 ± 0.037	$+0.484$
	MAD / explore	0.492 ± 0.041	0.996 ± 0.008	$+0.504$
	MAD / navigate	0.484 ± 0.053	1.000 ± 0.000	$+0.516$
	MAD / stitch	0.428 ± 0.107	0.996 ± 0.008	$+0.568$
	TDMAD / explore	0.296 ± 0.078	0.996 ± 0.008	$+0.700$
	TDMAD / navigate	0.236 ± 0.086	0.972 ± 0.039	$+0.736$
	TDMAD / stitch	0.300 ± 0.069	1.000 ± 0.000	$+0.700$
AntMaze Large	HILBERT / navigate	0.036 ± 0.043	0.652 ± 0.149	$+0.616$
	HILBERT / stitch	0.024 ± 0.023	0.484 ± 0.122	$+0.460$
	MAD / navigate	0.340 ± 0.063	0.980 ± 0.018	$+0.640$
	MAD / stitch	0.308 ± 0.045	0.992 ± 0.010	$+0.684$
	TDMAD / navigate	0.328 ± 0.041	0.868 ± 0.061	$+0.540$
	TDMAD / stitch	0.324 ± 0.008	1.000 ± 0.000	$+0.676$
AntMaze Giant	HILBERT / navigate	0.000 ± 0.000	0.548 ± 0.083	$+0.548$
	HILBERT / stitch	0.000 ± 0.000	0.368 ± 0.102	$+0.368$
	MAD / navigate	0.004 ± 0.008	0.620 ± 0.379	$+0.616$
	MAD / stitch	0.016 ± 0.015	0.988 ± 0.010	$+0.972$
	TDMAD / navigate	0.000 ± 0.000	0.572 ± 0.178	$+0.572$
	TDMAD / stitch	0.000 ± 0.000	0.984 ± 0.023	$+0.984$

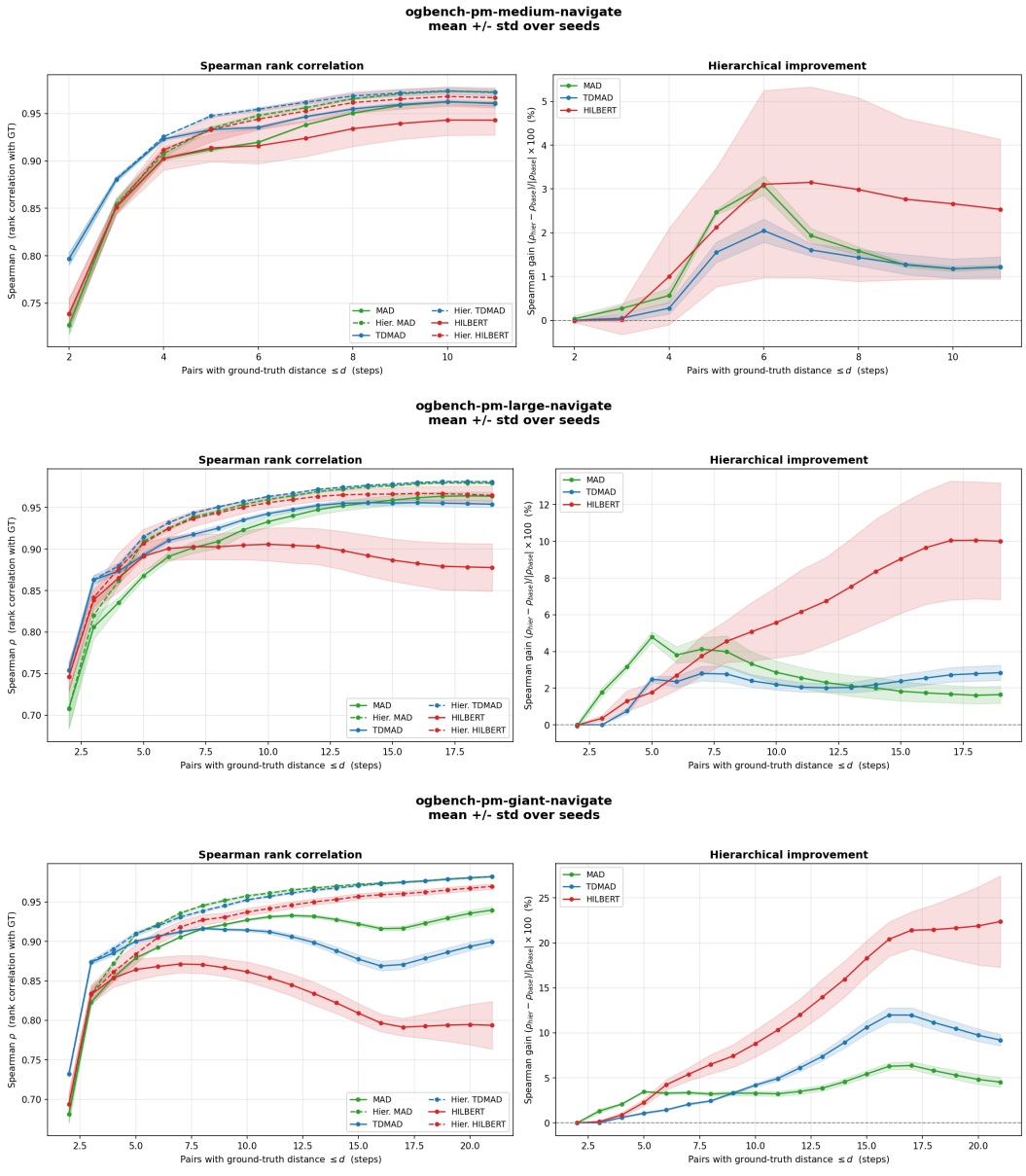


Figure 6: Spearman correlation with grid-geodesic distance as the evaluated range increases (left) and relative improvement from the hierarchy (right). Results are mean ± standard deviation over five seeds for all three base models. Dotted lines show the approximate trusted radius $\kappa\delta$, converted to grid-distance units using the mean ratio d_θ/d^*; they are illustrative rather than exact. PointMaze, *navigate* split, at medium, large, and giant sizes.

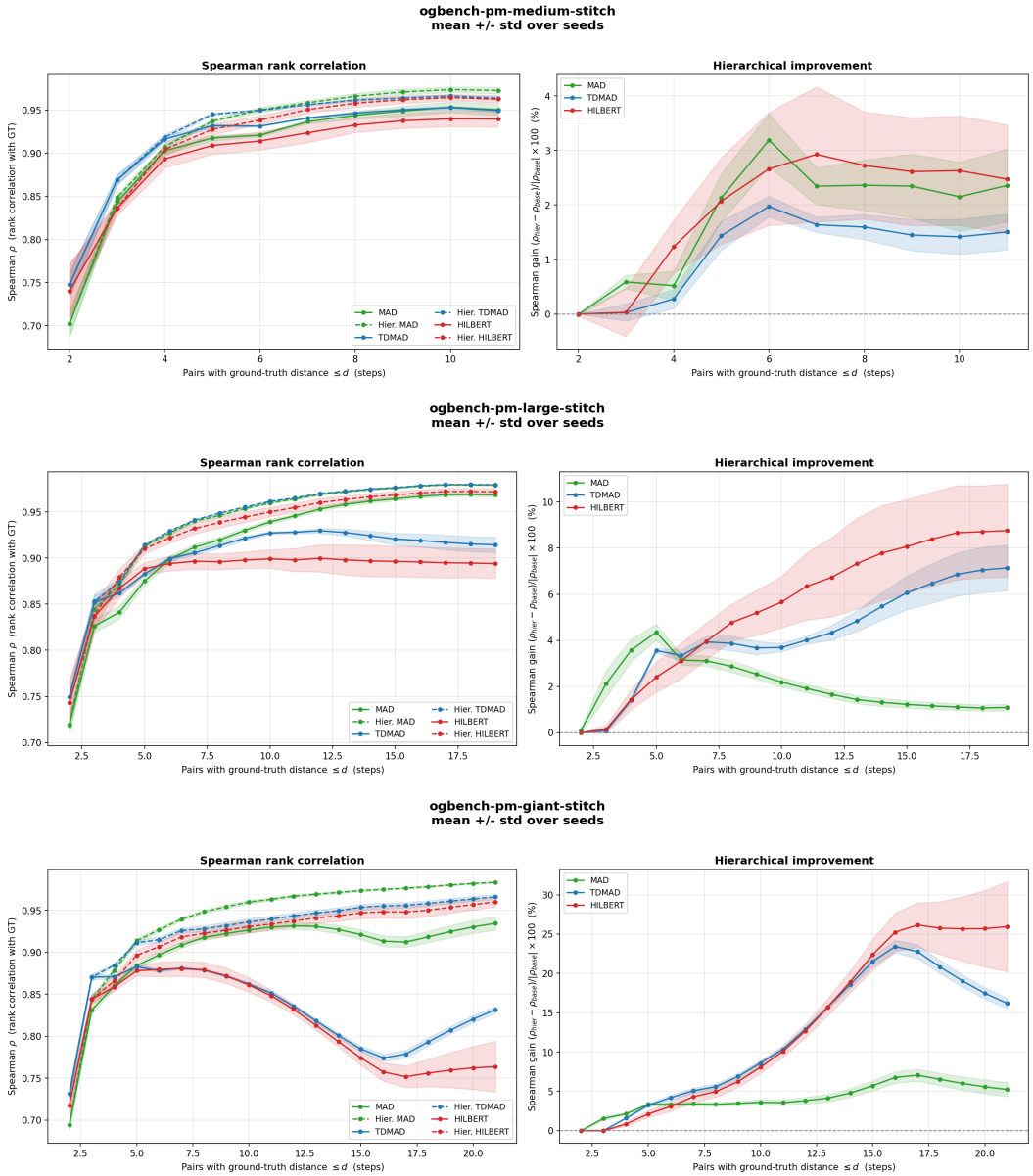


Figure 7: Spearman correlation with grid-geodesic distance as the evaluated range increases (left) and relative improvement from the hierarchy (right). Results are mean ± standard deviation over five seeds for all three base models. Dotted lines show the approximate trusted radius $\kappa\delta$ in grid-distance units. PointMaze, *stitch* split.

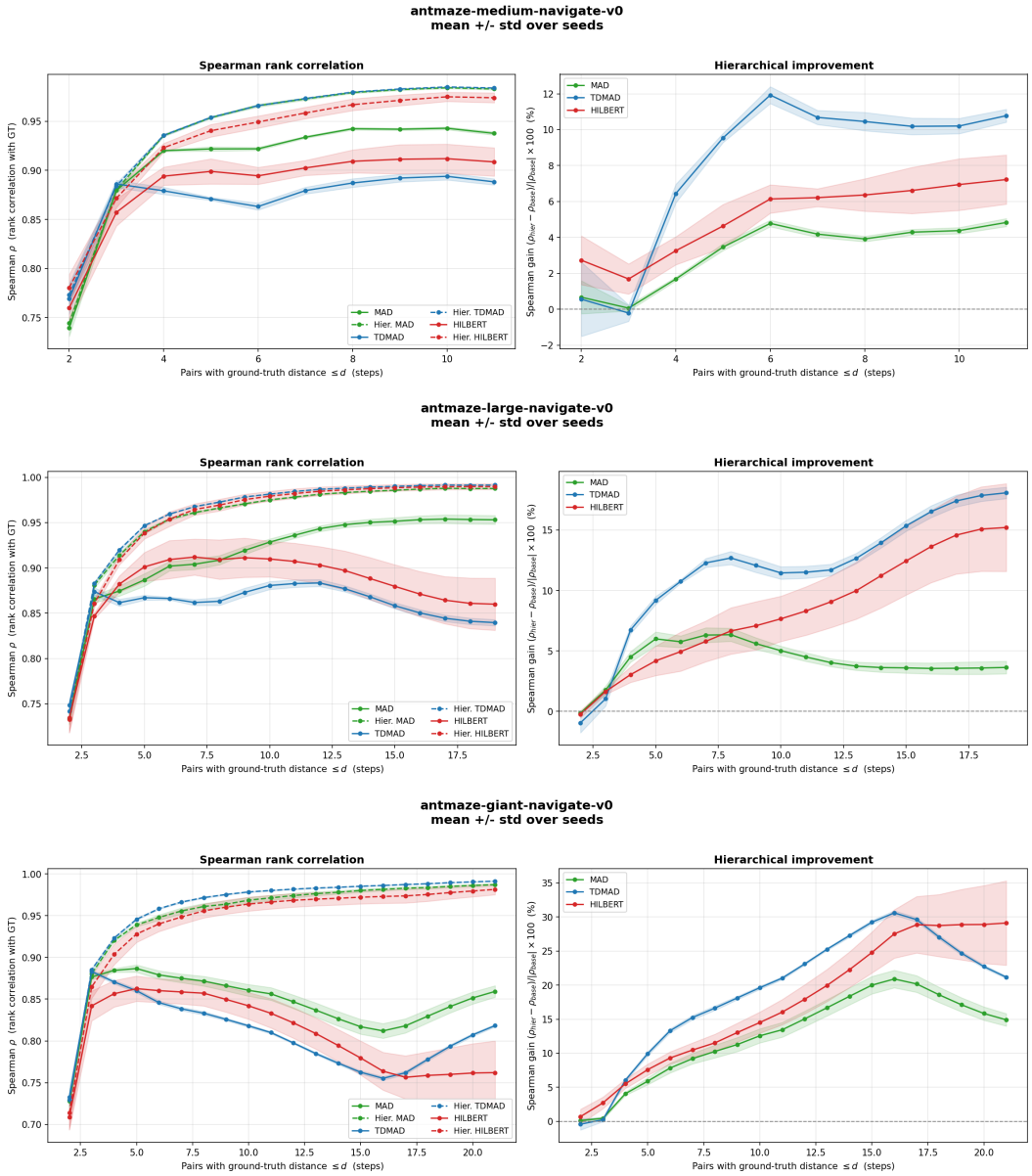


Figure 8: Spearman correlation with grid-geodesic distance as the evaluated range increases (left) and relative improvement from the hierarchy (right). Results are mean ± standard deviation over five seeds for all three base models. Dotted lines show the approximate trusted radius $\kappa\delta$ in grid-distance units. AntMaze, *navigate* split.

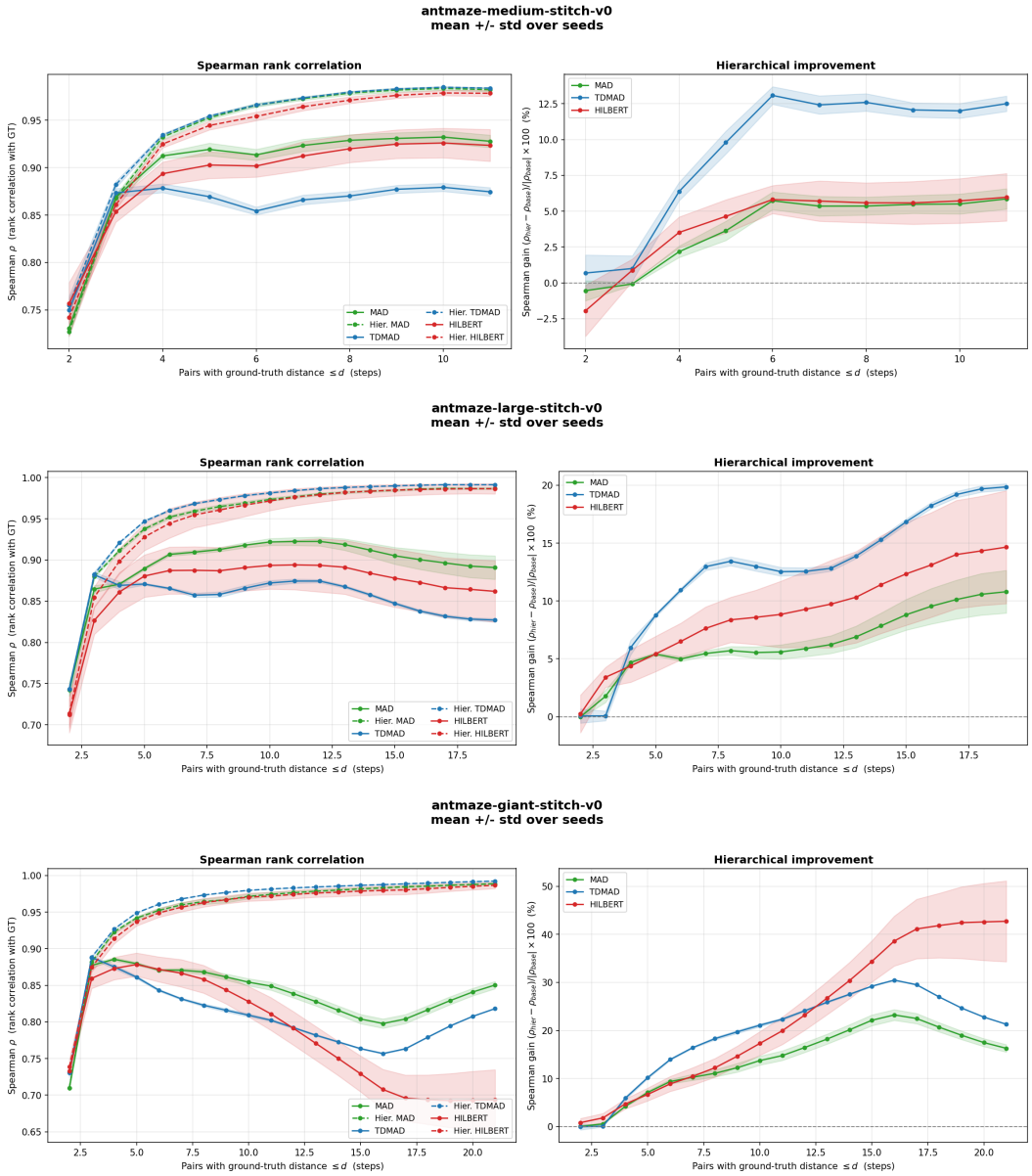


Figure 9: Spearman correlation with grid-geodesic distance as the evaluated range increases (left) and relative improvement from the hierarchy (right). Results are mean ± standard deviation over five seeds for all three base models. Dotted lines show the approximate trusted radius $\kappa\delta$ in grid-distance units. AntMaze, *stitch* split.

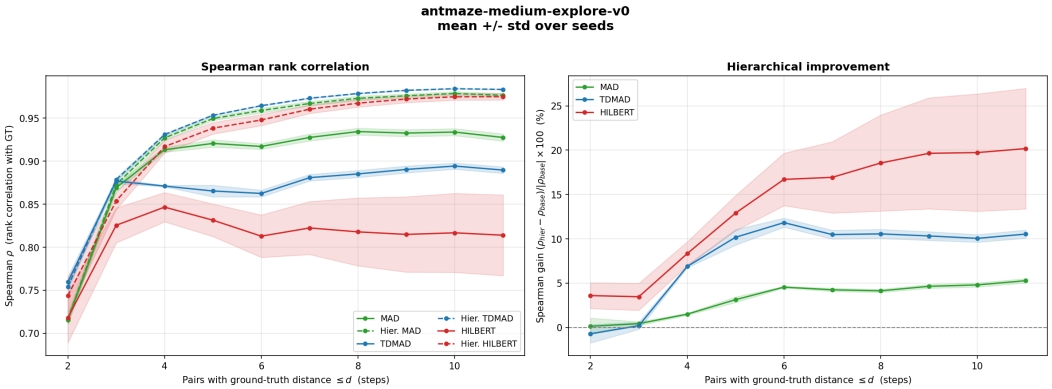


Figure 10: Spearman correlation with grid-geodesic distance as the evaluated range increases (left) and relative improvement from the hierarchy (right). Results are mean $\pm$ standard deviation over five seeds for all three base models. Dotted lines show the approximate trusted radius $\kappa\delta$ in grid-distance units. AntMaze Medium, *explore* split.