cuBCR: Batched GPU Evaluation for Counterfactual Responsibility

Anonymous authors
Paper under double-blind review

Abstract

Multi-agent systems coordinate multiple AI agents to complete tasks, but attributing responsibility becomes challenging when collaborative execution fails. In time-sensitive services, slow root-cause analysis delays mitigation and degrades service quality. Explaining team failure requires analyzing how individual actions constrain the overall system's ability to recover. In this paper, we formalize responsibility attribution in finite stochastic models: holding a subset of agent policies fixed, where a policy specifies how an agent chooses its actions, we compute the lowest failure probability achievable when the remaining agents coordinate recovery within a fixed number of decision steps. We use Shapley values to allocate the difference between the original failure risk and the minimum risk achievable when all agents may coordinate recovery. However, exhaustively evaluating every coalition of agent policies requires an exponentially large number of counterfactual recovery calculations. We introduce CUDA-accelerated Batched Counterfactual Responsibility (cuBCR), which estimates Shapley values from sampled agent orders and evaluates the required recovery calculations together on a GPU. On a synthetic ticketing benchmark, cuBCR achieves a median 233× speedup over exhaustive CPU attribution. By significantly reducing attribution latency, cuBCR makes iterative counterfactual analysis practical, so system designers can quickly evaluate how individual policy revisions change multi-agent failure dynamics.

1 Introduction

In a multi-agent system, a mistake by one agent can lead to failure if the mistake is not detected by other agents. Consider a ticketing service. The reservation agent reserves a seat that is already taken, and the payment agent accepts payment without detecting the problem. The service then issues an incorrect ticket. Correcting the reservation or catching the error during payment could prevent this failure. Responsibility therefore depends on both the initial mistake and the other agents' opportunities to prevent failure.

We study failure risk using a *finite stochastic model*, which specifies finitely many states and actions and the probabilities of their outcomes. Each agent follows an original *policy*, a rule that assigns action probabilities in each state. Following Mu & Najib (2026), we keep selected agents' policies fixed and allow the others to coordinate recovery. For each selection, we calculate the lowest probability of reaching a designated failure state within a fixed number of decision steps, called the *time horizon*. The recovering agents use the current state and the time remaining to choose their actions. This analysis averages over possible executions from a specified initial state.

An agent's contribution depends on which other agents may change their policies. Correcting a reservation, for example, provides little additional benefit if the payment agent can already prevent the invalid ticket. Measuring each agent's benefit separately can therefore count the same improvement more than once. We use *Shapley values* (Shapley, 1953) to allocate the total improvement. Starting with all original policies fixed, we allow agents to change one at a time and record each additional reduction in minimum failure risk. We average each agent's reduction over all possible orders. These calculation orders do not reorder decisions in the modeled system. The resulting shares sum to the difference between the original failure risk and the minimum risk when all agents may coordinate recovery. Any remaining risk is left unassigned. Figure 1 illustrates this allocation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

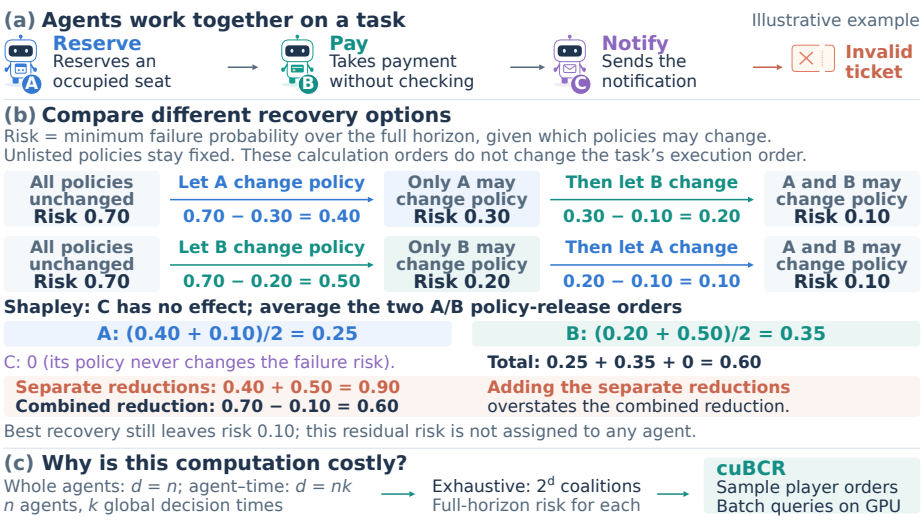


Figure 1: **Allocating avoidable failure risk.** (a) A reservation error passes through payment to an invalid ticket. (b) Illustrative minimum risks when the listed agents may change policies; other policies stay fixed. C has no effect, so the two A/B release orders represent all six agent orders equally. Shapley allocations sum to 0.60, leaving residual risk 0.10 unassigned. (c) Each of 2^d coalitions requires full-horizon recovery, with $d = n$ or nk. Appendix A.2 explains more details.

The main obstacle is computational cost. With n agents, exhaustive attribution considers 2^n combinations of fixed policies. If responsibility is assigned separately to each agent at each of k global decision times, there are nk policy restrictions and 2^{nk} possible combinations. Existing methods estimate Shapley values by sampling calculation orders and averaging the contributions measured in those orders (Castro et al., 2009; Mitchell et al., 2022). Sampling reduces the number of combinations evaluated, but each selected combination still requires a recovery calculation. This calculation finds the best available recovery actions across states and remaining decision times. Its cost can limit repeated analysis as policies change.

We present CUDA-accelerated Batched Counterfactual Responsibility (CUBCR), which combines sampling with computation on graphics processing units (GPUs). CUDA is NVIDIA's parallel computing platform. All recovery calculations share the transition probabilities, available actions, and original policies; only the fixed-policy choices differ. CUBCR stores the shared data once and evaluates many policy combinations together, maintaining separate risk values for each. It uses backward induction: starting at the final decision time, it uses the next time step's values to compute the current values. The GPU performs these updates in parallel across policy combinations and states. Sampling reduces the number of recovery calculations, while batching accelerates their execution. Uncertainty estimates guide when sampling stops.

Our evaluation separates sampling savings from GPU execution gains. On synthetic ticketing systems, identical query batches yield a median $5.99\times$ GPU speedup over a 32-core OpenMP/AVX-512 CPU implementation. Combining sampling and GPU execution yields a median $233\times$ speedup over exhaustive CPU attribution across 36 tasks. These timings exclude model construction and engine initialization. Controlled models also show that freeing one agent while keeping all others fixed can produce different responsibility rankings from Shapley attribution.

Our contributions are:

1. **Batched GPU recovery calculations.** We evaluate many policy combinations together, sharing transition data while keeping separate results. The solver supports responsibility attribution to whole agents and to agents at individual decision times.
2. **Sampling with uncertainty-based stopping.** We use confidence sequences, intervals designed to cover the target values throughout sampling, to decide when enough orders have been evaluated (Waudby-Smith & Ramdas, 2024).

3. **An evaluation of computational gains and interactions.** We separate the effects of sampling, GPU execution, and preprocessing, and examine when interactions between agents change the allocation of responsibility.

2 Related Work

Responsibility in sequential systems. Triantafyllou et al. (2021) allocate policy-improvement benefits using Shapley values in multi-agent Markov decision processes with an objective that sums rewards indefinitely while giving less weight to later rewards. Their whole-agent allocation matches our policy-pinning allocation when the objective and permitted recovery strategies are the same. Mu & Najib (2026) study bounded-horizon responsibility with randomized recovery whose action probabilities depend on the current state, without separately using time remaining. We allow recovery to use time remaining; this distinction disappears when time is encoded in the state. Verification methods also use Shapley values to allocate responsibility for temporal requirements (Mascle et al., 2021; Baier et al., 2024). TimeSHAP perturbs sequences to explain recurrent predictions (Bento et al., 2021), whereas Counterfactual Shapley Credit Assignment attributes rewards along observed trajectories (Li et al., 2026b).

Counterfactual queries. DoWhy-GCM supports graphical causal models (Blöbaum et al., 2024), and Counterfactual Influence in MDPs optimizes alternatives tied to an observed trajectory (Kazemi et al., 2025). We vary policy restrictions without conditioning on an observed failure. AgenTracer uses replay and TraceElephant provides executable environments for agent-failure analysis (Zhang et al., 2026; Chen et al., 2026); our engine requires explicit transitions and recovery actions.

Sampling and guarantees. Permutation sampling and reverse pairing are established Shapley estimators (Castro et al., 2009; Mitchell et al., 2022); alternatives include SVARM, k-ADD-SHAP, and Adalina (Kolpaczki et al., 2024; Pelegrina et al., 2023; Li et al., 2026c). We pair orders to obtain bounded observations for the confidence sequence of Waudby-Smith & Ramdas (2024). Certified Interventional Fidelity also uses adaptive bounds (Asiaee, 2026), while Oura & Ito (2025) give probabilistic guarantees for causality with uncertain transition parameters. Our conditional guarantee concerns sampling and numerical error for a fixed model, excluding model-fitting uncertainty.

GPU inference and reuse. GPU value iteration (Bošnački et al., 2011; Wu et al., 2016), probabilistic temporal-logic checking (Heemstra et al., 2025), and batched parameterized verification (Češka et al., 2016) precede this work. Tessa uses tensor computations for finite-horizon Markov-chain verification (Li et al., 2026a); PyJuice compiles probabilistic circuits for GPU inference (Liu et al., 2024). Our batches share a transition model while varying time-indexed policy restrictions, averaging pinned actions and minimizing over coordinated recovery actions.

3 Problem Definition

We study responsibility for reaching a failure set within k decision steps. Each agent has an original policy specifying its reference behavior. Recovery allows selected agents to change their choices; we ask how these restrictions affect minimum failure risk.

System and policies. We use a finite concurrent stochastic game with agents $N = \{1, \ldots, n\}$, state space $\mathcal{S}$, initial state $s_0 \in \mathcal{S}$, and finite, nonempty action sets $A_i(s)$. A state describes the system configuration. At time $t \in \{0, \ldots, k-1\}$, agents simultaneously choose $\mathbf{a} = (a_1, \ldots, a_n)$, where $a_i \in A_i(s_t)$; inactive agents have only a no-op action. The transition probability $P(s' \mid s, \mathbf{a})$ gives the probability of the next state s' after joint action $\mathbf{a}$ in state s.

Failure means reaching $F \subseteq \mathcal{S}$ at some $t \in \{0, \ldots, k\}$. We make F absorbing: once entered, it cannot be left. Avoiding F does not imply task completion; non-completion is failure only when included in F.

Agent i's original policy $\pi_i(a_i \mid s)$ is the supplied probability of choosing action a_i in state s. Pinning preserves this rule at every reached state, rather than replaying a recorded action. Pinned

actions are independent conditional on state. Transition and policy probabilities are nonnegative and normalized; Appendix B.3 states further numerical assumptions for coverage.

We use TicketBench, a synthetic ticket-processing benchmark, as an example. A state records the request, processing stage, retry count, and payment/inventory status; actions include proceeding, retrying, and rejecting. A seeded generator supplies the transition probabilities P and original policies π_i, while F represents issuing a ticket with inconsistent payment or inventory. The restricted Overcooked kitchen (Section 5.3) instead uses simulator transitions and policies fitted from human actions.

Players, coalitions, and recovery. A player is a unit to which responsibility is assigned. Let $\mathcal{P}$ be the player set and $d = |\mathcal{P}|$ its size. For whole-agent attribution, each player represents one agent throughout the horizon, so $\mathcal{P} = N$ and $d = n$. For decision-time attribution, each player represents one agent at one global decision time, so $\mathcal{P} = N \times \{0, \dots, k-1\}$ and $d = nk$. Selecting player (i, t) fixes agent i's original policy at time t in every state that may be reached then.

Figure 1 lists the agents allowed to change their policies. In the formal definition, a coalition $C \subseteq \mathcal{P}$ instead lists the players whose policies remain fixed. It pins the following agents at time t:

$$C_t = C \quad \text{(whole-agent)}, \qquad C_t = \{i : (i, t) \in C\} \quad \text{(decision-time)}.$$

The remaining agents $N \setminus C_t$ form one coordinated recovery controller. A strategy in $\Sigma(C)$ specifies their joint-action distribution given state and time, allowing deterministic or randomized choices using the remaining $k - t$ steps. The controller cannot observe pinned agents' simultaneous action draws; its draw is independent of theirs conditional on state and time.

For coalition C, write π_C for the original policies applied to agents in C_t at each time. The probability $\mathbb{P}_{s_0}^{\pi_C, \sigma}$ is taken over executions starting at s_0, with pinned policies π_C, recovery strategy σ, and transitions P. The coalition value $v(C)$ is the lowest failure risk achievable under these restrictions:

$$v(C) = \min_{\sigma \in \Sigma(C)} \mathbb{P}_{s_0}^{\pi_C, \sigma}\{\exists t \in \{0, \dots, k\} : s_t \in F\}, \qquad \Upsilon = v(\mathcal{P}) - v(\emptyset). \tag{1}$$

Here $v(\emptyset)$ allows all agents to coordinate recovery, whereas $v(\mathcal{P})$ is the original failure risk. Thus, Υ is the total risk difference to be allocated; the residual risk $v(\emptyset)$ remains unassigned.

Allocating the risk difference. For $p \notin C$, the marginal $v(C \cup \{p\}) - v(C)$ measures the increase from additionally pinning p. The Shapley value averages this increase over all player orders (Shapley, 1953). Let ω be a uniformly random permutation and P_p^ω the players preceding p:

$$\phi_p = \sum_{C \subseteq \mathcal{P} \setminus \{p\}} \frac{|C|!\,(d - |C| - 1)!}{d!} [v(C \cup \{p\}) - v(C)] = \mathbb{E}_\omega[v(P_p^\omega \cup \{p\}) - v(P_p^\omega)]. \tag{2}$$

The factorial coefficient is the fraction of all $d!$ orders in which exactly C precedes p, so coalitions are not uniformly weighted. These attribution orders do not change the chronological execution; restricting them to chronological order yields a different allocation (Faigle & Kern, 1992).

Pinning more players cannot reduce minimum risk: every marginal lies in $[0, \Upsilon]$ and $\sum_p \phi_p = \Upsilon$ (Proposition 1). This supplies the sampling range. A responsibility $\phi_p = 0.25$ allocates 25 percentage points of modeled failure probability, not a 25% probability that p caused an observed failure. Summed decision-time responsibilities need not equal whole-agent responsibility, because merging players changes the coalitions averaged.

For whole-agent attribution, let $S \subseteq N$ be the agents allowed to change. The release game $g(S) = v(N) - v(N \setminus S)$ gives the same Shapley allocation (Appendix A.1).

4 Methodology

We integrate paired permutation sampling with batched GPU coalition evaluation. The CPU constructs coalition queries from sampled player orders, the GPU returns their minimum failure risks, and the CPU updates responsibility estimates and checks the requested accuracy condition.

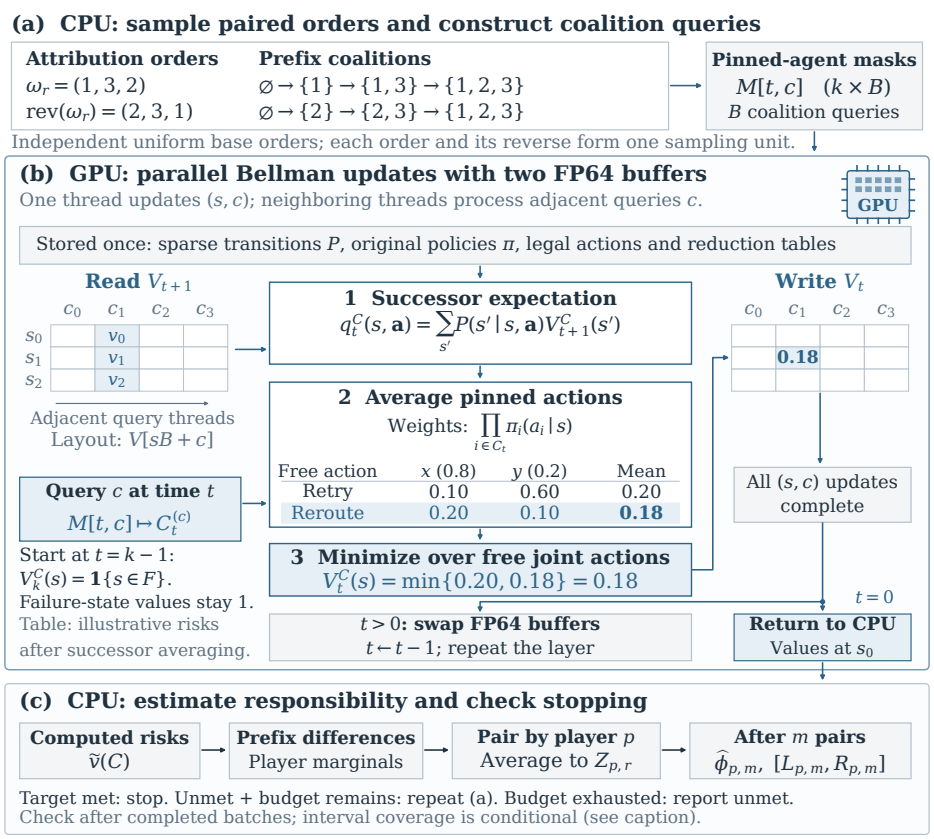


Figure 2: **cuBCR workflow.** (a) Paired orders produce time-indexed pinning masks. (b) The highlighted update uses $C = C^{(1)}$ and $v_j = V_{t+1}^C(s_j)$. Illustrative successor-averaged risks are averaged under pinned-action probabilities 0.8 and 0.2, then minimized over recovery actions: retry gives 0.20, reroute 0.18. Recovery cannot observe simultaneous pinned draws. All updates finish before buffer swapping. (c) Marginals are paired by player p into observations $Z_{p,r}$, then accumulated over m pairs. Coverage is conditional on Theorem 1.

4.1 Coalition recovery by backward induction

Let $V_t^C(s)$ be the minimum probability of failure by time k from state s at time t. Set $V_k^C(s) = \mathbf{1}\{s \in F\}$. Set $V_t^C(s) = 1$ for failure states at every time. For a nonfailure state,

$$q_t^C(s, \mathbf{a}) = \sum_{s' \in \mathcal{S}} P(s' \mid s, \mathbf{a}) V_{t+1}^C(s'), \tag{3}$$

$$V_t^C(s) = \min_{a_{N \setminus C_t}} \sum_{a_{C_t}} \left[\prod_{i \in C_t} \pi_i(a_i \mid s) \right] q_t^C(s, \mathbf{a}), \qquad v(C) = V_0^C(s_0). \tag{4}$$

Here $a_G \in \prod_{i \in G} A_i(s)$ is a joint action; an empty set has one empty action and an empty product equals one. Equation (3) averages the next-step failure risk over possible successor states. For each recovery action, Equation (4) then averages over the pinned agents' possible actions and selects the recovery action with the lowest average risk. Randomization cannot improve on the best joint recovery action. Applying these updates for $t = k - 1, \ldots, 0$ gives the coalition risk.

4.2 Shared-model GPU evaluation

A coalition query computes $v(C)$ for one choice of fixed policies. All queries share the transition model, original policies, and precomputed action-reduction tables. Each joint action is encoded as

one row number, and each row stores only successors with nonzero transition probability. For each pinning pattern, a reduction table groups rows with the same free joint action. The engine averages within groups under the pinned policies and minimizes across groups, implementing Equation (4).

For B queries, the pinning mask $M[t, c]$ records which physical agents are pinned at execution time t in query c. A decision-time player (i, t) changes its original time's mask; attribution order does not move decisions in execution time. Values use layout `V[s*B+c]`, making adjacent queries contiguous. Neighboring GPU threads process queries at the same state, sharing transition coefficients while reading adjacent successor values. Two FP64 buffers hold consecutive time layers: updates read V_{t+1}, write V_t, and exchange buffers after the layer completes. The engine returns the computed initial-state risks $\widetilde{v}(C)$; oversized batches are split to fit memory.

Under the fixed action and mask limits in Appendix B.1, let Z_{model} count sparse successor entries plus joint-action rows. A batch requires $O(kBZ_{\text{model}})$ work and $O(B|\mathcal{S}| + Z_{\text{model}} + kB)$ storage.

4.3 Paired permutation estimation

We estimate the expectation in Equation (2) by sampling player orders. For each independent uniform permutation ω_r, we evaluate the $d+1$ prefix coalitions by starting with the empty set and adding players in that order, then repeat this process in the reverse order (Castro et al., 2009; Mitchell et al., 2022). Adding a player between consecutive prefixes gives its marginal contribution; averaging its contributions from the two orders gives

$$D_{p,r} = \tfrac{1}{2}\{\Delta_p(\omega_r) + \Delta_p(\operatorname{rev}\omega_r)\}, \qquad \Delta_p(\omega) = v(P_p^\omega \cup \{p\}) - v(P_p^\omega). \tag{5}$$

Reversal preserves uniformity, so $\mathbb{E}D_{p,r} = \phi_p$; Proposition 1 gives $0 \le D_{p,r} \le \Upsilon$. The two orders are dependent and together form one independent sampling unit.

Let $Z_{p,r}$ be the pair contribution obtained from computed risks $\widetilde{v}(C)$ using floating-point subtraction and averaging. After m pairs, the reported estimate is the ordinary mean, computed in CPU arithmetic:

$$\widehat{\phi}_{p,m} = \frac{1}{m}\sum_{r=1}^{m} Z_{p,r}. \tag{6}$$

For fixed m, the corresponding exact-arithmetic mean of $D_{p,r}$ is unbiased and sums to Υ; adaptive stopping need not preserve unbiasedness. The two extra endpoint queries evaluate $v(\emptyset)$ and $v(\mathcal{P})$ to determine Υ. Before value reuse, m pairs request $2 + 2m(d+1)$ coalition occurrences, counting the endpoints again within each prefix sequence. Stopping is checked after each completed batch.

4.4 Adaptive stopping and its conditions

We use confidence sequences, intervals designed to cover each responsibility throughout sampling (Waudby-Smith & Ramdas, 2024). Appendix B.2 constructs these intervals and retains their intersection across completed batches, denoted $I_{p,m} = [L_{p,m}, R_{p,m}]$. Theorem 1 gives simultaneous coverage throughout sampling with probability at least $1-\alpha$ under its model, sampling, and numerical conditions. The implementation has not verified all numerical conditions, so the experiments report empirical interval checks rather than numerical certificates; Appendix B.3 specifies this distinction.

For finite estimates and finite, ordered, nonempty intervals, define

$$H_{p,m} = \max\{|\widehat{\phi}_{p,m} - L_{p,m}|, |R_{p,m} - \widehat{\phi}_{p,m}|\}. \tag{7}$$

For absolute tolerance $\epsilon > 0$, sampling stops when $\max_p H_{p,m} \le \epsilon$, using a computed radius that does not underestimate $H_{p,m}$. Primary whole-agent and Overcooked runs use $\epsilon = \widetilde{\Upsilon}/20$, where $\widetilde{\Upsilon} = \widetilde{v}(\mathcal{P}) - \widetilde{v}(\emptyset)$ is the computed attributable risk gap. The exact-gap qualification appears in Appendix B.3.

5 Experiments and Results

We first measure how much of the speedup comes from sampling and how much from the GPU, and check the estimates against exhaustive references. We then examine agent interactions, decision-time attribution, and policies fitted to human play.

We vary agent count, state-space size, and horizon to examine two costs: the number of policy combinations and the work per recovery calculation. The primary TicketBench grid combines $n \in \{16, 20, 24, 28, 32\}$, about 10^3 or 10^4 states (including states unreachable within the horizon), and $k \in \{50, 100\}$ into 20 configurations. Three generator seeds give 60 tasks, and three sampler seeds give 180 runs per engine. We use an A100 80 GB PCIe GPU and a 32-core OpenMP/AVX-512 CPU baseline, both in FP64. All timings below exclude model construction and engine initialization, which can dominate total time for small jobs.

5.1 Sampling savings and GPU gains

We start with the overall speedup over exhaustive attribution. All 180 sampled runs on each engine reach their stopping target, and on the GPU, runs take 0.07–0.19 s with about 10^3 states and 0.21–1.06 s with about 10^4 states, so each attribution on these models takes about a second or less. Figure 3 shows the combined speedup (a) and the part due to the GPU alone (b). Panel (a) compares sampled estimates, stopped at the target tolerance, with exhaustive FP64 enumeration.

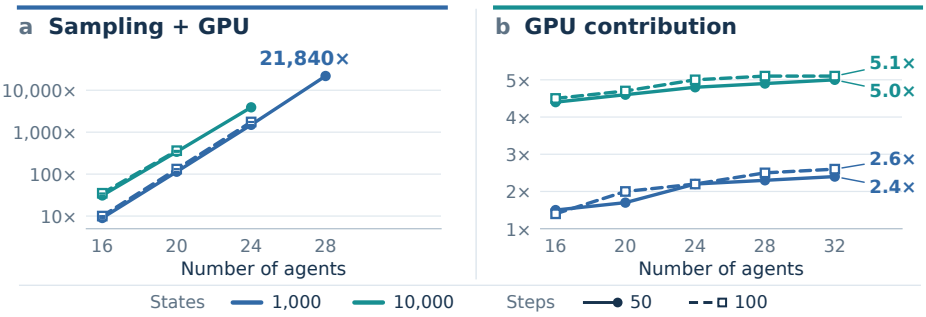


Figure 3: Separating sampling savings from GPU acceleration. (a) Exhaustive CPU time divided by sampled GPU time. (b) Sampled CPU time divided by sampled GPU time for identical queries. Points are medians of the matched ratios over three generator seeds and three sampler seeds. Panel (a) includes 12 completed configurations; seven were not run and one reached a process time limit. Panel (b) includes all 20 configurations. Sampling supplies most of the gain at larger agent counts, and GPU execution adds a further speedup.

At 28 agents, about 10^3 states, and 50 steps, the median sampled GPU time is 0.105 s, whereas exhaustive CPU attribution takes 2290 s (about 38 minutes). Over the 36 tasks whose exhaustive CPU runs completed, the median of the 108 matched runtime ratios (one per task and sampler seed) is 233×. This speedup combines sampling savings with GPU acceleration; we separate the two next.

With the GPU fixed, exhaustive time divided by sampled time is 1.9–6.7× at 16 agents and 4671–5454× in the measured 28-agent settings. Sampling therefore becomes the larger source of savings as the number of coalitions grows. With the sampler fixed, both engines run identical permutations, recovery calculations, and batch schedules in all 180 matched runs, and the per-configuration median CPU/GPU ratio is 1.4–5.1×.

We also compare the intervals with exhaustive allocations. Exhaustive GPU references are available for 42 tasks (126 sampled runs per engine), and in all 126 runs every reported interval contains its FP64 reference value.

GPU execution and batching. A separate study tests GPU execution on identical random coalition batches across 60 settings. These cover 8–24 agents, approximately 10^3–10^6 states, and horizons of 25, 100, or 400 steps. The overall median GPU gain is 5.99×. Figure 4, left, shows that the gain generally increases with the number of states. Only the smallest setting favors the CPU (0.7×), because it gives the GPU too little work. As an ablation, we fix the GPU and the exhaustive coalition set and compare batched evaluation with one coalition per launch, in eight settings with 8 or 12 agents, about 10^3 or 10^4 states, and 25 or 100 steps. Batching is 136–1836× faster, with the largest gains (1829× and 1836×) at 12 agents and about 10^3 states, because it avoids repeated launches and exposes more parallel work.

Matched reachability pruning. We also test whether GPU gains remain after both engines remove states unreachable within the horizon. Across 12 settings with identical queries, the median GPU gain is $5.57\times$ (range 2.33–$6.58\times$), so the GPU advantage does not come from processing unreachable states. Pruning time is excluded.

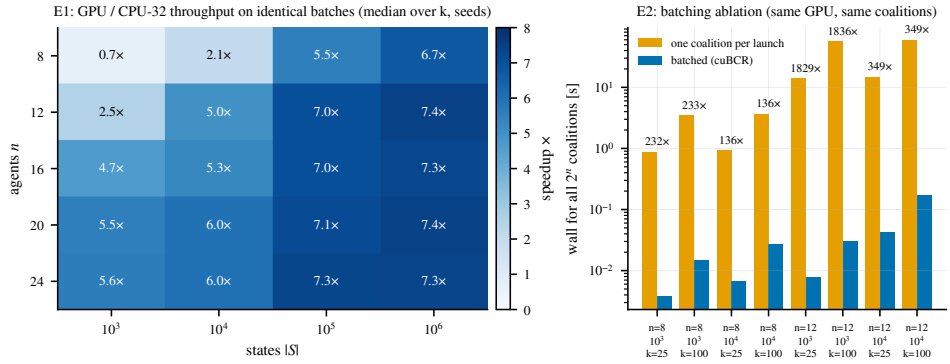


Figure 4: Testing GPU execution and coalition batching. Left: GPU throughput divided by 32-core CPU throughput for identical coalition batches. Cells show medians over horizons and seeds. GPU gains generally increase with the number of states; only the smallest setting favors the CPU. Right: time to evaluate all 2^n coalitions on the same GPU. Batching is 136–$1836\times$ faster than evaluating one coalition per launch. These results support evaluating many recovery queries together using the shared model.

5.2 Interactions and Estimator Quality

We next ask whether changing one agent at a time ranks agents the same way as Shapley attribution. The leave-one-out score $\ell_p = v(\mathcal{P}) - v(\mathcal{P} \setminus \{p\})$ lets only player p change, whereas the Shapley value averages the contribution of p over all sets of fixed policies.

We compare these scores on 90 controlled models with five interaction patterns. The models use 8, 12, or 16 agents, about 1000 states, two horizons, and three generator seeds. Median Kendall τ_a rank correlations (1 means identical rankings) are 0.95 for plain pipelines, 0.99 with an irrelevant agent, 0.81 for symmetric agents, and 0.83 for joint contributors, but only 0.08 for redundant contributors. The two methods select the same top player in only 17% of redundant models, with rank differences of up to 14 positions.

In these redundant models, either contributor can be enough to cause the failure. Allowing only one to change may therefore leave failure risk unchanged. Shapley attribution also considers cases where both may change. The results show why coalition context matters when agents have overlapping effects on failure. Rankings use unnormalized scores and break ties by player index.

Agreement across sampling budgets. To check sampling separately from recovery solving, we use 13 exhaustive coalition-value tables. Each query retrieves a stored risk, so this experiment removes Bellman evaluation from the comparison. Across eight requested budgets (2^9–2^{16}) and three seeds, every reported interval contains its corresponding exhaustive FP64 reference value in all 312 runs. Together with the primary reference checks, this supports empirical agreement across the tested budgets and models.

5.3 Decision-Time Attribution and Fitted Policies

Decision-time attribution asks when an agent's policy contributes to failure risk. Each agent at each global decision time becomes a separate player, giving $d = nk$. This increases the attribution problem without changing the underlying recovery calculation. Across six TicketBench models with 12–400 players, all three sampler seeds reach each tested absolute radius target in $\{.02, .01, .005\}$.

Table 1 reports the smallest target, .005, corresponding to 0.5 percentage points of failure probability. The two 400-player models take median GPU times of 0.5 s and 3.8 s, with 961 and 9985 states.

These results demonstrate computational feasibility for hundreds of decision-time players in the tested models.

The 300-player model takes 22.5 s because its model is much larger: it has 100,009 states, only 683 of which are reachable within the horizon, and we ran it without pruning. Player count alone therefore does not predict runtime.

Table 1: Decision-time attribution at absolute radius 0.005 (median GPU time over three sampler seeds).

Agents	States	Steps	Players	Queries	GPU (s)
3	55	4	12	2246	0.014
8	1009	25	200	342908	0.5
8	9985	25	200	302306	3.2
12	100009	25	300	225752	22.5
16	961	25	400	396190	0.5
16	9985	25	400	397794	3.8

Human-fitted Overcooked reference. TicketBench uses generated policies. We next test the attribution procedure with policies fitted from human actions in the restricted `cramped_room` kitchen of Overcooked-AI (Carroll et al., 2019). Two roles must load three onions before the deadline. A tick is one simulator decision step. At 10 ticks, the model has 20 decision-time players. Exhaustive evaluation of all 2^{20} coalitions takes 110.1 s, while sampled runs take 1.62–1.77 s. Every reported interval contains its corresponding exhaustive FP64 reference value in all three runs. This extends the runtime and numerical-agreement results to a simulator model with fitted policies. It does not establish agreement with human responsibility judgments, and the intervals exclude policy-estimation error.

5.4 Independent numerical checks

Agreement between our CPU and GPU engines could leave a shared implementation error undetected. We therefore also check coalition risks against independent solvers. The checks use nine adapted PRISM-games models (Kwiatkowska et al., 2018; 2020) and six small TicketBench models checked with Storm (Hensel et al., 2022).

The PRISM-games models contain 75–25624 states, use horizons of 6–25, and assign uniform original policies over legal actions. External solvers receive transitions after pinned-policy averaging and check reachability and minimization over free actions.

Maximum coalition-value differences are 1.86×10^{-9} for PRISM-games and 4.99×10^{-11} for Storm. Both are below the 10^{-6} acceptance threshold. These results support the numerical agreement of the recovery calculations underlying the responsibility estimates.

6 Conclusion

Attributing responsibility in collaborative multi-agent systems has long been computationally bottlenecked by the combinatorial explosion of counterfactual coalitions. We introduced cuBCR, a framework that transforms counterfactual responsibility attribution from an intractable offline problem into an interactive, near real-time analysis tool. By unifying adaptive paired-permutation sampling with massively parallel, batched GPU Bellman updates over a shared transition model, cuBCR achieves a median $233\times$ speedup over exhaustive CPU attribution. Beyond algorithmic acceleration, cuBCR demonstrates the practical necessity of cooperative game-theoretic attribution: when multiple agents possess backup recovery options, changing each agent in isolation conceals their actual role in preventing failure, whereas cuBCR correctly shares avoidable risk across these redundant actions. By reducing attribution latency from tens of minutes to seconds, cuBCR provides systems engineers and safety auditors with a deployable debugging engine to iteratively refine agent policies, diagnose cascade failures, and harden autonomous multi-agent workflows before deployment.

AI USE STATEMENT

We used generative AI tools to assist in editing the text, enhancing figure aesthetics, and providing draft critique. All content was reviewed and edited by the authors. The authors take responsibility for the final text, claims, and artifacts.

REFERENCES

Amir Asiaee. Certified interventional fidelity: Anytime-valid, adaptive evaluation of causal claims in mechanistic interpretability. In *Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence*, volume 337 of *Proceedings of Machine Learning Research*, pp. 210–227, 2026. URL https://proceedings.mlr.press/v337/asiaee26d.html.

Christel Baier, Roxane van den Bossche, Sascha Klüppelholz, Johannes Lehmann, and Jakob Piribauer. Backward responsibility in transition systems using general power indices. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 20320–20327, 2024. doi: 10.1609/aaai.v38i18.30013.

João Bento, Pedro Saleiro, André F. Cruz, Mário A. T. Figueiredo, and Pedro Bizarro. TimeSHAP: Explaining recurrent models through sequence perturbations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2565–2573, 2021. doi: 10.1145/3447548.3467166. URL https://arxiv.org/abs/2012.00073.

Patrick Blöbaum, Peter Götz, Kailash Budhathoki, Atalanti A. Mastakouri, and Dominik Janzing. DoWhy-GCM: An extension of DoWhy for causal inference in graphical causal models. *Journal of Machine Learning Research*, 25(147):1–7, 2024. URL https://jmlr.org/papers/v25/22-1258.html.

Dragan Bošnački, Stefan Edelkamp, Damian Sulewski, and Anton Wijs. Parallel probabilistic model checking on general purpose graphics processors. *International Journal on Software Tools for Technology Transfer*, 13:21–35, 2011. doi: 10.1007/s10009-010-0176-4. URL https://link.springer.com/article/10.1007/s10009-010-0176-4.

Micah Carroll, Rohin Shah, Mark K. Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-AI coordination. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/f5b1b89d98b7286673128a5fb112cb9a-Abstract.html.

Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009. doi: 10.1016/j.cor.2008.04.004.

Milan Češka, Petr Pilař, Nicola Paoletti, Luboš Brim, and Marta Kwiatkowska. PRISM-PSY: Precise GPU-accelerated parameter synthesis for stochastic systems. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 9636 of *Lecture Notes in Computer Science*, pp. 367–384, 2016. doi: 10.1007/978-3-662-49674-9_21.

Mengzhuo Chen, Junjie Wang, Fangwen Mu, Yawen Wang, Zhe Liu, Huanxiang Feng, and Qing Wang. Seeing the whole elephant: A benchmark for failure attribution in LLM-based multi-agent systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19888–19905. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.912. URL https://aclanthology.org/2026.acl-long.912/.

Ulrich Faigle and Walter Kern. The Shapley value for cooperative games under precedence constraints. *International Journal of Game Theory*, 21:249–266, 1992. doi: 10.1007/BF01258278.

Jan Heemstra, Muhammad Osama, and Anton Wijs. Towards end-to-end GPU acceleration of PCTL model checking. In *Principles of Verification: Cycling the Probabilistic Landscape*, volume 15261 of *Lecture Notes in Computer Science*, pp. 314–337. Springer, 2025. doi: 10.1007/978-3-031-75775-4_14.

Christian Hensel, Sebastian Junges, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. The probabilistic model checker Storm. *International Journal on Software Tools for Technology Transfer*, 24(4):589–610, 2022. doi: 10.1007/s10009-021-00633-z.

Milad Kazemi, Jessica Lally, Ekaterina Tishchenko, Hana Chockler, and Nicola Paoletti. Counterfactual influence in Markov decision processes. In *Proceedings of the Fourth Conference on Causal Learning and Reasoning*, volume 275 of *Proceedings of Machine Learning Research*, pp. 792–817, 2025. URL https://proceedings.mlr.press/v275/kazemi25a.html.

Patrick Kolpaczki, Viktor Bengs, Maximilian Muschalik, and Eyke Hüllermeier. Approximating the Shapley value without marginal contributions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 13246–13255, 2024. doi: 10.1609/aaai.v38i12.29225.

Marta Kwiatkowska, Gethin Norman, David Parker, and Gabriel Santos. Automated verification of concurrent stochastic games. In *International Conference on Quantitative Evaluation of Systems (QEST)*, volume 11024 of *Lecture Notes in Computer Science*, pp. 223–239. Springer, 2018. doi: 10.1007/978-3-319-99154-2_14. URL https://www.prismmodelchecker.org/papers/qest18csgs.pdf.

Marta Kwiatkowska, Gethin Norman, David Parker, and Gabriel Santos. PRISM-games 3.0: Stochastic game verification with concurrency, equilibria and time. In *International Conference on Computer Aided Verification (CAV)*, volume 12225 of *Lecture Notes in Computer Science*, pp. 475–487. Springer, 2020. doi: 10.1007/978-3-030-53291-8_25. URL https://www.prismmodelchecker.org/papers/cav20pg3.pdf.

Jianlin Li, Nick Guo, Peter Ye, and Yizhou Zhang. Tensor probabilistic model checking of finite-horizon Markov chains. In *International Conference on Computer Aided Verification (CAV)*, volume 16684 of *Lecture Notes in Computer Science*, pp. 502–525, 2026a. doi: 10.1007/978-3-032-32537-2_24.

Mingxuan Li, Kai-Zhan Lee, and Elias Bareinboim. Counterfactual shapley credit assignment. *Reinforcement Learning Journal*, 7, 2026b. URL https://rlj.cs.umass.edu/2026/papers/Paper45.html.

Weida Li, Yaoliang Yu, and Bryan Kian Hsiang Low. Adalina: Adaptive linear approximation for the Shapley value and beyond. In *International Conference on Machine Learning*, 2026c. URL https://openreview.net/forum?id=hMTCuqjCW8. Updated full version: arXiv:2604.08438v3.

Anji Liu, Kareem Ahmed, and Guy Van Den Broeck. Scaling tractable probabilistic circuits: A systems perspective. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 30630–30646, 2024. URL https://proceedings.mlr.press/v235/liu24a.html.

Corto Mascle, Christel Baier, Florian Funke, Simon Jantsch, and Stefan Kiefer. Responsibility and verification: Importance value in temporal logics. In *Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science*, pp. 1–14, 2021. doi: 10.1109/LICS52264.2021.9470597.

Rory Mitchell, Joshua Cooper, Eibe Frank, and Geoffrey Holmes. Sampling permutations for Shapley value estimation. *Journal of Machine Learning Research*, 23(43):1–46, 2022. URL https://jmlr.org/papers/v23/21-0439.html.

Chunyan Mu and Muhammad Najib. Counterfactual reasoning for causal responsibility attribution in probabilistic multi-agent systems, 2026. URL https://arxiv.org/abs/2605.13077v2. Accepted at IJCAI-ECAI 2026; full version, arXiv:2605.13077v2.

Ryohei Oura and Yuji Ito. Probability-raising causality for uncertain parametric Markov decision processes with PAC guarantees. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 286 of *Proceedings of Machine Learning Research*, pp. 3300–3321, 2025. URL https://proceedings.mlr.press/v286/oura25a.html.

Guilherme Dean Pelegrina, Leonardo Tomazeli Duarte, and Michel Grabisch. A k-additive choquet integral-based approach to approximate the SHAP values for local interpretability in machine learning. *Artificial Intelligence*, 325:104014, 2023. doi: 10.1016/j.artint.2023.104014.

Lloyd S. Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker (eds.), *Contributions to the Theory of Games*, volume II, pp. 307–317. Princeton University Press, 1953. doi: 10.1515/9781400881970-018.

Stelios Triantafyllou, Adish Singla, and Goran Radanovic. On blame attribution for accountable multi-agent sequential decision making. In *Advances in Neural Information Processing Systems*, volume 34, pp. 15774–15786, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/848c4965359e617d5e16c924b4a85fd9-Abstract.html.

Ian Waudby-Smith and Aaditya Ramdas. Estimating means of bounded random variables by betting. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 86(1):1–27, 2024. doi: 10.1093/jrsssb/qkad009.

Zhimin Wu, Ernst Moritz Hahn, Akın Günay, Lijun Zhang, and Yang Liu. GPU-accelerated value iteration for the computation of reachability probabilities in MDPs. In *European Conference on Artificial Intelligence (ECAI)*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pp. 1726–1727. IOS Press, 2016. doi: 10.3233/978-1-61499-672-9-1726.

Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. AgenTracer: Who is inducing failure in the LLM agentic systems? In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/134ed7a477770f227f12450ef0cbb8f4-Abstract-Conference.html.

A Allocation properties and recovery semantics

A.1 Monotonicity and the equivalent release game

Proposition 1. *For $C \subseteq D \subseteq \mathcal{P}$, $v(C) \leq v(D)$. Every marginal contribution in Equation (2) lies in $[0, \Upsilon]$, and $\sum_{p \in \mathcal{P}} \phi_p = \Upsilon$.*

Proof. The recovery controller under C can reproduce the behavior under D by imitating its additional pinned policies. Having fewer restrictions therefore cannot increase the minimum failure risk. Since all coalition values lie between $v(\emptyset)$ and $v(\mathcal{P})$, every marginal contribution lies in $[0, \Upsilon]$. Along any ordering, successive differences cancel, leaving $v(\mathcal{P}) - v(\emptyset) = \Upsilon$. Averaging over orderings preserves this sum. □

For whole-agent attribution, let $S \subseteq N$ be the agents allowed to change and define $g(S) = v(N) - v(N \setminus S)$. Reversing a pinning order gives a release order with identical player contributions, hence the same Shapley allocation. This matches Triantafyllou et al. (2021) when objectives and recovery strategies match.

A.2 A small sequential example

In Figure 1, reservation A, payment B, and notification C act at times 0, 1, and 2. Original policies are deterministic; recovery may choose an alternative. For reservation/payment pairs (alternative, alternative), (alternative, original), (original, alternative), and (original, original), failure probabilities are 0.10, 0.30, 0.20, and 0.70. These are illustrative inputs, not measurements; notification has no effect.

Thus $v(\emptyset) = 0.10$, $v(\{\mathrm{A}\}) = 0.20$, $v(\{\mathrm{B}\}) = 0.30$, and $v(\{\mathrm{A}, \mathrm{B}\}) = 0.70$; adding C changes nothing. Figure 1 lists released agents: releasing only A gives $v(\{\mathrm{B}, \mathrm{C}\}) = 0.30$.

Half the six pinning orders place A before B, giving respective marginal risk increases of 0.10 and 0.50; the others give increases of 0.40 for A and 0.20 for B. Hence $\phi_{\mathrm{A}} = (0.10 + 0.40)/2 = 0.25$, $\phi_{\mathrm{B}} = (0.50 + 0.20)/2 = 0.35$, and $\phi_{\mathrm{C}} = 0$. Their sum is $\Upsilon = 0.60$, leaving residual risk 0.10 unassigned. Executions always preserve reservation before payment.

A.3 Recovery classes and time dependence

Stationary recovery uses a fixed state-dependent action distribution, as in Mu & Najib (2026); our controller also observes remaining time. For whole-agent attribution, restricting recovery to stationary independent policies, then to deterministic stationary policies, gives

$$v_{\text{time}}(C) = v(C) \leq v_{\text{stationary}}(C) \leq v_{\text{deterministic stationary}}(C).$$

Encoding time in state removes the time-dependence distinction. Coalition-risk bounds do not generally bound individual stationary-policy Shapley values.

B Additional methodological details

B.1 Implementation limits and caching

Simultaneous implementation limits are 64 physical agents, three active agents per state, eight actions per active agent, and 64 joint-action rows per state. Time-indexed masks permit $d = nk > 64$ decision-time players.

Coalition values are reused across batches for whole-agent games and decision-time games with at most 24 players. Larger decision-time games evaluate every requested occurrence.

B.2 Confidence-sequence construction and numerical conditions

For $\alpha \in (0, 1)$, prespecify valid bounds

$$|Z_{p,r} - D_{p,r}| \leq \beta, \qquad U \geq \Upsilon$$

for every player and possible pair, with $\beta \geq 0$ covering coalition-evaluation and pair-arithmetic errors. If $U = 0$, all exact Shapley values are zero and no sampling is needed.

For $U > 0$, set $a = -\beta$, $b = U + \beta$. Each rounded contribution must be a fixed function of its base ordering; its mean $\mu_p = \mathbb{E}Z_{p,r}$ satisfies $|\mu_p - \phi_p| \leq \beta$. Normalize:

$$X_{p,r} = \frac{Z_{p,r} - a}{b - a} \in [0, 1].$$

Choose predictable weights $\lambda_{p,r} \in (0, 1)$ and predictions $\widehat{x}_{p,r} \in [0, 1]$ using only completed pairs before r, excluding current-pair contributions and unrevealed permutations even if already generated.

With $h(\lambda) = -\log(1 - \lambda) - \lambda$, Waudby-Smith & Ramdas (2024, Theorem 2) gives

$$\begin{aligned} W_{p,m} &= \sum_{r \leq m} \lambda_{p,r}, \qquad c_{p,m} = \frac{\sum_{r \leq m} \lambda_{p,r} X_{p,r}}{W_{p,m}}, \\ q_{p,m} &= \frac{\log(2d/\alpha) + \sum_{r \leq m} h(\lambda_{p,r})(X_{p,r} - \widehat{x}_{p,r})^2}{W_{p,m}}, \\ J_{p,m} &= a + (b - a)[c_{p,m} - q_{p,m},\ c_{p,m} + q_{p,m}]. \end{aligned} \tag{8}$$

The logarithmic term allocates error across d players and both tails; the weighted center generally differs from the ordinary mean in Equation (6).

Require a numerical enclosure $J^{\text{enc}}_{p,j} \supseteq J_{p,j}$ of the exact-arithmetic interval. For completed-batch counts $\mathcal{B}_m$ up to m, retain

$$I_{p,m} = [L_{p,m}, R_{p,m}] = \bigcap_{j \in \mathcal{B}_m} \left(J^{\text{enc}}_{p,j} + [-\beta, \beta] \right). \tag{9}$$

Expansion by β transfers coverage from μ_p to ϕ_p; all interval operations must preserve exact-arithmetic containment.

B.3 Numerical error and conditional coverage

Theorem 1 (Conditional simultaneous coverage)**.** *Fix a valid stochastic model and independently sample uniform base permutations. Suppose each rounded pair-observation vector is produced by the same fixed function of its base permutation. Assume that the bounds β and U are valid, weights and predictions use only earlier completed pairs, and numerical interval calculations satisfy the containment requirements above. Then, with probability at least $1 - \alpha$,*

$$\phi_p \in I_{p,m} \quad \textit{for every player } p \textit{ and every completed-batch count } m.$$

Proof. Independent base permutations give independent rounded pair observations from a fixed distribution. Equation (8) covers their mean throughout sampling for each player. Combining the error probabilities gives simultaneous coverage for all players; independence between players is unnecessary. Since $|\mu_p - \phi_p| \leq \beta$, enlarging each interval by β transfers coverage to the exact Shapley value. Numerical containment, intersection over completed batches, and adaptive stopping preserve this event. □

Probability-input validity, uniform error bounds, and conservative interval arithmetic remain unverified in the implementation. Experiments assess agreement and runtime; recorded interval checks are not numerical certificates.

Primary whole-agent and Overcooked runs use $\epsilon = \widetilde{\Upsilon}/20$, with computed full-minus-empty coalition risk $\widetilde{\Upsilon}$; certifying $\Upsilon/20$ requires a verified lower bound on the exact gap. Coverage conditions on supplied transitions, policies, and permitted recovery strategies, excluding uncertainty in fitted policies or omitted behavior.