Dispatchable Coordination Envelopes for Embodied Collaboration Under Sequential Uncertainty

Yuchen Wang[1] **Fanqi Kong**[1] **Ke Shi**[4] **Zhipeng Liu**[1]
Xiaofei Yue[2] **Zhaoxuan Li**[3] **Tingting Li**[1*] **Ziming Zhao**[1*]
[1]Zhejiang University
[2]Beijing Institute of Technology
[3]State Key Laboratory of Cyberspace Security Defense,
Institute of Information Engineering, Chinese Academy of Sciences
[4]University of Science and Technology of China

Abstract

Embodied AI systems can pass planning and still fail in execution when a delay, resource conflict, synchronization change, or candidate invalidation turns a valid point schedule into broken commitments. We introduce **FLECAR**, which gives the executor a dispatchable coordination envelope instead of fixed start times. Offline, Flexible Envelope Synthesis builds executable windows, coupling constraints, and a dispatch graph; online, Controllability-Aware Repair updates that envelope after events, dispatching within the retained region when possible and escalating from localized to wider repair when needed. On generated coordination instances with shared event streams and budgets, FLECAR raises post-event feasibility from 0.3880 to 0.8400 and terminal success from 0.5800 to 0.8400, while reducing normalized repair burden from 0.5357 to 0.2964 relative to legacy point-schedule repair.

1 Introduction

Execution in embodied AI systems can fail even when high-level planning succeeds. A plan may assign the right subtasks to the right agents, yet become unusable after a delayed collaborator, a blocked resource, a changed object state, or a temporary agent outage. The semantic plan can still be sensible while the schedule passed to the executor has no legal next move. Point schedules make this failure mode sharper. They turn an optimizer output into exact-time commitments, so a small disturbance can invalidate the chosen candidate and push conflicts downstream. An envelope keeps more of the unfinished schedule alive, so when an event removes one option, the executor can spend remaining slack or switch within the admissible region before invoking repair.

We introduce **FLECAR**, an execution-layer method that carries such an envelope through the offline-online loop. Flexible Envelope Synthesis (FES) builds executable windows, coupling constraints, and a dispatch graph. Controllability-Aware Repair (CAR) updates the envelope after each event, checks validity and dispatchability, and re-synthesizes a local region when the current envelope cannot be dispatched or the gate fails. The method turns the execution object itself into the thing being maintained, including selected starts, usable windows, coupled commitments, and local alternatives that remain after partial execution. Figure 1 summarizes the shift from point commitments to an executable region.

We test whether this carried envelope makes sequential disturbances cheaper to absorb and repair. On generated coordination instances, all methods share event streams, trigger-time sampling, budgets,

*Co-corresponding authors: Tingting Li (litt2020@zju.edu.cn) and Ziming Zhao (zhaoziming@zju.edu.cn).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

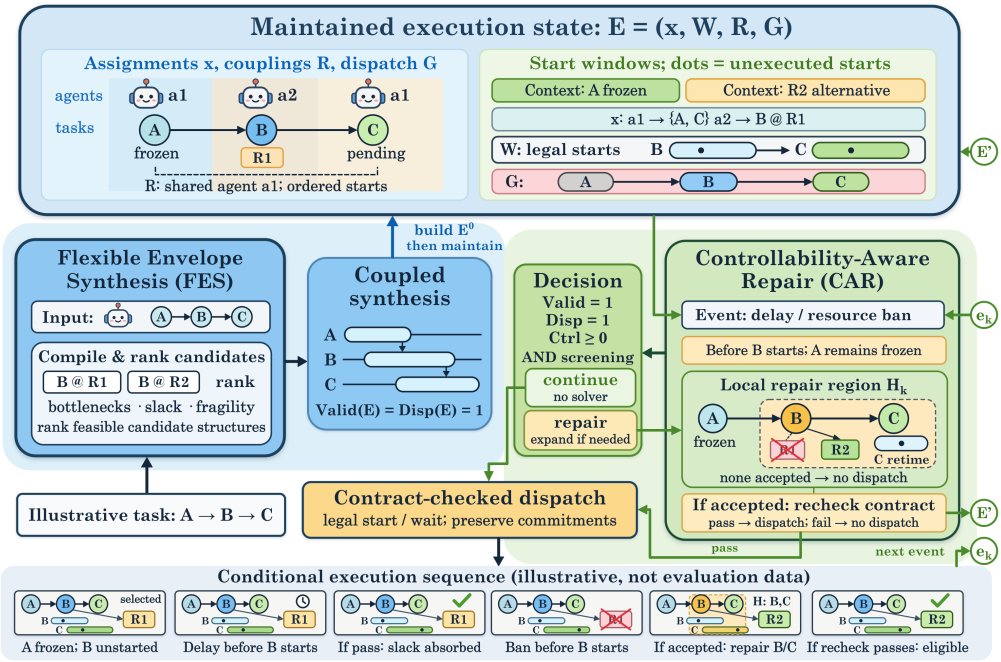


Figure 1: Mechanism overview. FLECAR replaces brittle point commitments with a dispatchable envelope that keeps slack, coupling constraints, and local repair choices available.

seeds, and evaluation metrics. Against legacy point-schedule repair, FLECAR raises post-event feasibility from 0.3880 to 0.8400 and terminal success from 0.5800 to 0.8400, while reducing normalized repair burden from 0.5357 to 0.2964. Static feasible rate also rises from 0.4653 to 0.6600 relative to the point-schedule compiler baseline. A matched point baseline receives CAR-like repair terms and aligned event streams. It slightly exceeds FLECAR on pooled one-step survivability, while FLECAR retains higher post-event feasibility, lower repair burden, positive repair-free absorption in the main online setting, and better tight-budget stability. An 800-event matched comparison examines the effect of the maintained execution object. Together, these evaluations test whether disturbances can be absorbed within retained choices and how much of the unfinished schedule must change when repair is necessary. This paper makes the following contributions:

- We introduce dispatchable coordination envelopes as execution-layer state, retaining executable windows, coupling constraints, and dispatch choices throughout collaborative execution.
- We develop FLECAR, combining offline flexible envelope synthesis with online controllability-aware repair to absorb disturbances and revise affected coordination decisions.
- We evaluate FLECAR under sequential disturbances and external task settings, showing gains against point-based repair and generic replanning under the same online contract.

2 Related Work

2.1 Embodied Coordination

Savva et al.'s Habitat [29], Kolve et al.'s AI2-THOR [15], Deitke et al.'s RoboTHOR [8], and Xia et al.'s Interactive Gibson [35] establish interactive embodied platforms where navigation, manipulation, object state, and resource access become execution-relevant conditions rather than abstract background assumptions. Szot et al.'s Habitat 2.0 [33] emphasizes rearrangement for home assistants; Puig et al.'s VirtualHome [27] represents household activities as programs; Shridhar et al.'s ALFRED [30] and ALFWorld [31] connect everyday tasks to grounded instructions and text-environment interaction; Padmakumar et al.'s TEACh [23] adds task-driven dialogue; and BEHAVIOR [32] and BEHAVIOR-1K [17] stress everyday household activities and ecological preconditions at scale. Chang et al.'s PARTNR [4] makes the multi-agent pressure explicit because collaborators share resources, synchronize progress, and must continue execution after coupled

task-state changes. These works motivate the execution pressures in the generated contract. FLECAR focuses on the object handed from task reasoning to execution. Instead of committing agents to a point schedule, it maintains a dispatchable envelope whose slack, coupling constraints, dispatch graph, and local repair choices remain available when admissibility changes arrive.

2.2 Temporal Dispatchability

Dechter, Meiri, and Pearl [7] introduced temporal constraint networks as the classical language for consistency among timepoint constraints. Vidal and Fargier [34], Morris, Muscettola, and Vidal [21], and Morris [20] develop controllability and dispatchability for temporal plans whose durations are observed during execution rather than fully controlled in advance. Comin, Posenato, and Rizzi [6] extend temporal networks with disjunctive constraints. Later algorithms and tools by Cairo and Rizzi [2], Comin and Rizzi [5], Hunsberger and Posenato [12, 13], and Cairo, Comin, and Rizzi [3], together with Posenato's CSTNU Tool [26], support exact checking for temporal models including STNUs and CSTNs. FLECAR uses this machinery where its assumptions apply. Once assignments and resource orders are fixed, the selected execution object induces a temporal core with timepoints, windows, contingent durations, precedence, synchronization, and fixed non-overlap constraints. Event-driven candidate invalidation, resource bans, object-state filtering, temporary agent outages, and reselection mutate an outer admissibility layer, which is evaluated operationally.

2.3 Scheduling Comparators

Gerkey and Mataric [10] formalize task allocation, Parker [24] studies fault-tolerant multirobot cooperation, Dias et al. [9] survey market-based coordination, and Korsah et al. [16], Nunes et al. [22], and Khamis et al. [14] map MRTA variants with allocation, ordering, temporal, and robustness constraints. Constraint-based scheduling, integrated optimization, and CP-SAT-style solvers provide strong machinery for compiling or rebuilding feasible schedules under constraints [1, 11, 25]. Related orchestration work jointly optimizes function placement and resource allocation in distributed systems [36, 37]. HyFaaS [38] models workflow dependencies and coordinates computation and communication resources during execution. In quantum services, Moirai [18] coordinates device allocation and deployment, while QCDeploy [19] ranks candidate deployment strategies; time-aware calibration work [40] schedules calibration experiments while accounting for parameter dependencies and drift. QuaMap [39] provides a multi-backend dataset for evaluating quantum circuit mapping and learning-based compilers. These studies address allocation and scheduling in different settings, while FLECAR maintains executable coordination choices during embodied task execution. Classical scheduling methods are natural comparators because they can improve assignment quality, initial schedule quality, or repair strength. The studied baselines maintain a point schedule or a newly replanned point solution, whereas FLECAR maintains a dispatchable envelope across events. The resulting comparison separates initial compilation, online repair, and the representation maintained between events. This distinction drives the comparator taxonomy in Section 4. Point compilation tests whether a stronger initial schedule is enough, matched repair tests whether CAR-like repair explains the gain, and library-based replanners test whether generic scheduling-style repair closes the same gap under the fixed online contract.

3 Method

FLECAR is an execution-interface method that maintains a structured object whose feasible dispatch choices, coupling constraints, and repair locality remain explicit as uncertainty is revealed. Offline, Flexible Envelope Synthesis (FES) replaces a single point schedule with executable windows, coupling constraints, and a dispatch structure. Online, Controllability-Aware Repair (CAR) assimilates disturbances, tests the event-updated dispatch contract, and either continues execution through the current envelope or re-synthesizes a localized region. Figure 2 summarizes this workflow.

3.1 Problem Setup

We study the execution-layer decision problem induced by embodied coordination under sequential uncertainty. An instance is

$$I = (A, T, P, X, K, C_{res}, C_{spa}, C_{sync}, D, \Omega, \Xi),$$

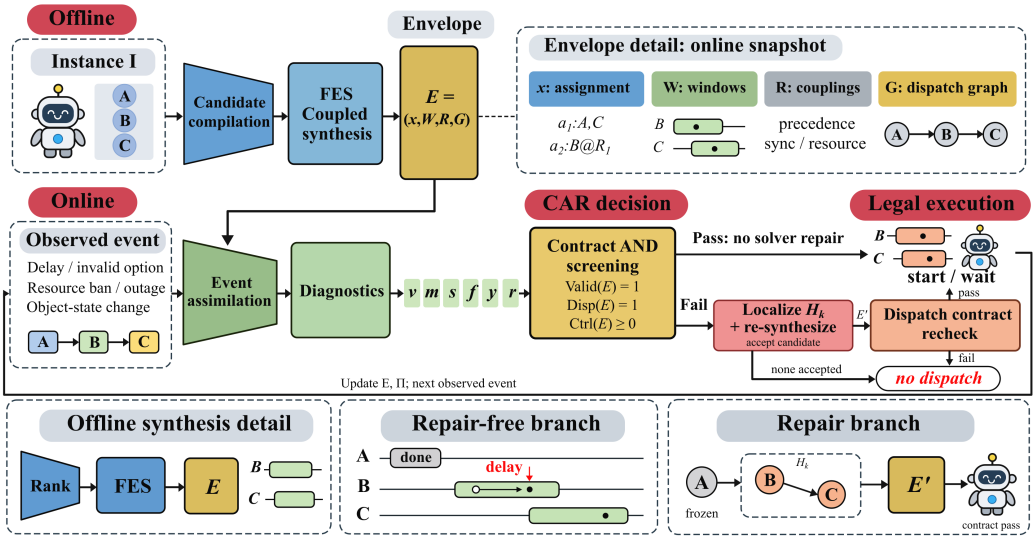


Figure 2: FLECAR workflow. FES builds $E = (x, W, R, G)$ offline; CAR assimilates events, tests validity, dispatchability, and controllability, then dispatches or re-synthesizes locally.

where agents, tasks, candidate assignments, precedence, synchronization, resource, spatial, duration, event, and disturbance-prior information define the benchmark contract; K counts event steps. The method outputs a **dispatchable coordination envelope**

$$E = (x, W, R, G).$$

Here x is the assignment decision, $W = \{[l(i), u(i)] : i \in \mathcal{T}\}$ is the set of executable windows, $R = R^{prec} \cup R^{sync} \cup R^{res}$ collects precedence, synchronization, and resource constraints, and $G = (V_G, E_G)$ is the dispatch structure. These components play different online roles. The assignment x determines which candidates remain in play, the windows W determine how much temporal freedom remains after disturbances, the couplings R determine which commitments must still be preserved jointly, and the dispatch graph G determines whether the next feasible move is local or repair is needed.

The benchmark contract has two layers. The fixed temporal core $\mathcal{T}(O)$ of an execution object O contains the timepoints, windows, contingent duration intervals, precedence, synchronization, and fixed resource-order constraints needed for dispatchable temporal execution. The embodied admissibility layer $\mathcal{B}(O, e)$ contains event-triggered candidate changes such as selected-candidate invalidation, resource-wide bans, temporary agent unavailability, and object-state filtering. FLECAR keeps these layers in the same execution object. The fixed temporal core can be checked exactly once choices are fixed, while the admissibility layer can be updated and repaired online without discarding the dispatchable structure.

For exact-core analysis, each fixed execution object is compiled to a CSTNU Tool STNU $\Gamma_{\text{exact}}(O)$; official dynamic controllability is then a property of $\mathcal{T}(O)$. The full benchmark remains operational because event assimilation can mutate candidate admissibility and require discrete reselection. For example, after a resource becomes unavailable, the admissibility layer can remove affected candidates while the temporal core retains the ordering and synchronization constraints of the unfinished tasks. The frozen prefix preserves commitments that have already been executed. Keeping these updates within one execution object lets the coordinator distinguish a change in what may be selected from a change in when selected tasks can run. A resource ban can remove an assignment without discarding valid temporal relations among the remaining tasks. Conversely, a delay can consume slack while leaving the assignment structure intact. This separation gives CAR a precise repair target and helps preserve unaffected decisions when only part of the plan has been invalidated.

The online checks follow this separation. CAR first asks whether the realized event leaves selected candidates admissible, then whether the remaining graph can dispatch legal starts, and finally whether the temporal margin is nonnegative. Repair is triggered after the maintained execution object fails this contract or the repair gate, not merely because an event arrived. The order of these checks avoids

solving a new scheduling problem for every event. Admissibility determines whether the selected choices are still available, dispatchability determines whether a legal next action can be issued, and the margin summarizes how much temporal room remains. When all three conditions hold, CAR can continue from the current state. When a condition fails, the same diagnostics identify which connected commitments should enter local repair.

Operationally, these layers are exposed through three predicates used by the online loop. $Valid(E) = 1$ means the selected candidates remain admissible after event assimilation and all frozen-prefix, precedence, synchronization, resource, and window constraints are internally consistent. $Disp(E) = 1$ means the current dispatch graph can still choose legal next starts for all unfinished tasks without violating those constraints. The controllability summary is the minimum margin.

$$Ctrl(E) = \min\{m^{prec}, m^{sync}, m^{res}, m^{win}\}.$$

Dispatch is allowed when $Valid(E) = 1$, $Disp(E) = 1$, and $Ctrl(E) \geq 0$. These checks see both the temporal structure and the event-assimilated admissibility state.

At event step k, the state is

$$z_k = (E^{k-1}, \Pi^{k-1}, e_k), \qquad g_k = Diag(z_k) = [v_k, m_k, s_k, f_k, y_k, r_k] \in [0, 1]^6.$$

The diagnostic vector records normalized gate evidence comprising validity violations v_k, admissibility mutations m_k, synchronization/resource exposure s_k, frozen-prefix pressure f_k, forward-risk pressure y_k, and repair-scope pressure r_k. CAR uses this vector to decide whether to continue dispatch, expand the local repair region, or escalate to a wider repair scope.

3.2 Flexible Envelope Synthesis

FES constructs the offline execution object $E = (x, W, R, G)$. The assignment x fixes the selected candidate structure, W gives each task an executable window, R records precedence, synchronization, and resource constraints, and G stores the dispatch structure used online. The synthesis problem is to choose coupled slack that remains dispatchable under shared temporal and resource constraints. Candidate compilation ranks feasible structures before envelope optimization; acceptance is determined by the validity and dispatchability constraints.

$$score(c) = base(c) + \alpha_b b(c) + \alpha_c q(c) + \alpha_s s(c) + \alpha_d r(c) - \alpha_f g(c) + \alpha_q h(c),$$

where the feature terms summarize bottleneck pressure, conflict exposure, slack risk, structural reuse, flexibility gain, and controllability fragility. The score determines which candidate structures are considered first; acceptance still requires the validity and dispatchability constraints below. For reproducibility, all main runs fix $(\alpha_b, \alpha_c, \alpha_s, \alpha_d, \alpha_f, \alpha_q) = (0.25, 0.30, 0.15, 0.10, 0.20, 0.20)$. In the default protocol, these coefficients are not tuned per split, event family, or comparator. They freeze a single ordering protocol so the comparisons test representation and repair behavior rather than per-setting retuning.

Given the ordered candidate set, FES selects windows and coupling structure over the feasible-dispatchable set by minimizing the following objective.

$$\begin{aligned} E^\star = \arg\min_{E \in \mathcal{E}(\mathcal{C})} \quad & \lambda_{obj} L_{obj}(E) - \lambda_{flex} \Phi(E) + \lambda_{risk} \Psi(E) \\ \text{s.t.} \quad & Valid(E) = 1, \quad Disp(E) = 1. \end{aligned}$$

$\Phi(E)$ rewards executable volume and slack expansion, while $\Psi(E)$ penalizes negative critical margins and dispatch-failure exposure. The fixed weights $\lambda_{obj} = 1.0, \lambda_{flex} = 0.4, \lambda_{risk} = 0.35$ define the main protocol. The constraint set defines the method guarantee. FES accepts only a feasible-dispatchable execution object, and the objective chooses among accepted objects by nominal cost, retained slack, and predicted fragility.

3.3 Controllability-Aware Repair

CAR is the online half of FLECAR. A centralized coordinator maintains the joint envelope and invokes repair. Each event first induces an event-assimilated admissibility update, after which CAR asks whether the current envelope still satisfies the dispatch contract. The gate checks validity, dispatchability, post-ban feasibility, and bounded absorbability. Let b_{fail} count impacted subtasks

whose backup candidate set is empty after assimilation. CAR skips solver repair when either no impacted subtask is unbacked or both the unbacked fraction and remaining controllability margin satisfy the screening condition below.

$$b_{fail} = 0 \quad \text{or} \quad \left(\frac{b_{fail}}{\max(1, |\text{impacted subtasks}|)} \leq 0.34 \ \wedge \ Ctrl(E) \geq -0.15 \right).$$

CAR evaluates the screening condition together with the validity and dispatchability contract. If both hold, execution continues without solver repair. Otherwise, CAR localizes the repair region

$$H_k = Localize(E^{k-1}, e_k, g_k)$$

by expanding from impacted subtasks over temporal, shared-resource/agent, and synchronization adjacency. The default radius is one hop; high forward risk or synchronization exposure escalates the scope. This localization lets CAR preserve frozen prefixes and unaffected assignments instead of reconstructing a full point solution after every event. Within H_k, CAR minimizes edit and replanning costs while rewarding controllability recovery and limiting forward risk. Repair candidates preserve the frozen prefix and admissible choices outside H_k, and satisfy the following conditions.

$$\begin{aligned} \min_{E' \in \widehat{\mathcal{E}}(H_k)} \quad & \lambda_e C_{edit} + \lambda_r C_{replan} - \lambda_c \Delta Ctrl + \lambda_m FutureRisk \\ \text{s.t.} \quad & Valid(E') = 1, \quad Disp(E') = 1. \end{aligned}$$

Among candidates that meet these conditions, CAR selects the one with the lowest objective within the permitted repair scope. To make edit cost, replanning cost, controllability recovery, and forward risk comparable across event types, CAR normalizes diagnostic terms to $[0, 1]$ before combining them; raw margins remain intermediate signals. The fixed weights $\lambda_e = 4.0, \lambda_r = 1.0, \lambda_c = 2.0, \lambda_m = 1.5$ define the online repair protocol used in all reported runs.

3.4 Repair Scope

The dispatch contract also determines repair scope. After a gate failure, CAR localizes the affected subgraph by breadth-first expansion over temporal, shared-resource/agent, and synchronization adjacency in $O(|V_{loc}| + |E_{loc}|)$, then evaluates local, expanded, and global repair scopes. The implementation allocates a fixed fraction of the event budget to each solver call according to the following rule.

$$t_{call} = \max(0.05, \ t_{budget}/3).$$

This rule specifies the implementation budget allocation; the 0.05s tight-budget condition is an external stress setting, not a guarantee that every internal solver call finishes below 0.05s. The same fixed-object view supplies the temporal core used by official STNU checks. Once candidate choices and resource orders are fixed, the execution object induces an STNU-checkable temporal core; event-driven candidate invalidation, resource bans, temporary agent unavailability, object-state filtering, repair-free absorption, and reselection are handled by the online loop.

4 Experiments

4.1 Evaluation Design

The experiments test whether carrying a dispatchable envelope into execution improves task outcomes, whether this benefit persists against a point baseline with comparable repair, and whether generic replanners recover the same execution profile. The primary study uses generated coordination instances under a shared event contract. External evaluations use Watch-And-Help programs and VirtualHome state transitions. Within each study, methods share the task inputs, event stream, solver access, time budget, and evaluator. The generated event contract covers task delays, task failures, resource occupation, object-state changes, and temporary agent unavailability. We compare initial schedule quality, post-event behavior, repair effort, and final episode outcomes. Tables 1 and 2 report the numerical results; the discussion below focuses on what those changes reveal about the state maintained by the executor. The metrics follow the execution process rather than collapsing it into one score. Static feasibility measures the quality of the initial schedule, post-event feasibility records whether legal dispatch remains after each disturbance, and terminal success is a binary episode

Table 1: Results on generated instances.

Comparison	Metric	Point baseline	FLECAR
Static compiler	Feasibility ↑	0.4653	0.6600
Legacy repair	Post-event feasibility ↑	0.3880	0.8400
Legacy repair	Terminal success ↑	0.5800	0.8400
Legacy repair	Repair burden ↓	0.5357	0.2964
Matched repair	Post-event feasibility ↑	0.7800	0.8400
Matched repair	Repair burden ↓	0.3287	0.2964
Matched repair	One-step survivability ↑	0.1720	0.1670

Table 2: Additional results. Differences are FLECAR minus the point baseline.

Evaluation / baseline	Metric	Difference
Strict match / CAR-matched	Post-event feasibility ↑	+0.1027
	All-attempt feasibility ↑	+0.0528
	Repair-free absorption ↑	+0.0787
	Repair burden ↓	−0.0449
	Terminal feasibility ↑	+0.0400
Single event / CAR-matched	Overall feasibility ↑	+0.1052
	Delay feasibility ↑	+0.3024
	Failure feasibility ↑	+0.0728
	State-change feasibility ↑	+0.0456
VirtualHome / CAR-like	Task success ↑	+0.160
	Goal completion ↑	+0.085
	Unfinished goals ↓	−0.510
	Post-event feasibility ↑	+0.118
WAH / CAR-like point	Success ↑	+0.139
	Completion ↑	+0.0714
	Unfinished goals ↓	−0.086
WAH / legacy repair	Success ↑	+0.179
	Completion ↑	+0.1444
	Unfinished goals ↓	−0.216
50 seeds / matched point	Feasibility ↑	+0.1969
	One-step survivability ↑	+0.0581
	Repair burden ↓	−0.0558

outcome after the final event. Repair burden measures the changes required to maintain that trajectory. A method can therefore begin with a feasible plan yet struggle after events, or preserve feasibility by making large repairs. Reading these outcomes together shows whether an execution representation supports both continuity and economical recovery.

4.2 Task-Facing Performance

Figure 3 summarizes the generated-instance comparison. FLECAR improves static feasibility, post-event feasibility, and terminal success while lowering normalized repair burden. The clearest separation appears after disturbances, when the executor can spend retained slack and switch among admissible alternatives before rebuilding a large suffix. This pattern links the initial schedule representation to the quality of the state available for continued execution. The initial-feasibility gain establishes a stronger starting point, while the larger post-event gain shows that the advantage grows as execution proceeds. The terminal result indicates that this retained flexibility carries through to task completion instead of improving only an intermediate feasibility check. Lower repair burden adds a complementary explanation because the coordinator often has useful alternatives available in the maintained envelope and can resolve an event without rebuilding as much of the unfinished schedule. Together, these results support the envelope as persistent execution state.

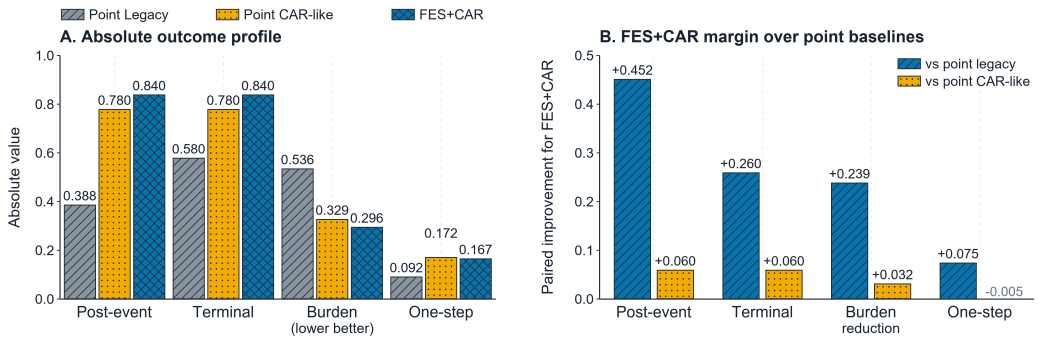


Figure 3: Online comparison under the shared contract. The right panel shows FLECAR's margins against legacy and CAR-like point baselines. Lower repair burden is better.

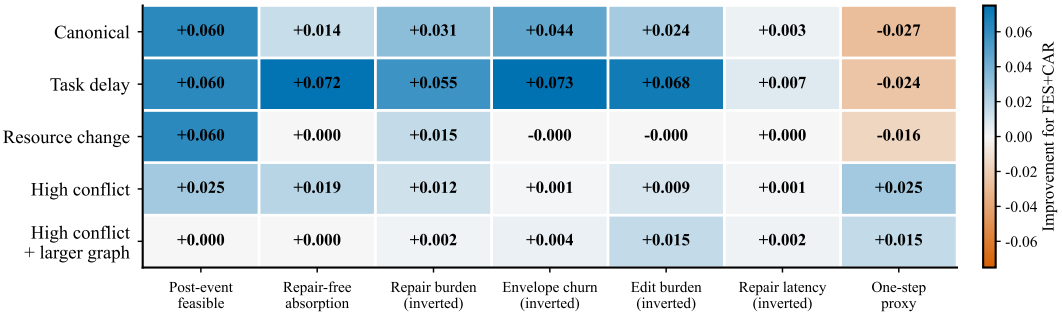


Figure 4: Slice-wise FLECAR advantage over the CAR-like point baseline. Positive cells favor FLECAR; cost metrics are sign-inverted so higher values indicate improvement.

The result is not explained by stronger initial compilation alone. Matched repair gives the point baseline CAR-like repair terms and aligned event streams, yet FLECAR preserves a further gain in post-event feasibility and uses less repair effort. One-step survivability is slightly higher for the point baseline when pooled across settings, while the slice-level pattern in Figure 4 favors FLECAR on feasibility and cost in the higher-conflict cases. Together, these outcomes show that an isolated one-step score and the quality of the evolving execution state capture different aspects of performance. This distinction is visible across event regimes. In a canonical setting, a point schedule can be effective for the next immediate action because its selected start is already optimized for that local decision. As conflicts accumulate, however, the executor benefits from retaining a region of legal choices and the couplings that coordinate them. The slice-level results show that this advantage is strongest where resource pressure or graph complexity makes a single selected start more brittle. The one-step proxy remains informative about a local dispatch decision, while the broader metrics reflect the ability to continue the full episode.

Across pooled settings, the point baseline leads the one-step proxy by 0.005. FLECAR leads on feasibility and cost in the high-conflict slices. In the main setting, it raises post-event feasibility by 0.0600 and reduces repair burden by 0.0322. A separate 800-event matched comparison aligns the solver, budget, event stream, frozen-prefix handling, and repair objective. FLECAR preserves more controllability while making fewer envelope and schedule edits and completing repair faster. Table 3 reports the full comparison, including the near-parity future-feasibility result. The shared repair machinery therefore leaves the carried execution representation as the main source of the observed difference. The external replanning comparison reinforces this interpretation. General scheduling solvers can rebuild feasible suffixes and provide a strong recovery mechanism, but rebuilding begins after the event has exposed which earlier choices were discarded. An envelope lets the executor first test the slack and alternatives already preserved around the affected subtasks, then reserve wider replanning for cases that require it. The resulting profile reflects how much usable structure remains available when repair begins. Table 4 reports the corresponding task outcomes and burden.

Table 3: Matched repair over 800 events. Differences are FLECAR minus point.

Metric	FLECAR	Point	Difference
Post-event feasibility ↑	0.8500	0.7750	+0.0750
Future feasibility ↑	0.1738	0.1700	+0.0038
Repair-free absorption ↑	0.0325	0.0175	+0.0150
Repair burden ↓	0.2929	0.3329	−0.0401
Envelope churn ↓	0.0316	0.0939	−0.0622
Edit burden ↓	0.0888	0.1375	−0.0488
Repair latency (s) ↓	0.0112	0.0152	−0.0040
Controllability preserved ↑	0.4250	0.3625	+0.0625

Table 4: Comparison with library-based replanning.

Method	Post-event feasibility ↑	Terminal success ↑	Repair burden ↓
FLECAR	0.8000	0.8000	0.3032
Matched point repair	0.7400	0.7400	0.3399
NetworkX + PuLP	0.3550	0.5333	0.5039

4.3 Matched Repair Across Task Programs

Table 5 reports absolute results alongside the differences in Table 2. The strict comparison covers 72 tasks, 50 seeds, and four events, with the same initial candidates, event payloads, CP-SAT backend, repair thresholds, objectives, frozen-prefix rule, and per-event deadline. The point method collapses timing windows after each event, while FLECAR retains them. On the common initialization-feasible population, post-event feasibility improves by 0.1027. Its post-event feasibility advantage remains positive across all 50 seeds, and it also appears when initialization failures are counted. This directly supports the role of retained execution choices under matched repair conditions. A separate 50-seed analysis also favors FLECAR on post-event feasibility and repair burden for every seed. Its one-step result is computed on a different sample from the original pooled comparison. The single-event control starts each event from the same initial state. FLECAR improves feasibility across task delays, task failures, and object-state changes, complementing the sequential evaluation by separating event effects from their order.

The VirtualHome evaluation uses 10 programs and five seeds in single-event and sequential modes, yielding 100 paired rollouts, and extends this comparison to simulator state transitions. Selected actions execute in the evolving graph, updated states return to the coordinator, and native goal predicates determine outcomes. FLECAR raises task success and goal completion while reducing unfinished goals and increasing post-event feasibility. These results connect the maintained envelope to successful action execution in an external environment. VirtualHome covers food preparation, storing items, table setting, and dishwasher loading. Both methods use the same initial graph, action interface, event information, and goal evaluator. After each action, the changed graph is passed back to the coordinator before the next dispatch or repair decision. This loop makes the outcome sensitive to accumulated state changes because a choice that remains feasible on paper must still correspond to an executable action in the updated environment. The gain in native task success therefore complements schedule-level feasibility with direct evidence from simulated execution. Among 100 paired rollouts, 20 succeed only under FLECAR, four only under the point baseline, 68 under both, and eight under neither. Task success improves by 0.160 and goal completion by 0.085, while unfinished goals decrease by 0.510 and post-event feasibility improves by 0.118.

The Watch-And-Help evaluation [28] uses 100 official task programs and five seeds with shared event streams, candidate sources, and online budgets. FLECAR improves success and completion against both the CAR-like point baseline and legacy point repair. Against CAR-like point repair, task success improves by 0.139 and goal completion by 0.0714; against legacy repair, the corresponding gains are 0.179 and 0.1444. Unfinished goals also decrease against both baselines. The direction agrees with VirtualHome while testing task structures that were not generated by FLECAR. The two external evaluations probe transfer in different ways. Watch-And-Help varies the task programs presented to the coordinator, while VirtualHome executes selected actions and returns the resulting symbolic state.

Table 5: Absolute results for matched and external evaluations.

Evaluation / baseline	Metric	Point	FLECAR
Strict / CAR-matched	Post-event feasibility ↑	0.3724	0.4751
VirtualHome / CAR-like	Task success ↑	0.720	0.880
	Goal completion ↑	0.842	0.927
	Unfinished goals ↓	0.948	0.438
	Post-event feasibility ↑	0.796	0.914
WAH / CAR-like	Task success ↑	0.437	0.576
	Goal completion ↑	0.7850	0.8564
	Unfinished goals ↓	0.734	0.648
WAH / legacy	Task success ↑	0.397	0.576
	Goal completion ↑	0.7120	0.8564
	Unfinished goals ↓	0.864	0.648

Agreement across them suggests that the measured benefit is tied to preserving executable alternatives through events rather than to one generated task schema or one simulator scoring interface. The matched comparisons also keep the repair mechanism and downstream evaluation aligned, so the result remains interpretable as an effect of the carried execution representation.

5 Discussion and Conclusion

FLECAR shows that the object carried into execution matters. Across the full online contract, replacing point commitments with a dispatchable envelope raises post-event feasibility and terminal success while reducing repair burden. The matched-repair comparison gives the point baseline a much stronger repair policy, yet the envelope still preserves more controllability, changes less structure, and repairs faster from the same event streams. The comparison is therefore not just a stronger-baseline check; it diagnoses what information the executor carries forward after execution has begun. The paired comparisons provide evidence because, when repair strength, events, budgets, and frozen-prefix handling are aligned, the method that keeps windows, couplings, dispatch structure, and alternatives live gives the executor stronger post-disturbance state. The strict matched evaluation makes this distinction concrete. Both methods use the same repair machinery and event payloads, but differ in whether executable windows persist after each event. The program-level and symbolic closed-loop evaluations extend the comparison to externally specified tasks and simulator state transitions. The broader lesson is to treat the execution interface as a maintained state, not a one-time schedule export. Passing an envelope leaves slack to spend, alternatives to select, and coupled constraints to localize repair, allowing the executor to use choices the planner deliberately kept alive. More reliable coordination can improve continuity in shared embodied tasks, while changes to shared resources may affect safety and access in deployment contexts. This view also clarifies how the components work together. FES determines which temporal and assignment choices can remain available, while CAR decides when those choices are sufficient to continue and when they should be revised. Their shared object gives the offline planner and online executor a common representation of progress. That handoff makes the response to uncertainty a continuation of the same coordination process rather than a succession of unrelated replanning steps.

Limitations. FLECAR is evaluated on generated instances, Watch-And-Help programs, and symbolic VirtualHome execution. The exact STNU/CSTNU grounding applies to the fixed temporal core after candidate and resource-order decisions are fixed, while event-driven admissibility mutations and discrete reselection are handled by the online procedure. The current implementation has been tested at moderate coordination scale and online budgets. The external task programs and simulator state transitions extend the evidence beyond generated instances.

Acknowledgments and Disclosure of Funding

This work was supported by the Ningbo Yongjiang Talent Programme (2025A-387-G), the Zhejiang Provincial Natural Science Foundation of China (Grant No. LQN26F020008), the CCF-Kuaishou Large Model Explorer Fund (No. CCF-KuaiShou 2025001), the CAAI-Ant Group Research Fund (2025CAAI-ANT-15), the Science and Technology Innovation Project of Ningbo Global Innovation Center, Zhejiang University (Grant No. S02J2604), and the China Postdoctoral Science Foundation under Grant No. GZB20260393. The authors declare no competing interests.

References

[1] Philippe Baptiste, Philippe Laborie, Claude Le Pape, and Wim Nuijten. Constraint-based scheduling and planning. In *Handbook of Constraint Programming*, Foundations of Artificial Intelligence, pages 761–799. Elsevier, 2006.

[2] Massimo Cairo and Romeo Rizzi. Dynamic controllability of simple temporal networks with uncertainty: Simple rules and fast real-time execution. *Theor. Comput. Sci.*, 797:2–16, 2019.

[3] Massimo Cairo, Carlo Comin, and Romeo Rizzi. Instantaneous reaction-time in dynamic consistency checking of conditional simple temporal networks. *J. Log. Algebraic Methods Program.*, 113:100542, 2020. doi: 10.1016/J.JLAMP.2020.100542. URL `https://doi.org/10.1016/j.jlamp.2020.100542`.

[4] Matthew Chang, Gunjan Chhablani, Alexander Clegg, Mikael Dallaire Cote, Ruta Desai, Michal Hlavac, Vladimir Karashchuk, Jacob Krantz, Roozbeh Mottaghi, Priyam Parashar, Siddharth Patki, Ishita Prasad, Xavier Puig, Akshara Rai, Ram Ramrakhya, Daniel Tran, Joanne Truong, John M. Turner, Eric Undersander, and Tsung-Yen Yang. PARTNR: A benchmark for planning and reasoning in embodied multi-agent tasks. In *ICLR*. OpenReview.net, 2025.

[5] Carlo Comin and Romeo Rizzi. Dynamic consistency of conditional simple temporal networks via mean payoff games: A singly-exponential time dc-checking. In *TIME*, pages 19–28. IEEE Computer Society, 2015.

[6] Carlo Comin, Roberto Posenato, and Romeo Rizzi. Hyper temporal networks - A tractable generalization of simple temporal networks and its relation to mean payoff games. *Constraints An Int. J.*, 22(2):152–190, 2017. doi: 10.1007/s10601-016-9243-0.

[7] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artif. Intell.*, 49(1-3): 61–95, 1991.

[8] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. Robothor: An open simulation-to-real embodied AI platform. In *CVPR*, pages 3161–3171. Computer Vision Foundation / IEEE, 2020.

[9] M. Bernardine Dias, Robert Zlot, Nidhi Kalra, and Anthony Stentz. Market-based multirobot coordination: A survey and analysis. *Proc. IEEE*, 94(7):1257–1270, 2006.

[10] Brian P. Gerkey and Maja J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *Int. J. Robotics Res.*, 23(9):939–954, 2004.

[11] John N. Hooker. *Integrated methods for optimization*, volume 100 of *International series in operations research and management science*. Springer, 2007.

[12] Luke Hunsberger and Roberto Posenato. Simpler and faster algorithm for checking the dynamic consistency of conditional simple temporal networks. In *IJCAI*, pages 1324–1330. ijcai.org, 2018.

[13] Luke Hunsberger and Roberto Posenato. Faster dynamic-consistency checking for conditional simple temporal networks. In *ICAPS*, pages 152–160. AAAI Press, 2020.

[14] Alaa Khamis, Ahmed Hussein, and Ahmed Elmogy. Multi-robot task allocation: A review of the state-of-the-art. In *Cooperative Robots and Sensor Networks 2015*, volume 604 of *Studies in Computational Intelligence*, pages 31–51. Springer, 2015. doi: 10.1007/978-3-319-18299-5_2.

[15] Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: an interactive 3d environment for visual AI. *CoRR*, abs/1712.05474, 2017. URL https://arxiv.org/abs/1712.05474v1.

[16] G. Ayorkor Korsah, Anthony Stentz, and M. Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *Int. J. Robotics Res.*, 32(12):1495–1512, 2013.

[17] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, Mona Anvari, Minjune Hwang, Manasi Sharma, Arman Aydin, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R. Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Silvio Savarese, Hyowon Gweon, C. Karen Liu, Jiajun Wu, and Li Fei-Fei. BEHAVIOR-1K: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation. In *CoRL*, volume 205 of *Proceedings of Machine Learning Research*, pages 80–93. PMLR, 2023.

[18] Tingting Li and Ziming Zhao. Moirai: Optimizing quantum serverless function orchestration via device allocation and circuit deployment. In *IEEE International Conference on Web Services*, pages 707–717, 2024. doi: 10.1109/ICWS62655.2024.00090.

[19] Tingting Li, Ziming Zhao, and Jianwei Yin. Empowering quantum serverless circuit deployment optimization via graph contrastive learning and learning-to-rank co-designed approaches. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pages 9250–9258, 2025. doi: 10.24963/ijcai.2025/1028.

[20] Paul Morris. A structural characterization of temporal dynamic controllability. In *CP*, Lecture Notes in Computer Science, pages 375–389. Springer, 2006.

[21] Paul H. Morris, Nicola Muscettola, and Thierry Vidal. Dynamic control of plans with temporal uncertainty. In Bernhard Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 494–499. Morgan Kaufmann, 2001.

[22] Ernesto Nunes, Marie D. Manner, Hakim Mitiche, and Maria L. Gini. A taxonomy for task allocation problems with temporal and ordering constraints. *Robotics Auton. Syst.*, 90:55–70, 2017.

[23] Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramuthu, Gökhan Tür, and Dilek Hakkani-Tür. Teach: Task-driven embodied agents that chat. In *AAAI*, pages 2017–2025. AAAI Press, 2022.

[24] Lynne E. Parker. ALLIANCE: an architecture for fault tolerant multirobot cooperation. *IEEE Trans. Robotics Autom.*, 14(2):220–240, 1998.

[25] Laurent Perron, Frédéric Didier, and Steven Gay. The CP-SAT-LP solver (invited talk). In *CP*, LIPIcs, pages 3:1–3:2. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi: 10.4230/LIPIcs.CP.2023.3.

[26] Roberto Posenato. CSTNU tool: A java library for checking temporal networks. *SoftwareX*, 17: 100905, 2022.

[27] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *CVPR*, pages 8494–8502. Computer Vision Foundation / IEEE Computer Society, 2018.

[28] Xavier Puig, Tianmin Shu, Shuang Li, Zilin Wang, Yuan-Hong Liao, Joshua B. Tenenbaum, Sanja Fidler, and Antonio Torralba. Watch-And-Help: A challenge for social perception and human-ai collaboration. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.09890.

[29] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied AI research. In *ICCV*, pages 9338–9346. IEEE, 2019.

[30] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *CVPR*, pages 10737–10746. Computer Vision Foundation / IEEE, 2020.

[31] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *ICLR*. OpenReview.net, 2021.

[32] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, C. Karen Liu, Silvio Savarese, Hyowon Gweon, Jiajun Wu, and Li Fei-Fei. BEHAVIOR: benchmark for everyday household activities in virtual, interactive, and ecological environments. In *CoRL*, volume 164 of *Proceedings of Machine Learning Research*, pages 477–490. PMLR, 2022.

[33] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John M. Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel X. Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 251–266, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/021bbc7ee20b71134d53e20206bd6feb-Abstract.html.

[34] Thierry Vidal and Hélène Fargier. Handling contingency in temporal constraint networks: from consistency to controllabilities. *J. Exp. Theor. Artif. Intell.*, 11(1):23–45, 1999.

[35] Fei Xia, William B. Shen, Chengshu Li, Priya Kasimbeg, Micael Tchapmi, Alexander Toshev, Roberto Martín-Martín, and Silvio Savarese. Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics Autom. Lett.*, 5(2):713–720, 2020.

[36] Xiaofei Yue, Song Yang, Liehuang Zhu, Stojan Trajanovski, and Xiaoming Fu. Demeter: Fine-grained function orchestration for geo-distributed serverless analytics. In *IEEE INFOCOM 2024*, pages 2498–2507, 2024. doi: 10.1109/INFOCOM52122.2024.10621303.

[37] Xiaofei Yue, Song Yang, Liehuang Zhu, Stojan Trajanovski, Fan Li, and Xiaoming Fu. Exploiting wide-area resource elasticity with fine-grained orchestration for serverless analytics. *IEEE Transactions on Networking*, 33(1):398–413, 2025. doi: 10.1109/TNET.2024.3486788.

[38] Xiaofei Yue, Song Yang, Fan Li, Liehuang Zhu, Xu Wang, Zhen Feng, and Fernando A. Kuipers. HyFaaS: Accelerating serverless workflows by unleashing hybrid resource elasticity. *IEEE Transactions on Parallel and Distributed Systems*, 37(1):272–286, 2026. doi: 10.1109/TPDS.2025.3632089.

[39] Ziming Zhao, Tingting Li, and Jianwei Yin. QuaMap: A multi-backend benchmark dataset for quantum circuit mapping and learning-based compiler evaluation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2913–2923, 2026. doi: 10.1145/3770854.3785697.

[40] Ziming Zhao, Tingting Li, Xiaofei Yue, Zhaoxuan Li, and Jianwei Yin. Time-aware active calibration and scheduling for n-qubit systems via bayesian optimization and reinforcement learning. In *Proceedings of the 63rd ACM/IEEE Design Automation Conference*, pages 289:1–289:7, 2026. URL https://63dac.conference-program.com/presentation/?id=RESEARCH2805&sess=sess135.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state a scoped execution-layer contribution and distinguish task-facing results, matched-repair evidence, and exact-core STNU grounding.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The discussion and method sections state the exact-core/full-contract boundary, the larger-graph scale boundary, and incomplete-probe handling.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not introduce new theorem/proof claims. Formal support is stated as scoped exact-core CSTNU/STNU checking for fixed temporal objects, with the method section defining the fixed-object scope and the project repository documenting checker coverage.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The project repository gives package contents, command templates, seeds, instance counts, event counts, configured budgets, CAR call-allocation semantics, package/tool versions, exact-core checker settings, and numeric provenance. It includes FLECAR source, baseline utilities, experiment and export scripts, frozen configs, generated instances, frozen exported tables, exact-core output tables, README/runbook instructions, requirements, environment disclosure, license/version notes, and numeric provenance for the main evidence chain.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The FLECAR GitHub repository contains FLECAR source, baseline utilities, experiment and export scripts, configs, generated instances with manifest metadata, frozen exported tables, exact-core tables, runbook, requirements, README, environment disclosure, license/version notes, and numeric provenance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental protocol reports instances, seeds, event rows, event families, comparator roles, metrics, formulas, top-k, density scale, time budget, and event effects.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: The results report paired differences and pairing units for the primary comparisons.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The project repository reports the local operating system, CPU, memory class, GPU status, software versions, suite sizes, configured solver and event budgets, CAR call allocation, and exact-core checker timeout.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The manuscript describes no human-subject, privacy-sensitive, or high-risk released model/data activity.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The discussion notes that more reliable coordination can improve continuity in shared embodied tasks and that changes to shared resources may affect safety and access in deployment contexts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk models, scraped personal datasets, or dual-use assets requiring special safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [No]

Justification: Existing assets are credited, but complete license and terms-of-use details for all assets are not specified in the paper.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The project repository documents generated FLECAR instances, exported result tables, exact-core outputs, scripts, configs, README/runbook instructions, requirements, environment disclosure, license/version notes, and the numeric-provenance manifest. These are reproduction assets rather than a claimed public benchmark release.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not describe crowdsourcing or human-subject experiments.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not describe human-subject experiments requiring IRB or equivalent review.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: No LLM is part of the core method or experimental procedure. Writing/editing assistance is outside the method contribution.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.