Reachability at Scale for State-Constrained Offline RL: Retrieve-then-Certify Graphs with End-to-End Guarantees

Liz Lemma Future Detective

January 20, 2026

Abstract

State-constrained offline RL replaces batch constraints on (s, a) with constraints on states: policies may take out-of-distribution actions so long as they land in in-distribution states. The practical bottleneck is reachability estimation—naively checking which of N dataset states are reachable from each state is quadratic. Building on the state-constrained framework and StaCQ's model-based reachability witness, we propose Retrieve-then-Certify (RTC): a two-stage pipeline that (1) retrieves a small candidate set of next states per state using approximate nearest-neighbor search in a learned reachability embedding, and (2) certifies each candidate edge with a calibrated model-consistency test that yields a probabilistic certificate of approximate executability. RTC produces a sparse reachability graph enabling graph-based SCQL/StaCQ training with sublinear query time and GPU-friendly batching. We provide (i) tight time/space complexity bounds (matching an $\Omega(Nk)$ output-size lower bound), (ii) a high-probability guarantee that no spurious edges are admitted under a uniform error certificate, and (iii) an end-to-end value degradation bound for state-constrained offline RL driven by the certified edge tolerance ε. Experiments (to strengthen the claims) would demonstrate scaling to tens of millions of transitions on robotics/driving logs and pixel observations, with ablations isolating retrieval recall, certification tightness, and RL performance.

Table of Contents

1. 1. Introduction: state-constrained offline RL, why reachability is the scaling bottleneck, limitations of quadratic reachability and prior heuristics (e.g., R-tree pruning), and overview of Retrieve-then-Certify (RTC).
2. 2. Preliminaries and Setup: MDP/offline RL notation; state-constrained policies; reachability witness via forward/inverse models; definition

of approximate executability; statement of computational goals (subquadratic reachability construction).

3. 3. Problem Formulation: Certified Reachability Graph Construction: formal input/output; admissible edge semantics; certification risk parameter α; embedding and retrieval assumptions; output sparsity parameter k.

4. 4. Algorithm: Retrieve-then-Certify (RTC): (i) learn reachability embedding; (ii) ANN retrieval; (iii) calibration step for forward-model error bounds; (iv) batch certification; (v) caching/incremental updates; integration into graph-SCQL/StaCQ.

5. 5. Theoretical Guarantees I — Certification: conditions under which RTC certifies approximate executability; union bounds across all queried pairs; discussion of calibration methods (split conformal / empirical Bernstein) and what they do and do not guarantee.

6. 6. Theoretical Guarantees II — End-to-End RL Impact: value loss bounds for graph-based SCQL under approximate executability and Lipschitz value; robustness to missing edges (false negatives) vs extra edges; optional pessimistic variant to mitigate residual uncertainty.

7. 7. Complexity Landscape: time and memory of RTC; output-size lower bound $\Omega(Nk)$; conditional lower bounds for exact neighborhood graph construction in high dimension; why approximation/retrieval is necessary.

8. 8. Experimental Plan (Implementation-Strengthening Section): scaling benchmarks; large-log robotics/driving; pixel observations; throughput/memory; end-to-end returns vs baselines; ablations (k, embedding dimension, certification thresholds, caching).

9. 9. Related Work: offline RL (BCQ/TD3+BC/IQL), model-based uncertainty and conformal prediction, ANN search and graph construction, state similarity/bisimulation, and trajectory stitching.

10. 10. Discussion and Limitations: when certificates fail (OOD queries, representation collapse), what assumptions are needed for correctness, and future directions (state-only datasets, multi-step reachability).

11. 11. Conclusion: summary of RTC and what it enables for 2026-scale state-constrained offline RL.

1 Introduction

Offline reinforcement learning (offline RL) seeks to compute a policy using only a fixed dataset of logged transitions, without further interaction with the environment. In many safety- or compliance-critical deployments, it is natural to impose an additional *state constraint*: the learned policy should remain within the support of the dataset, or at least within a controlled neighborhood thereof. A convenient formalization is to restrict decision-making to the set of dataset states and to allow transitions only along edges that are (approximately) executable by some action available in the environment. This viewpoint reduces a large portion of the offline RL problem to (i) constructing a directed reachability graph on dataset states and then (ii) optimizing a standard discounted control objective on that graph, as is done by graph-based variants of conservative or state-constrained Q-learning.

The dominant obstacle is that reachability is, at scale, a graph construction problem. Given n unique dataset states, an exact approach that tests all ordered pairs (s, s') for executability incurs $\Theta(n^2)$ candidate edges before any sparsification. This quadratic barrier is not a constant-factor nuisance: even when certification of a single candidate edge is moderately cheap, n^2 quickly becomes prohibitive for datasets of modern size. The difficulty is structural: executability is not purely a geometric property of states, but a property of *existence of an action* that induces a transition from s to (near) s'. Consequently, the naive fallback "connect each state to its nearest neighbors under a metric" is neither sufficient nor necessary, and exact reachability cannot be reduced to a single kNN query without additional modeling assumptions.

Prior work has therefore relied on a mixture of heuristics for pruning candidates. Typical strategies include distance-threshold graphs in raw state space, approximate kNN over hand-crafted features, tree-based spatial indices (e.g., R-tree variants) to avoid a full scan, and various forms of local connectivity expansion (grow the graph outward from observed transitions). Such methods can reduce wall-clock time in favorable low-dimensional regimes, but they do not directly address the core bottleneck: the need to identify, for each source state, a small set of *plausible* target states that might be reachable by some action, and then to *validate* these candidates. Moreover, purely geometric pruning can fail catastrophically in high-dimensional observations (images, proprioception histories) or when the environment dynamics are highly non-Euclidean; conversely, using a learned representation without a principled acceptance test tends to produce graphs with spurious edges, which in turn invalidates any downstream policy guarantee.

We adopt a two-stage design that separates candidate generation from correctness: *Retrieve-then-Certify* (RTC). The guiding principle is that retrieval should be fast and permissive, while certification should be slow(er) but reliable. Concretely, we first build an approximate nearest-neighbor

(ANN) index over a learned embedding $\phi(s)$ of dataset states. For each source state s, we retrieve at most $k \ll n$ candidate targets $s' \in S_D$ by a single ANN query. This stage enforces the sublinear access pattern we require for scalability: no full scan over S_D is permitted when forming adjacency lists. Retrieval alone, however, is not a correctness mechanism; it is merely a proposal distribution over potential next states that is intended to have high recall for truly executable transitions.

The second stage certifies approximate executability by leveraging learned forward/inverse dynamics models. Given a proposed edge $(s \to s')$, we use an inverse model to propose an action $\hat{a} = \hat{I}(s, s')$, and then we check whether the forward prediction $\hat{f}(s, \hat{a})$ lands close to s' according to a metric d. This yields a reachability witness residual

$$\widehat{w}(s, s') \;:=\; d\big(\hat{f}(s, \hat{I}(s, s')),\, s'\big),$$

which is then combined with a calibrated predictive error bound for $\hat{f}$ to obtain a sufficient condition for the true transition to land within a tolerance ε. The key point is that certification is *global*: we allocate a failure probability budget α across all tested edges and demand that, with probability at least $1-\alpha$, every accepted edge is ε-approximately executable in the environment. In this way, retrieval errors affect only *recall* (which edges we consider), whereas certification controls *precision* (which edges we accept).

This decomposition yields both computational and algorithmic benefits. Computationally, RTC constructs a sparse directed graph with out-degree at most k per state, hence at most nk total candidates and at most nk certifications; its running time is essentially the cost of n ANN queries plus nk model-based checks, which is the best we can hope for up to logarithmic factors given that writing down nk edges already requires $\Omega(nk)$ time. Algorithmically, the resulting graph is a drop-in substrate for state-constrained offline RL methods: we can optimize a discounted objective on the graph, interpret each accepted edge as an approximately executable option, and then relate the performance of the learned policy to that of an oracle constrained policy under mild regularity (e.g., Lipschitz) conditions. The remainder of the paper formalizes this construction, states the certification event precisely, and quantifies both the end-to-end value degradation induced by ε-approximate executability and the tightness of the resulting construction-time guarantees.

2 Preliminaries and Setup

We work with a discounted Markov decision process (MDP) $M = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(\cdot \mid s, a)$ is the transition kernel, $r(s, a) \in \mathbb{R}$ is the one-step reward, and $\gamma \in (0, 1)$ is the discount factor. For notational clarity we will often consider the deterministic case, in which there exists a transition function $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ such that $s^+ = T(s,a)$; all definitions below are stated in terms of a metric and therefore extend verbatim to stochastic transitions by interpreting $T(s,a)$ as a suitable representative (e.g., mean next state) or by working with high-probability neighborhoods.

In offline RL we are given a fixed dataset of transitions

$$D = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^{N},$$

collected by some (unknown) behavior policy. We denote by

$$S_D \;:=\; \{\, s : \exists (s,a,r,s'') \in D \,\}$$

the set of unique dataset states, and write $n := |S_D|$. Our primary goal is to compute a policy with high discounted return while respecting a *state constraint* tied to the empirical support S_D. We treat the dataset as read-only: in particular, our constructions may preprocess D and train auxiliary models, but they do not query the environment.

To formalize the state constraint, we equip $\mathcal{S}$ with a metric $d(\cdot,\cdot)$ (defaulting to ℓ_2 on a chosen representation). Fix a tolerance $\varepsilon \geq 0$. For $s, s' \in S_D$, we say that s' is *ε-approximately executable from s* if there exists an action $a \in \mathcal{A}$ such that

$$d\big(T(s,a),\, s'\big) \leq \varepsilon. \tag{1}$$

When $\varepsilon = 0$ this reduces to exact executability. This notion induces a (generally unknown) directed relation on S_D, and it is convenient to view it as a directed graph G^* on vertex set S_D with edge $s \to s'$ whenever (1) holds. A *state-constrained* policy is then any policy that, when started at a dataset state, selects actions so that its realized state sequence remains within (or near, up to ε) the dataset support. In this work we operationalize the constraint via edges on S_D: downstream algorithms will only be permitted to attempt transitions from s to states s' that are certified to satisfy (1).

Since the environment dynamics are unknown, executability cannot be tested directly. We therefore introduce learned dynamics models: a forward model $\hat{f}$ that predicts the next state from a state–action pair, and an inverse model $\hat{I}$ that proposes an action intended to realize a desired state transition. Concretely, $\hat{f} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ and $\hat{I} : \mathcal{S} \times \mathcal{S} \to \mathcal{A}$ are trained from D (or from a replay buffer derived from D) using standard supervised objectives; the details of training are orthogonal to the subsequent graph construction, provided we can later calibrate predictive error. Given a candidate pair (s, s'), we form the *reachability witness residual*

$$\widehat{w}(s,s') \;:=\; d\big(\hat{f}(s, \hat{I}(s,s')),\, s'\big). \tag{2}$$

Intuitively, $\widehat{w}(s,s')$ is small when (i) $\hat{I}$ can identify an action that should move s toward s' and (ii) $\hat{f}$ agrees that executing that action lands near

s'. Crucially, $\widehat{w}$ alone is not a guarantee of true executability; it must be combined with a forward-model error certificate to control the probability of accepting spurious edges. We will phrase this certificate as a bound of the form $d(\hat{f}(s,a), T(s,a)) \leq U(s,a)$ holding simultaneously over all queried pairs (s,a), where U is obtained by calibration on held-out data and a global risk budget $\alpha \in (0,1)$.

The preceding definitions clarify the computational bottleneck. If we attempted to construct G^* (or even its ε-approximate variant) by enumerating all ordered pairs $(s,s') \in S_D \times S_D$ and testing (1) via model-based certification, then we would incur $\Theta(n^2)$ candidate checks, which is infeasible for modern offline datasets. At the same time, purely geometric graphs (e.g., connecting each s to its nearest neighbors under d) are not semantics-preserving: proximity in d is neither sufficient nor necessary for the existence of an action realizing the transition. Our aim is therefore to separate *candidate generation* from *acceptance*: we seek a procedure that (a) proposes, for each $s \in S_D$, a small set of plausible targets without scanning all of S_D, and (b) certifies that any accepted proposal is ε-approximately executable with a controlled global failure probability.

Formally, we will construct a sparse directed graph $\widehat{G}$ on vertex set S_D with out-degree at most $k \ll n$ per state. Candidate generation must be sublinear in n per source state, typically implemented via approximate nearest-neighbor retrieval in a learned embedding. Certification may be more expensive but is applied only to the $O(nk)$ proposed edges. The ensuing sections make these input–output requirements explicit, specify the admissible semantics of an edge in $\widehat{G}$ (as an ε-approximate executability claim), and introduce the risk parameter α governing simultaneous correctness of all accepted edges.

3 Problem Formulation: Certified Reachability Graph Construction

We now formalize the intermediate problem solved prior to planning or value optimization: from the finite set of dataset states S_D and learned models $(\hat{f}, \hat{I})$, we wish to construct a sparse directed graph whose edges have an explicit, certifiable semantics in the underlying MDP. The construction must avoid the $\Theta(n^2)$ cost of enumerating all ordered pairs of vertices, and it must control the probability of admitting any spurious edge across the entire output.

Inputs and outputs. The input consists of (i) the vertex set $S_D = \{s^{(1)}, \ldots, s^{(n)}\}$ (and optionally the full transition dataset D, used only for training/calibration of auxiliary models), (ii) a tolerance $\varepsilon \geq 0$ defining approximate executability, (iii) learned models $\hat{f}$ and $\hat{I}$ defining the witness residual $\widehat{w}$ in (2), (iv) a learned embedding map $\phi : \mathcal{S} \to \mathbb{R}^m$ together with an approximate nearest-neighbor (ANN) data structure $\mathcal{I}$ built over $\{\phi(s) : s \in S_D\}$, (v) an out-degree budget $k \in \mathbb{N}$, and (vi) a global risk budget $\alpha \in (0, 1)$ for certification. The output is a directed graph

$$\widehat{G} \;=\; (S_D,\, E(\widehat{G})), \qquad \widehat{\mathrm{Reach}}(s) \;:=\; \{\, s' \in S_D : (s \to s') \in E(\widehat{G})\,\},$$

such that $|\widehat{\mathrm{Reach}}(s)| \le k$ for all $s \in S_D$. We emphasize that $\widehat{G}$ is a *constructed* object: it is not assumed to coincide with the (unknown) true executability graph G^*, but its edges must carry a certified meaning.

Admissible edge semantics. An edge $(s \to s')$ in $\widehat{G}$ is intended to assert that s' is ε-approximately executable from s in the sense of (1). Since we cannot evaluate T directly, we restrict attention to edges whose executability is witnessed by the inverse model action proposal

$$\hat{a}(s, s') \;:=\; \hat{I}(s, s') \in \mathcal{A}.$$

Accordingly, we say that an edge $(s \to s')$ is *model-witnessed ε-executable* if

$$d\big(T(s, \hat{a}(s, s')),\; s'\big) \;\le\; \varepsilon. \tag{3}$$

This definition is asymmetric in (s, s') and is consistent with our downstream use: planning algorithms will interpret $(s \to s')$ as permission to attempt the action $\hat{a}(s, s')$ when at s, with the promise that the resulting next state lies within ε of s'.

Certification with a global risk budget. The witness residual $\widehat{w}(s, s')$ in (2) measures agreement between $\hat{f}$ and $\hat{I}$ but does not alone imply (3). We therefore introduce a calibrated forward-model error bound $U(s, a)$ such that, over a specified set of queried pairs (s, a), the event

$$\mathcal{E} \;:=\; \Big\{\forall (s, a) \in \mathcal{Q} :\; d(\hat{f}(s, a), T(s, a)) \le U(s, a)\Big\} \tag{4}$$

holds with probability at least $1 - \alpha$. Here $\mathcal{Q}$ denotes the (random, data-dependent) set of state–action pairs on which we will ever invoke certification during graph construction; in our setting $\mathcal{Q}$ is induced by retrieval (defined below) and inverse-model proposals. Under $\mathcal{E}$, triangle inequality yields the soundness implication

$$\widehat{w}(s, s') + U\big(s, \hat{a}(s, s')\big) \;\le\; \varepsilon \quad\Longrightarrow\quad d\big(T(s, \hat{a}(s, s')),\; s'\big) \;\le\; \varepsilon,$$

and thus motivates the certification rule used by our algorithm. The key point is that α is *global*: it bounds the probability that *any* accepted edge fails to satisfy (3), rather than providing a per-edge guarantee. Operationally, $\mathcal{E}$ may be obtained either by a simultaneous calibration procedure designed for a known upper bound on $|\mathcal{Q}|$ (e.g., $|\mathcal{Q}| \le nk$), or by a union bound over per-query bounds that allocate risk $\alpha/|\mathcal{Q}|$; our subsequent analysis is agnostic to the particular calibration mechanism provided (6) holds.

Embedding-based candidate generation and retrieval assumptions. To avoid exhaustive pairwise testing, we enforce that candidate next-states are generated by sublinear retrieval in a learned embedding. For each source state $s \in S_D$, the candidate set is defined as

$$C(s) \;:=\; \mathrm{ANN}_k(\mathcal{I}, \phi(s), k) \subseteq S_D, \qquad |C(s)| \leq k,$$

where ANN_k returns up to k approximate nearest neighbors of $\phi(s)$ among $\{\phi(\tilde{s}) : \tilde{s} \in S_D\}$. We do *not* assume that Euclidean proximity under d suffices for executability; rather, ϕ is intended to organize S_D so that truly executable targets are retrieved with high probability/recall. The only hard computational constraint we impose is that building $\mathcal{I}$ is near-linear in n (up to logarithmic factors) and that each query runs in time $C_{\mathrm{ann}} = \tilde{O}(1)$ or $\tilde{O}(\log n)$, so that all candidate generation across S_D is $\tilde{O}(n\, C_{\mathrm{ann}})$ rather than $\Theta(n^2)$.

Sparsity parameter k and the certified graph construction objective. The out-degree budget k simultaneously controls (i) the memory footprint $O(nk)$ of adjacency lists and (ii) the number of certification checks, which is at most nk. The problem is therefore to design a procedure that, for each s, uses a single ANN query to obtain $C(s)$, and then selects a subset $\widehat{\mathrm{Reach}}(s) \subseteq C(s)$ such that every accepted edge is sound under the global risk budget α. Subject to these constraints, we seek to maximize the inclusion of truly ε-executable targets (recall) while keeping the construction cost dominated by n retrievals and at most nk certifications. The next section instantiates this formulation via a concrete Retrieve-then-Certify algorithm and specifies how the calibration step realizes (6) in practice.

4 Algorithm: Retrieve-then-Certify (RTC)

We now instantiate the construction objective of the previous section via a concrete two-stage routine, *Retrieve-then-Certify* (RTC). RTC takes as input the dataset vertex set S_D, a retrieval budget k, learned models $(\hat{f}, \hat{I})$ defining the witness residual $\widehat{w}$, an embedding ϕ equipped with an ANN index $\mathcal{I}$, and calibration parameters (α, ε). It outputs adjacency lists $\widehat{\mathrm{Reach}}(s)$ that are sparse by construction and amenable to downstream graph-constrained planning and value optimization.

Stage A: learning representations and models. RTC presumes that we have trained three auxiliary components from D: (i) a forward model $\hat{f}(s, a)$, (ii) an inverse model $\hat{I}(s, s')$ (interpreted as an action proposal that intends to move s toward s'), and (iii) an embedding $\phi(s) \in \mathbb{R}^m$ used only for retrieval. The learning objective for ϕ is not required to coincide with the metric d used in certification; rather, ϕ is a statistical device for concentrating truly executable targets among the top-k retrieved candidates. Concretely, we may train ϕ by a temporal-contrastive objective on transitions in D (making s close to s' when $(s, \cdot, \cdot, s') \in D$ and far from negatives), by goal-conditioned imitation signals, or by any representation that empirically improves candidate recall under ANN queries. Since the certification step ultimately gates edges, the role of ϕ is to reduce the number of pairs we ever test.

Stage B: building the ANN index and performing candidate retrieval. Given ϕ and S_D, we build an ANN index $\mathcal{I}$ over $\{\phi(s) : s \in S_D\}$. For each source state $s \in S_D$, we generate a candidate set

$$C(s) \;=\; \mathrm{ANN}_k(\mathcal{I}, \phi(s), k) \subseteq S_D, \qquad |C(s)| \leq k,$$

using a single ANN query. This step enforces the sublinear candidate-generation constraint: we never scan all n potential targets for a given s. We also allow implementation-dependent filters at retrieval time (e.g., excluding s itself, enforcing diversity among candidates, or restricting to candidates within a coarse embedding radius), provided $|C(s)| \leq k$ remains intact.

Stage C: calibration of a forward-model error certificate. Before certifying any candidate edges, RTC computes a predictive error bound $U(s, a)$ for the forward model that is valid *simultaneously* over the set of queried pairs $\mathcal{Q}$ in (6) with failure probability at most α. Algorithmically, we treat calibration as a black box:

$$\textsc{Calibrate}(\hat{f}, \alpha, \text{data}) \;\longrightarrow\; U(\cdot, \cdot),$$

where the calibration data is disjoint from any data used to fit $\hat{f}$ (e.g., a held-out split of D). In the simplest uniform form, $U(s, a) \equiv U$ is a scalar chosen so that $d(\hat{f}(s, a), T(s, a)) \leq U$ holds for all $(s, a) \in \mathcal{Q}$ with probability $\geq 1 - \alpha$. More refined variants let U depend on (s, a) through stratification, quantile regression, or uncertainty estimates, but RTC only requires that the resulting U satisfies the simultaneous guarantee (6). The next section details concrete calibration mechanisms and their guarantees; here we only record that calibration is performed once per graph build (or once per refresh, if models are updated).

Stage D: batched certification and graph assembly. For each $s \in S_D$ and each candidate $s' \in C(s)$, we compute the proposed action and the witness residual

$$\hat{a}(s, s') \;=\; \hat{I}(s, s'), \qquad r_{\text{hat}}(s, s') \;=\; d\big(\hat{f}(s, \hat{a}(s, s')),\; s'\big) \;=\; \widehat{w}(s, s').$$

We accept $(s \to s')$ into $\widehat{G}$ if and only if

$$r_{\text{hat}}(s, s') \; + \; U\big(s, \hat{a}(s, s')\big) \; \le \; \varepsilon. \tag{5}$$

The resulting adjacency list is

$$\widehat{\text{Reach}}(s) \; = \; \Big\{ s' \in C(s) : (5) \text{ holds} \Big\},$$

which automatically satisfies $|\widehat{\text{Reach}}(s)| \le |C(s)| \le k$. In practice, we implement this loop in batches: we query ANN for a batch of sources, form all candidate pairs (s, s') in that batch, run $\hat{I}$ and $\hat{f}$ as large batched forward passes (GPU), and then apply the scalar test (5). Batching does not change the semantics of acceptance; it only reduces the amortized constant in C_{cert}.

Caching and incremental refresh. RTC admits a natural caching strategy. For each accepted edge we may store $(s, s', \hat{a}, r_{\text{hat}})$, and optionally also store the full candidate list $C(s)$. If $(\hat{f}, \hat{I}, \phi)$ are frozen, the graph is built once. If these components are trained jointly with downstream RL, we may refresh the graph periodically: every R gradient steps, rebuild $\mathcal{I}$ (or update it incrementally), re-retrieve $C(s)$, and re-certify. When only $\hat{f}$ changes, one may keep cached $C(s)$ and re-run certification; when ϕ changes, one must at least re-run retrieval.

Integration into graph-constrained offline RL. The output $\widehat{G}$ is consumed by graph-based state-constrained methods (e.g., graph-SCQL / StaCQ) by restricting the policy's feasible next-state choices at s to $\widehat{\text{Reach}}(s)$. Operationally, choosing an outgoing edge $(s \to s')$ corresponds to executing the recovered action $\hat{a}(s, s')$. In offline value optimization, backups are likewise restricted to successors in $\widehat{\text{Reach}}(s)$, ensuring that value propagation occurs only along edges that have passed the certification test. Thus RTC provides a computationally efficient interface between continuous-control dynamics (handled by $(\hat{f}, \hat{I})$ and calibration) and discrete graph planning (handled by $\widehat{G}$).

5 Theoretical Guarantees I: Certification

We formalize the sense in which RTC is a *sound* graph-construction routine: whenever it accepts an edge, that edge is guaranteed (with high probability) to be ε-approximately executable in the environment. The key point is that soundness is achieved by a *simultaneous* predictive error certificate for the forward model $\hat{f}$, combined with a union bound (or an equivalent multiple-testing control) over all state–action pairs that RTC ever evaluates.

Queried set and the simultaneous event. Fix the learned components $(\hat{f}, \hat{I}, \phi)$ and the retrieved candidate sets $\{C(s)\}_{s \in S_D}$. RTC evaluates $\hat{f}$ only on the finite set of pairs

$$\mathcal{Q} \;:=\; \big\{(s, \hat{I}(s, s')) :\; s \in S_D,\; s' \in C(s)\big\}. \tag{6}$$

In particular, $|\mathcal{Q}| \le nk$. We say that a function $U(s,a)$ is a *simultaneous error certificate* on $\mathcal{Q}$ at level α if

$$\mathbb{P}\Big(\forall (s,a) \in \mathcal{Q} :\; d\big(\hat{f}(s,a), T(s,a)\big) \le U(s,a)\Big) \;\ge\; 1-\alpha,$$

where the probability is over the calibration sample used to compute U (and any algorithmic randomness), while conditioning on the trained models and the retrieved candidates.

Soundness of the RTC acceptance test. Assume deterministic transitions T. For any candidate edge $(s \to s')$ we form $\hat{a} = \hat{I}(s,s')$ and compute the residual $r_{\text{hat}}(s,s') = d(\hat{f}(s,\hat{a}), s')$. If the test

$$r_{\text{hat}}(s,s') + U(s,\hat{a}) \le \varepsilon$$

accepts the edge, then on the simultaneous event we have, by the triangle inequality,

$$d\big(T(s,\hat{a}), s'\big) \;\le\; d\big(T(s,\hat{a}), \hat{f}(s,\hat{a})\big) + d\big(\hat{f}(s,\hat{a}), s'\big) \;\le\; U(s,\hat{a}) + r_{\text{hat}}(s,s') \;\le\; \varepsilon.$$

Thus, *conditional on the certificate holding on all queried pairs*, every accepted edge is ε-approximately executable. The entire question of correctness therefore reduces to constructing U so that this simultaneous event holds with probability at least $1-\alpha$.

Union bounds and global risk allocation. A simple way to obtain the simultaneous guarantee is to enforce a per-query failure probability δ and apply a union bound. Concretely, if we ensure that for each fixed $(s,a) \in \mathcal{Q}$,

$$\mathbb{P}\big(d(\hat{f}(s,a), T(s,a)) > U(s,a)\big) \;\le\; \delta,$$

then

$$\mathbb{P}\Big(\exists (s,a) \in \mathcal{Q} :\; d(\hat{f}(s,a), T(s,a)) > U(s,a)\Big) \;\le\; |\mathcal{Q}|\delta \;\le\; \alpha$$

whenever $\delta = \alpha/|\mathcal{Q}|$ (or any smaller choice). This accounting makes explicit why RTC isolates candidate generation: we may set δ using $|\mathcal{Q}| \le nk$ rather than n^2, thereby avoiding a catastrophic multiple-testing penalty.

Split conformal calibration (distribution-free, marginal). One practically robust mechanism is split conformal calibration on a held-out set $\mathcal{D}_{\text{cal}} = \{(s_j, a_j, s'_j)\}_{j=1}^{N_{\text{cal}}}$, disjoint from the data used to fit $\hat{f}$. Define calibration scores

$$e_j \;:=\; d\big(\hat{f}(s_j, a_j),\, s'_j\big),$$

and let $\hat{q}_{1-\delta}$ be the empirical $(1-\delta)$-quantile of $\{e_j\}$. Under exchangeability of the calibration triples with future test triples, split conformal yields

$$\mathbb{P}\big(d(\hat{f}(s,a), T(s,a)) \le \hat{q}_{1-\delta}\big) \;\ge\; 1-\delta$$

for a fresh (s, a) drawn from the same distribution. Setting $U(s, a) \equiv \hat{q}_{1-\delta}$ and choosing $\delta = \alpha/|\mathcal{Q}|$ gives a global $1-\alpha$ guarantee by the union bound above. We emphasize what is (and is not) obtained here: the guarantee is *marginal* over the calibration/test sampling distribution and is only as credible as the assumption that queried pairs resemble the calibration distribution.

Empirical Bernstein-style bounds (model-based, potentially tighter). When we are willing to assume bounded or sub-Gaussian errors, we may replace quantile calibration by concentration bounds that exploit empirical variance. For example, if the scores e_j are almost surely bounded by b and approximately i.i.d., an empirical Bernstein inequality yields an upper confidence bound on the mean error of the form

$$\bar{e} \;+\; \sqrt{\frac{2\hat{\sigma}^2 \log(2/\delta)}{N_{\text{cal}}}} \;+\; \frac{3b \log(2/\delta)}{N_{\text{cal}}},$$

and analogous constructions can be applied to conservative upper bounds on high quantiles via tail-modeling assumptions. Such bounds can be substantially less conservative than worst-case or high-quantile conformal bounds, but they are correspondingly less assumption-free.

Limitations and non-guarantees. First, certification controls *false positives* (unsound edges), not *false negatives*: RTC may reject executable edges if $\hat{I}$ fails to propose a suitable action, if retrieval omits the true target, or if U is overly conservative. Second, the guarantee is tied to the metric d and the tolerance ε; it does not imply semantic equivalence of states beyond $d(\cdot, \cdot)$. Third, distribution shift matters: if the queried pairs $\mathcal{Q}$ are atypical relative to calibration data, split conformal guarantees may degrade. Finally, if the environment is stochastic, the deterministic statement must be replaced by a probabilistic one (e.g., bounding $d(s'_{\text{env}}, s')$ in high probability), and the certificate must account for both model error and inherent transition noise.

6 Theoretical Guarantees II: End-to-End RL Impact

We now quantify how ε-approximate executability of graph edges translates into end-to-end performance guarantees for any offline RL procedure that plans or optimizes over the constructed graph $\widehat{G}$. The guiding observation is that, once the policy is constrained to select edges that can be realized up to d-error at most ε, the induced mismatch between the *intended* dataset-state successor and the *actual* environment successor can be propagated through the value function by a Lipschitz continuity assumption.

Oracle graph and graph-constrained policies. Let $G^\star$ denote the directed graph on vertex set S_D containing exactly the executable dataset edges: $(s \to s') \in E(G^\star)$ if and only if there exists an action a such that $T(s,a) = s'$ (or $d(T(s,a), s') = 0$ under our metric). A graph-constrained policy is any mapping π that, at state $s \in S_D$, selects an outgoing edge (equivalently, a target dataset successor) among the available adjacency list, and then executes the recovered action $\hat{I}(s,s')$. We write $\pi_{\widehat{G}}$ for the policy returned by graph-SCQL (or any other graph-optimizing method) when restricted to $\widehat{G}$, and $\pi_{G^\star}$ for the corresponding oracle policy when restricted to $G^\star$.

Value loss under approximate executability. Assume that every edge $(s \to s') \in E(\widehat{G})$ is ε-approximately executable in the sense that executing $\hat{a} = \hat{I}(s,s')$ yields an environment successor $\tilde{s}'$ satisfying $d(\tilde{s}', s') \le \varepsilon$. Assume further that the oracle value function is L-Lipschitz:

$$|V^{\pi_{G^\star}}(s) - V^{\pi_{G^\star}}(\tilde{s})| \;\le\; L\, d(s,\tilde{s}) \qquad \text{for all } s, \tilde{s} \in \mathcal{S}.$$

Then the performance degradation due solely to approximate landing can be bounded uniformly over dataset states.

Theorem 6.1 (Value loss from approximate executability)**.** *Under the assumptions above, for all $s \in S_D$,*

$$V^{\pi_{G^\star}}(s) - V^{\pi_{\widehat{G}}}(s) \;\le\; \frac{\gamma L \varepsilon}{1-\gamma}.$$

Proof sketch (coupling). We couple two trajectories, both starting at the same $s \in S_D$. The oracle trajectory follows $\pi_{G^\star}$ and lands exactly at the intended next dataset state $s_1^\star$. The approximate trajectory follows $\pi_{\widehat{G}}$ and intends some s_1 but lands at $\tilde{s}_1$ with $d(\tilde{s}_1, s_1) \le \varepsilon$. Writing Bellman recursions and subtracting, the immediate rewards cancel under the standard modeling choice that rewards are evaluated at the pre-transition state (or are otherwise Lipschitz and can be treated similarly), leaving a one-step discrepancy bounded by $\gamma |V^{\pi_{G^\star}}(s_1) - V^{\pi_{G^\star}}(\tilde{s}_1)| \le \gamma L\varepsilon$. Iterating along the coupled rollouts yields a geometric series $\sum_{t\ge 1} \gamma^t L\varepsilon = \gamma L\varepsilon/(1-\gamma)$.

False negatives versus extra edges. Theorem 6.1 isolates the effect of *edge imprecision* (approximate executability) and does not require $\widehat{G}$ to contain all edges of $G^\star$. Missing edges (false negatives) are therefore qualitatively different from extra edges (false positives):

- *Missing edges* shrink the feasible action set and can only reduce the optimal achievable value under a graph constraint, even if every retained edge is sound. Any bound must therefore depend on a coverage condition describing how well $\widehat{G}$ approximates the oracle reachable set.
- *Extra edges* are benign *provided they are sound*: enlarging the feasible set cannot hurt the optimal value of the constrained planning problem. Thus, once certification ensures that accepted edges are ε-executable, adding more such edges can only help optimization (though it may increase variance or computation in practice).

A simple sufficient condition for robustness to false negatives is an approximate successor coverage property: for each $s \in S_D$ and each oracle edge $(s \to s^\star) \in E(G^\star)$, there exists $\hat{s} \in \widehat{\mathrm{Reach}}(s)$ such that $d(s^\star, \hat{s}) \le \varepsilon_{\mathrm{cov}}$. Under the same Lipschitz assumption, repeating the coupling argument with ε replaced by $\varepsilon + \varepsilon_{\mathrm{cov}}$ yields the bound

$$V^{\pi_{G^\star}}(s) - V^{\pi_{\widehat{G}}}(s) \;\le\; \frac{\gamma L(\varepsilon + \varepsilon_{\mathrm{cov}})}{1-\gamma},$$

where ε accounts for approximate execution and $\varepsilon_{\mathrm{cov}}$ accounts for approximating the oracle successor by a nearby available node.

A pessimistic variant for residual uncertainty. If we are unwilling to treat ε as fully deterministic (e.g., because certificates are conservative but not absolute, or because transitions are mildly stochastic), we may incorporate pessimism directly into graph-based backups. Concretely, for an edge intending successor s', the next state is known only to lie in the ball $B(s', \varepsilon)$, so the worst-case continuation value is at least

$$\inf_{\tilde{s} \in B(s', \varepsilon)} V(\tilde{s}) \;\ge\; V(s') - L\varepsilon.$$

Thus, a conservative Bellman backup may subtract $\gamma L\varepsilon$ per step (or use edge-dependent radii $\varepsilon(s, s')$ when available). This modification preserves the same functional form of the bound while reducing sensitivity to occasional miscalibration, at the cost of additional pessimism when ε is large.

7 Complexity Landscape: what RTC can and cannot do at scale

We now isolate the computational bottlenecks of Retrieve–then–Certify (RTC) and place them in a broader landscape of lower bounds which explain why subquadratic graph construction requires approximation or additional structure.

Bookkeeping and dominant operations. Let $n := |S_D|$ denote the number of unique dataset states, let m be the embedding dimension for ϕ, and let k be the out-degree budget per state. RTC consists of three primitive operations: (i) building an ANN index $\mathcal{I}$ over $\{\phi(s) : s \in S_D\}$, (ii) performing n ANN queries to retrieve k candidate successors per state, and (iii) certifying up to nk candidate edges via a constant number of evaluations of $(\hat{f}, \hat{I})$ plus a distance computation in d. Denoting by C_{ann} the amortized cost of one ANN query and by C_{cert} the amortized cost of certifying one candidate edge, we obtain the construction time

$$\tilde{O}(n\,C_{\text{ann}} \;+\; nk\,C_{\text{cert}})\,, \tag{7}$$

where $\tilde{O}(\cdot)$ suppresses polylogarithmic factors from index maintenance, batching overheads, and (when applicable) approximate search parameters such as $(1+\eta)$-approximation.

The quantity C_{cert} is model-dependent but conceptually constant with respect to n: certification requires computing $a_{\text{hat}} = \hat{I}(s, s')$, then $\hat{f}(s, a_{\text{hat}})$, then the residual $\widehat{w}(s, s') = d(\hat{f}(s, a_{\text{hat}}), s')$, and finally evaluating the acceptance test against ε after accounting for the calibration bound. In practice, $nk\,C_{\text{cert}}$ is highly parallelizable across candidate edges and typically dominates wall-clock time once n is large.

Memory footprint. RTC stores (a) the ANN index and embeddings, (b) the adjacency lists, and (c) model parameters. The index requires $\tilde{O}(nm)$ memory in typical implementations (e.g., graph-based ANN or IVF variants) plus any auxiliary quantization tables. The output graph occupies $O(nk)$ memory (up to constant factors for storing integer vertex ids and, optionally, edge metadata such as $\widehat{w}(s, s')$ or the recovered action). Hence the overall memory is

$$\tilde{O}(nm) \;+\; O(nk) \;+\; \text{size}(\hat{f}, \hat{I}), \tag{8}$$

which should be contrasted with the $O(n^2)$ adjacency representation implicitly targeted by exact all-pairs scanning.

Output-size lower bound and tightness. The upper bound (7) is essentially optimal once we insist on explicitly outputting k outgoing edges per state. Indeed, in the word-RAM model, writing down nk edges already forces $\Omega(nk)$ time, irrespective of how candidates are generated or verified. Formally, any algorithm that outputs adjacency lists $\widehat{\mathrm{Reach}}(s)$ with $|\widehat{\mathrm{Reach}}(s)| = k$ for all $s \in S_D$ must write $\Theta(nk)$ machine words (vertex ids), implying a time lower bound

$$\Omega(nk). \tag{9}$$

Consequently, when $C_{\mathrm{cert}} = \Theta(1)$ (or bounded independently of n via constant-depth networks and fixed-precision arithmetic), RTC matches the output-size lower bound up to logarithmic factors and any additional constant overhead from approximate retrieval.

Why exact neighborhood construction is conditionally quadratic. One might ask whether the ANN stage is merely an implementation choice, and whether we could instead compute, for every $s \in S_D$, the *exact* set of dataset successors within a geometric tolerance, e.g.,

$$\{s' \in S_D : \ d(s', T(s,a)) \le \varepsilon \text{ for some } a\},$$

or even the exact ε-neighborhood graph of the embedded points $\{\phi(s)\}$. In high ambient dimension m, such exact constructions are widely believed to require essentially quadratic time in the worst case. A representative conditional statement is that for $m = \omega(\log n)$, deciding all pairs (i, j) with $\|\phi(s_i) - \phi(s_j)\|_2 \le \varepsilon$ (the exact ε-graph) cannot be done in $n^{2-o(1)}$ time under fine-grained conjectures such as SETH via reductions from Orthogonal Vectors. While the reductions are geometric and do not use RL structure, they serve as a computational barrier: if we demand worst-case exactness in high dimension, we should not expect to beat exhaustive pairwise checks by more than subpolynomial factors.

Necessity of approximation and retrieval assumptions. The preceding considerations justify the architectural separation in RTC:

- *Candidate generation must be sublinear in n per state.* ANN retrieval provides this by exploiting empirical structure (e.g., low intrinsic dimension of $\phi(S_D)$ or favorable clusterability), yielding $C_{\mathrm{ann}} \ll n$ in regimes of interest.
- *Correctness must not depend on retrieval being perfect.* Since high-dimensional exact neighborhood queries are hard, we treat retrieval as a *recall* mechanism and place all soundness requirements in the certification stage. In particular, missed neighbors (false negatives) affect coverage but do not invalidate any accepted edge.
- *Certification should be constant-time per candidate.* This is precisely what makes the overall time essentially proportional to the number of tested edges, nk, which is unavoidable by (9).

In this sense, approximation is not a superficial optimization but a prerequisite for scalability: without approximate near-neighbor search (or strong additional assumptions on the geometry of S_D), any method that even *identifies* a useful set of candidate successors for each of n states risks reverting to $\Omega(n^2)$ probing in the worst case. RTC expends its computational budget where it matters—on certifying a small, explicitly represented set of edges—and thereby remains compatible with modern large-scale offline RL datasets.

8 Experimental Plan (Implementation-Strengthening)

We design experiments to validate RTC along three axes: (i) scaling of graph construction time and memory as a function of n, m, and k; (ii) soundness/coverage tradeoffs induced by certification and retrieval; and (iii) end-to-end offline RL performance when graph-SCQL optimizes over the certified graph $\widehat{G}$.

Benchmarks and data regimes. We select domains that expose both low-level control and long-horizon stitching needs. Concretely, we consider (a) standard continuous-control offline RL suites with state observations (e.g., locomotion and manipulation) at multiple dataset sizes; (b) large-log robotics and driving regimes in which n is large and trajectory diversity is high (e.g., multi-task manipulation logs and logged driving trajectories with diverse initial conditions); and (c) pixel-observation variants where s is an image (or image-history) and ϕ must be learned. For each domain we construct S_D by deduplicating states up to an application-appropriate tolerance (exact equality for discrete/quantized states; clustering for continuous states when raw logs contain near-duplicates), and we report both N and $n = |S_D|$.

Implementation of retrieval and certification. We implement ϕ as either (i) an MLP encoder for state vectors, (ii) a convolutional encoder for pixels, or (iii) a frozen pretrained encoder (when available) followed by a learned projection to $\mathbb{R}^m$. We instantiate $\mathcal{I}$ with at least two ANN backends (e.g., HNSW and IVF-PQ) to separate algorithmic effects from index engineering, and we record build time, query time C_{ann}, and peak memory. Certification uses learned $(\hat{f}, \hat{I})$ trained on D with held-out calibration to obtain a simultaneous error bound U at risk budget α. We ensure that calibration and evaluation splits are disjoint at the trajectory level to avoid temporal leakage. We batch certification over candidate edges to saturate GPU throughput and treat C_{cert} as an empirically measured quantity (including model forward passes and metric computation).

Scaling benchmarks: time and memory. To test the predicted scaling $\tilde{O}(nC_{\text{ann}} + nkC_{\text{cert}})$, we sweep n by subsampling trajectories and/or taking prefixes of large logs, and we sweep $k \in \{4, 8, 16, 32, 64\}$ and $m \in \{16, 32, 64, 128, 256\}$. For each configuration we report: index build time, total retrieval time (all n queries), total certification time (all tested edges), and total wall-clock time with fixed hardware. We also report memory usage decomposed into index memory $\tilde{O}(nm)$, adjacency memory $O(nk)$, and model parameters. As a sanity check, we include a naive baseline that scans all $s' \in S_D$ for a small subset of source states (due to cost) to illustrate the empirical gap to quadratic construction.

Soundness and coverage diagnostics. We evaluate the certification layer independently of policy learning by sampling accepted edges $(s \to s') \in \widehat{G}$ and executing the recovered action $\hat{a} = \hat{I}(s, s')$ in the environment (or a trusted simulator) to measure realized deviation $d(\tilde{s}', s')$. We report the empirical violation rate of the ε-executability condition and compare it to the target risk budget α (with confidence intervals). Separately, we estimate *coverage/recall* by defining a set of "ground-truth" executable targets for a manageable subset of sources using extensive action sampling or short-horizon planning, and measuring what fraction appear in $C(s)$ and what fraction are ultimately certified. This separates retrieval failures (false negatives in $C(s)$) from certification conservatism (false negatives due to U or ε).

End-to-end offline RL evaluation. We compare graph-SCQL using $\widehat{G}$ against strong offline RL baselines that do not explicitly construct reachability graphs (e.g., value-based and actor-critic methods with behavioral regularization), using standardized evaluation protocols and reporting normalized return and median over multiple random seeds. We also include internal baselines to isolate contributions: (i) graph-SCQL with retrieval-only (no certification), (ii) graph-SCQL with certification but using a non-learned embedding (e.g., raw state or PCA), and (iii) graph-SCQL with a dense candidate set for small n (approximating full scan) to estimate an upper envelope. For long-horizon tasks we additionally report success rates and trajectory-level constraint violations when applicable.

Ablations and engineering choices. We perform targeted ablations to quantify sensitivity and guide deployment: (a) out-degree budget k (compute–coverage tradeoff); (b) embedding dimension m (index cost vs retrieval quality); (c) certification threshold parameters, including ε and the calibration risk allocation α (soundness–acceptance tradeoff); (d) choice of metric d (raw ℓ_2 in state space versus latent-space distances); and (e) caching strategies. For caching, we evaluate (i) memoizing $\hat{I}(s, s')$ for repeated candidate pairs across index refreshes, (ii) caching $\phi(s)$ and batched ANN queries, and (iii) storing per-edge metadata (e.g., $\widehat{w}(s, s')$ and $U(s, \hat{a})$) to avoid recomputation during RL updates. We report the resulting changes in throughput (edges certified per second), peak memory, and final return, thereby clarifying which costs are one-time preprocessing versus iterative training overhead.

Reporting. Across all experiments we log acceptance rates $|\widehat{\mathrm{Reach}}(s)|/k$, degree distributions, and the empirical distribution of $\widehat{w}(s, s')$ for candidates and accepted edges. These diagnostics allow us to attribute performance differences to retrieval, certification, or downstream optimization, rather than to incidental hyperparameter choices.

Offline RL with behavioral regularization. A large fraction of offline RL work addresses the distribution-shift pathology by constraining the learned policy toward the dataset behavior. Representative methods include BCQ **?** (implicit action constraints via a generative model), TD3+BC **?** (explicit behavior cloning regularization in actor updates), and IQL **?** (implicitly conservative policy improvement via expectile regression). Closely related are CQL-style conservative value learning **?** and advantage-weighted regression variants (e.g., AWAC **?**). These methods typically avoid explicit combinatorial reasoning over dataset states; instead, they control extrapolation in action space or in the value function. Our construction is orthogonal: we build an explicit sparse directed graph over dataset states and enforce state-constraint structure through certified approximate executability of edges. In particular, the key distinction is that our guarantee is stated on *state reachability* for accepted edges, rather than on policy divergence from the behavior distribution.

Model-based offline RL and uncertainty. Model-based offline RL methods such as MOPO **?**, MOReL **?**, and COMBO **?** use a learned dynamics model together with uncertainty penalties or pessimism to mitigate compounding model error. The common thread is to restrict planning to regions where the model is believed to be accurate, typically by penalizing high-uncertainty rollouts (often estimated by ensembles). Our use of a learned forward model $\hat{f}$ is more narrowly scoped: we do not attempt long-horizon rollout under $\hat{f}$; rather, we use $\hat{f}$ only to *certify one-step approximate executability* of candidate edges $(s \to s')$ proposed by an inverse model $\hat{I}$. This design reduces the certificate target to a one-step statement of the form $d(T(s, \hat{I}(s, s')), s') \leq \varepsilon$, and it allows us to allocate a global risk budget α across a finite queried set $\mathcal{Q}$ of state-action pairs. In this sense, our certificate layer is compatible with pessimism-based model learning, but it requires a different calibration object: a simultaneous bound on one-step prediction error over the queried set, rather than an uncertainty heuristic integrated into multi-step planning.

Conformal prediction and calibration for certificates. The statistical structure of our certification step is aligned with conformal prediction and post-hoc calibration, which provide finite-sample, distribution-free coverage guarantees under exchangeability assumptions **??**. Recent work has applied conformal ideas to dynamics models and safe decision-making by calibrating prediction sets or error radii, yielding guarantees that hold with a user-specified failure probability. In our setting, we require a bound $U(s, a)$ such that $d(\hat{f}(s, a), T(s, a)) \leq U(s, a)$ holds simultaneously over all queried $(s, a) \in \mathcal{Q}$ with probability at least $1 - \alpha$. While a union bound over individually calibrated pairs is conceptually sufficient, it is often loose; more refined simultaneous calibration techniques (e.g., split-conformal with max-residual calibration over the query set) can yield tighter thresholds at the same α. Our contribution here is not a new conformal method per se, but an algorithmic placement of such calibration within a retrieve-then-certify pipeline whose output is a sparse reachability graph used by downstream state-constrained optimization.

Approximate nearest neighbor search and graph construction. Our candidate generation stage is an instance of approximate nearest neighbor (ANN) retrieval in an embedding space, using index structures such as HNSW, IVF-PQ, or LSH **???**. ANN has been extensively used to accelerate kNN classification, retrieval-augmented models, and approximate geometric graph construction. From the perspective of graph algorithms, our procedure resembles building a directed kNN graph on S_D in ϕ-space, except that we do not interpret proximity as reachability; instead, proximity is only a heuristic to propose candidates that are then filtered by a dynamics-based certificate. This separation is essential: exact ε-neighborhood graphs in high-dimensional ℓ_2 are known to be computationally expensive in the worst case, and ANN methods trade worst-case exactness for practical sublinear queries. Our analysis accordingly treats retrieval quality (recall) as an empirical factor affecting coverage, while the correctness of accepted edges is delegated to certification.

State similarity, representation learning, and bisimulation. A related line of work studies state abstractions and similarity metrics (including bisimulation metrics) that preserve value-relevant structure **??**. These methods aim to learn representations in which metric proximity implies similarity of transition and reward structure, enabling generalization and compression. Our embedding ϕ serves a different purpose: it is a computational instrument for retrieving a small candidate set $C(s)$ from a large discrete set S_D. We do not assume that ϕ induces a bisimulation metric, nor do we require that $d(\phi(s), \phi(s'))$ upper bound true one-step reachability. Indeed, any such assumption would be fragile under representation collapse or task mismatch;

instead, the role of ϕ is to improve candidate recall under a fixed budget k, while certification enforces the reachability condition in the original metric $d(\cdot, \cdot)$ (or another explicitly chosen metric).

Trajectory stitching, planning over datasets, and skill composition. Finally, our graph perspective connects to trajectory stitching and dataset-based planning, where one seeks to compose short behavioral fragments into long-horizon behavior by selecting intermediate states that are "connectable" **??**. Prior approaches often use learned models, goal-conditioned policies, or nearest-neighbor heuristics to link states across trajectories, sometimes without formal executability guarantees. Our retrieve-then-certify construction can be viewed as providing a principled substrate for stitching: the certified edges define admissible one-step connectors between dataset states, and graph-SCQL (or other graph-based methods) can then optimize long-horizon objectives under these admissibility constraints. The novelty is the combination of (i) sublinear candidate generation at scale, (ii) explicit global risk control via calibration, and (iii) a value-loss guarantee that depends only on ε and a Lipschitz constant, rather than on unrolled model accuracy.

9 Discussion and Limitations

When and why certificates can fail. Our executability guarantee is conditional: it holds on the event that the forward-model error bound is simultaneously valid over the queried set $\mathcal{Q} = \{(s, \hat{I}(s, s')) : s \in S_D,\ s' \in C(s)\}$. If the calibration procedure underestimates the worst-case residual on $\mathcal{Q}$, then the test $\widehat{w}(s, s') + U(s, \hat{I}(s, s')) \leq \varepsilon$ may accept an edge whose true landing state $T(s, \hat{I}(s, s'))$ is farther than ε from s'. This is the only failure mode relevant to the stated correctness statement: retrieval errors or inverse-model errors affect which edges are *considered* or *accepted*, but they do not, by themselves, invalidate an accepted edge unless they push $\mathcal{Q}$ into regions where the certificate is miscalibrated.

Out-of-distribution (OOD) queries and risk allocation. The simultaneous guarantee is strongest when the calibration residuals and the queried residuals are exchangeable (or when the calibration scheme explicitly targets worst-case coverage on a specified query family). In practice, $\mathcal{Q}$ is induced by the learned inverse model and by the ANN retrieval policy; both can drift during training, and both can generate state–action pairs that are poorly represented in the calibration split. This creates an OOD problem in *action space* even if the states remain within S_D: $\hat{I}(s, s')$ may propose actions outside the dataset support at s, and $\hat{f}$ may be least accurate precisely there. A second, more prosaic issue is risk allocation: even if each query were individually controlled at level $\alpha/|\mathcal{Q}|$, the resulting bound can be too loose to admit useful edges. Thus the empirical utility of RTC depends on simultaneously (i) keeping $\mathcal{Q}$ within the regime where $\hat{f}$ is accurate and (ii) calibrating U tightly enough to permit nontrivial out-degree.

Representation collapse and retrieval-induced blind spots. The ANN stage is deliberately heuristic: ϕ need only be useful for candidate recall under a fixed budget k. If ϕ collapses (e.g., maps many states to near-identical embeddings) or becomes misaligned with one-step controllability, then $C(s)$ will omit most truly executable neighbors. In this case the method remains *sound*—certification can still reject false positives—but it becomes incomplete, yielding a graph $\widehat{G}$ with small or disconnected components. This incompleteness is not repaired by calibration, since calibration controls false acceptance rather than false rejection. Consequently, RTC inherits an implicit coverage assumption: for each s of interest there exist executable successors in S_D that are retrieved with non-negligible probability by ANN at budget k. When this assumption fails, the downstream optimizer is constrained by an impoverished feasible set regardless of value-function accuracy.

Assumptions needed for the stated correctness statements. Theorems that assert $d(T(s, \hat{I}(s, s')), s') \leq \varepsilon$ for accepted edges require (a) a deterministic transition map T (or an explicitly chosen notion of stochastic executability), (b) a metric d on states (or latent states) for which ε-closeness is semantically meaningful for control, and (c) a valid simultaneous prediction-error certificate for $\hat{f}$ on $\mathcal{Q}$ at global level $1 - \alpha$. None of these is innocuous. In partially observed settings, d in observation space can be misleading; in such cases one must specify a representation in which one-step control closeness implies downstream value closeness, or else the Lipschitz assumption used in value-loss bounds is not justified. Likewise, for stochastic dynamics, a pointwise bound $d(\hat{f}(s, a), T(s, a)) \leq U(s, a)$ is not well-posed unless $T(s, a)$ is interpreted as a conditional mean; a more appropriate target is a high-probability statement $\mathbb{P}(d(S', s') \leq \varepsilon \mid s, a) \geq 1 - \delta$, which changes both calibration and downstream guarantees.

Practical limitations at 2026 scale. While construction is subquadratic, the constant factors can be material: certification entails forward passes through $\hat{f}$ and $\hat{I}$ for up to nk candidates, and memory must accommodate the index and adjacency lists. Moreover, the graph is only as good as the state set S_D: if the dataset provides sparse coverage of the reachable manifold, then even perfect retrieval and perfect certification yield a graph whose best paths are long detours or do not exist. Finally, if the environment admits action constraints or discontinuities (e.g., contact dynamics), small forward-model residuals may not correlate with true executability unless the model class is sufficiently expressive and trained with appropriate objectives; RTC does not remove the need for careful model learning.

Future directions: state-only datasets and multi-step reachability. A natural extension is to settings where the dataset records only states (or observations) without actions. One approach is to replace $\hat{I}$ with an *action proposal* module trained from additional interaction, from a simulator, or from weakly supervised signals (e.g., temporal adjacency and controllability priors), and to reinterpret certification as verifying the existence of an action achieving $s \to s'$ within ε. Another direction is to move beyond one-step edges. Multi-step reachability can be handled by composing certified edges, but naive composition accumulates slack: a path of length H can incur $O(H\varepsilon)$ deviation without additional structure. Tight multi-step guarantees likely require contraction or incremental re-anchoring to dataset states (e.g., re-projecting to the nearest certified node at each step), as well as calibration procedures that control error over adaptively chosen query sequences rather than a fixed finite set. These extensions preserve the same organizing principle: retrieve a small candidate set, certify with explicit risk control, and expose a sparse combinatorial object that downstream optimization can exploit.

10 Conclusion

We have presented a *Retrieve–then–Certify* (RTC) construction for turning a large offline dataset D into a sparse directed reachability graph $\widehat{G}$ over the unique dataset states S_D. The construction is deliberately modular: an embedding ϕ and ANN index $\mathcal{I}$ provide a sublinear candidate-generation mechanism, while learned dynamics surrogates $(\hat{f}, \hat{I})$ supply a quantitative witness of one-step executability,

$$\widehat{w}(s,s') \; := \; d\big(\hat{f}(s, \hat{I}(s,s')),\, s'\big),$$

that can be combined with an explicit prediction-error certificate U to decide edge acceptance. The resulting object $\widehat{G}$ is a combinatorial surrogate for one-step controllability on the support of the data, and it is precisely this surrogate that enables state-constrained planning and optimization at scales where an all-pairs construction is infeasible.

The core technical point is that the soundness of accepted edges is separated from the heuristic nature of retrieval. RTC is permitted to be incomplete—it may miss many executable neighbors when k is small—yet it can still be *correct* in the sense that every accepted edge comes with a global risk-controlled certificate. Concretely, under a simultaneous forward-model error event over the queried set

$$\mathcal{Q} \; = \; \{(s, \hat{I}(s,s'')) : \; s \in S_D, \; s'' \in C(s)\},$$

the acceptance test $\widehat{w}(s, s'') + U(s, \hat{I}(s, s'')) \leq \varepsilon$ implies $d(T(s, \hat{I}(s, s'')), s'') \leq \varepsilon$ for all edges $(s \to s'') \in \widehat{G}$, with overall failure probability at most α. Thus RTC provides a simple interface: any improvement to retrieval quality affects recall and downstream performance, but it does not alter the meaning of an accepted edge so long as the certificate remains valid on $\mathcal{Q}$.

This certificate-centric view makes the graph useful for downstream state-constrained offline RL. Once we restrict attention to policies that move along edges of $\widehat{G}$, dynamic programming or Q-learning variants (such as graph-SCQL) can optimize over an explicit feasible set of state transitions, rather than relying on unconstrained action selection that may leave the dataset support. Moreover, when the oracle value function is L-Lipschitz in the metric d, approximate executability of edges translates directly into an end-to-end value-loss bound of order $\frac{\gamma L \varepsilon}{1-\gamma}$. In this way, the parameter ε plays a transparent algorithmic role: it is the tolerance at which we trade off graph connectivity against conservative soundness, and it enters the final performance guarantee in a quantitatively interpretable manner.

From a systems perspective, RTC is designed to match the computational realities of 2026-scale datasets. The construction cost decomposes into an index build over $\{\phi(s) : s \in S_D\}$, n ANN queries to obtain candidate sets $C(s)$ of size at most k, and at most nk certifications. This yields total expected time $\tilde{O}(n\, C_{\text{ann}} + nk\, C_{\text{cert}})$ and space $\tilde{O}(nm + nk)$, with the expensive component—evaluation of $(\hat{f}, \hat{I})$ during certification—being embarrassingly parallel. In particular, the method avoids the quadratic barrier inherent in any exact all-pairs neighborhood construction in high dimension. At the same time, the output is a graph with $O(nk)$ edges, which is the correct size for a downstream optimizer that must, at minimum, read and write nk adjacency entries.

Indeed, the construction time is tight in the natural word-RAM model: any procedure that produces k outgoing edges for each of n states must spend $\Omega(nk)$ time simply to materialize the output, irrespective of how candidates are generated. RTC therefore attains the optimal scaling up to logarithmic factors when the amortized certification cost is constant (or dominated by a constant number of model forward passes). This tightness is not merely aesthetic; it clarifies where further improvements can and cannot come from. Speedups must target constant factors (better batching, cheaper witnesses, lighter models) or problem structure (smaller effective k, shared candidates, intrinsic low dimension), rather than hoping for an asymptotic improvement over the nk term.

More broadly, RTC formalizes a pathway from continuous control to tractable combinatorial optimization on dataset support. By exposing a sparse, certified reachability structure, we can decouple concerns that are typically entangled in offline RL: representation learning affects retrieval, model learning affects witness quality and certification slack, and policy learning operates on a constrained graph whose edges have an explicit operational meaning. This decoupling is what makes the approach adaptable: one may swap ANN backends, refine ϕ to improve recall, replace the witness with alternative consistency checks, or allocate risk α differently, without changing the basic retrieve–certify–optimize loop.

We view the principal contribution as providing a scalable and analyzable primitive for state-constrained offline RL: construct $\widehat{G}$ subquadratically, certify its edges with explicit risk control, and optimize on the resulting sparse object with performance guarantees parameterized by $(\varepsilon, \alpha, L, \gamma)$. In regimes where datasets are large but not dense enough to support unconstrained action optimization, this primitive supplies an actionable middle ground: it preserves the computational advantages of graph-based planning while retaining a clear, probabilistic notion of executability tied to the environment dynamics.