ADA: Resolving Attribution Ambiguity in End-to-End Power System Dispatch via Two-Time-Scale Stochastic Approximation

Mengqi Han[1,2,3] **Bo Yang**[1,2,3]* **Qi Liu**[1,2,3] **Mengshuo Jia**[1,2,3]
Chen Gong[1,2,3] **Yuxiang Liu**[1,2,3] **Sicheng Liu**[1,2,3,4] **Mingxuan Cai**[1,2,3]
[1]School of Automation and Intelligent Sensing, State Key Laboratory of Submarine Geoscience, Shanghai Jiao Tong University, Shanghai, China
[2]Key Laboratory of System Control and Information Processing, Ministry of Education of China, Shanghai, China
[3]Shanghai Engineering Research Center of Industrial Intelligent Control and Management, Shanghai, China
[4]Huaneng Clean Energy Research Institute and the National Energy R&D Center for Efficient Utilization of Hydropower & Dam Safety, Beijing 102209, China
{hanmengqi, bo.yang, liuqi.sjtu.iwinc2, m.jia, chen.gong, liu953983860, sichengliu, caimingxuan}@sjtu.edu.cn

Abstract

Large Language Models (LLMs) present a novel paradigm for power system dispatch (PSD), enabling operators to derive optimal dispatch strategies from multi-objective requirements and grid dynamics. However, isolated paradigms focusing on modeling or solving often yield decision bias or physical infeasibility. Meanwhile, end-to-end paradigms couple modeling-solving with reflection-feedback loops, obscuring whether failures stem from inaccurate modeling, invalid solutions, or insufficient feedback, thereby causing ineffective iterations. To address these challenges, we propose Agile Dispatch Agent (ADA), an agent based on two-time-scale evolution that decouples the optimization process into two asynchronous loops. On the fast time scale, the Planner actively acquires information to clarify complex requirements, while the Solver selects algorithms to adapt to the problem mathematical characteristics. On the slow time scale, the Summarizer module updates the modeling feasible region based on evaluations from the Judger. Under a finite-horizon Planner and standard two-time-scale stochastic approximation assumptions, our analysis provides local tracking guarantees for the fast decision process and the slow knowledge evolution. Experiments on L2RPN benchmarks with ambiguous requirements and renewable dynamics show that ADA outperforms state-of-the-art baselines.

1 Introduction

In modern power systems, operators must rapidly respond to stochastic renewable energy injections [1] and the risks associated with cascading contingencies (e.g., N-k faults) [2]. However, Power System Dispatch (PSD) faces conflicting optimization goals, such as economic efficiency versus resiliency. Additionally, evolving system states impose dynamic physical constraints. Therefore, transforming these complex, time-varying needs into rigorous formulations continues to rely on domain expert insight [3]. Leveraging breakthrough advancements in intent understanding, logical reasoning and

*Corresponding author: Bo Yang

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

mathematical solving [4], recent research is deploying Large Language Models (LLMs) to assist the PSD, specifically via automated modeling [5] and heuristic algorithm design [6].

Despite the promising potential of LLMs, existing efforts to apply them to power systems constrained by ambiguity, limited observability, and dynamic conditions remain imperfect. On one hand, modeling-centric studies primarily focus on prompt engineering or retrieval-augmented agents to clarify ambiguous requirements and enhance the accuracy of mathematical formulations [7–9]. However, missing environmental observations or data noise often result in unreliable physical constraints, yielding dispatch problems that are formally complete yet computationally intractable. On the other hand, solver-centric studies typically treat LLMs as black-box optimizers [10] or heuristic strategy generators [11]. While these studies significantly optimize search efficiency for specific tasks, they rely heavily on static assumptions; consequently, if objectives or constraints shift beyond the LLM's capability boundary, the resulting decisions frequently become physically infeasible.

Meanwhile, end-to-end (E2E) frameworks such as LLM4AD [12] and HeurAgenix [13] integrate modeling and solving, utilizing execution feedback or self-reflection as supervisory signals to drive iteration. While these studies show automation potential, they couple the modeling-solving loop of the optimization problem with the reflection-evolution loop of LLMs, which leads to severe *Attribution Ambiguity*: when solutions fail, the system struggles to distinguish between a mismatch in the modeling-solving process and a deficiency in reflective capability. The ambiguity often traps the system in futile search or induces logical hallucinations, lowering evolutionary efficiency [14].

To address these challenges, we propose Agile Dispatch Agent (ADA), an E2E framework based on two-time-scale evolution. ADA models the entire process from scene perception to decision generation as a decoupled evolutionary process of task knowledge and action knowledge. By reconstructing agent modeling boundaries on a slow time scale and adapting unstructured requirements and solution strategies on a fast time scale, we achieve efficient automation of operations research.

The main contributions of this paper are as follows:

- To address the attribution ambiguity of E2E paradigms for PSD, we introduce ADA, which decouples the transition from unstructured requirements to feasible solutions into a two-time-scale evolutionary process. Under finite-horizon Planner and TTSA assumptions, this asynchronous update admits local stabilization and tracking guarantees for the embedded AK/TK knowledge dynamics.
- To address the modeling ambiguity arising from ambiguous requirements and time-varying systems, we propose an Active State Augmentation mechanism. Leveraging Monte Carlo Tree Search, this mechanism precisely maps ambiguous observations to rigorous mathematical formulations, effectively ensuring modeling accuracy and physical feasibility.
- To address the inductive bias of single algorithms, we design a dynamic matching strategy based on dispatch features. This approach automatically retrieves and adapts the optimal solver according to the mathematical characteristics of the dispatch problem, significantly enhancing the robustness of the system when handling dynamic scenarios.

2 Motivation

In this section, we use L2RPN (real-world French grid data) [15] to expose bottlenecks in E2E paradigms. IJCNN-2019 tests fuzzy requirements, requiring trade-offs among cost, stability, and utility without explicit formulations, while WCCI-2020 tests implicit constraints under time-varying renewables and storage, requiring agents to infer dynamic feasibility boundaries.

Modeling Ambiguity Optimization modeling in complex systems is inherently ambiguous. On one hand, fuzzy requirements lack precise mathematical definitions, introducing vagueness into optimization objectives. On the other hand, the time-varying nature of physical parameters, compounded by limited observability, results in highly dynamic constraint boundaries. Hence, existing modeling paradigms that rely on fixed optimization templates or time-invariant assumptions generate significant distorted modeling, leading to intractable or physically infeasible solutions. As shown in Fig. 1, mainstream agents on WCCI-2020 suffer a sharp drop in constraint formulation accuracy from 95.2% under static conditions to 70.3% under renewable-driven dynamics.

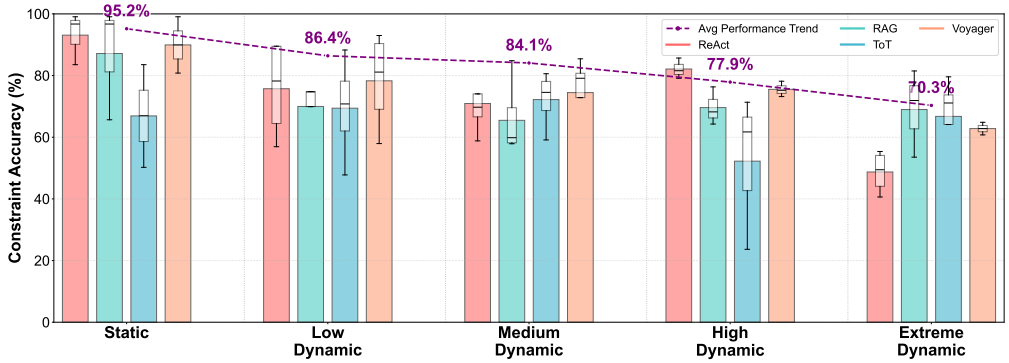


Figure 1: Constraint Formulation Accuracy under Varying Scene Dynamicity.

Algorithm Inductive Bias Dispatch problems often exhibit mathematical characteristics that evolve dynamically with the grid state. The inherent preferences of specific solvers cannot universally cover these shifting solution space. Consequently, relying on a fixed solver can cause performance collapse or physical infeasibility. We validate this by evaluating six algorithm families across tasks with different characteristics, including **Convexity** (e.g., CNP1), **Discreteness** (e.g., KNAPSACK), **Multiple solutions** (e.g., BRANIN), and **Numerical Instability** (ILL_COND). We measure **Solution Accuracy** as $\max(0, 1 - \frac{|f(x)-f^*|}{|f^*|+\epsilon}) \times 100\%$. As shown in Fig. 2, gradient-based, MILP, and Bayesian methods perform well in their preferred regimes but fail to generalize across changing problem characteristics. This motivates the Adaptive Algorithm Selection mechanism in Section 3.2.

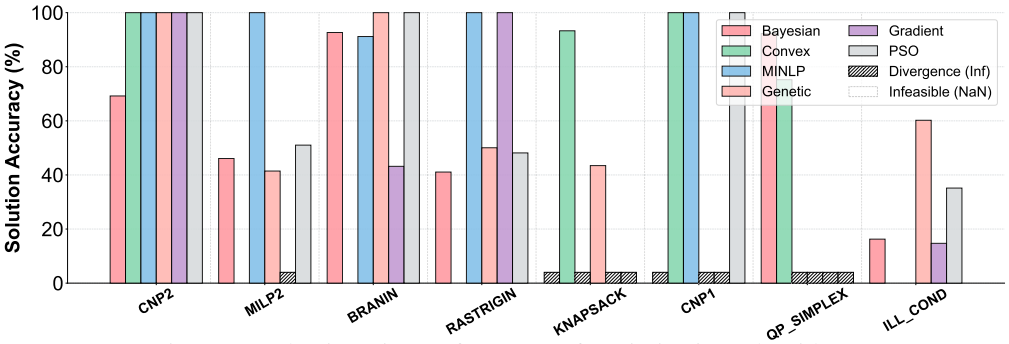


Figure 2: Inductive Bias Performance of Optimization Algorithms.

E2E Attribution Ambiguity In existing E2E paradigms, reflection and iterative correction couple the modeling-solving process with the LLM's cognitive updates within a single feedback loop. When the system encounters infeasible solutions or suboptimal performance, this coupling obscures the specific cause of failure. The lack of a clear update direction triggers ineffective random search or logical hallucinations, ultimately causing system evolution to stagnate. We analyzed the trajectory of the grid entering an Emergency Phase in the WCCI-2020, as shown in Fig. 3. Although Refinement-MILP adjusts parameters to reduce grid load, the lack of explicit update guidance causes severe search latency and wastes the critical response window. These findings demonstrate the urgent need for an efficient evolutionary architecture capable of achieving convergent system evolution.

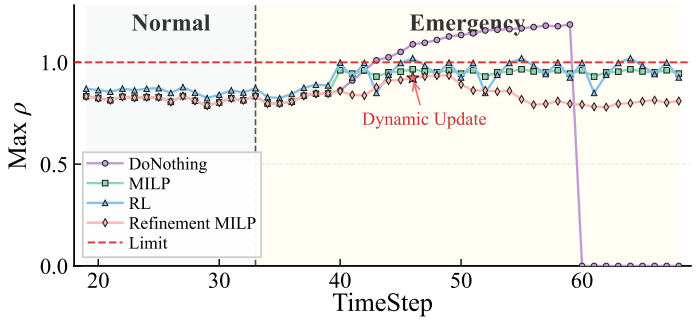


Figure 3: Impact of knowledge evolution on search efficiency in WCCI-2020 Scenario 182.

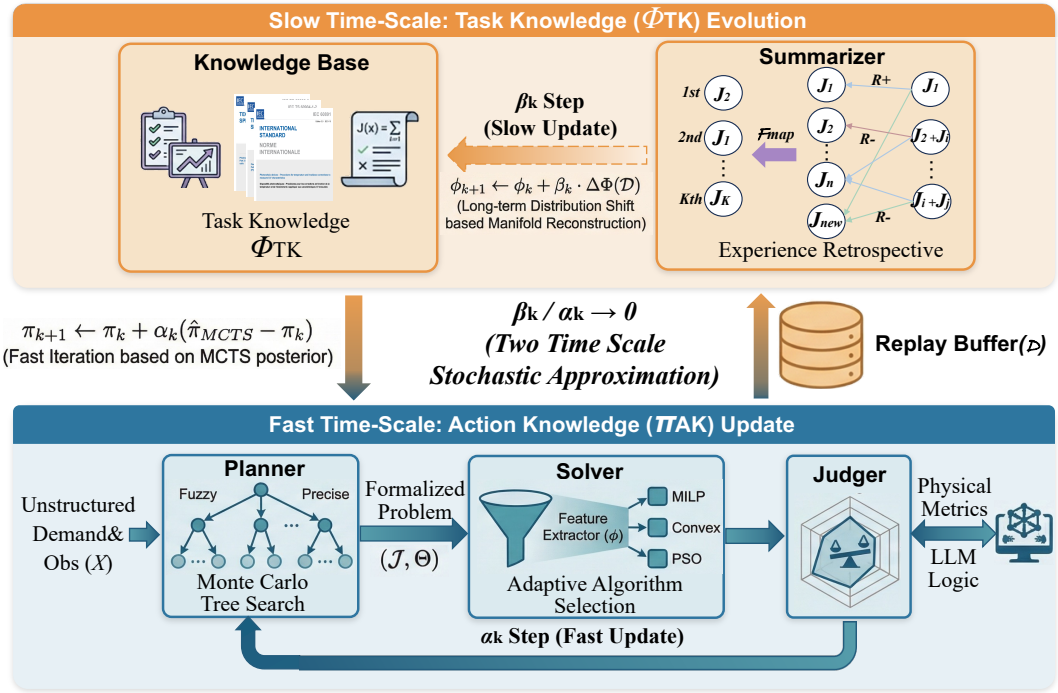


Figure 4: **The Agile Dispatch Agent framework architecture.** The system consists of four core modules (Planner, Solver, Judger, and Summarizer) that work together to transform unstructured scenarios into formal optimization problems and generate executable solutions. The framework is driven by Task Knowledge ($\mathcal{TK}$) and Action Knowledge ($\mathcal{AK}$), enabling knowledge-driven dynamic evolution for adaptive decision-making in complex physical systems.

3 Methods

As shown in Fig. 4 and Alg. 1, ADA addresses attribution ambiguity through explicit module interfaces and separated knowledge updates. The Planner uses Action Knowledge ($\mathcal{AK}$) to acquire missing state context and Task Knowledge ($\mathcal{TK}$) to construct the dispatch formulation $\langle \mathcal{J}, \Theta \rangle$. The Solver matches its problem features to algorithm capabilities and records the selected solver and execution status. The Judger retains physical and logical feedback separately before aggregation. These interfaces provide distinct diagnostic signals for context or formulation errors, solver-selection or execution failures, and Judge–physics conflicts. The Summarizer uses verified trajectories to revise $\mathcal{AK}$ tool-chain priors and reconstruct $\mathcal{TK}$ objectives, constraints, and solver references on separate time scales, supporting module-level diagnosis and targeted knowledge correction.

3.1 Planner

The core function of the Planner is to resolve the ambiguity identified in Sec. 2. We model the entire process from seeking the optimal toolchains $\mathcal{H}^*$ to enhance observations $\mathcal{X}$, to deducing the optimal formal mathematical dispatch problem $\langle \mathcal{J}, \Theta \rangle$ as a Belief Tree Search process.

First, we define the augmented state space $\mathcal{S} = \{\mathcal{X}, \mathcal{H}\}$, where $\mathcal{H} = (t_0, t_1, \ldots, t_n)$ represents the history of tool invocations selected from the toolset $\mathcal{T}$ ($t_i \in \mathcal{T}$). $\mathcal{TK}$ characterizes the form of dispatch problems for a specific scenario, defined as $\Phi_{\mathcal{TK}} = \langle \Lambda_J, \Lambda_\Theta, \mathcal{F}_{map} \rangle$. Here, Λ_J and Λ_Θ define the parameterized families of objective functions and physical constraints, respectively, while the mapping operator $\mathcal{F}_{map} : \mathcal{S} \to (\mathbf{p}, \mathbf{q}, \mathcal{I}_{active})$ extracts specific physical parameters from the augmented state and determines the subset of active constraints.

Initialization: Action Knowledge ($\mathcal{AK}$) utilizes historical states $\mathcal{S}_0$ to initialize tree nodes and the initial value estimate $Q_0(t)$ for action t:

$$N_0(\mathcal{X}, t) = \sum_{s \in \mathcal{S}_0} \sum_{(\mathcal{X}', t) \in s} \kappa_{\mathcal{AK}}(\mathcal{X}, \mathcal{X}'), \qquad Q_0(\mathcal{X}, t) = \frac{1}{N_0(\mathcal{X}, t)} \sum_{s \in \mathcal{S}_0} \sum_{(\mathcal{X}', t) \in s} \kappa_{\mathcal{AK}}(\mathcal{X}, \mathcal{X}') \cdot R(s), \tag{1}$$

where $\kappa_{\mathcal{AK}}(\mathcal{X}, \mathcal{X}')$ is the similarity weight between the current state and historical state under $\mathcal{AK}$, and $R(s)$ denotes the realized reward of trajectory s.

Selection: The Planner selects the next tool t^* using a UCB criterion with the $\mathcal{AK}$ prior:

$$t^* = \arg\max_{t \in \mathcal{T}} \left[Q(\mathcal{X}, t) + \varepsilon \cdot P(t|\mathcal{X}) \frac{\sqrt{\sum_{t'} N(\mathcal{X}, t')}}{1 + N(\mathcal{X}, t)} \right], \tag{2}$$

where $N(\mathcal{X}, t)$ counts visits, $P(t|\mathcal{X})$ denotes the historical tool-use prior, and ε controls exploration.

Expansion: The selected tool t^* is executed to update the search state from $s_d = (\mathcal{X}_d, \mathcal{H}_d)$ to $s_{d+1} = (\mathcal{X}_{d+1}, \mathcal{H}_d \cup \{t^*\})$. This process repeats until the termination condition is met or the maximum depth is reached.

Generation: Finally, leveraging the policy knowledge $\mathcal{TK}$, the Planner rigorously translates the unstructured environmental semantics $\mathcal{X}^*$ into a mathematical model $\langle \mathcal{J}, \Theta \rangle$ processing by the solver:

$$\mathcal{J}(\mathbf{u}) = J(\mathbf{u}; \hat{\mathbf{p}}), \qquad \Theta = \{ g_m(\mathbf{u}; \hat{\mathbf{q}}) \leq 0 \mid m \in \hat{\mathcal{I}} \}, \qquad (\hat{\mathbf{p}}, \hat{\mathbf{q}}, \hat{\mathcal{I}}) = \mathcal{F}_{map}(\mathcal{X}^*). \tag{3}$$

Backpropagation: On the fast time scale, the generated dispatch problem is evaluated by the Solver and Judger, producing the reward signal R defined in Eq. 8 for toolchain updates. This reward serves as the backup value for the selected toolchain and is propagated to all nodes along the path:

$$N_{i+1}(\mathcal{X}, t) = \sum_{s \in \mathcal{S}_i \oplus \delta s} \sum_{(\mathcal{X}', t) \in s} \kappa_{\mathcal{AK}}(\mathcal{X}, \mathcal{X}'), \qquad Q_{i+1}(\mathcal{X}, t) = (1 - \lambda) Q_i(\mathcal{X}, t) + \lambda R(s), \tag{4}$$

where δs denotes the newly expanded trajectory segment and λ is the update factor.

3.2 Solver

Different optimization problems exhibit varying sensitivities to solution algorithms (Section 2), so we propose a Problem–Algorithm Alignment mechanism. We define the problem feature space $\mathcal{G}$ as an n-dimensional metric space of quantifiable attributes, where each dimension g_i corresponds to a property such as non-convexity, constraint stiffness, or scale. The projection ϕ maps the dispatch tuple $\langle \mathcal{J}, \Theta \rangle$ into this space as the retrieval query, while each algorithm $A \in \mathcal{A}$ maintains a capability vector $\psi(A)$ encoding its inherent preferences:

$$\mathcal{G} = \left\{ \mathbf{g} \in \mathbb{R}^n_{\geq 0} \mid \mathbf{g} = [g_1, \ldots, g_n]^\top \right\}, \qquad \phi(\langle \mathcal{J}, \Theta \rangle), \psi(A) \in \mathcal{G}, \tag{5}$$

where $\phi(\langle \mathcal{J}, \Theta \rangle)$ are provided in Appendix C.2 and $\psi(A)$ is initialized with the fixed values in Table 6.

We define an alignment function $G(\psi, \phi)$ to quantify the fitness between the algorithm and the task:

$$G(\psi, \phi) = \sum_{i=1}^{n} \phi_i w_i \frac{\psi_i(A)(k_1 + 1)}{\psi_i(A) + K(A)}, \qquad w_i = \ln \left(1 + \frac{\sum_{A' \in \mathcal{A}} \sum_{j=1}^{n} \psi_j(A')}{\sum_{A \in \mathcal{A}} \psi_i(A) + \epsilon} \right), \tag{6}$$

where $K(A) = k_1(1 - b + b\frac{L_A}{L_{avg}})$ with standard parameters ($k_1 = 1.2, b = 0.75$). $L_A = \sum \psi_i(A)$.

Specifically, this mechanism aligns the solver through three distinct levers: the feature intensity ϕ_i and global weight w_i jointly anchor the retrieval to the problem's dominant and rare characteristics, while the regularization operator $K(A)$ penalizes excessive versatility (L_A). This design effectively filters out "jack-of-all-trades" candidates, ensuring the selection prioritizes specialized solvers strictly customized to the current optimization landscape. Ultimately, the optimal algorithm A^* is retrieved by maximizing the alignment score:

$$A^* = \arg\max_{A \in \mathcal{A}} G(A, \langle \mathcal{J}, \Theta \rangle). \tag{7}$$

Feedback updates $\mathcal{TK}$ and thereby changes future formulations and alignment scores, while $\psi(A)$ and G remain fixed (Appendix C.1).

3.3 Judger

ADA separates synchronous action verification from asynchronous Judger feedback. Before reward scoring, a deterministic physics gate verifies each candidate dispatch action against the current simulator state and grid operating constraints. Infeasible or unverifiable actions are rejected and trigger the retry mechanism described in Sec. 4.3.

For feasible actions, relying only on short-term grid improvement can be insufficient, since an action may reduce immediate loading while violating the formalized dispatch priority or increasing delayed operational risk. The Judger therefore computes a hybrid reward that combines physical dispatch quality with semantic decision consistency:

$$R = (1-\alpha)R_{\mathrm{PHY}} + \alpha R_{\mathrm{LLM}}, \tag{8}$$

where R_{PHY} captures measurable grid effects such as stability and cost, while R_{LLM} evaluates formulation–action consistency, scenario-priority alignment, and delayed risk. When the two rewards assign conflicting preferences to feasible candidates, α mediates this mismatch in the feedback signal rather than changing action admissibility. The resulting reward links dispatch actions to scenario conditions, providing the Summarizer with a direction signal for knowledge evolution.

3.4 Summarizer

The Summarizer transforms evaluated dispatch trajectories into reusable knowledge for later planning and formulation. $\mathcal{AK}$ is updated from MCTS posterior statistics to refine tool selection and suppress invalid tool chains, whereas $\mathcal{TK}$ is reconstructed from historical traces as decision priors over dispatch objectives, active constraints, and solver preferences. Because tool-level and formulation-level evidence become reliable at different rates, mismatched updates may introduce stale or conflicting records. Since they are symbolic memories, we therefore model their coupled evolution with Two-Time-Scale Stochastic Approximation (TTSA) and Robbins–Monro form [16]:

$$\pi_{\mathcal{AK}}^{k+1} = \pi_{\mathcal{AK}}^{k} + \alpha_k\big(\hat{\pi}_{\mathrm{MCTS}}(\mathcal{E}_k) - \pi_{\mathcal{AK}}^{k}\big), \qquad \phi_{\mathcal{TK}}^{k+1} = \phi_{\mathcal{TK}}^{k} + \beta_k\,\Delta\Phi\big(\mathcal{J}, \Theta, \mathcal{D}_{\mathrm{buffer}}\big). \tag{9}$$

Here, $\hat{\pi}_{\mathrm{MCTS}}$ is the empirical tool-level posterior induced by the Planner, and $\Delta\Phi$ is the replay-buffer reconstruction operator for $\mathcal{TK}$. The step sizes α_k and β_k control the fast AK update and the slow TK update, respectively.

We impose the standard time-scale separation condition [17], under which $\phi_{\mathcal{TK}}$ is quasi-static on the fast scale and $\pi_{\mathcal{AK}}$ is nearly equilibrated on the slow scale:

$$\sum_k \alpha_k = \infty, \qquad \sum_k \beta_k = \infty, \qquad \lim_{k\to\infty} \frac{\beta_k}{\alpha_k} = 0. \tag{10}$$

We instantiate this separation with polynomial step sizes $\alpha_k = \alpha_0/(k+1)^a$ and $\beta_k = \beta_0/(k+1)^b$, where $1/2 < a < b \le 1$ and $2b - a > 1$, so that $\beta_k/\alpha_k \to 0$ and $\sum_k \beta_k^2/\alpha_k < \infty$. This matches evidence availability: $\mathcal{AK}$ updates after finite-horizon MCTS searches, while $\mathcal{TK}$ waits for multiple trajectories to support reliable formulation updates. Full proofs are provided in Appendix I.

Assumption 3.1 (Regularity and boundedness)**.** The reconstruction operator $\Delta\Phi$ is Lipschitz continuous under the TK embedding and the AK policy metric defined in Appendix F.4. The iterates of $\pi_{\mathcal{AK}}$ and $\phi_{\mathcal{TK}}$ remain in compact sets.

Assumption 3.2 (Finite-horizon Planner stability)**.** For a fixed TK state Φ, the Planner induces a finite-horizon, finite-action, bounded-reward improvement operator T_Φ over the AK policy space. We assume that, in a neighborhood of its local fixed point $\pi^*(\Phi)$, this idealized operator satisfies a contraction-style stability condition.

Proposition 3.3 (Fast-scale tracking under fixed TK)**.** *Under Assumptions 3.1–3.2, if $\beta_k/\alpha_k \to 0$, then on the fast time scale $\phi_{\mathcal{TK}}$ is quasi-static. The AK update tracks the local fixed point of the Planner-induced improvement operator:*

$$\lim_{k\to\infty} \big\|\pi_{\mathcal{AK}}^{k} - \pi^*(\phi_{\mathcal{TK}}^{k})\big\| = 0 \quad \textit{a.s.}. \tag{11}$$

Proof sketch. Fix $\phi_{\mathcal{TK}}^{k} = \Phi$. Then the AK update in Eq. (9) can be written as a stochastic approximation to the Planner-induced update:

$$\hat{\pi}_{\mathrm{MCTS}}(\mathcal{E}_k) = T_\Phi(\pi_k) + \xi_{k+1}, \qquad \pi_{k+1} = \pi_k + \alpha_k\big(T_\Phi(\pi_k) - \pi_k + \xi_{k+1}\big). \tag{12}$$

By Assumption 3.2, T_Φ is locally stable around $\pi^*(\Phi)$. Using the Lyapunov function $L(\pi) = \frac{1}{2}\|\pi - \pi^*(\Phi)\|^2$, the expected drift satisfies

$$\mathbb{E}\big[L(\pi_{k+1}) - L(\pi_k) \mid \mathcal{F}_k\big] \le -c\alpha_k\|\pi_k - \pi^*(\Phi)\|^2 + \mathcal{O}(\alpha_k^2), \tag{13}$$

for some $c > 0$. Together with $\sum_k \alpha_k = \infty$, $\sum_k \alpha_k^2 < \infty$, and square-integrable martingale noise, the Robbins–Siegmund convergence theorem gives almost-sure convergence to the local fixed point. For the slowly moving target $\pi^*(\phi^k_{\mathcal{TK}})$, the additional perturbation is controlled by $\sum_k \beta_k^2/\alpha_k < \infty$, so the same tracking argument remains valid.

Proposition 3.4 (Slow-scale mean-field tracking)**.** *On the slow time scale, the AK variable is viewed as equilibrated at $\pi^*(\phi_{\mathcal{TK}})$. Under Assumption 3.1, the TK trajectory asymptotically tracks the mean-field ODE:*

$$\dot{\phi}(t) = \mathbb{E}\big[\Delta\Phi\big(\mathcal{J}, \Theta, \mathcal{D} \mid \pi^*(\phi(t))\big)\big]. \tag{14}$$

Proof sketch. The slow update in Eq. (9) can be decomposed as

$$\phi_{k+1} = \phi_k + \beta_k\big(\underbrace{\mathbb{E}[\Delta\Phi \mid \pi^*(\phi_k)]}_{\text{mean field}} + \underbrace{\Delta\Phi_k - \mathbb{E}[\Delta\Phi_k \mid \mathcal{F}_k]}_{\text{martingale noise}} + \underbrace{\mathbb{E}[\Delta\Phi \mid \pi_k] - \mathbb{E}[\Delta\Phi \mid \pi^*(\phi_k)]}_{b_k}\big). \tag{15}$$

Here, the martingale term vanishes because $\sum_k \beta_k^2 < \infty$ and the noise is square-integrable. The transient bias vanishes only under time-scale separation. By Proposition 3.3, $\|\pi_k - \pi^*(\phi_k)\| \to 0$ on the fast scale, and the Lipschitz continuity of the mean-field reconstruction map gives

$$\|b_k\| \le L_\pi\|\pi_k - \pi^*(\phi_k)\|. \tag{16}$$

Thus $b_k \to 0$ when $\beta_k/\alpha_k \to 0$, because $\mathcal{AK}$ reaches its local response before $\mathcal{TK}$ makes a significant update. If $\mathcal{TK}$ were updated on the same or a faster time scale, this transient bias would not necessarily vanish; the slow recursion would then follow a biased drift.

4 Experiments

4.1 Benchmark

We evaluate ADA on four L2RPN [15, 18](details in Appendix A.4). **ijcnn-2019** tests fuzzy redispatch on an IEEE14 grid, where safety, cost, and intervention scope must be balanced. **wcci-2020** introduces planned maintenance and renewable uncertainty on an IEEE36 grid, requiring inference of time-varying feasibility boundaries. **neurips-2020** scales to an IEEE118 grid with extreme disturbances and adversarial line disconnections, testing robustness beyond myopic overload relief. **wcci-2022** further adds storage coordination, requiring multi-step decisions with delayed grid effects.

4.2 Baseline Methods

We compare ADA with three groups of state-of-the-art baselines. **Static rules and optimization** include the passive *Do-Nothing* policy, the greedy optimization approach *GreedyOpt* [19], and the optimization-based *MILPAgent* [20]. **Reinforcement Learning (RL)** include *SAC* [21], the SOTA solution for IJCNN-2019, and *HRL* [22]. **LLM-based Agents** include *ToT* [23], *ReAct* [24], and the *ORLM* [3], an LLM fine-tuned for operations research domains. For fairness, ToT, ReAct, ORLM, and ADA use identical prompt templates, inference settings, power-flow calculators, timeout limits, and retry policies, and can select from the same six solver families through a shared interface.

4.3 Experimental Settings

All methods use the same simulator backend, action space, and hardware. Each decision must finish within the 5min dispatch interval; invalid, missing, or unverifiable actions are uniformly replaced by the *Do-Nothing* fallback used for benchmark evaluation. This convention is not a claim that *Do-Nothing* is universally safe in a live grid. We repeat all experiments over five random seeds and report mean $\pm$ standard deviation. Runtime is measured under identical settings, with detailed latency and resource accounting in Appendix G.

We report completion rate (Rate), average survival-step reward (AvgR), and grid overload rate (Over). Let M be the number of evaluated episodes, T_i the survived steps of episode i, H_i its full horizon, $r_{i,t}$ the reward (Appendix D.1.1), and $\bar{\rho}_{i,t} = \max_\ell \rho_{\ell,i,t}$ the maximum line loading:

$$\text{Rate} = \frac{1}{M}\sum_i \mathbb{I}[T_i = H_i], \qquad \text{AvgR} = \frac{1}{M}\sum_i \frac{1}{T_i}\sum_t r_{i,t}, \qquad \text{Over} = \frac{\sum_i \sum_t \mathbb{I}[\bar{\rho}_{i,t} > 1]}{\sum_i T_i}. \tag{17}$$

For LLM-based methods, we use Qwen3-32B [25] with temperature 0.7, identical inference settings, and the same 10-tool set of scene summarization, power-flow calculators, and heuristic rules (Appendix B.1). For ADA, the Planner runs depth-6 finite-horizon MCTS over this set; the Solver chooses among six algorithm families and calls CVXPY for generated mathematical programs (Appendix C.1). The Judger combines physical rewards and an LLM-as-a-Judge signal to evaluate consistency and reliability, with $\alpha = 0.5$ in Eq. (8); setup and calibration are in Appendix D.1.

4.4 Main Results

Overall Performance. Table 1 confirms ADA's superiority across L2RPN benchmarks, achieving near-100% completion even in high-dynamic scenarios (NeurIPS-2020, WCCI-2022). By leveraging the **Planner** to map unstructured states into rigorous mathematical constraints within the physical feasible region, the system minimizes overload rates to 0.10%, significantly outperforming static optimization and RL baselines. Moreover, while standard LLM agents (ReAct, ToT) falter in long-horizon tasks (achieving $< 8.1\%$ completion in WCCI-2022), ADA maintains strategic consistency via Action Knowledge ($\mathcal{AK}$) accumulated by the **Summarizer**.

Table 1: **Main Results on L2RPN Benchmarks.** We report 3 metrics: **Rate, ↑**, **Avg. R, ↑**, and **Overload, ↓**. Best results are in **bold**, second-best are underlined.

Method	ijcnn-2019			wcci-2020			neurips-2020			wcci-2022		
	Rate ↑	Avg R. ↑	Over ↓	Rate ↑	Avg R. ↑	Over ↓	Rate ↑	Avg R. ↑	Over ↓	Rate ↑	Avg R. ↑	Over ↓
Do-Nothing	13.3%	14.32 ± 2.30	14.50%	58.5%	16.24 ± 0.12	0.27%	80.2%	15.21 ± 0.01	0.05%	7.5%	21.49 ± 4.00	17.62%
GreedyOpt	**100.0%**	$\mathbf{15.04 \pm 0.04}$	0.36%	62.0%	16.28 ± 0.08	0.30%	**100.0%**	15.19 ± 0.02	0.05%	14.5%	22.10 ± 0.52	7.30%
MILPAgent	28.7%	14.28 ± 2.31	14.46%	54.6%	16.32 ± 0.13	0.28%	**100.0%**	15.22 ± 0.01	**0.00%**	46.2%	18.80 ± 8.63	1.39%
SAC	26.8%	14.02 ± 2.18	17.89%	24.9%	14.93 ± 2.67	0.16%	90.8%	15.03 ± 0.29	2.55%	36.8%	21.29 ± 3.88	16.57%
HRL	13.3%	14.31 ± 2.30	14.56%	54.8%	$\mathbf{16.36 \pm 0.16}$	0.48%	**100.0%**	15.22 ± 0.01	**0.00%**	26.8%	$\mathbf{22.88 \pm 0.57}$	1.57%
ToT	13.3%	14.25 ± 2.28	14.50%	21.0%	16.25 ± 0.32	0.45%	85.8%	15.19 ± 0.05	0.12%	8.1%	21.72 ± 2.86	17.74%
ReAct	51.5%	14.78 ± 0.26	0.16%	35.6%	16.26 ± 0.25	0.95%	89.5%	15.21 ± 0.04	0.09%	7.9%	21.35 ± 3.89	18.07%
ORLM	25.9%	14.30 ± 2.30	14.45%	64.7%	16.32 ± 0.10	0.30%	**100.0%**	15.22 ± 0.01	**0.00%**	15.0%	19.16 ± 8.67	1.97%
ADA	97.3%	14.96 ± 0.23	**0.03%**	**100.0%**	16.35 ± 0.07	**0.04%**	**100.0%**	$\mathbf{15.25 \pm 0.01}$	**0.00%**	**95.0%**	22.20 ± 0.15	**0.10%**

Decision Efficiency. We quantify action cost $\mathcal{I}$ and effective safety capacity $\mathcal{S}_{\text{eff}}$ as:

$$\mathcal{I} = \frac{1}{T}\sum_{t=1}^{T}\left(2 \cdot \mathbf{a}_t^{\text{redisp}} + 10 \cdot \mathbb{I}(\mathbf{a}_t^{\text{topo}})\right), \quad \mathcal{S}_{\text{eff}} = \frac{1}{T}\sum_{t=1}^{T}\left(1 - \max_{l \in \mathcal{L}} \rho_{t,l}\right), \tag{18}$$

where $\mathbf{a}_t^{\text{redisp}}$ and $\mathbf{a}_t^{\text{topo}}$ are the redispatch and topology action costs, respectively. ADA establishes a Pareto frontier (Fig. 5), maximizing safety at minimal cost through a selective intervention mechanism $G(\psi, \phi)$ driven by asynchronous knowledge updates. By pruning topologically redundant actions and penalizing expensive redispatching when simpler switching suffices, the agent reduces intervention intensity by an order of magnitude ($\mathcal{I} : 1.5 \to 0.2$) while preserving safety margins.

4.5 Ablation Studies

Knowledge Evolution Mechanism. Table 2 tests whether ADA's gains come from knowledge evolution, TTSA, or extra compute. *Static ADA*, which uses fixed priors, drops to 65.1% completion with 0.34% overload on WCCI-2022, showing that static knowledge fails under grid changes. *Static-ADA-High* further biases the same fixed priors toward higher-cost tools, but still does not improve over *Static ADA*, ruling out extra compute as the main source of gains. Updating only AK every 10 steps or only TK every 100 steps improves performance but remains below full ADA, indicating that fast tool-policy adaptation and slow formulation-memory evolution address complementary failures. Moreover, *Syn-ADA-Slow/Fast*, which update AK and TK synchronously under every 10/100 steps, still trail ADA on WCCI-2022 (88.9%/86.7% vs. 95.0%), showing that update frequency alone is insufficient. Thus, ADA performs best because it decouples short-term action adaptation from slower

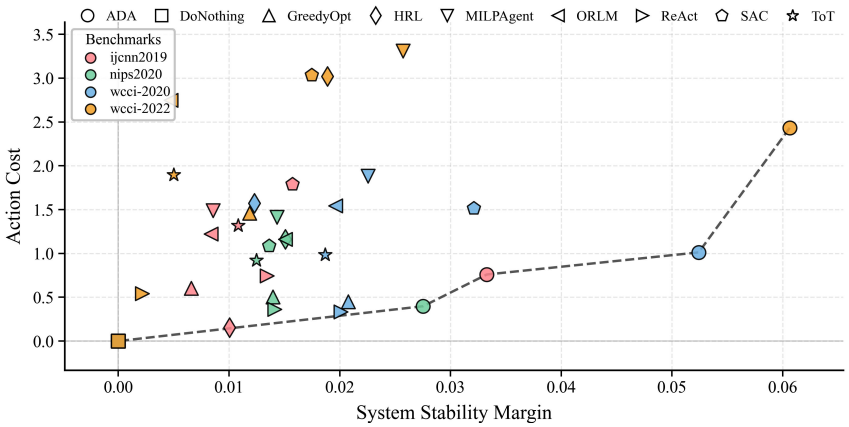


Figure 5: **Pareto Frontier.** Axes show per-benchmark relative safety and cost coordinates normalized against the Do-Nothing reference; absolute $\mathcal{S}_{\text{eff}}$ and $\mathcal{I}$ values are reported in Appendix G.

feasible-region reconstruction, allowing the Planner to react quickly while the Summarizer maintains stable task priors. Appendix E.2 further analyzes the knowledge accumulation effect.

Table 2: **Ablation Study on Knowledge Evolution and Time-Scale Controls.** We report 3 metrics: **Rate,** ↑, **Avg. R,** ↑, and **Overload,** ↓. Best results are in **bold**, second-best are underlined.

Method	**ijcnn-2019**			**wcci-2020**			**neurips-2020**			**wcci-2022**		
	Rate ↑	Avg R. ↑	Over ↓	Rate ↑	Avg R. ↑	Over ↓	Rate ↑	Avg R. ↑	Over ↓	Rate ↑	Avg R. ↑	Over ↓
Static ADA	70.9%	14.66 ± 0.35	0.11%	76.3%	16.16 ± 0.06	0.07%	**100.0%**	15.21 ± 0.01	**0.00%**	65.1%	22.01 ± 0.21	0.34%
AK Update Only	91.0%	14.87 ± 0.20	**0.03%**	89.6%	16.18 ± 0.09	0.05%	**100.0%**	15.21 ± 0.01	**0.00%**	84.2%	22.15 ± 0.20	0.28%
TK Update Only	81.2%	14.71 ± 0.20	0.05%	90.9%	16.22 ± 0.07	0.05%	**100.0%**	15.20 ± 0.01	**0.00%**	90.8%	22.12 ± 0.10	0.19%
Static-ADA-High	71.4%	14.65 ± 0.32	0.12%	76.9%	16.15 ± 0.03	0.07%	**100.0%**	15.20 ± 0.01	**0.00%**	64.7%	22.01 ± 0.25	0.37%
Syn-ADA-Slow	89.4%	14.84 ± 0.18	0.04%	92.1%	16.20 ± 0.08	0.05%	**100.0%**	15.22 ± 0.01	**0.00%**	88.9%	22.08 ± 0.16	0.23%
Syn-ADA-Fast	88.7%	14.85 ± 0.19	0.04%	90.8%	16.21 ± 0.07	0.05%	**100.0%**	15.21 ± 0.01	**0.00%**	86.7%	22.13 ± 0.14	0.24%
ADA	**97.3%**	14.96 ± 0.23	**0.03%**	**100.0%**	16.35 ± 0.07	**0.04%**	**100.0%**	15.25 ± 0.01	**0.00%**	**95.0%**	22.20 ± 0.15	**0.10%**

Role of the Judger. The Judger ablation studies the effects of physical rewards, LLM-Judger rewards, and their hybrid on PSD, especially feedback quality and future operating risk. We report disagreement rate, completion rate, and future maximum line loading. The disagreement rate measures preference directions between the LLM and physical reward for the action, defined as:

$$D = \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}[\Delta R_{\text{LLM}} \cdot \Delta R_{\text{PHY}} < 0] \,. \tag{19}$$

Table 3 shows that the LLM Judge provides a distinct but complementary signal to the physical reward. Under the calibrated hybrid setting ($\alpha = 0.5$), the disagreement rate remains 29.52%–36.14% across benchmarks, indicating that the two rewards assess different aspects of dispatch quality because actions with similar short-term loading reductions can still differ in formulation–action consistency, scenario priority, or delayed operational risk. Meanwhile, the coefficient sweep shows that either single signal is insufficient: on WCCI-2022, both physical-only and LLM-only setting increase line loading 40% over the hybrid setting and reduce completion rate.

Table 3: **Ablation Study on Judger.** We report D, Rate and Future 10 Max Line Loading (ρ_{10}). Best results are in **bold**, second-best are underlined. Additional curves are provided in Appendix H.3.

Judger α	**ijcnn-2019**			**wcci-2020**			**neurips-2020**			**wcci-2022**		
	$D \downarrow$	$\rho_{10} \downarrow$	Rate ↑	$D \downarrow$	$\rho_{10} \downarrow$	Rate ↑	$D \downarrow$	$\rho_{10} \downarrow$	Rate ↑	$D \downarrow$	$\rho_{10} \downarrow$	Rate ↑
$\alpha = 0.00$	70.12%	1.12	88.5%	69.35%	1.08	90.2%	71.04%	1.00	99.7%	68.53%	1.18	84.0%
$\alpha = 0.25$	**33.02%**	**0.81**	95.8%	56.84%	0.84	98.6%	37.55%	0.87	**100.0%**	41.07%	0.85	91.8%
$\alpha = 0.50$	35.46%	**0.81**	**97.3%**	35.51%	**0.82**	**100.0%**	36.14%	**0.85**	**100.0%**	**29.52%**	**0.84**	**95.0%**
$\alpha = 0.75$	46.71%	0.82	96.8%	**25.06%**	0.83	99.2%	**35.88%**	0.86	**100.0%**	32.11%	**0.84**	94.2%
$\alpha = 1.00$	69.78%	1.05	86.0%	70.46%	1.10	87.5%	70.21%	1.12	88.0%	68.96%	1.20	81.5%

5 Related Work

LLM for optimization modeling. OptiMUS formulates LP/MILP models and repairs solver code using execution errors and solution tests [7]. Chain-of-Experts coordinates specialized agents through forward reasoning and backward reflection [26]. Astorga et al. explore formulation components with hierarchical MCTS, LLM evaluation, and symbolic pruning [8], while DRoC decomposes constraints for retrieval and routes between retrieval and self-debugging [9]. ORLM learns optimization modeling from OR-Instruct synthetic data [3]. For dynamic dispatch, missing state observations or evolving constraints can still lead to formally valid yet physically infeasible models.

LLM for optimization solving and algorithm selection. OPRO generates candidates conditioned on previously evaluated solutions and objective values [10]. LLM4AD combines LLM sampling, search strategies, and sandboxed evaluation for algorithm design [12]. HeurAgenix evolves heuristics from contrastive solution trajectories and selects them online through state information and test-time search [13]. Feature-based algorithm selection exploits complementary solver strengths across problem instances [27]. Even with adaptive selection, algorithm-specific inductive biases remain: shifts in convexity, variable types, or constraint stiffness can make a previously suitable solving strategy ineffective.

End-to-end optimization and agentic reasoning. ReAct interleaves reasoning with actions and environmental observations [24], while Tree of Thoughts evaluates branching reasoning states with lookahead and backtracking [23]. OR-LLM-Agent separates mathematical modeling, Gurobi code generation, and execution-driven repair of code or models [5]. In tightly coupled feedback loops, poor outcomes may stem from formulation errors, solver mismatch, or misleading reflection, leaving the failure source and correction direction ambiguous.

LLM agents for power systems. GridMind integrates conversational agents with deterministic solvers for AC optimal power flow and contingency analysis [1]. X-GridAgent organizes planning, coordination, and tool execution for grid analysis, combining domain-specific servers with retrieval and memory [2]. These systems demonstrate how language interfaces orchestrate engineering analyses. Maintaining consistent objectives, feasible constraints, and reliable feedback over long dispatch horizons remains challenging as renewable injections and network topology change.

6 Conclusion

End-to-End LLM agents for power system dispatch often couple modeling, solving, and reflection, making failures hard to attribute to formulation errors, solver mismatch, or misleading feedback. To address this ambiguity, ADA provides a structured connection between semantic reasoning, mathematical optimization, and physical feasibility. It separates dispatch reasoning into coupled but asynchronous processes, using finite-horizon planning to formalize ambiguous dispatch states as optimization problems while updating task knowledge on a slower time scale. Experiments on L2RPN benchmarks show that ADA improves completion, reduces overload, and achieves better safety–cost trade-offs over optimization, RL, and LLM-agent baselines. Ablations confirm that these gains come from asynchronous AK/TK evolution rather than extra compute. Nevertheless, ADA remains limited when grid dynamics fluctuate faster than slow knowledge updates can track; outside this separation regime, the local AK/TK tracking analysis provides no guarantee. The Planner's MCTS search also introduces overhead that may require further optimization for stricter real-time deployment. Finally, the simulator evaluation does not validate unattended live-grid operation. We envision tiered supervised autonomy in which only low-impact, physically verified actions within an operator-defined envelope may execute automatically, while distribution shifts, narrow margins, verification failures, Judge–physics conflicts, and high-impact topology changes require operator review and a certified operational fallback.

Acknowledgments and Disclosure of Funding

This work was supported in part by the National Natural Science Foundation of China under Grants 62325306 and 62273237. This work was also supported in part by the Key Research and Development Program of Xizang Department of Science and Technology under Grant XZ202502ZY0065.

The authors declare no competing interests.

References

[1] Hongwei Jin, Kibaek Kim, and Jonghwan Kwon. Gridmind: Llms-powered agents for power system analysis and operations, 2025.

[2] Yihan, Wen, and Xin Chen. X-gridagent: An llm-powered agentic ai system for assisting power grid analysis, 2025. URL https://arxiv.org/abs/2512.20789.

[3] Chenyu Huang, Zhengyang Tang, Shixi Hu, Ruoqing Jiang, Xin Zheng, Dongdong Ge, Benyou Wang, and Zizhuo Wang. Orlm: A customizable framework in training large models for automated optimization modeling. *Operations Research*, 73(6):2986–3009, 2025.

[4] Cheryl Li, Tianyuan Xu, and Yiwen Guo. Reasoning-as-logic-units: Scaling test-time reasoning in large language models through logic unit alignment, 2025.

[5] Bowen Zhang, Pengcheng Luo, Genke Yang, Boon-Hee Soong, and Chau Yuen. Or-llm-agent: Automating modeling and solving of operations research optimization problems with reasoning llm, 2025. URL https://arxiv.org/abs/2503.10009.

[6] Xinyu Jiang, Haoyu Zhang, Mengyi Sha, Zihao Jiao, Long He, Junbo Zhang, and Wei Qi. Rideagent: An llm-enhanced optimization framework for automated taxi fleet operations, 2026.

[7] Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. Optimus: Optimization modeling using mip solvers and large language models, 2023.

[8] Nicolás Astorga, Tennison Liu, Yuanzhang Xiao, and Mihaela Van Der Schaar. Autoformulation of mathematical optimization models using llms, 2024.

[9] Xia Jiang, Yaoxin Wu, Chenhao Zhang, and Yingqian Zhang. Droc: Elevating large language models for complex vehicle routing via decomposed retrieval of constraints. In *13th International Conference on Learning Representations, ICLR 2025*. OpenReview. net, 2025.

[10] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers, 2024. URL https://arxiv.org/abs/2309.03409.

[11] Rui Zhang, Fei Liu, Xi Lin, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Understanding the importance of evolutionary search in automated heuristic design with large language models, 2024. URL https://arxiv.org/abs/2407.10873.

[12] Fei Liu, Rui Zhang, Zhuoliang Xie, Rui Sun, Kai Li, Qinglong Hu, Ping Guo, Xi Lin, Xialiang Tong, Mingxuan Yuan, et al. Llm4ad: A platform for algorithm design with large language model, 2024.

[13] Xianliang Yang, Ling Zhang, Haolong Qian, Lei Song, and Jiang Bian. Heuragenix: Leveraging llms for solving complex combinatorial optimization challenges, 2025. URL https://arxiv.org/abs/2506.15196.

[14] Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change, 2023.

[15] RTE, EPRI, Chalearn, Tailor, and European Union. L2rpn challenge. https://l2rpn.chalearn.org/, 2024. Chalearn; Accessed 20-08-2024.

[16] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[17] Adrian Hauswirth, Saverio Bolognani, Gabriela Hug, and Florian Dörfler. Timescale separation in autonomous optimization. *IEEE Transactions on Automatic Control*, 66(2):611–624, 2020.

[18] Antoine Marot, Benjamin Donnot, Gabriel Dulac-Arnold, Adrian Kelly, Aidan O'Sullivan, Jan Viebahn, Mariette Awad, Isabelle Guyon, Patrick Panciatici, and Camilo Romero. Learning to run a power network challenge: a retrospective analysis, 2021.

[19] Antoine Marot, Benjamin Donnot, Sami Tazi, and Patrick Panciatici. Expert system for topological remedial action discovery in smart grids. In *Mediterranean Conference on Power Generation, Transmission, Distribution and Energy Conversion (MEDPOWER 2018)*, pages 1–6. IET, 2018.

[20] RTE France. Milp-agent. https://github.com/Grid2op/grid2op-milp-agent, 2022. GitHub; Accessed 15-08-2024.

[21] Tu Lan, Jiajun Duan, Bei Zhang, Di Shi, Zhiwei Wang, Ruisheng Diao, and Xiaohu Zhang. Ai-based autonomous line flow control via topology adjustment for maximizing time-series atcs. In *2020 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5. IEEE, 2020.

[22] Blazej Manczak, Jan Viebahn, and Herke van Hoof. Hierarchical reinforcement learning for power network topology control, 2023.

[23] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL `https://arxiv.org/abs/2305.10601`.

[24] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL `https://arxiv.org/abs/2210.03629`.

[25] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

[26] Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Jessica Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, and Gang Chen. Chain-of-experts: When LLMs meet complex operations research problems. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=HobyL1B9CZ`.

[27] Pascal Kerschke, Holger H Hoos, Frank Neumann, and Heike Trautmann. Automated algorithm selection: Survey and perspectives, 2019.

[28] Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, et al. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*, 2023.

[29] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 2511–2522, 2023.

[30] Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, Nitesh V. Chawla, and Xiangliang Zhang. Justice or prejudice? quantifying biases in LLM-as-a-judge. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2410.02736. URL `https://arxiv.org/abs/2410.02736`.

A Experimental details and Dispatch Workflow

This appendix provides the implementation details used for all L2RPN experiments and expands the settings summarized in Sec. 4.3.

A.1 Hardware

All experiments were conducted in PyTorch (v2.8.0) with the `torch-npu` backend on Huawei Ascend 910B NPUs under CANN v8.2. The platform is a Huawei Atlas 900 RCK A2 compute node with 8 Ascend-series accelerator cards, 4 HiSilicon Kunpeng-920 CPUs (192 CPU cores in total, `aarch64`), and 1.5 TiB host memory.

A.2 Experimental settings

All chat models are served through an OpenAI-compatible chat-completions endpoint configured by `base_url`, `api_key`, and `model`. The temperature is fixed to 0.7 and the maximum generation length is 4096 tokens. The same simulator backend, action space, solver interfaces, power-flow calculators, prompt templates, and retry policy are used across LLM-based baselines and ADA. We repeat each experiment over five random seeds $[41, 42, 43, 44, 45]$. Each online dispatch decision must be returned within the 5-minute dispatch interval; invalid, missing, unverifiable, or timed-out actions are replaced by the uniform *Do-Nothing* benchmark fallback.

The baselines choose from this pool through their native reasoning, whereas ADA matches problem features to solver capability vectors. All solver attempts, including retries, remain subject to the shared timeout, retry limit, five-minute dispatch deadline, and benchmark fallback policy.

A.3 Open Access, Reproducibility Assets, and Licenses

The reproduction package is available at `https://github.com/kongfuguagua-iwin/Agile-Dispatch-Agent.git`. All benchmark data are obtained from the official Grid2Op/L2RPN distribution channels rather than redistributed in this paper. Reproducers can obtain the environments through Grid2Op (`https://github.com/Grid2op/grid2op`; documentation: `https://grid2op.readthedocs.io/`) and the L2RPN Challenge portal (`https://l2rpn.chalearn.org/`). The L2RPN baseline implementations used for reference agents are available from `https://github.com/Grid2op/l2rpn-baselines`. For faster power-flow simulation, the Grid2Op-compatible LightSim2Grid backend can be obtained from `https://github.com/Grid2op/lightsim2grid`. The reproduction package specifies the four benchmark identifiers used in the experiments, the five seeds $[41, 42, 43, 44, 45]$.

A.4 Benchmark Dispatch Priorities

The four L2RPN benchmarks in Sec. 4.1 form a progression from fuzzy redispatch to renewable uncertainty, extreme disturbances, and storage coordination. Table 4 provides a compact summary.

IJCNN-2019: fuzzy redispatch. IJCNN-2019 is built on an IEEE 14-bus system and mainly evaluates redispatch-only control under network constraints. The agent cannot rely on topology switching or storage coordination, so it must adjust generator outputs while balancing overload risk, operating cost, and intervention scope. This benchmark tests whether the Planner can convert fuzzy operator preferences into explicit objective weights and feasible redispatch constraints.

WCCI-2020: constraint inference under maintenance and renewable uncertainty. WCCI-2020 extends the dispatch problem to an IEEE 36-bus grid with planned line outages and uncertain photovoltaic and wind generation. Planned maintenance changes available transmission paths, while renewable fluctuations shift line flows and feasibility boundaries over time. This benchmark tests whether the agent can identify active constraints and construct optimization problems that remain physically meaningful under maintenance and renewable uncertainty.

NEURIPS-2020: robustness under extreme disturbances. NEURIPS-2020 uses an IEEE 118-bus system and emphasizes robustness under extreme weather and adversarial topological disturbances. Short-term overload reduction alone is insufficient because a locally beneficial action may reduce the

system's ability to recover from later disturbances. This benchmark tests whether ADA avoids brittle, myopic decisions and maintains safety, recovery, and stability under out-of-distribution stress.

WCCI-2022: storage coordination with delayed effects. WCCI-2022 introduces large-scale battery storage into the IEEE 118-bus system. Storage actions have delayed temporal effects: charging or discharging may improve the current state while harming future overload margins, or vice versa. This benchmark tests whether the dispatch policy can coordinate storage, topology, and redispatch over multiple steps, which is directly aligned with ADA's fast adaptation and slow task-knowledge evolution.

Table 4: **Compact summary of L2RPN benchmark dispatch priorities.**

Benchmark	Grid scale	Main stressor	Core dispatch priority
IJCNN-2019	IEEE 14-bus	Fuzzy redispatch	Balance safety, cost, and intervention scope
WCCI-2020	IEEE 36-bus	Maintenance + renewables	Infer time-varying feasibility boundaries
NEURIPS-2020	IEEE 118-bus	Extreme disturbances	Maintain safety, recovery, and robustness
WCCI-2022	IEEE 118-bus	Storage coordination	Coordinate multi-step storage and grid stability

A.5 ADA Algorithm

Algorithm 1 Agile Dispatch Agent Framework with Retry and Benchmark Fallback

Require: Initial observation $\mathcal{X}_0$, toolset $\mathcal{T}$, knowledge bases $\mathcal{TK}, \mathcal{AK}$, max retries $K = 3$, solver timeout $\tau = 30$s, update frequency N_{freq}
Ensure: Dispatch action S^* and asynchronously updated knowledge

Initialize: Experience buffer $\mathcal{D} \leftarrow \emptyset$, async feedback queue $\mathcal{Q}_{\text{async}} \leftarrow \emptyset$, step $t \leftarrow 0$
while system is running **do**
  $t \leftarrow t + 1$, $Valid \leftarrow$ false
  for $r = 1$ to K **do**
    {— Module 1: Planner (active state augmentation) —}
    $\mathcal{X} \leftarrow \text{Planner}(\mathcal{X}_0, \mathcal{AK}, \mathcal{TK}, \mathcal{T})$
    Formulate problem: $\langle \mathcal{J}, \Theta \rangle \leftarrow \mathcal{TK}(\mathcal{X})$
    {— Module 2: Solver (adaptive selection) —}
    Extract features $\phi \leftarrow \text{FeatureMap}(\langle \mathcal{J}, \Theta \rangle)$
    Select solver $A^* \leftarrow \arg\max_{A \in \mathcal{A}} G(A, \phi)$ ▷ Eq. (6)
    Generate candidate $S_{sol} \leftarrow A^*(\langle \mathcal{J}, \Theta \rangle)$ with timeout τ
    {— Synchronous physical gate —}
    $Valid \leftarrow \text{CheckPhysics}(S_{sol})$
    if $Valid$ **then**
      break
    end if
    Record violation source and refine constraints for the next retry
  end for
  if $\neg Valid$ **then**
    $S^* \leftarrow \text{DoNothing}()$
    Store fallback trace in $\mathcal{D}$, enqueue it to $\mathcal{Q}_{\text{async}}$, and execute S^*
    continue
  end if
  $S^* \leftarrow S_{sol}$
  Execute S^* and store verified trace $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathcal{X}, t_{seq}, A^*, S^*)\}$
  Enqueue the verified trace to $\mathcal{Q}_{\text{async}}$ for Judger–Summarizer feedback
  {— Asynchronous feedback loop (non-blocking) —}
  if Judger scores are available for completed traces **then**
    Compute reward $R \leftarrow (1 - \alpha) R_{PHY} + \alpha R_{LLM}$ ▷ Eq. (8)
    Update $\mathcal{AK}$ from MCTS posterior statistics and Judger feedback on the fast scale
    if $t \mod N_{freq} = 0$ **then**
      Update $\mathcal{TK}$ from high-reward and physically verified replay traces
    end if
  end if
end while

B Planner Module

B.1 Planner Configuration

The Planner uses the same 10-tool interface made available to LLM-based methods. Each tool is deterministic except for the LLM summarization step, and each call appends structured fields to the augmented observation $\mathcal{X}$. Table 5 lists the tool set.

Table 5: **Planner tool set.** The same 10 tools are exposed to ADA and LLM-based baselines; ADA searches over them with finite-horizon MCTS.

Tool	Output field	Function
StateParser	$\mathcal{X}_{raw}$	Normalizes Grid2Op observations, including loads, generators, line loading ρ, topology, storage state of charge, cooldown counters, and maintenance flags.
SceneSummarizer	$\mathcal{S}_{text}$	Converts the normalized state into a compact natural-language scene summary for LLM reasoning and records the dominant operational risk.
FaultScanner	$\mathcal{F}_{risk}$	Detects overloaded lines, disconnected lines, maintenance outages, and N-k-style contingency patterns.
PowerFlow	$\hat{\rho}, \hat{v}$	Runs post-action power-flow simulation for candidate redispatch, topology, and storage actions.
Sensitivity	PTDF, LODF	Estimates line-flow sensitivity to generator redispatch and topology changes, used to rank candidate interventions.
RenewableBand	$[\hat{p}^-, \hat{p}^+]$	Builds short-horizon PV/wind uncertainty bands from recent injections and forecast deltas.
RedispatchTemplate	Θ_{redisp}	Extracts generator ramp limits, redispatch bounds, and marginal-cost terms.
TopologyTemplate	Θ_{topo}	Enumerates legal busbar switching actions under cooldown and substation constraints.
StorageTemplate	Θ_{stor}	Constructs battery charge/discharge limits, state-of-charge bounds, and terminal reserve constraints.
ObjectiveComposer	$\langle \mathcal{J}, \Theta \rangle$	Maps benchmark priorities and active constraints into the final optimization objective, constraint set, and solver query.

In ADA, the Planner performs finite-horizon MCTS with maximum depth $D = 6$ over this tool set. Each decision uses $M = 100$ MCTS simulations; a single simulation expands at most one new tool node, so the per-decision expansion budget is also bounded by 100. A search state is $s_d = (\mathcal{X}_d, \mathcal{H}_d)$, where $\mathcal{H}_d$ is the tool history. The action is the next tool $t_d \in \mathcal{T}$, and the backup is the Solver-Judger reward. The selection rule follows the UCB form in Sec. 3.1, while expansion executes the selected tool and appends its structured output.

Algorithm 2 Planner tool-level search and formulation

Require: Observation $\mathcal{X}_0$, tool set $|\mathcal{T}| = 10$, depth $D = 6$, simulations $M = 100$, action knowledge $\mathcal{AK}$, task knowledge $\mathcal{TK}$

1: Initialize root state $s_0 = (\mathcal{X}_0, \emptyset)$ with priors from $\mathcal{AK}$
2: **for** MCTS simulation $m = 1, \ldots, M$ **do**
3: Select tools by UCB until depth D or a terminal feasible formulation is reached
4: Execute selected tool t_d and update $\mathcal{X}_{d+1} \leftarrow t_d(\mathcal{X}_d)$
5: Compose candidate $\langle \mathcal{J}, \Theta \rangle \leftarrow \mathrm{ObjectiveComposer}(\mathcal{X}_{d+1}, \mathcal{TK})$
6: Evaluate the candidate with Solver and Judger to obtain reward R
7: Backpropagate R to update visit counts and tool values
8: **end for**
9: Return the best augmented observation and formal problem $\langle \mathcal{J}, \Theta \rangle$

C Solver Module

C.1 Solver Configuration

The Solver selects among six solver/problem families according to the formalized tuple $\langle \mathcal{J}, \Theta \rangle$ and the features in Appendix C.2. Let u denote the full dispatch decision, including redispatch Δp, topology variables z, and storage action b when available. Let $\rho_\ell(u)$ be the post-action loading ratio of line ℓ estimated by the simulator or linearized flow model. The six candidate families are:

1. Greedy verified local search. Used when the candidate action set is small and physical simulation is cheap:

$$u^* = \arg \min_{u \in \mathcal{C}_K(\mathcal{X})} \left[C(u) + \lambda_o \sum_\ell \max(0, \rho_\ell(u) - 1)^2 + \lambda_t \|z - z_{ref}\|_0 \right]. \tag{20}$$

Here $\mathcal{C}_K(\mathcal{X})$ is a top-K candidate set generated from sensitivity and heuristic-rule tools.

2. Linear redispatch program. Used when a DC approximation is sufficient:

$$\min_{\Delta p, s} \; c^\top \Delta p + \lambda_s \mathbf{1}^\top s \quad \text{s.t.} \quad \mathbf{1}^\top \Delta p = 0,\; \underline{r} \le \Delta p \le \bar{r},\; |f^0 + \mathrm{PTDF}\Delta p| \le f^{\max} + s,\; s \ge 0. \tag{21}$$

3. Quadratic redispatch program. Used when smooth cost and ramp penalties dominate:

$$\min_{\Delta p, s} \; \frac{1}{2} \Delta p^\top H \Delta p + c^\top \Delta p + \lambda_s \|s\|_2^2 \quad \text{s.t.} \quad \mathbf{1}^\top \Delta p = 0,\; \underline{r} \le \Delta p \le \bar{r},\; |f^0 + \mathrm{PTDF}\Delta p| \le f^{\max} + s. \tag{22}$$

4. Mixed-integer topology program. Used when busbar switching or line-status decisions are active:

$$\min_{\Delta p, z, s} \; c^\top \Delta p + \kappa^\top z + \lambda_s \mathbf{1}^\top s \quad \text{s.t.} \quad A\Delta p + Bz \le b + s,\; z \in \{0,1\}^m,\; z \in \Theta_{cooldown}. \tag{23}$$

5. Particle-swarm black-box optimization. Used for non-convex or simulator-only objectives:

$$u^* = \arg\min_{u \in \Omega} \mathcal{J}_{sim}(u), \qquad v_j^{r+1} = \omega v_j^r + c_1 \xi_1 (p_j^r - u_j^r) + c_2 \xi_2 (g^r - u_j^r), \quad u_j^{r+1} = \Pi_\Omega(u_j^r + v_j^{r+1}). \tag{24}$$

6. Bayesian black-box optimization. Used when each power-flow evaluation is expensive and the search dimension is moderate:

$$u_{r+1} = \arg\max_{u \in \Omega} a(u; \mathcal{D}_r), \qquad \mathcal{D}_{r+1} = \mathcal{D}_r \cup \{(u_{r+1}, \mathcal{J}_{sim}(u_{r+1}))\}, \tag{25}$$

where $a(\cdot)$ is the acquisition function. LP/QP/MILP forms are modeled through CVXPY and solved with the available numerical backend; greedy, particle-swarm, and Bayesian variants call the same physical verification layer before an action is accepted.

Table 6: **Initial Solver Capability Vectors.** Each entry of $\psi(A) \in [0,1]^4$ follows the feature order used by ϕ: non-convexity, constraint stiffness, variable complexity, and objective evaluation cost. The vectors are fixed across all benchmarks.

Solver family	Non-convexity	Constraint stiffness	Variable complexity	Eval. cost
Greedy verified local search	0.50	0.70	0.20	0.60
Linear redispatch program	0.10	0.80	0.40	0.90
Quadratic redispatch program	0.20	0.80	0.50	0.80
Mixed-integer topology program	0.30	0.90	0.90	0.50
Particle-swarm black-box optimization	0.90	0.50	0.60	0.40
Bayesian black-box optimization	0.80	0.40	0.40	0.90

C.2 Solver Feature Extraction Methodology

To enable the Adaptive Algorithm Selection (AAS) mechanism in the Solver module, we map the formalized optimization tuple $\langle \mathcal{J}, \Theta \rangle$ into a rigorous feature vector $\phi \in \mathbb{R}^d$. The extraction process utilizes both symbolic analysis and excitation testing.

1. Local Convexity Estimation (f_{cvx}): We employ a Monte Carlo-based excitation test to estimate the non-convexity of the objective landscape. By sampling N_s pairs of points (x_1, x_2) within the variable bounds and checking Jensen's inequality:

$$f_{cvx} = \frac{1}{N_s} \sum_{i=1}^{N_s} \mathbb{I}\Big[\mathcal{J}(\lambda x_1 + (1-\lambda)x_2) > \lambda \mathcal{J}(x_1) + (1-\lambda)\mathcal{J}(x_2)\Big], \tag{26}$$

where $\mathbb{I}$ is the indicator function and $\lambda \sim U(0,1)$. A value closer to 0 indicates high convexity.

2. Constraint Stiffness (f_{stiff}): This feature quantifies the tightness of the feasible region defined by Θ without measuring zero-volume equality-constrained sets in the ambient bounding box. Let the equality constraints in Θ be written as $Ex = h$, let x_p be a feasible point satisfying them, and let N be an orthonormal basis of $\mathrm{null}(E)$; if no equality constraints are present, $N = I$. Hit-and-Run directions are sampled in this affine hull as $d_j = Nu_j/\|Nu_j\|_2$, with u_j drawn from a standard normal distribution. For a current feasible point x_j, define

$$\begin{aligned} &I_B(x_j, d_j) = \{\gamma \mid \mathbf{LB} \le x_j + \gamma d_j \le \mathbf{UB}\}, \\ &I_\Theta(x_j, d_j) = \{\gamma \in I_B(x_j, d_j) \mid x_j + \gamma d_j \text{ satisfies all inequalities in } \Theta\}. \end{aligned} \tag{27}$$

The stiffness estimate is the average feasible chord ratio:

$$f_{stiff} = 1 - \frac{1}{N_{\text{sample}}} \sum_{j=1}^{N_{\text{sample}}} \frac{|I_\Theta(x_j, d_j)|}{|I_B(x_j, d_j)| + \epsilon}. \tag{28}$$

Thus sampling is performed within the equality manifold rather than uniformly over the ambient box. Higher f_{stiff} implies a narrower feasible chord structure and a more constrained problem requiring solvers with stronger constraint-handling capability.

3. Solving Complexity Quantification (f_{var}): We quantify the complexity based on the scale and type of decision variables. The metric aggregates the dimensionality of continuous variables (N_{cont}) and integer variables (N_{int}):

$$f_{var} = \log\left(1 + N_{cont} + \beta \cdot N_{int} \cdot 2^{N_{int}}\right), \tag{29}$$

where $\beta = 1$ in all experiments is the fixed weight factor reflecting the NP-hard nature of integer programming. This feature helps distinguish between LP/QP and combinatorial problems.

4. Objective Computational Complexity (f_{ops}): To estimate the cost of evaluating the objective function $\mathcal{J}$, we construct a computational graph and count the floating-point operations (FLOPs):

$$f_{ops} = \frac{\sum_{o \in \mathcal{O}_{\text{op}}} \omega_o \cdot N_{\text{op},o}}{T_{\text{ref}}}, \tag{30}$$

where $\mathcal{O}_{\text{op}}$ is the set of operator types in the computational graph, $N_{\text{op},o}$ is the occurrence count of operator o, and ω_o is its weighted FLOP cost. We use $\omega_{\text{add}} = \omega_{\text{mul}} = 1$, $\omega_{\text{exp}} = \omega_{\log} = \omega_{\text{trig}} = 20$, and $T_{\text{ref}} = 10^3$ weighted FLOPs. The symbol $\pi_{\mathcal{AK}}$ is reserved for the AK tool-policy distribution and is not used for operator sets. This feature serves as a proxy for the per-iteration cost of gradient-based or population-based algorithms.

D Judger Module

D.1 Judger Configuration

ADA separates the synchronous physical gate from asynchronous Judger scoring. The physical gate checks executability before the action is sent to the environment, while the Judger computes feedback for later AK/TK updates by combining physical outcomes with soft logical assessment:

1. A synchronous physics checker verifies whether the Solver output satisfies simulator constraints, including thermal limits, topology validity, maintenance/cooldown restrictions, and action executability. Invalid or unverifiable actions are rejected before dispatch and trigger the retry mechanism described in Sec. 4.3.

2. For dispatched or fallback traces, the asynchronous Judger computes a physical reward R_{PHY} from physical dispatch metrics.
3. The LLM-as-a-Judge produces a rubric-based logical score R_{LLM}, which is fused with R_{PHY} only in the non-blocking feedback loop used by the Summarizer.

Thus, the LLM never directly selects or executes grid actions, and its latency is not part of the synchronous dispatch path; physical feasibility determines admissibility, while the LLM term provides complementary semantic feedback for policy and knowledge evolution.

D.1.1 Physical Reward

After the synchronous physical gate accepts an action or records a fallback trace, the asynchronous Judger computes the physical reward used in Sec. 4.3 by aggregating four physical signals aligned with the reported dispatch metrics:

$$\mathcal{R}_{PHY} = 0.25\mathcal{R}_{disp} + 0.30\mathcal{R}_{bridge} + 0.25\mathcal{R}_{overflow} + 0.20\mathcal{R}_{distance}. \tag{31}$$

RedispReward $\mathcal{R}_{disp}$ penalizes unnecessary generation adjustment cost, BridgeReward $\mathcal{R}_{bridge}$ measures remaining safety margin, CloseToOverflowReward $\mathcal{R}_{overflow}$ penalizes operation near or beyond thermal limits, and DistanceReward $\mathcal{R}_{distance}$ penalizes excessive deviation from the reference topology:

$$\begin{aligned} \mathcal{R}_{disp} &= 1 - \frac{\sum_{g\in\mathcal{G}} C(P_g^t) - C(P_g^0)}{\max(C_{\mathrm{limit}}, \epsilon)}, \\ \mathcal{R}_{bridge} &= \min_{\ell\in\mathcal{L}} \max(0, 1-\rho_\ell)^p, \\ \mathcal{R}_{distance} &= \exp(-\lambda \|\mathbf{T}_t - \mathbf{T}_{ref}\|_0). \end{aligned} \tag{32}$$

$$\mathcal{R}_{overflow} = \frac{1}{|\mathcal{L}|}\sum_{\ell\in\mathcal{L}} \phi(\rho_\ell), \qquad \phi(\rho) = \begin{cases} 1, & \rho < \tau_{\mathrm{warn}}, \\ \frac{1-\rho}{1-\tau_{\mathrm{warn}}}, & \tau_{\mathrm{warn}} \le \rho \le 1, . \\ -\exp(\rho - 1), & \rho > 1 \end{cases} \tag{33}$$

The discontinuity at $\rho = 1$ is intentional: the middle branch reaches zero at the warning-limit boundary, while any strict thermal-limit violation enters the negative penalty branch and is also caught by the hard feasibility checker before action acceptance. We use $\tau_{\mathrm{warn}} = 0.9$, $p = 2$, $\lambda = 1$, and $\epsilon = 10^{-6}$. The normalization constant C_{limit} is the benchmark-specific maximum feasible redispatch cost increase under generator ramp and capacity bounds. For the compact three-term notation, the bridge and overflow terms jointly form overload safety, so $w_d = 0.25$, $w_o = 0.55$, and $w_t = 0.20$. These weights are kept fixed across methods so that AvgR is comparable across baselines and ADA.

D.1.2 LLM-as-a-Judge Reward

The LLM-as-a-Judge evaluates aspects of dispatch quality that are not fully captured by scalar physical rewards. Examples include whether the formalized optimization problem matches the stated dispatch priority and whether the action creates avoidable future risk. The LLM Judge receives four structured inputs:

1. the current grid state summary, including maximum loading, overloaded lines, maintenance flags, renewable uncertainty, and storage state;
2. the proposed optimization formulation $\langle \mathcal{J}, \Theta \rangle$;
3. the candidate dispatch action and simulated physical effects;
4. the benchmark-specific scenario-priority mapping in Appendix A.4.

The prompt is fixed across benchmarks and calibrated as a rubric:

> **Judge prompt skeleton.** You are given four structured inputs for evaluating a dispatch decision.
>
> **[Input 1: Grid State Summary]**
> {grid state summary, including maximum loading, overloaded lines, maintenance flags, renewable uncertainty, and storage state}

[Input 2: Optimization Formulation]
{proposed optimization problem $\langle \mathcal{J}, \Theta \rangle$, including objective terms, constraints, and key hyperparameters}

[Input 3: Candidate Action and Simulated Effects]
{candidate dispatch action and simulated physical effects, including post-action loading, violations, topology changes, and storage changes}

[Input 4: Scenario-Priority Mapping]
{benchmark-specific dispatch priorities from Appendix A.4}

Based only on the provided inputs, score the candidate dispatch plan on three rubric dimensions: *Rationality*, *Safety*, and *Sustainability*, each from 0 to 5. *Rationality* measures whether the action is consistent with the proposed optimization formulation and the benchmark-specific priority. *Safety* measures whether the action respects thermal limits, maintenance or cooldown constraints, and critical topology requirements. *Sustainability* measures whether the action avoids unnecessary future risk, including avoidable overload propagation, storage infeasibility, or degradation of future controllability. Penalize any plan that violates physical constraints, ignores maintenance or cooldown restrictions, disconnects critical topology, or worsens future storage feasibility. Return structured fields: scores, supporting evidence, detected failure mode, and a one-sentence reliability summary. Do not propose a new control action.

Let the returned rubric scores be s_{rat}, s_{safe}, and s_{sus} for *Rationality*, *Safety*, and *Sustainability*, respectively, each in $\{0, 1, \ldots, 5\}$. The scalar logical reward used in Eq. (8) is the normalized equally weighted average:

$$R_{\mathrm{LLM}} = \frac{s_{\mathrm{rat}} + s_{\mathrm{safe}} + s_{\mathrm{sus}}}{15}. \tag{34}$$

This score is computed only after the physical feasibility gate and is used for asynchronous feedback; infeasible actions are rejected before dispatch, not by an LLM scoring threshold.

D.2 Detailed Judge Coefficient

Figure 6 shows that the safety margin follows a U-shaped pattern. Pure physical scoring ($\alpha = 0$) and pure logical scoring ($\alpha = 1$) both increase future maximum loading, whereas the hybrid region around $\alpha \in [0.5, 0.75]$ gives the lowest Future 10 loading. This supports the main-text conclusion: the LLM Judge should provide semantic ranking information, but it should not replace physical verification.

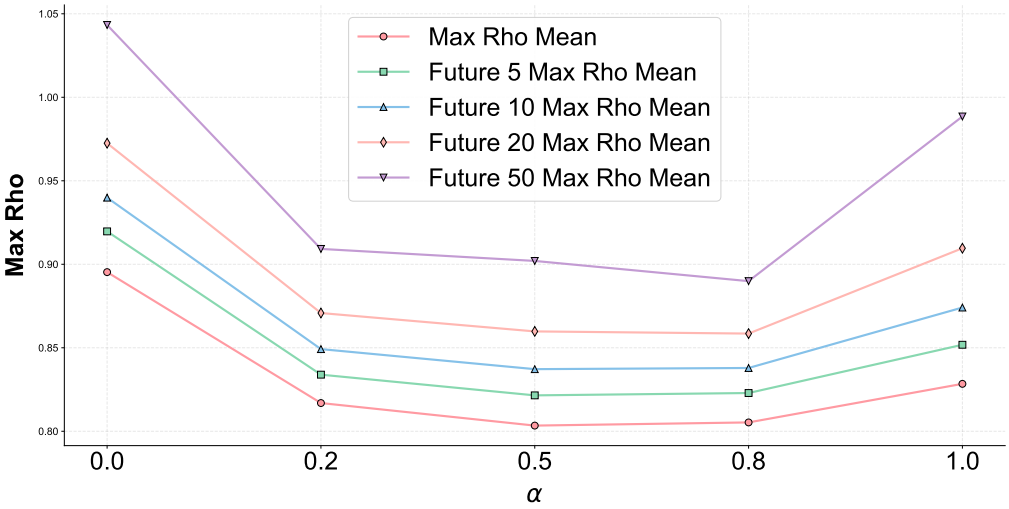


Figure 6: **Sensitivity Analysis of Judger Coefficient.**

E Summarizer Module

E.1 Update Frequencies

ADA updates AK and TK at different frequencies because they track different levels of non-stationarity. AK biases short-term tool selection in the Planner, so it is updated every 10 dispatch steps from MCTS posterior statistics, matching the depth-6 search budget over 10 tools and allowing rapid adaptation to recent grid-state changes. TK stores slower-changing task structure, including objective templates, active constraints, and solver preferences. Updating TK too often would make the feasible-region estimate sensitive to noisy trajectories, while updating it too rarely would retain stale templates under distribution shift. We therefore update TK every 100 dispatch steps; in the 5-minute dispatch setting, this corresponds to roughly 8 hours and aligns with intra-day renewable, load, and storage-regime changes. This schedule lets AK absorb short-term fluctuations while TK reconstructs reusable formulation priors only after sufficient evidence accumulates.

E.2 Asynchronous Update Efficiency

Efficiency Gains from Asynchronous Updates: Figure 7 measures LLM call count. Total token consumption is the product of call count and the per-call budgets in Table 10. The call count can be larger than the number of dispatch decisions because planning tools, MCP-connected scripts, waiting for tool results, and callback functions may trigger multiple LLM-facing calls within one dispatch step. Across datasets, LLM calls decrease from 46.73 to 18.62 on average, and WCCI-2022 drops from 81.00 to 34.10. This shows that the Summarizer turns repeated dispatch experience into reusable memory: as $\mathcal{AK}$ and $\mathcal{TK}$ accumulate, the Planner makes fewer erroneous tool calls and suffers fewer tool-induced reasoning interruptions.

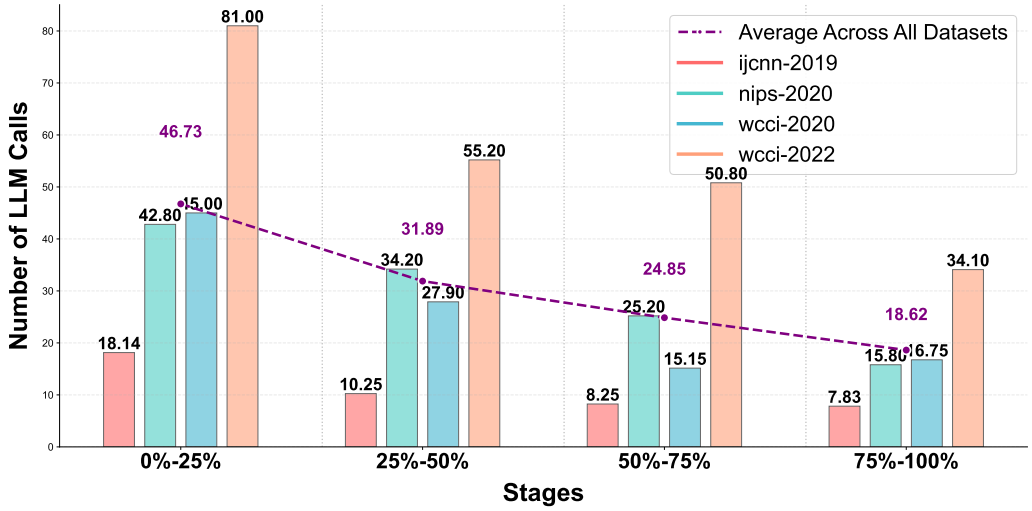


Figure 7: **Evolution of LLM Call Frequency.** The figure reports call count, not token volume; token usage is obtained by multiplying calls by the per-call budgets in Table 10.

F Knowledge Module: TK and AK

F.1 Action Knowledge Configuration

Action Knowledge ($\mathcal{AK}$) stores reusable tool-chain statistics for fast Planner adaptation. Each AK record contains a state signature (benchmark id, time bucket, max-ρ bucket, overloaded lines, maintenance flags, renewable regime, storage state, and cooldown status), a tool-chain prefix, visit count, average Solver-Judger reward, invalid-action count, timeout count, last-used step, and the posterior tool probability induced by MCTS.

F.2 Task Knowledge Configuration

TK record schema. Each TK entry is a structured record containing the grid condition, optimization setup, execution effect, and update metadata. The record fields are as follows.

1. **State key.** Benchmark id, time bucket, max-ρ bucket, overloaded lines, maintenance flags, renewable regime, storage SoC regime, cooldown status.
2. **Grid summary.** Operating mode, dominant stressor, and active risk pattern extracted by Planner tools.
3. **Optimization setup.** Objective template, active constraints, variable types, redispatch/storage/topology bounds, preferred solver family.
4. **Action trace.** Tool chain, formalized problem $\langle J, \Theta \rangle$, selected action, solver status, feasibility result.
5. **Effect summary.** 0–50 step effects: completion status, max-ρ trend, overload change, redispatch cost, topology distance, storage reserve change.
6. **Usage metadata.** Visit count, last-used step, average Judger reward, conflict count, stale score.
7. **Rationale.** Concise explanation of why the template is reusable under similar conditions.

F.3 Knowledge Storage and Update

Storage. The knowledge base is bounded to 500 entries. Records are partitioned by max-ρ regime, i.e., low, medium, high, and overload, and indexed by a B+tree over the state key. Each entry is about 3KB, giving about 1.5MB total storage. At dispatch time, the Planner retrieves top-k records from the matched max-ρ partition and ranks them by state-key overlap, effect similarity, recency, and historical utility.

Update. At each slow update, recent high-quality replay traces are converted into candidate AK/TK records and inserted into the corresponding partition. Records are scored by usage, 0–50 step effect, recency, Judger reward, and conflict penalty. The bottom 20% in each partition are removed as stale. Near-duplicates are merged, missing fields are filled from replay traces, and conflicts are resolved by keeping the record with better downstream physical effects and lower conflict count. The memory cap is then enforced.

Algorithm 3 Summarizer update for AK/TK

Require: Replay buffer $\mathcal{D}$, knowledge base $\mathcal{K}$, memory cap $B = 500$
if slow update step or distribution shift detected **then**
  Convert high-quality traces in $\mathcal{D}$ into candidate records
  Insert candidates into max-ρ partitions and update B+tree indices
  Score records by usage, 0–50 step effect, recency, reward, and conflict penalty
  Remove the bottom 20% records as stale
  Merge near-duplicates and fill missing fields
  Resolve conflicts by downstream physical effects
  Enforce $|\mathcal{K}| \leq B$
end if

F.4 TK Update Metric and Theoretical Assumption

The logical reconstruction operator Δ_Φ is the slow-scale update that revises TK templates from bounded replay summaries, rather than an append-only memory expansion. In implementation, $\Delta_\Phi(K, \mathcal{D})$ first converts high-quality replay traces in $\mathcal{D}$ into candidate TK records with state keys, optimization templates, action traces, and 0–50 step physical effects. Within each max-ρ partition, ADA ranks existing and candidate records by retrieval utility, downstream physical effect, recency, Judger reward, and conflict penalty. The bottom 20% records are removed as stale, near-duplicates are merged, missing fields are filled from replay traces, and conflicts are resolved by retaining the record with better future outcomes and lower conflict count. The resulting update revises reusable objective templates, active constraints, and solver priors while enforcing the bounded TK capacity.

Each TK record K is embedded into a continuous representation $z(K) \in \mathbb{R}^m$ by concatenating normalized numeric state features, categorical embeddings for benchmark/stressor/solver fields, and a sentence embedding of the concise rationale. We normalize $z(K)$ to unit L_2 norm and define

$$d(K, K') = \|z(K) - z(K')\|_2. \tag{35}$$

This metric only measures similarity between TK records. For AK policies over the finite tool set $\mathcal{T}$, we use the total-variation equivalent ℓ_1 metric

$$\|\pi - \pi'\|_1 = \sum_{t \in \mathcal{T}} |\pi(t) - \pi'(t)|. \tag{36}$$

Let $\bar{\Delta}_\Phi(\phi, \pi)$ denote the mean-field reconstruction map obtained by averaging the empirical replay update Δ_Φ under TK embedding ϕ and AK policy π. The regularity assumption used in Appendix I is

$$\|\bar{\Delta}_\Phi(\phi, \pi) - \bar{\Delta}_\Phi(\phi', \pi')\| \le L_\phi \|\phi - \phi'\| + L_\pi \|\pi - \pi'\|_1. \tag{37}$$

Operationally, this assumption means that nearby TK templates and nearby AK tool policies should induce nearby revisions of objectives, constraints, and solver priors, provided that the replay buffer is bounded and physically verified. The bounded TK capacity, stale-record removal, and conflict consolidation in Appendix F are the implementation mechanisms that keep the empirical discrete update close to this regularized embedded surrogate.

G Efficiency, Runtime, and Stability Analysis

G.1 Decision-Efficiency

This subsection reports the absolute numerical decision-efficiency metrics summarized by the relative Pareto frontier in Fig. 5. The experiment decomposes efficiency into action cost, effective safety capacity, and timely decision rate, so that low-cost behavior is not conflated with reliable dispatch. We define timely decision rate as

$$\mathcal{P} = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}(\tau_n \le 300\text{s}), \tag{38}$$

where τ_n is the wall-clock latency of decision n. Table 7 shows that ADA's main advantage is not the cheapest individual action, but the highest safety capacity under the 300s operational deadline.

Table 7: **Decision-Efficiency Results on L2RPN Benchmarks.** We report 3 metrics: **Action Cost ($\mathcal{I}$, $\downarrow$)**, **Effective Safety Capacity ($\mathcal{S}_{\text{eff}}$, $\uparrow$)**, and **Timely Decision Rate ($\mathcal{P}$, $\uparrow$)**, i.e., the proportion of decisions completed within the 300s dispatch interval. Best results are in **bold**, second-best are underlined.

Method	ijcnn-2019			wcci-2020			neurips-2020			wcci-2022		
	$\mathcal{I}\downarrow$	$\mathcal{S}_{\text{eff}}\uparrow$	$\mathcal{P}$ (%)$\uparrow$	$\mathcal{I}\downarrow$	$\mathcal{S}_{\text{eff}}\uparrow$	$\mathcal{P}$ (%)$\uparrow$	$\mathcal{I}\downarrow$	$\mathcal{S}_{\text{eff}}\uparrow$	$\mathcal{P}$ (%)$\uparrow$	$\mathcal{I}\downarrow$	$\mathcal{S}_{\text{eff}}\uparrow$	$\mathcal{P}$ (%)$\uparrow$
GreedyOpt	0.040	0.181	**100.0**	0.028	0.253	**100.0**	0.032	0.233	**100.0**	0.289	0.146	**100.0**
MILPAgent	0.307	0.183	**100.0**	0.763	0.255	**100.0**	0.258	0.233	**100.0**	20.374	0.160	**100.0**
SAC	0.616	0.190	**100.0**	0.327	0.264	**100.0**	0.122	0.232	**100.0**	10.811	0.152	**100.0**
HRL	**0.014**	0.184	**100.0**	0.374	0.245	**100.0**	0.145	0.234	**100.0**	10.497	0.153	**100.0**
ToT	0.207	0.185	84.3	0.197	0.251	87.1	0.083	0.231	92.6	0.791	0.139	71.4
ReAct	0.156	0.188	96.8	**0.022**	0.253	97.5	**0.023**	0.233	98.2	**0.035**	0.136	93.1
ORLM	0.166	0.183	**100.0**	0.349	0.252	**100.0**	0.145	0.234	**100.0**	5.581	0.139	**100.0**
ADA	0.057	**0.208**	**100.0**	0.103	**0.285**	**100.0**	0.025	**0.246**	**100.0**	2.704	**0.195**	**100.0**

G.2 Runtime and Compute Budget

We report runtime from the perspective of online dispatch: each decision must be issued within the 5-minute L2RPN control interval. All methods are evaluated under the same hardware and simulator settings as in Sec. 4.3. For ADA, the synchronous dispatch path consists of Planner $\rightarrow$ Solver $\rightarrow$ physical executability gate $\rightarrow$ L2RPN interaction, while LLM Judger scoring and Summarizer-based knowledge updates are reported separately because they do not block the immediate action execution.

Inference Latency. Tables 8 and 9 test whether ADA's MCTS-driven Planner remains usable for online dispatch. ADA's blocking path takes 109.33–133.09 s per step, and the retry-enabled end-to-end bound remains below the 300 s L2RPN control interval. This bound is measured after enabling the retry and timeout policy, so it already contains retry handling and should not be multiplied again by K. Most latency comes from adaptive formulation in the Planner; Solver and L2RPN interaction add only a few seconds. The comparison therefore separates computational cost from dispatch utility: optimization and RL baselines are faster but less reliable, while ADA spends more planning time to obtain higher long-horizon completion without missing the operational deadline.

Table 8: **Synchronous runtime and dispatch-time bound.** Times are seconds per decision step. Total is the realized blocking cost of Planner, Solver, the physical executability gate, and L2RPN interaction. The E2E bound is obtained from retry-enabled end-to-end tests with the same $K = 3$ and $\tau = 30$ s policy; it already includes retry handling.

Benchmark	Total/s	Planner/s	Solver/s	Env./Gate/s	E2E bound/s
IJCNN-2019	109.33	107.93	1.38	0.02	199.33
WCCI-2020	129.47	127.85	1.60	0.02	219.47
NEURIPS-2020	133.09	129.57	3.51	0.01	223.09
WCCI-2022	121.76	118.25	3.48	0.03	211.76

Table 9: **Cross-method wall-clock overhead.** WC denotes wall-clock seconds per decision step. ADA's WC is kept consistent with the module-level decomposition in Table 8. Best results are in **bold**, second-best are underlined.

Method	IJCNN-2019		WCCI-2020		NeurIPS-2020		WCCI-2022	
	Rate ↑	WC ↓	Rate ↑	WC ↓	Rate ↑	WC ↓	Rate ↑	WC ↓
MILPAgent	28.7%	**1.06**	54.6%	**1.23**	**100.0%**	**2.70**	46.2%	**2.68**
HRL	13.3%	2.46	54.8%	2.78	**100.0%**	3.05	26.8%	3.18
ReAct	51.5%	92.14	35.6%	90.37	89.5%	98.22	7.9%	176.56
ADA	**97.3%**	109.33	**100.0%**	129.47	**100.0%**	133.09	**95.0%**	121.76

Asynchronous compute budget. Table 10 reports the token budget of LLM-based modules. Planner and LLM Judger entries are per-call budgets, so total token consumption equals call count times the corresponding per-call budget. The Judger+Summarizer column is reported per 100 steps because it runs as a batched asynchronous update. Table 11 reports the corresponding wall-clock time for the asynchronous Judger+Summarizer update with four parallel workers. When triggered, Judger+Summarizer is more expensive, reflecting its role in scoring trajectories, compressing traces, removing stale knowledge, and maintaining the long-term knowledge base rather than producing the current dispatch action.

Table 10: **LLM token budget.** Planner and Judger are reported per call; Judger+Summarizer is reported per 100 steps due to asynchronous batched invocation.

Benchmark	Planner / call	Judger / call	Judger+Summarizer / 100 steps
IJCNN-2019	1034	972	236418
WCCI-2020	987	1016	242073
NEURIPS-2020	1061	948	239265
WCCI-2022	1012	1035	244801

Overall, the runtime study supports the asynchronous architecture. The latency-critical Planner–Solver path with the physical gate issues actions within the 5-minute window, while Judger scoring and Summarizer updates improve verification and long-term adaptation without blocking the current dispatch action.

Table 11: **Asynchronous Judger+Summarizer wall-clock budget.** Times are seconds per 100 dispatch steps with four parallel workers. These updates do not block the current dispatch action.

Benchmark	IJCNN-2019	WCCI-2020	NEURIPS-2020	WCCI-2022
Judger+Summarizer / 100 steps	1548.37	1647.55	1572.48	1660.92

G.3 Long-term Stability and Visualization

To further evaluate the resilience of ADA under continuous perturbations, we visualize the step-wise system load margin (ρ) trajectory in a representative long-horizon episode (Scenario `april_006`), as shown in Figure 8.

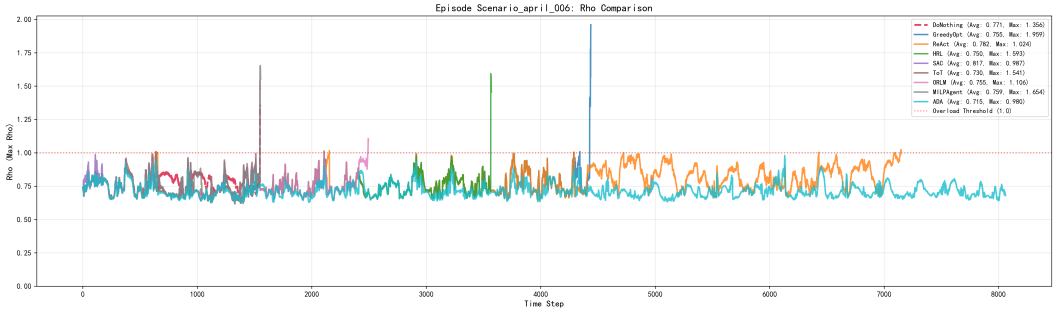


Figure 8: **Trajectory analysis of maximum line loading (ρ) across different agents over a long-horizon episode (Scenario `april_006`).** The red dashed line indicates the thermal limit threshold ($\rho = 1.0$). Baselines such as DoNothing, SAC, and GreedyOpt fail prematurely due to constraint violations. While ReAct survives longer, it exhibits high variance and operates dangerously close to the limit. In contrast, ADA maintains a stable safety corridor with the lowest average load ($\bar{\rho} = 0.715$), demonstrating superior robust control capabilities.

Trajectory result. This experiment complements the aggregate completion rates by plotting the maximum line loading over a full long-horizon episode. Most baselines terminate early or operate close to the thermal limit, indicating brittle recovery under accumulated topology and injection shifts. ReAct survives longer than other prompt-only agents but has higher variance and a larger average maximum loading ($\bar{\rho} \approx 0.782$).

Implication for ADA. ADA completes the 8000-step horizon with the lowest average maximum loading ($\bar{\rho} \approx 0.715$) and keeps the peak below the hard limit ($\max \rho = 0.980$). This suggests that the main-table gains are not only endpoint survival effects: the Planner constructs more stable optimization formulations over time, the synchronous physics gate excludes invalid or unverifiable actions, and asynchronous Judger feedback together with the evolving TK/AK memory improves subsequent decisions under persistent distribution shifts.

H Extended Ablation Studies

H.1 LLM Backbone Ablation

To isolate the sensitivity of ADA to the underlying chat model, we evaluate the same Planner-Solver-Judger-Summarizer pipeline with different OpenAI-compatible LLM backbones. All rows use the same prompts, tools, solver, physical verification, retry policy described in Appendix A.

Table 12: **LLM Backbone Ablation on L2RPN Benchmarks.** We report the same metrics as Table 1: **Rate,** ↑, **Avg. R,** ↑, and **Overload,** ↓. Best results are in **bold**, second-best are underlined.

Backbone	ijcnn-2019			wcci-2020			neurips-2020			wcci-2022		
	Rate ↑	Avg R. ↑	Over ↓	Rate ↑	Avg R. ↑	Over ↓	Rate ↑	Avg R. ↑	Over ↓	Rate ↑	Avg R. ↑	Over ↓
KIMI2.5	93.5%	14.86 ± 0.32	0.12%	96.8%	16.31 ± 0.11	0.09%	**100.0%**	15.23 ± 0.02	0.02%	89.4%	22.02 ± 0.28	0.38%
MINIMAX-M2.5	87.6%	14.72 ± 0.41	0.42%	91.2%	16.22 ± 0.16	0.25%	96.5%	15.20 ± 0.04	0.08%	82.7%	21.86 ± 0.43	1.05%
CHATGPT-5.3-CODEX	**98.9%**	$\mathbf{15.01 \pm 0.18}$	**0.01%**	**100.0%**	$\mathbf{16.38 \pm 0.05}$	**0.02%**	**100.0%**	$\mathbf{15.26 \pm 0.01}$	**0.00%**	**97.2%**	$\mathbf{22.35 \pm 0.12}$	**0.06%**
QWEN3-32B	97.3%	14.96 ± 0.23	0.03%	**100.0%**	16.35 ± 0.07	0.04%	**100.0%**	15.25 ± 0.01	**0.00%**	95.0%	22.20 ± 0.15	0.10%

H.2 Planner Ablation

To isolate whether formulation errors originate from the Planner rather than the downstream Solver or Judger, we evaluate the generated tuple $\langle \mathcal{J}, \Theta \rangle$ before optimization. The evaluation freezes the Solver/Judger interface and checks only the action family and model-construction structure produced by the Planner. **Scene–Action Match** measures the percentage of decision states for which the action family implied by $\langle \mathcal{J}, \Theta \rangle$ matches the required intervention family of the scene. For example, if a scene requires a discrete topology or line-switching intervention but the generated formulation only contains continuous load or redispatch variables, the sample is counted as a mismatch. **Formulation Validity** measures whether $\langle \mathcal{J}, \Theta \rangle$ is parseable, has consistent variable and constraint dimensions, and successfully constructs the corresponding `CVXPY` or model object. Table 13 reports this Planner-level check for direct LLM formulation, two search-only MCTS variants, and the full ADA Planner. MCTS-SHALLOW keeps the same tool set and formulation composer but reduces the search depth to $D = 3$ with $M = 100$ simulations. MCTS-UCT keeps $D = 6$ and $M = 100$ but replaces the prior-guided UCB rule with the classic UCT score $Q(\mathcal{X}, t) + c\sqrt{\log(1 + \sum_{t'} N(\mathcal{X}, t'))/(1 + N(\mathcal{X}, t))}$.

Table 13: **Planner-level formulation matching and validity.**

Method	IJCNN-2019		WCCI-2020		NeurIPS-2020		WCCI-2022	
	Match ↑	Validity ↑	Match ↑	Validity ↑	Match ↑	Validity ↑	Match ↑	Validity ↑
Direct LLM Formulation	78.4%	81.7%	64.2%	70.5%	49.8%	67.9%	38.6%	62.1%
MCTS-Shallow ($D = 3$)	92.6%	89.4%	93.1%	83.8%	68.4%	89.2%	78.7%	74.5%
MCTS-UCT	87.3%	93.2%	80.5%	90.1%	76.8%	87.6%	69.4%	83.8%
ADA Planner	**96.5%**	**98.7%**	**93.8%**	**97.4%**	**91.2%**	**96.5%**	**89.3%**	**95.2%**

H.3 LLM Judge Ablation on Calibration

We further ablate the LLM-as-a-Judge prompt while fixing the judge backbone to QWEN3-CODER from the Qwen3 family [25]. The three prompt variants differ only in calibration style: **Rubric-anchored scoring** assigns explicit 0–5 anchors to *Rationality*, *Safety*, and *Sustainability*, following customized-rubric evaluator design [28]; **Evidence-first form filling** requires the judge to write evidence and checklist fields before producing scores, following CoT and form-filling evaluation practice [29]; and **Bias-aware counterfactual consistency** anonymizes method identity, neutralizes ordering and length cues, and asks the judge to check whether the score would change under counterfactual prompt order, following recent bias analyses for LLM-as-a-Judge [30]. Table 14 reports the prompt-ablation results. *Rubric-anchored scoring*, which is the prompt used in our main results, achieves the lowest disagreement D and Future-10 loading ρ_{10} across all four benchmarks, indicating that explicit 0–5 anchors on *Rationality*, *Safety*, and *Sustainability* keep the judge tightly aligned with the physical verifier. *Evidence-first form filling* performs second-best: requiring evidence before scoring helps the judge stay grounded, but the longer CoT introduces additional variance. *Bias-aware counterfactual consistency* is consistent across re-orderings, yet anonymizing method identity removes the scenario-priority cues that the rubric relies on, leading to the highest D and the worst ρ_{10}. The gap, however, is much smaller than the α ablation in Table 3, confirming that physical verification (the α blend) dominates over prompt phrasing.

Table 14: **Prompt Ablation for LLM-as-a-Judge.** The judge model is fixed to QWEN3-CODER and the verifier weight to $\alpha = 0.5$. We report the same metrics as Table 3: disagreement rate D, Future 10 Max Line Loading (ρ_{10}), and completion Rate. Best results are in **bold**, second-best are underlined.

Prompt calibration	Judge model	ijcnn-2019			wcci-2020			neurips-2020			wcci-2022		
		$D\downarrow$	$\rho_{10}\downarrow$	Rate ↑	$D\downarrow$	$\rho_{10}\downarrow$	Rate ↑	$D\downarrow$	$\rho_{10}\downarrow$	Rate ↑	$D\downarrow$	$\rho_{10}\downarrow$	Rate ↑
Rubric-anchored scoring	QWEN3-CODER	**35.46%**	**0.81**	**97.3%**	**35.51%**	**0.82**	**100.0%**	**36.14%**	**0.85**	**100.0%**	**29.52%**	**0.84**	**95.0%**
Evidence-first form filling	QWEN3-CODER	42.18%	0.86	94.5%	41.27%	0.87	98.7%	43.55%	0.89	99.4%	37.81%	0.88	91.6%
Bias-aware counterfactual consistency	QWEN3-CODER	49.63%	0.93	90.2%	47.85%	0.91	97.1%	48.42%	0.94	98.5%	44.07%	0.95	87.4%

I Full Theoretical Assumptions and Proofs

This section formalizes the assumptions and proofs behind the two-time-scale result used in Sec. 3. The result is local: it applies to the policy class represented by the finite-horizon Planner, the finite tool set, and the TK/AK parameterization used by ADA.

Finite-horizon MCTS produces the empirical AK update target $\hat{\pi}_{\mathrm{MCTS}}$, whose conditional-mean improvement map is modeled by T_Φ. Solver–Judger returns evaluate tool chains, while the Summarizer

implements $\Delta\Phi$ by reconstructing TK templates from verified replay traces. These correspondences identify the embedded AK/TK recursion analyzed here; its local guarantees remain conditional on the stated noise, regularity, and stability assumptions. The 10/100-step schedule specifies when memories are revised and is distinct from the diminishing step-size conditions used in the proofs.

Assumption I.1 (Bounded stochastic approximation model)**.** The AK policy iterates π_k and TK parameters ϕ_k remain in compact sets Π and Φ. Planner rewards and Solver-Judger scores are uniformly bounded. The noise sequences in the AK and TK recursions are martingale differences with uniformly bounded second moments with respect to the natural filtration $\mathcal{F}_k$.

Assumption I.2 (Step sizes and time-scale separation)**.** The step sizes satisfy

$$\sum_k \alpha_k = \sum_k \beta_k = \infty, \qquad \sum_k \alpha_k^2 + \sum_k \beta_k^2 < \infty, \qquad \sum_k \frac{\beta_k^2}{\alpha_k} < \infty, \qquad \lim_{k\to\infty} \frac{\beta_k}{\alpha_k} = 0. \quad (39)$$

Assumption I.3 (Local Planner stability)**.** For every fixed TK state ϕ in a neighborhood of the operating point, the finite-horizon Planner induces an improvement map $T_\phi : \Pi \to \Pi$ with a locally unique fixed point $\pi^*(\phi)$. There exist $q \in (0,1)$ and neighborhoods $\mathcal{U}_\phi$ such that

$$\|T_\phi(\pi) - \pi^*(\phi)\| \le q\|\pi - \pi^*(\phi)\|, \qquad \pi \in \mathcal{U}_\phi. \quad (40)$$

The fixed-point map $\pi^*(\phi)$ is locally Lipschitz in ϕ.

Assumption I.4 (TK reconstruction regularity)**.** The mean-field TK reconstruction map $\bar{\Delta}_\Phi(\phi, \pi)$ is locally Lipschitz in both TK embeddings and AK policies under the metrics in Appendix F.4. The mean-field ODE

$$\dot{\phi} = g(\phi) := \bar{\Delta}_\Phi(\phi, \pi^*(\phi)) \quad (41)$$

has a locally asymptotically stable equilibrium ϕ^* in the considered neighborhood.

Proposition I.5 (Fast-scale AK tracking)**.** *Under Assumptions I.1–I.3, with ϕ_k viewed as quasi-static on the fast scale, the AK recursion tracks the local Planner fixed point:*

$$\lim_{k\to\infty} \|\pi_k - \pi^*(\phi_k)\| = 0 \quad \text{a.s.} \quad (42)$$

Proof. Fix a TK state ϕ over the fast time scale and write $h_\phi(\pi) = T_\phi(\pi) - \pi$. The AK recursion has the stochastic approximation form

$$\hat{\pi}_{\mathrm{MCTS}}(\mathcal{E}_k) = T_\phi(\pi_k) + \xi_{k+1}, \qquad \pi_{k+1} = \pi_k + \alpha_k\big(h_\phi(\pi_k) + \xi_{k+1}\big), \quad (43)$$

where $\mathbb{E}[\xi_{k+1} \mid \mathcal{F}_k] = 0$ and $\mathbb{E}[\|\xi_{k+1}\|^2 \mid \mathcal{F}_k] \le C$ by Assumption I.1. Let $V_k = \|\pi_k - \pi^*(\phi)\|^2$. Using the contraction property of T_ϕ, there exists $c > 0$ in the local neighborhood such that

$$\langle \pi_k - \pi^*(\phi), h_\phi(\pi_k)\rangle \le -c\|\pi_k - \pi^*(\phi)\|^2. \quad (44)$$

Expanding V_{k+1} and conditioning on $\mathcal{F}_k$ gives

$$\mathbb{E}[V_{k+1} \mid \mathcal{F}_k] \le (1 - 2c\alpha_k)V_k + C'\alpha_k^2. \quad (45)$$

Since $\sum_k \alpha_k = \infty$ and $\sum_k \alpha_k^2 < \infty$, the Robbins–Siegmund supermartingale convergence theorem implies $V_k \to 0$ almost surely for fixed ϕ. When ϕ_k varies, the target $\pi^*(\phi_k)$ also moves. Let $e_k = \pi_k - \pi^*(\phi_k)$. The Lipschitz continuity of $\pi^*(\phi)$ and boundedness of the TK update imply $\|\pi^*(\phi_{k+1}) - \pi^*(\phi_k)\| \le C_\phi\beta_k$. Expanding $\|e_{k+1}\|^2$ adds a moving-target perturbation bounded by Young's inequality as $c\alpha_k\|e_k\|^2 + C''\beta_k^2/\alpha_k$. Assumption I.2 makes $\sum_k \beta_k^2/\alpha_k < \infty$, so this perturbation is summable and the fixed-target Robbins–Siegmund argument carries over to the slowly moving target $\pi^*(\phi_k)$. □

Proposition I.6 (Slow-scale TK tracking)**.** *Under Assumptions I.1–I.4, the interpolated TK trajectory tracks the ODE*

$$\dot{\phi} = g(\phi) = \bar{\Delta}_\Phi(\phi, \pi^*(\phi)). \quad (46)$$

Proof. Write the slow recursion as

$$\phi_{k+1} = \phi_k + \beta_k\big(g(\phi_k) + M_{k+1} + b_k\big), \quad (47)$$

where

$$M_{k+1} = \Delta\Phi_k - \mathbb{E}[\Delta\Phi_k \mid \mathcal{F}_k] \quad (48)$$

is a martingale-difference term and

$$b_k = \bar{\Delta}_\Phi(\phi_k, \pi_k) - \bar{\Delta}_\Phi(\phi_k, \pi^*(\phi_k)) \tag{49}$$

is the transient bias from using the current AK policy rather than its fast-scale equilibrium. The martingale term is asymptotically negligible because $\sum_k \beta_k^2 < \infty$ and the second moments are bounded. By the AK-policy Lipschitz part of Assumption I.4, there is $L_\pi > 0$ such that

$$\|b_k\| \le L_\pi \|\pi_k - \pi^*(\phi_k)\|_1. \tag{50}$$

Proposition I.5 implies $\|b_k\| \to 0$ almost surely. Therefore the slow recursion is an asymptotic pseudo-trajectory of the ODE $\dot{\phi} = g(\phi)$. The ODE method for stochastic approximation then gives convergence of the interpolated TK trajectory to the internally stable invariant set of this ODE; under Assumption I.4, this set is the local equilibrium ϕ^*. □

Theorem I.7 (Local two-time-scale convergence of knowledge dynamics)**.** *Under Assumptions I.1–I.4, ADA's coupled AK/TK recursion converges almost surely to the local internally stable equilibrium*

$$(\pi_k, \phi_k) \to (\pi^*(\phi^*), \phi^*) \quad \text{a.s.} \tag{51}$$

This is a convergence claim for the embedded TK/AK knowledge-evolution dynamics, not a reward-maximization claim.

Proof. By Proposition I.5, the fast AK variable tracks $\pi^*(\phi_k)$ for the current slowly changing TK state. By Proposition I.6, the slow TK variable tracks the ODE driven by this equilibrated AK response and converges to its locally asymptotically stable equilibrium ϕ^*. The Lipschitz continuity of $\pi^*(\phi)$ then yields $\pi_k \to \pi^*(\phi^*)$ almost surely. At this pair, the AK policy is a fixed point of the finite-horizon Planner improvement operator and the TK reconstruction has zero local mean-field drift. Therefore the coupled embedded knowledge dynamics reach a locally internally stable equilibrium. This theorem does not assert that the expected Solver-Judger reward is maximized inside or outside the finite tool set, finite horizon, and local neighborhoods specified in the assumptions. □

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 ADA's two-time-scale design, active state augmentation, adaptive solver selection, theoretical convergence claim, and L2RPN evaluation scope. The experiments and limitations sections qualify these claims for the evaluated power-system dispatch settings.

   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 conclusion discusses limitations from the time-scale separation assumption, MCTS computational overhead, solver feature quality, and real-world deployment risks from probabilistic LLM outputs.

   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 against the paper, a worse outcome might be that important limitations are discovered only after submission. 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.

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: [Yes].

Justification: Sec. 3 states the regularity, finite-horizon Planner stability, and two-time-scale step-size assumptions, gives proof sketches for the fast and slow updates, and Appendix I provides the full assumptions and formal proofs for fast AK tracking, slow TK ODE tracking, transient-bias vanishing, and local two-time-scale convergence.

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 experiments section specifies the L2RPN benchmarks, reward metrics, baselines, LLM backbone, solver backends, retry budget, and five-seed reporting protocol, with additional agent configuration and reward details in the appendix.

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: Appendix A.3 provides a GitHub repository containing the ADA code; it also specifies that the benchmark data are obtained through the official Grid2Op/L2RPN channels. The same appendix and experimental details report the seeds, OpenAI-compatible API configuration, hardware, runtime budget, retry policy, and benchmark setup needed to reproduce the main results.

   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 experiments section specifies the benchmark suite, baselines, LLM configuration, solvers, retry budget, random seeds, and evaluation metrics needed to understand the reported results.

   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 main results and ablation tables report mean ± standard deviation across five random seeds, and the experimental settings state the repeated-run protocol.

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: [No].

   Justification: Appendix A.1 specifies the hardware platform and host memory, and Appendix G.2 reports per-step runtime and LLM token budgets. However, total compute for each experimental run and for the full research project is not reported.

   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 work uses public benchmark environments and focuses on decision support; the conclusion discusses human oversight, audit trails, and physical verification safeguards for operational use.

   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 introdction and conclusion summarize positive impacts for safer adaptive dispatch and negative operational risks from incorrect automated decisions, along with proposed safeguards.

    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 a pre-trained language model, image generator, scraped dataset, or similar asset with high misuse risk; deployment safeguards are discussed for the dispatch decision-support setting.

    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: [Yes].

Justification: Appendix A.3 lists the external assets and terms: Grid2Op, L2RPN baselines, and LightSim2Grid under MPL-2.0; L2RPN benchmark data under the official challenge/data access terms; and provider-hosted LLM API models under their provider terms. The paper cites the L2RPN/Grid2Op benchmarks and baseline methods and does not redistribute third-party model weights or benchmark data.

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: Appendix A.3 documents the new assets introduced by the paper: the ADA implementation, module prompts, configuration files, evaluation scripts, and processed result logs/metadata. These assets are released in the GitHub repository under MPL-2.0 with setup, API, benchmark, and limitation information; no pretrained model checkpoint, scraped dataset, or human-subject data are introduced.

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 report crowdsourcing experiments or research involving human subjects.

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 report crowdsourcing experiments or research involving human subjects, so IRB or equivalent approval is not applicable.

    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: [Yes].

    Justification: LLMs are a core component of ADA's planner, judger, and agent baselines; the methods and experiments sections describe the LLM-based modules and Qwen3-32B reasoning backbone. Gemini 3.1 Pro was used only to refine the visual presentation of Figure 4; the architecture, module relationships, labels, and scientific content were designed and verified by the authors.

    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.