DYNASTO: VALIDITY-AWARE DYNAMIC–STATIC PARAMETER OPTIMIZATION FOR AUTONOMOUS DRIVING TESTING

Dmytro Humeniuk, Mohammad Hamdaqa, Houssem Ben Braiek,
Amel Bennaceur, Foutse Khomh

Polytechnique Montréal · The Open University

Presented by **Houssem Ben Braiek**

ICST 2026
IEEE International Conference on
Software Testing, Verification and Validation

19 May 2026

POLYTECHNIQUE MONTRÉAL

Robotaxi market is expected to grow from ~$3 billions in 2025 to ~$33 billion in 2030



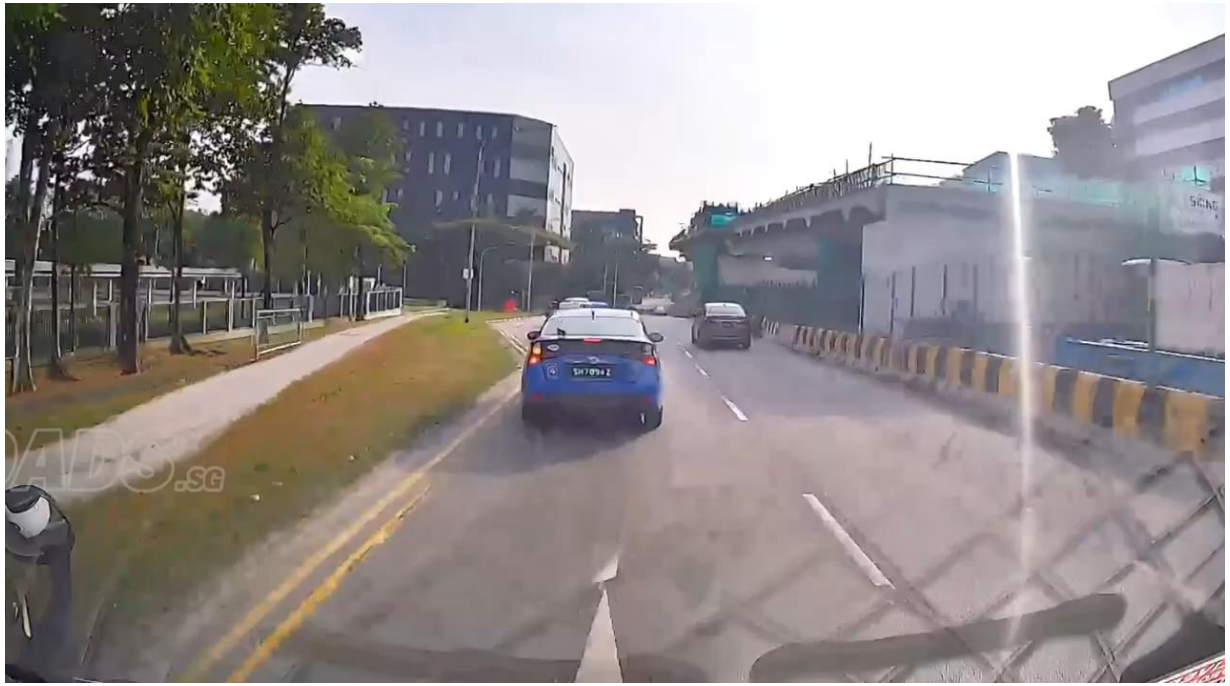
Source: The Business Research Company, Robotaxi Global Market Report (2025)

681 crashes reported for autonomous vehicles during testing in California in 2020 - 2025



Source: Berkeley TIMS Autonomous Vehicle Safety Dashboard

Collisions often arise from dynamic interactions between the traffic participants



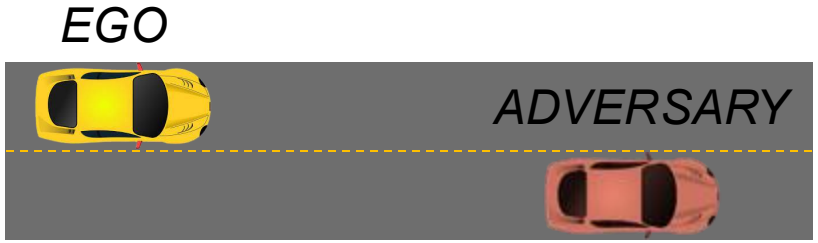
Pokorny, Petr, and Alena Høye. "Descriptive analysis of reports on autonomous vehicle collisions in California: January 2021–June 2022." *Traffic Safety Research* 2 (2022): 000011-000011.

Why Simulation-Based ADS Testing Needs Dynamic Scenarios

• Autonomous driving systems **operate** in **environments** with **moving traffic** participants.

• **Safety-critical** cases often **emerge** from **interactions**: cut-ins, braking, lane changes, and following behavior.

• **Simulation** lets us repeatedly **generate** rare and **risky scenarios** without **real-world danger**.



Modeling Dynamic Scenarios is Challenging

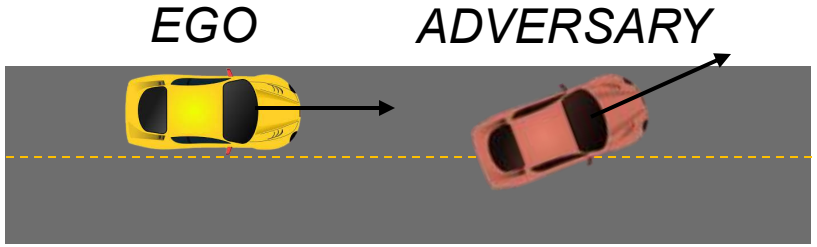
Initialization



$x_{ego}(0), y_{ego}(0)$
$v_{ego}(0), \varphi_{ego}(0)$

$x_{adv}(0), y_{adv}(0)$
$v_{adv}(0), \varphi_{adv}(0)$

Execution



$x_{ego}(t), y_{ego}(t)$
$v_{ego}(t), \varphi_{ego}(t)$

$x_{adv}(t), y_{adv}(t)$
$v_{adv}(t), \varphi_{adv}(t)$

Solution families

Static parameter search

- Produce **initial states** and/or **adversary trajectory** prior to execution
- Receive feedback **after scenario execution**

Dynamic parameter search

- Produce dynamic **adversarial behaviors**
- Receive feedback **after each action step**

Dynamic Behavior Alone is not Enough

- **Failures** can **depend** strongly on **initial** positions, lane assignments, speeds, headings, and target lanes.
- A **behavior** that **fails** under **one initialization** may reveal **many** related failures under **other initializations**.

Dynamic optimization

- What the adversary does during execution
- Lane change, brake, accelerate, etc.

Static optimization

- Where the scenario starts
- Positions, lanes, speeds, etc.

> *Dynasto combines both: learn what to do, then search where that behavior is most revealing.*

Pitfalls of Adversarial Reward Hacking

- Many adversarial testing approaches **reward any collision**.
- This can encourage **unrealistic maneuvers**: close cut-ins, impossible braking, or rear hits.
- The result may be a **high crash count** but **low diagnostic value**.

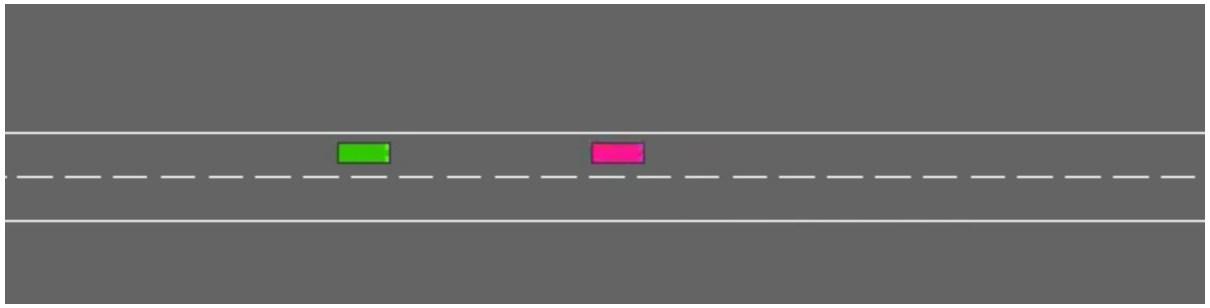
Question Dynasto asks

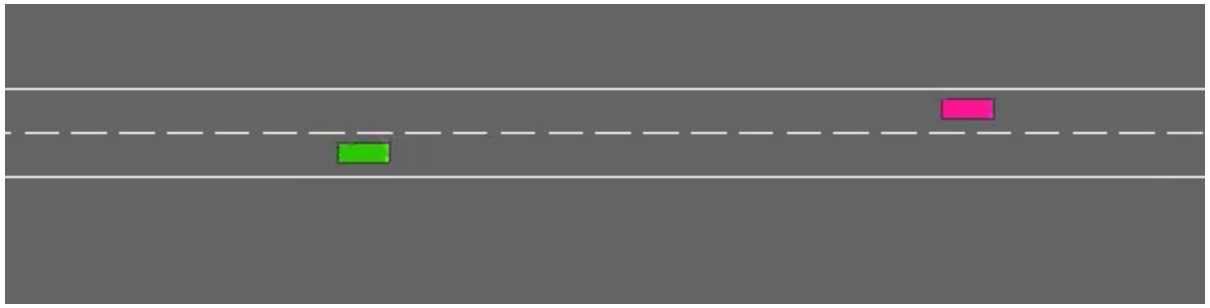
Was the collision caused by a real weakness in the ego controller, or by invalid adversarial behavior?

Only Valid Dynamic Action Sequences Count

ego vehicle

adversarial vehicle





Invalid failure

Adversary violates safe distance; the ego has no realistic opportunity to react.

Valid failure

Adversary starts from plausible headway; collision exposes an ego-controller weakness.

Paper statement and contributions

Dynamic adversarial testing becomes more effective when it accounts for both validity and initial conditions.

1 **Validity semantics**
STL rules separate genuine ADS weaknesses from unrealistic adversarial crashes.

2 **Two-step generation**
RL learns dynamic adversarial behavior; GA searches static initial conditions.

3 **Failure-mode analysis**
Event sequences + kNN graph + Leiden clustering summarize many failures into modes.

4 **Empirical evidence**
HighwayEnv evaluation on two SUTs shows 60–70% more valid failures than RL-only.

Validity Modeling: Discard Unrealistic Crashes



- ISO 34502 standardizes highway test scenarios for autonomous vehicles.
- ISO 34502 defines 24 critical scenarios, including 21 two-vehicle cases.
- We use it as inspiration for two-vehicle safety-relevant interactions.

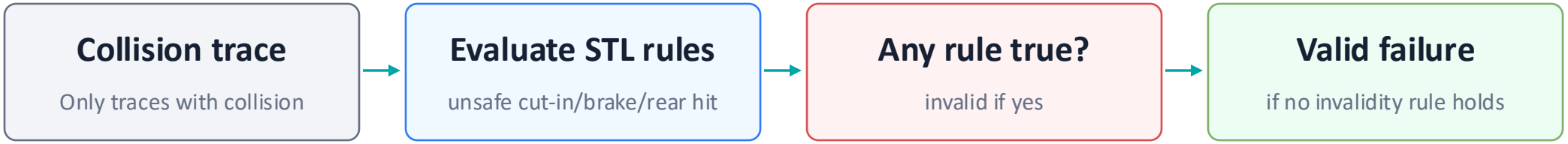
Three invalidity rules:

1. **Unsafe cut-in**
   Adversary changes lane while violating longitudinal or lateral safe distance.

2. **Unsafe braking**
   Adversary brakes ahead of the ego in the same lane from unsafe headway.

3. **Rear hit**
   Adversary accelerates into the ego from behind in the same lane.

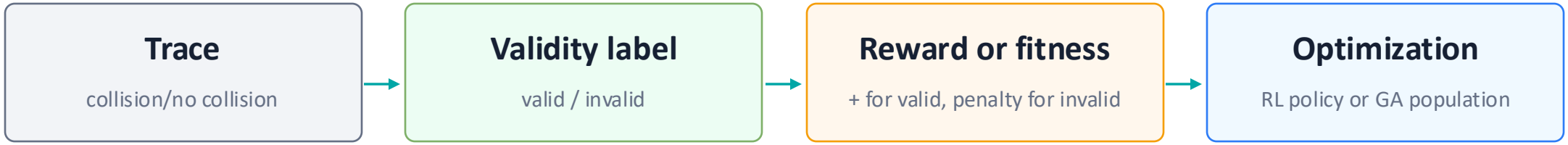
Valid vs. Invalid Failure: Decision Logic



Decision rule: if any invalidity rule is true, discard the collision; otherwise label it valid.

> *The validity rules are designed to exclude cases where the adversary's maneuver itself makes the crash unavoidable.*

Why Validity Matters for Rewarding



- Without validity, the adversary can maximize reward by producing unfair crashes.
- With validity, the adversary is pushed toward challenging but plausible interactions.
- The same validity check is reused in both RL training and GA evaluation.

Core principle: reward the adversary for exposing ego weakness, not for cheating.

Step 1: Dynamic Optimization with DQN

Observation

2 × 4 matrix [x, y, vx, vy] for ego and adversary

captures relative position, lane, and closing speed

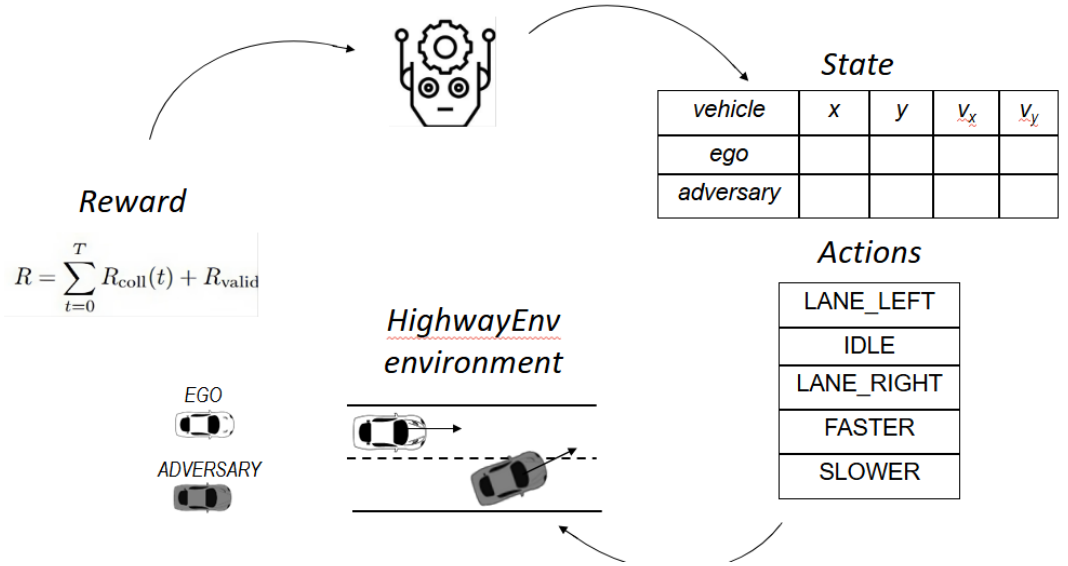
Actions

- LANE_LEFT
- IDLE
- LANE_RIGHT
- FASTER
- SLOWER

Algorithm

DQN in CleanRL

Episode ends at collision or 40 steps

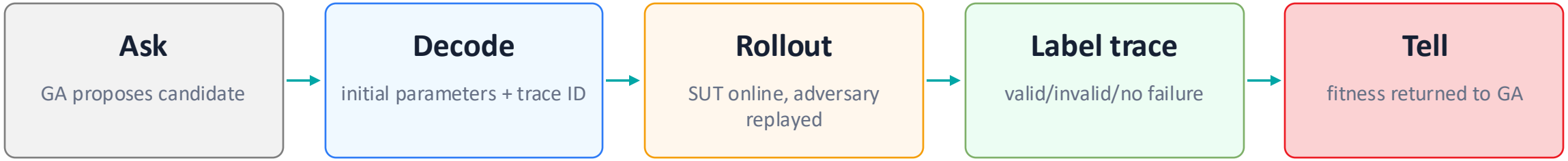


State → action → reward feedback trains an adversary that reacts online to the ego vehicle.

Step 2: Static Optimization with GA

Chromosome includes 11 normalized variables

ego/adv position · lane · heading · speed · target lane · failure-trace ID



- Fitness uses the same reward structure as RL.
- The candidate is evaluated by a full simulation rollout.
- The adversary behavior comes from a recorded Step 1 action sequence.

Important nuance: replay is fixed for adversary actions, but the ego vehicle still acts online.

Why sequential Dynasto instead of joint co-evolution?

Dynasto sequential setup



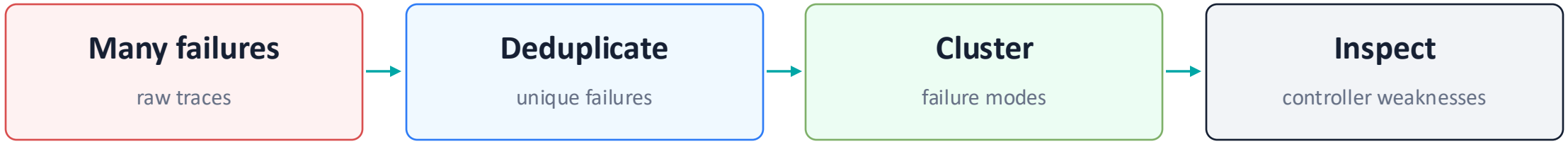
- optimizes one component at a time
- more stable search signal
- reuses strong failure-inducing behaviors

Co-evolutionary setup



- policy and initialization shift together
- can find failures in some runs
- less stable and more initialization-sensitive

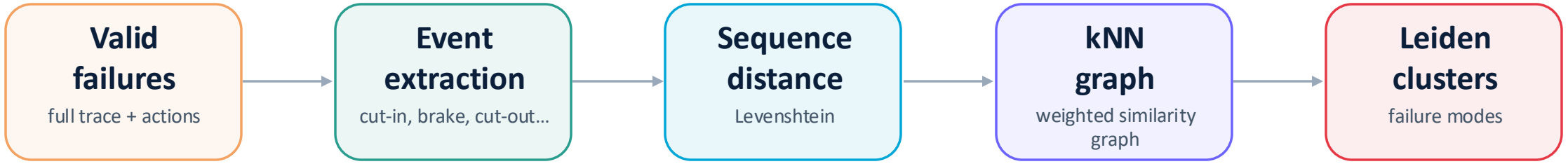
Why de-duplication and clustering are needed



- Scenario generation can produce many minor variations of the same underlying crash.
- De-duplication removes near-identical failures.
- Clustering reveals higher-level failure modes that a human can inspect.

Question answered: are we finding genuinely different failures or repeated variants of the same crash?

Post-hoc analysis: from many crashes to interpretable modes



Example event vector

[0, 0, 0, 6, 0, 0, 7, 7, 999]

(6) adversary cut-in → (7) same-lane braking → (999) collision

Rationales

- Semantic events reduce sensitivity to small trajectory variations
- Levenshtein distance measures the number of insertions, deletions, and substitutions between event sequences.
- Leiden assigns every failure to a mode rather than producing many outliers

Experimental setup



HighwayEnv

2-lane highway · ego SUT + adversarial vehicle · DQN ego controllers

1 Two systems under test

SUT1 trained with one standard IDM vehicle;
SUT2 trained with four defensive IDM vehicles.

3 Methods

RS	Random adversarial action at each step	Naive exploration
GA-only	Precomputes fixed action sequence	Open-loop: cannot adapt online
Base	DQN adversary from Doreste et al.	Rewards all failures equally
VARL	Validity-aware RL adversary	Step 1 only; no initial-condition search
Dynasto	VARL + GA over initial conditions	Full proposed method

2 Fixed budget

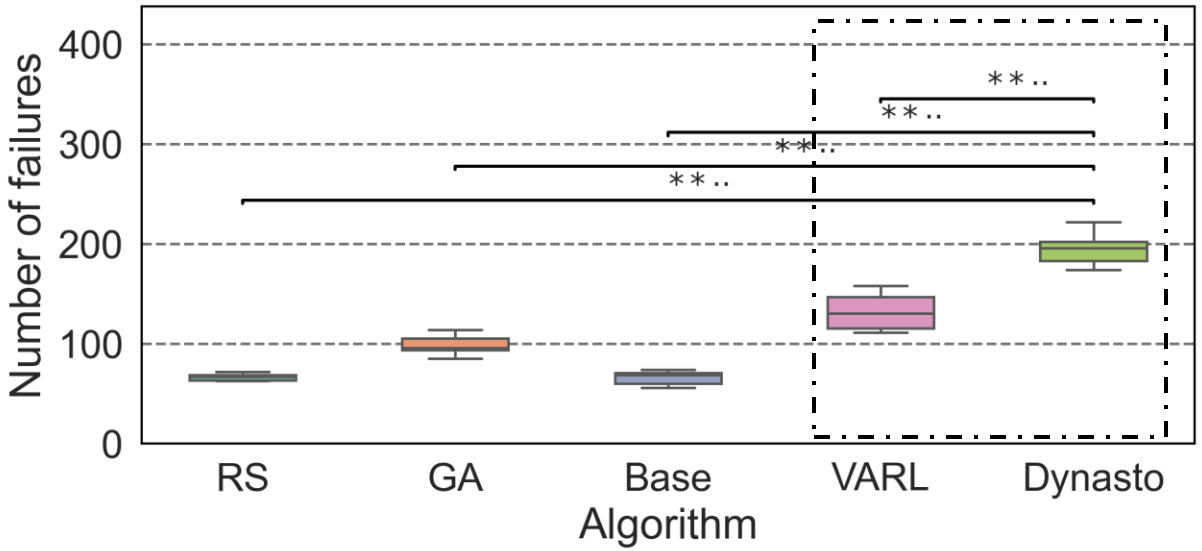
4,000 total steps per method; Dynasto uses 3,000 for RL and 1,000 for GA search.

4 Statistical protocol

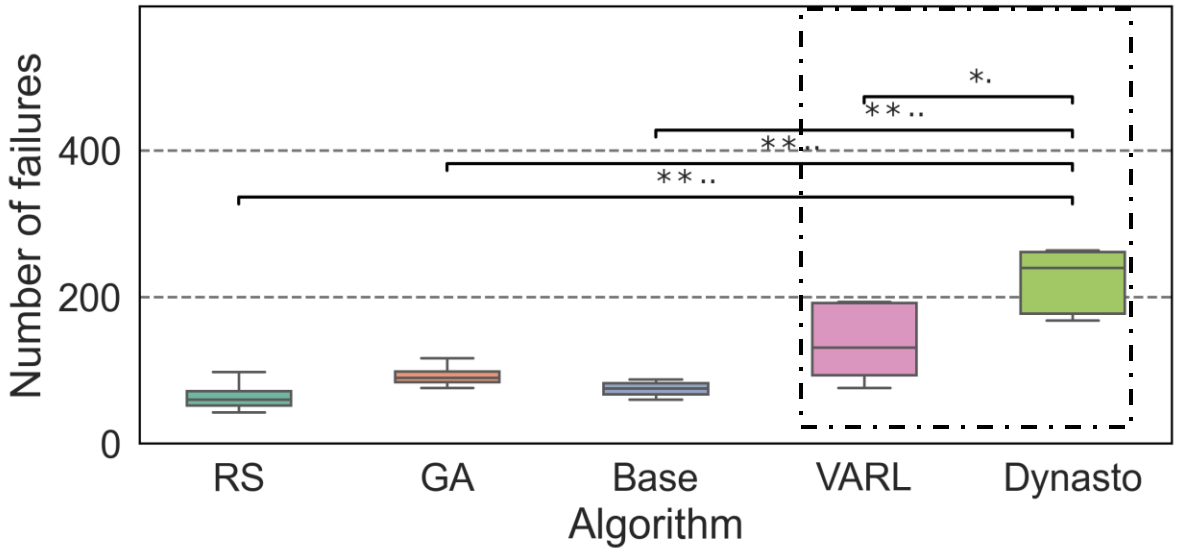
≥10 repetitions; Mann–Whitney U tests; Cliff's delta effect sizes.

RQ1: Dynasto finds more valid failures than the baselines

Use case 1



Use case 2



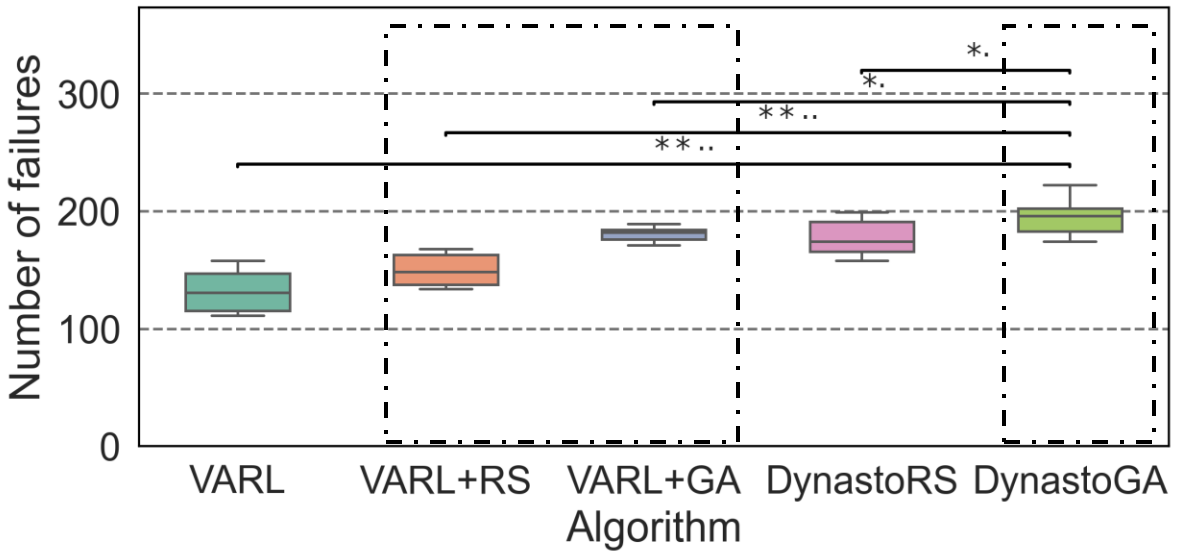
Dynasto finds up to 70% more valid failures than the validity-aware RL adversary (VARL).

- Validity-aware RL improves over all-failure RL by focusing the reward on useful failures.
- GA over initial conditions gives an additional boost beyond dynamic behavior alone.

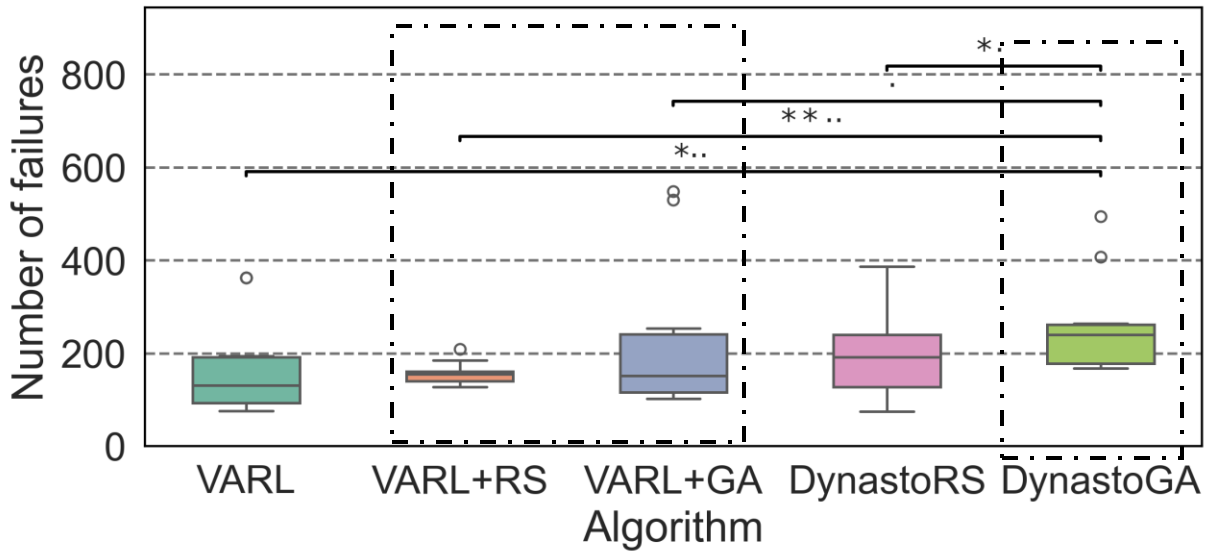
Significance notation: ** p<0.01, * p<0.05; effect sizes: .. large, . medium.

RQ2: sequential Dynasto is more consistent than co-evolution

Use case 1



Use case 2



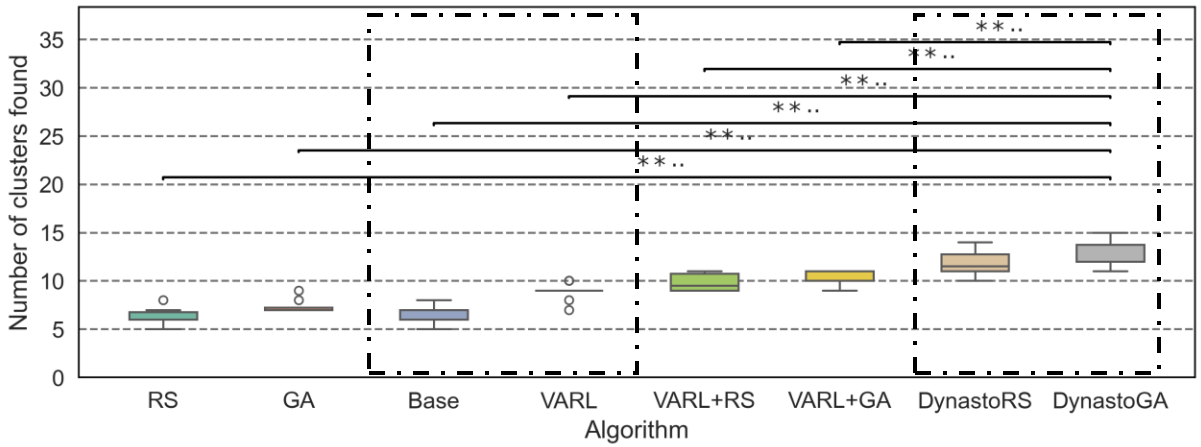
Dynasto(GA) is the strongest configuration across both use cases.

- UC1: Dynasto(GA) uncovers 11.4% more failures than co-evolutionary VARL+GA.
- UC2: Dynasto(GA) reveals about 78% more failures than baseline/co-evolutionary setups.
- GA search is more effective than random replay search, especially in UC2.

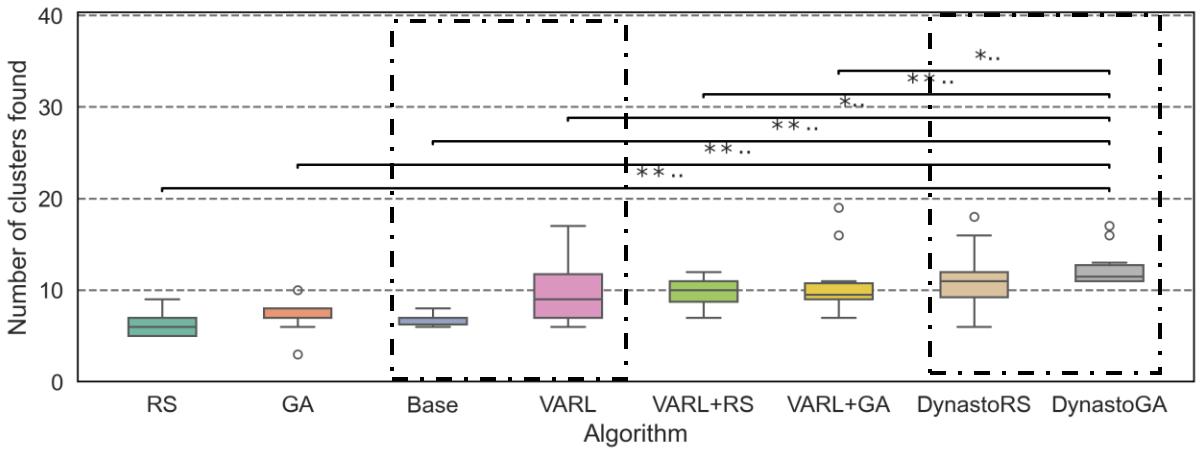
Significance notation: ** p<0.01, * p<0.05; effect sizes: .. large, . medium.

RQ3: Dynasto discovers more diverse failure modes

Use case 1



Use case 2

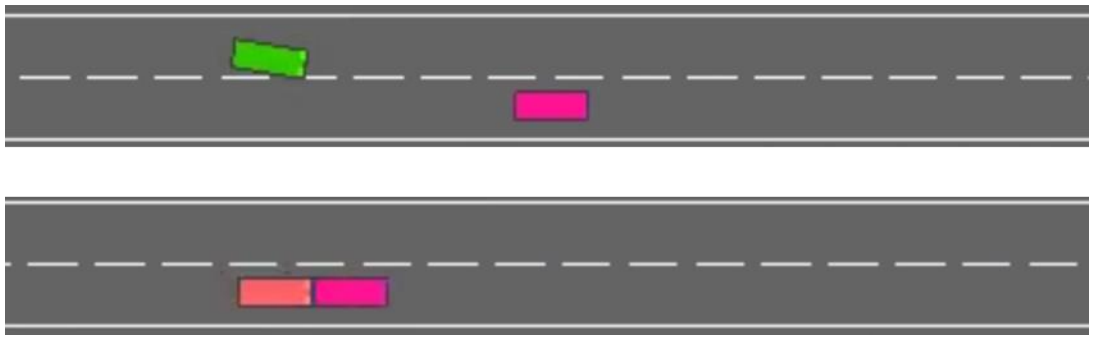


Clustering compresses ~200 unique failures into ~12 representative patterns per SUT.

- Dynasto finds about 20% more clusters than the co-evolutionary setup.
- Compared with DQN-only, Dynasto finds about 50% more clusters across both SUTs.

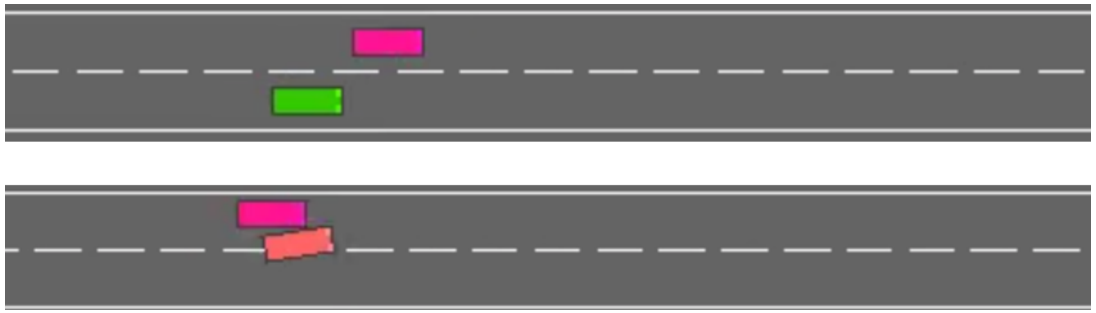
Significance notation: ** p<0.01, * p<0.05; effect sizes: .. large, . medium.

What the clusters reveal: lateral safety weaknesses



Unsafe lane changes

Ego initiates a lateral maneuver while the adversary is still nearby.

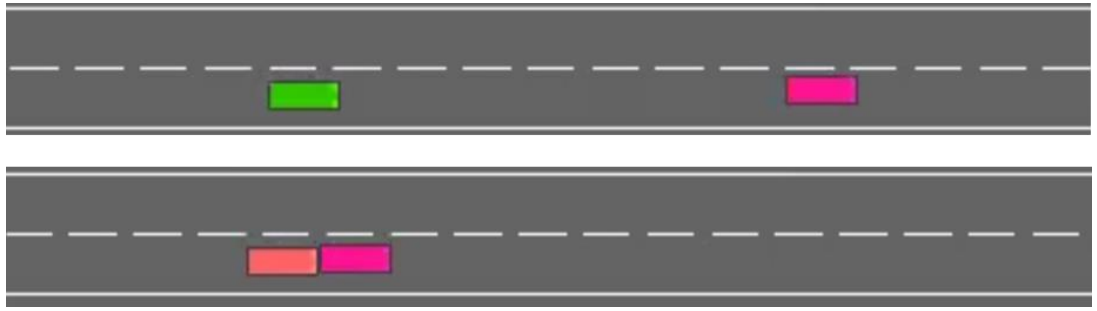


Unsafe cut-ins

Ego merges with insufficient headway, exposing unsafe gap selection.

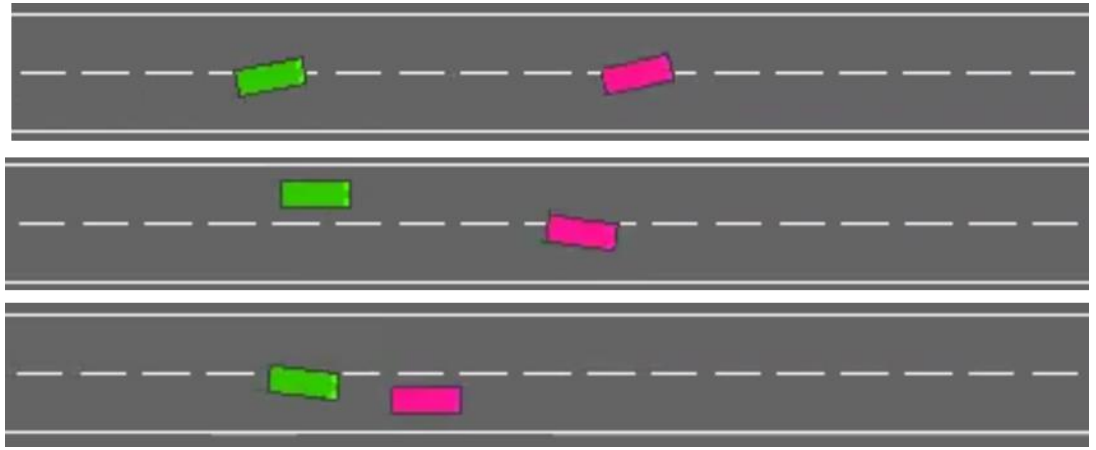
Common controller weakness: poor clearance assessment before lane changes.

What the clusters reveal: longitudinal response weaknesses



Failure to brake

Ego does not adapt speed when the lead adversarial vehicle slows down.

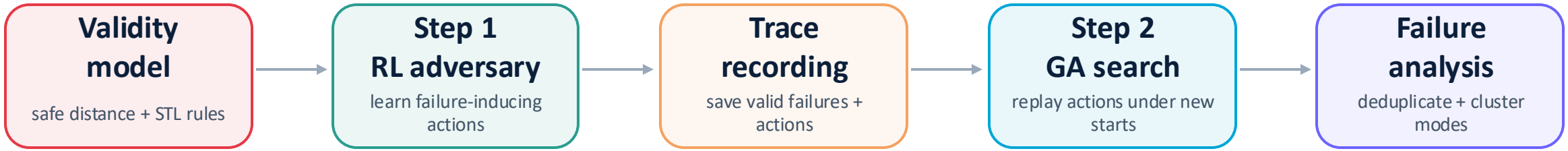


Following behavior

Ego mirrors the adversary's lane change too closely and loses safe separation.

Common controller weakness: delayed or inadequate response to evolving traffic context.

Why Dynasto improves failure discovery



1. **Validity changes the search target**
2. **RL captures interactive behavior**
3. **GA expands scenario coverage**
4. **Clustering makes results human-usable**

Key takeaways

1. **Validity matters**

   Not every collision is a meaningful ADS failure. Dynasto filters invalid adversarial crashes.

2. **Dynamic + static beats dynamic-only**

   Dynasto finds 60–70% more valid failures than an RL-only adversary under the same budget.

3. **Diversity becomes explainable**

   Event-based clustering compresses ~200 unique failures into ~12 representative modes per SUT.

Dynasto exposes realistic, diverse, and actionable ADS weaknesses.

Thank you

Try Dynasto here

Questions?