GROUPBRAIN LABS · FELLOWS' TALK SERIES

Credit assignment in RL

Which action earned the reward? From reward decomposition to LLM reasoning

Ashmit Janjirkar · DJSCE, Mumbai

15/11/2026

INTRODUCTION

Your team wins by one run. Which of the 300 balls won it?

An RL agent faces the same question: **one reward at the end, hundreds of actions before it.**

Credit assignment = working out which actions actually caused the outcome.

INTRODUCTION

Two kinds of credit

Temporal credit

Which **step in time** mattered? The key picked up 50 steps before the door opened.

THIS TALK

Structural credit

Which **agent or component** mattered? One player in a team, one module in a pipeline.

INTRODUCTION

Sparse and delayed are different problems

Sparse

The reward is rarely non-zero. The agent has to **find** success at all.

Main fix: better exploration

Delayed

The reward arrives long after the action that caused it. The agent has to **connect** them.

Main fix: credit assignment

Extreme case: **episodic feedback**, one number for the whole episode.

INTRODUCTION

Why standard RL struggles with delay

TD learning

Passes credit back **one step per update**. With a long delay, the signal crawls back slowly.

Problem: bias, fixed only exponentially slowly in the delay

Monte Carlo

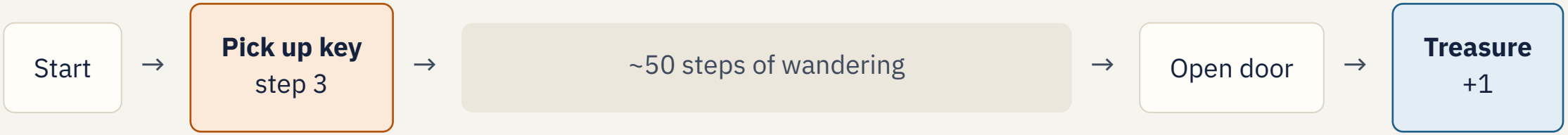
Uses the **full return** for every step. Unbiased, but every step gets the same noisy number.

Problem: variance grows with long episodes

The idea behind this talk: **move the reward closer to the action that caused it.** (Arjona-Medina et al., 2019)

RUNNING EXAMPLE

Key, door, treasure



What the agent sees

Zero, zero, zero … +1 at the very end.

What actually mattered

Picking up the key, about 50 steps earlier.

The goal

Put most of the +1 on the key step, not the wandering.

PART 2

Reward redistribution

Spread the final reward back over the steps that earned it

REWARD REDISTRIBUTION · THE RULE

The one equation in this talk

$$\hat{r}_1 + \hat{r}_2 + \ldots + \hat{r}_{\square} = R$$

new per-step rewards must add up to the episode’s return

Same total

So the best policy doesn’t change.

Dense feedback

The agent learns from every step, not just the last.

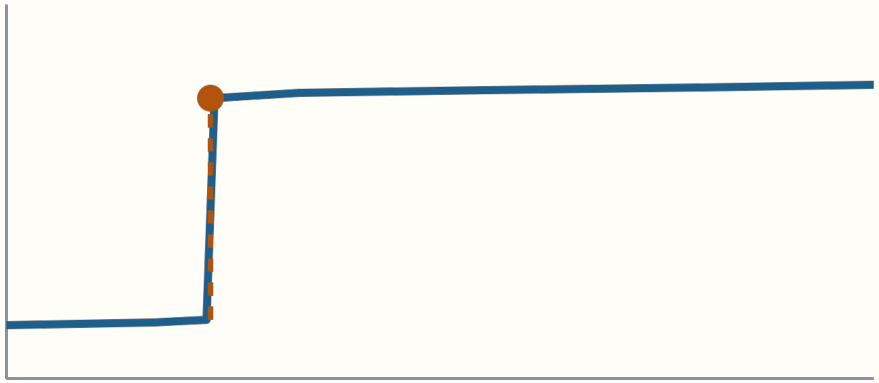
The only question

How do you split R?

REWARD REDISTRIBUTION · RUDDER

RUDDER: credit where the prediction jumps

Predicted final return, step by step



Big jump at the key → most of the credit goes there

1. Train an **LSTM to predict the final return** from the trajectory so far.
2. Credit at step t = **how much the prediction changed** at t.
3. Train the agent on these redistributed rewards.

Big idea: turn RL credit into a **regression problem**, which deep learning is good at. (Arjona-Medina et al., NeurIPS 2019)

REWARD REDISTRIBUTION · RUDDER IN DEPTH

How RUDDER builds its rewards

1

Keep the wins

A “lessons” buffer stores episodes where the delayed reward was seen, so rare successes are replayed often.

2

Predict the return

An LSTM reads the episode so far and predicts the final return at every step.

3

Take differences

A step’s reward is how much the prediction moved at that step.

4

Train the policy

PPO learns from these dense rewards instead of the single one at the end.

$$\hat{r}_t = g(\text{episode up to } t) - g(\text{episode up to } t-1)$$

g = the LSTM’s predicted return · any leftover goes on the last step, so the rewards add up to R

Why it works: a step that makes success likely causes a jump. Steps that change nothing get about zero.

Watch out: reward lands where the predictor *becomes sure*, which can be later than the action itself.

REWARD REDISTRIBUTION · IRCR IN DEPTH

IRCR: score a step by its episodes

Episode	Has key step	Has wandering	Return
1	yes	yes	1
2	yes	yes	1
3	no	yes	0

1. Give every step in an episode **that episode's return**. No model to train.
2. Training samples steps from a large buffer, so a step's reward becomes the **average return of the episodes it appears in**.
3. The agent (SAC) learns from these "guidance rewards".

KEY STEP

1.0

average of episodes 1 and 2

WANDERING STEP

0.67

average of episodes 1, 2 and 3

Inside one episode, the wandering steps get as much credit as the key. Steps only separate when you average **across many episodes**. (Gangwani et al., NeurIPS 2020)

REWARD REDISTRIBUTION · RRD IN DEPTH

RRD: fit the return from random pieces

$$R \approx (T \div K) \times [\ \hat{r}(s, a) \text{ summed over } K \text{ random steps }]$$

loss = (estimate − R)², averaged over episodes and random draws · T = episode length

K = T

Full decomposition

Every step in every update. Exact, but slow on long episodes.

SMALL K

RRD

A cheap, random estimate of the same loss. Scales to long episodes.

K = 1

Near uniform

Each step alone must explain R ÷ T. Pushes towards the even split.

Per-step model: r̂ sees only the current state and action, never the history. This matters in Finding 2. (Ren et al., ICLR 2022)

REWARD REDISTRIBUTION · DIASTER IN DEPTH

Diaster: cut, score, then difference

1

Cut

Pick a random cut point and split the episode into a prefix and a suffix.

2

Score both

A GRU scores each piece, trained so that prefix + suffix ≈ R.

3

Difference

A step's reward is the prefix score with the step, minus without it.

4

Make it per-step

Fit r̂(s, a) to those differences, then train SAC on r̂.

$$\text{score}(\text{prefix}) + \text{score}(\text{suffix}) \approx R \rightarrow \hat{r}\square \approx \text{score}(0\ldots t+1) - \text{score}(0\ldots t)$$

a prefix is scored for its own share of R, not for the whole return

Versus RUDDER: both difference prefix scores. RUDDER's prefix predicts all of R; Diaster's predicts only the prefix's part.

Cuts: one or a few work best. A cut at every step loses the sequence information. (Lin et al., AAAI 2024)

REWARD REDISTRIBUTION · SUMMARY

Same rule, different guesses

Method	How it splits R	Watch out for
IRCR	The episode’s return to every step	Irrelevant steps get equal credit
RUDDER	Where a return predictor’s output jumps	Needs a good LSTM predictor
RRD	Learned per-step reward, trained on random sub-pieces	Assumes reward adds up step by step
Diaster	Differences of sub-episode scores, fitted to a per-step reward	Return model can fail on long delays
GRD	Causal model of which states and actions produce reward	More modelling assumptions

RUDDER, RRD, Diaster and GRD keep the total fixed; IRCR does not. They all differ in **which steps they think mattered.**

REWARD REDISTRIBUTION · A SECOND LENS

What each model gets to see

SEES THE CURRENT STEP

Per-step: (s, a)

- **IRCR**: average return of the step's episodes
- **RRD**: learned $\hat{r}(s, a)$
- **Diaster's final reward**: fitted to (s, a)

SEES THE EPISODE SO FAR

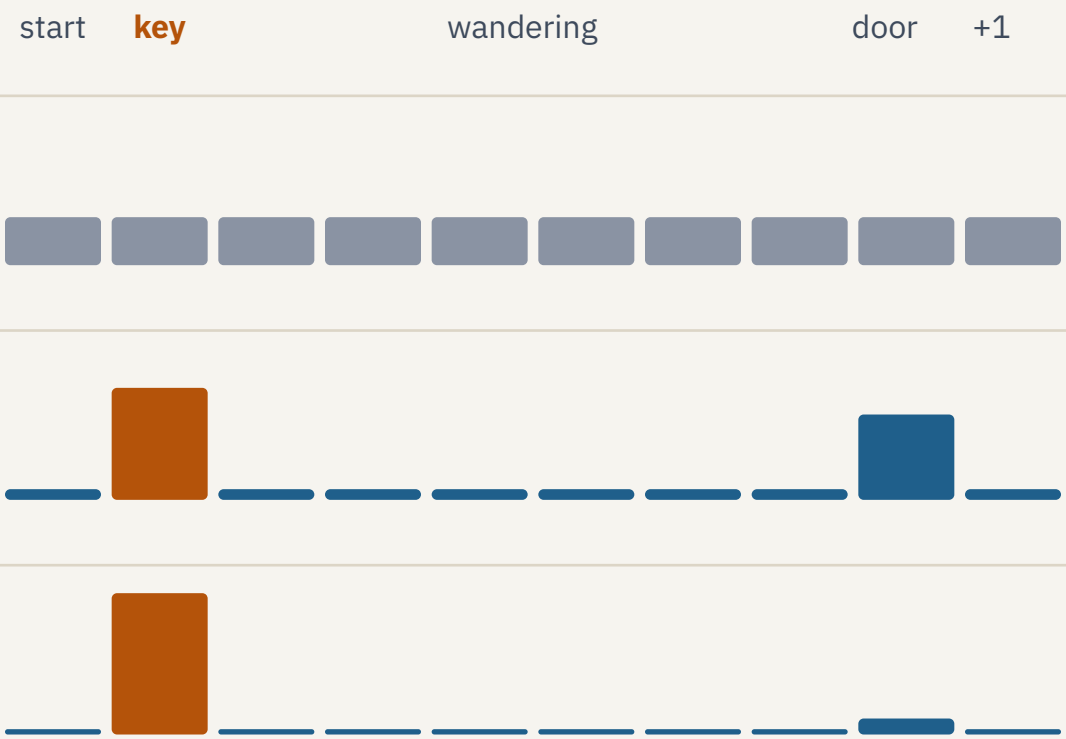
History

- **RUDDER**: LSTM over the prefix
- **Causal transformer**: attention over the prefix
- **Diaster's scorer**: GRU over sub-episodes
- **RLBR**: attention within a bag

If the true reward depends on history ("was a key already taken?"), a per-step model **cannot represent it**, however long it trains.

RUNNING EXAMPLE · ILLUSTRATIVE

One winning episode, three splits



Which split is “right” depends on what you mean by **cause**. That is the question in Part 5.

PART 3

Beyond one reward per episode

Rewards in chunks, and RL as sequence modelling

BEYOND EPISODES · RLBR

RLBR: rewards that arrive in bags

BAG SIZE 1

Standard RL

A reward every step.

BAG OF K STEPS

RLBR

One reward per chunk of steps.

BAG = EPISODE

Episodic

One reward at the end.

Real examples: a score per lap, a rating per meeting, an hourly electricity meter.

Method: a **Reward Bag Transformer** reads the whole bag with bidirectional attention and splits its reward across the steps. (Tang et al., TMLR 2025)

BEYOND EPISODES · RLBR IN DEPTH

Inside the Reward Bag Transformer

Steps of a bag
The (state, action) pairs, and one reward for the whole bag.

→

Causal transformer
One embedding per step, built from earlier steps only.

→

Bidirectional attention
Each step looks at the whole bag, before and after it.

→

Two heads
A reward for each step, and a prediction of the next state.

$$\text{loss} = \left(\Sigma\, \hat{r}_{\square} \text{ in the bag} - \text{bag reward}\right)^2 + \beta \cdot \left\| \hat{s}_{\square+1} - s_{\square+1} \right\|^2$$

first term: rewards add up per bag · second term: learn how the world moves

Then: relabel every step in the replay buffer with r̂ and train SAC on it.

Guarantee: if each bag reward is the sum of its step rewards, both tasks share the same optimal policies.

BEYOND EPISODES · DTRD

DTRD: decomposition for Decision Transformers

SETTING

A Decision Transformer treats RL as **sequence prediction**, conditioned on the return still to come.

PROBLEM

With long delays, that conditioning **picks up bias**: the model can't tell which tokens earned the return.

FIX

Learn the decomposition and the policy **together** (bi-level optimization).

The split is learned for **what helps the policy**, not just to fit the return. (Zhu et al., IJCAI 2023)

BEYOND EPISODES · DTRD IN DEPTH

How DTRD fixes return-to-go

The return-to-go a DT is given

One reward at the end

R	R	R	R	R

Same number at every step: no signal about which step mattered.

With redistributed rewards

R	R−$\hat{r}_1$	…	small	≈0

Drops as credit is earned, so the DT sees progress.

Two-level training

1. **Lower level:** train the Decision Transformer on a training split, using the shaped return-to-go.
2. **Upper level:** update the reward model f(s, a) so the trained DT does better on a held-out split.
3. **Keep the total:** add $\lambda \cdot (\Sigma \hat{r} - R)^2$ so the rewards still sum to R.
4. **Shortcut:** one gradient step stands in for full inner training.

The split is judged by **how well the policy predicts actions**, not only by how well it fits R. (Zhu et al., IJCAI 2023)

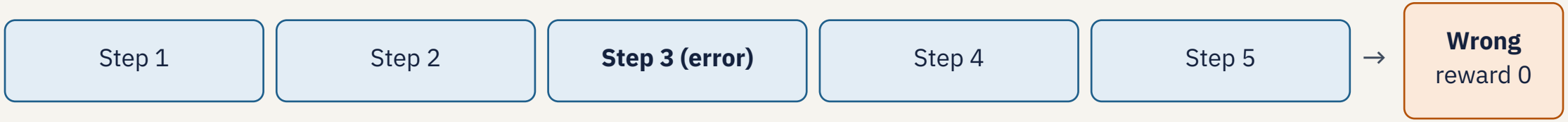
PART 4

Same problem, new place: LLMs

Why credit assignment is now central to post-training

LLMS · OUTCOME REWARDS

GRPO is uniform credit at LLM scale



Every step gets the same update, the four correct steps included.

What RLVR does

One verified outcome reward, spread equally across every token of the reasoning.

Why it’s familiar

That is the uniform credit of IRCR from Part 2, on traces thousands of tokens long.

LLMS · FINER CREDIT

How people are fixing it

Rollouts from the middle

Sample several continuations from an intermediate step to estimate how good it was.

VinePPO

Reset at the mistake

The model locates its own wrong step, resamples from there, and compares outcomes.

SRPO, 2026

Score each step

Process reward models rate every step, but need step-level labels or a model to make them.

PRMs

A 2026 survey calls credit assignment the **shared bottleneck** of both reasoning and agentic LLM RL.

PART 5

Is the credit right?

My research: checking these methods against exact ground truth

MY RESEARCH · THE QUESTION

Fitting the return is not finding the cause

Fit

Do the predicted rewards **add up to R**?

What methods optimize

Fidelity

Does credit land on the **actions that caused R**?

What we actually want

My research tests this directly: small environments where the true causal steps are known, with **depth** (delay to reward) and **breadth** (number of ways to succeed) varied separately.

Headline: a model can fit the return and still give a necessary action **0–4% of the credit** when its true share is 15–48%.

MY RESEARCH · SETUP

Two worlds where the right answer is known

ENVIRONMENT A

Redundant keys

Any one of several key slots is enough to succeed. **Breadth** is the number of interchangeable slots.

ENVIRONMENT B

A branch, then its key

The agent must choose a route early, then take that route's key. **Both** actions are needed.

Depth

Delay before the reward: 10 to 200 steps.

Breadth

Ways to succeed: 1, 2, 4 or 8.

Data

1,000 offline episodes per setting, five seeds.

Methods

IRCR, RRD, RUDDER, a causal transformer, Diaster.

MY RESEARCH · THE MEASURE

Credit, measured by changing one action

credit at step t = predicted return − average predicted return
when the action at step t is replaced

Credit mass

Share of the credit that lands on the truly causal steps. 1 is perfect.

Split error

With two necessary actions, is the credit divided between them correctly?

Calibration

Is the credit the right size, not just in the right place?

It looks only at how the predicted return depends on each action, so it does not matter **where** a model books its reward.

MY RESEARCH · FINDING 1

More training changes the picture

TRANSFORMER · LONGEST DELAY

0.16 → 0.82

Credit mass, original budget against four times the training.

RRD · LONGEST DELAY

0.07 → 0.34

Better, but still low.

RUDDER · EVERY DELAY

0.88–0.96

Already high at the original budget.

A registered test of depth against breadth **flips sign** with the longer budget. The flip does not survive resampling the grid's settings, so I report it as fragile.

Lesson: part of the long-delay failure was under-training. Report the budget.

MY RESEARCH · FINDING 2

What the model can see explains the gap

With several interchangeable keys, a second key adds nothing. A model that sees **only the current step** cannot know whether a key was already taken.

Credit mass in Environment A after longer training (1 is perfect)

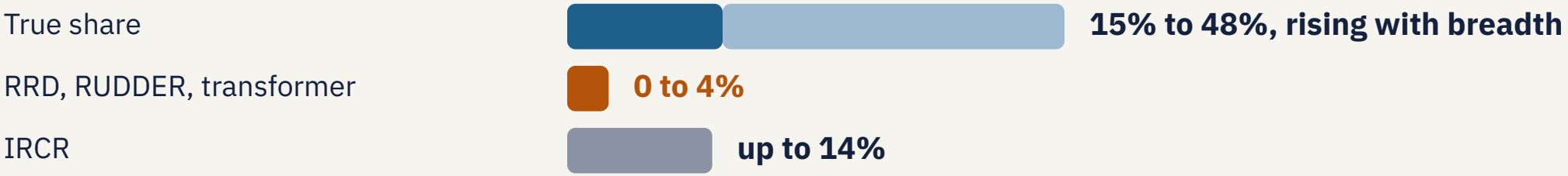


One extra bit closes the gap. The history models' lead here is **missing information**, not a better architecture.

MY RESEARCH · FINDING 3

A necessary action gets almost no credit

Share of the causal credit given to the branch, Environment B, delays up to 50



Fix 1: show it failing

Add episodes where the wrong branch led to failure: its share rises to 0.30–0.45. It needs enough data, 4,000 episodes, not 1,000.

Fix 2: tell the model the route

A per-step model given the opened route credits the branch 0.36–0.39, from the original data.

Why: every success took a correct branch, so successes alone never show that it mattered.

MY RESEARCH · FINDING 4

Better credit, barely faster learning

PPO ON EACH REWARD · LONGEST DELAY

Transformer's credit mass: **0.16 → 0.82**

Its steps to solve the task: **346k → 313k**

Return spread evenly, no credit information at all: **326k**

ROBOT DATA · MUJOCO WINDOWS

Window returns are fit almost exactly (error at most 0.03).

Agreement with the recorded per-step reward: **about 0 for RUDDER, up to 0.99 for RRD.**

Predicting the return, crediting the right actions and helping a policy learn are **three different tests**. A model can pass one and fail another.

MY RESEARCH · TAKEAWAYS

What a credit evaluation should report

1 **The training budget**
And whether the model was still improving when it stopped.

2 **What each model can observe**
A gap between model families may be a missing input.

3 **Which failures the data contains**
A necessary action only shows up when its absence is seen.

4 **A uniform baseline**
So that denser reward is not mistaken for accurate credit.

Limits: these are my own re-implementations on small synthetic tasks, not the authors' code.

PART 6

Open problems

Does it hold up in the real world, and where is the field heading?

OPEN PROBLEMS · REAL-WORLD REWARDS

Real rewards are delayed and aggregated

Delayed

A building's thermal mass: pre-cooling at 2 PM pays off at 5 PM.

Aggregated

Energy is metered per hour or per day, not per action.

Not a sum

A peak-demand charge depends on the single worst interval, which breaks the sum-to-R rule.

Building simulators (Sinergym, CityLearn) log the true per-step reward, so credit can be **checked**, not just assumed.

FUTURE SCOPE

Where the field is heading

1

LLMs as credit priors

Use a language model’s knowledge to judge which steps mattered (LaRe, CALM).

2

Long-horizon and multi-agent credit

Tool-using agents with 100+ step trajectories; teams where several agents share one reward.

3

Benchmarks with ground-truth credit

Report whether credit was right, not just the final score, and on real-world feedback.

REFERENCES

Further reading

- Arjona-Medina et al., RUDDER: Return Decomposition for Delayed Rewards, NeurIPS 2019
- Gangwani et al., Learning Guidance Rewards with Trajectory-space Smoothing (IRCR), NeurIPS 2020
- Ren et al., Randomized Return Decomposition (RRD), ICLR 2022
- Zhang et al., Interpretable Reward Redistribution in RL: A Causal Approach (GRD), NeurIPS 2023
- Tang et al., Reinforcement Learning from Bagged Reward (RLBR), TMLR 2025
- Zhu et al., DTRD: reward redistribution for Decision Transformers, IJCAI 2023
- Lin et al., Episodic Return Decomposition by Difference of Implicitly Assigned Sub-Trajectory Reward (Diaster), AAAI 2024
- Pignatelli et al., A Survey of Temporal Credit Assignment in Deep RL, TMLR 2024
- Meulemans et al., Counterfactual Contribution Analysis (COCOA), NeurIPS 2023
- Kazemnejad et al., VinePPO, 2024
- Credit Assignment with Resets in LM Reasoning (SRPO), 2026
- From Reasoning to Agentic: Credit Assignment in RL for LLMs (survey), 2026
- Qu et al., Latent Reward (LaRe); Pignatelli et al., CALM

CONCLUSION

Every method splits the same reward. The real question is whether it splits it onto the right actions.

Delayed rewards in robotics, buildings and LLM reasoning all come back to this one problem.

Questions?

Ashmit Janjirkar · [contact]