LAMF-MCTS: LEARNING-AUGMENTED MULTI-FIDELITY MONTE CARLO TREE SEARCH FOR EXPENSIVE COMBINATORIAL BLACK-BOX OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Expensive stochastic simulations limit how broadly an optimizer can explore a combinatorial design space. We introduce LAMF-MCTS, a learning-augmented multi-fidelity Monte Carlo tree search framework that coordinates search and simulation effort. The key idea is to exploit information available before a simulation finishes: a partial trajectory can inform both a design's eventual performance and the direction of subsequent search. The framework combines observed partial returns with learned predictions of the remaining return, penalizing uncertain estimates. Repeatedly selected designs are progressively promoted to higher fidelities, while side information from the same trajectories provides action priors that guide tree expansion. Experiments on an application-derived combinatorial black-box optimization problem show that LAMF-MCTS attains the highest final full-fidelity objective among six application-specific planning baselines and, under the same simulator-call budget as full-fidelity MCTS, reduces cumulative simulation cost by 35.2% while attaining a higher running-best objective.

1 INTRODUCTION

Many design problems involve discrete decisions whose performance can only be evaluated through expensive stochastic simulations. This is particularly challenging when performance depends on sequential operations, random demand, congestion, or adaptive system behavior, and the objective lacks a tractable analytical form. The result is a black-box optimization problem, and the difficulty is amplified by large combinatorial design spaces and long stochastic rollouts. Under a limited computational budget, exhaustive full-fidelity evaluation is infeasible, making the allocation of simulation effort central to optimization performance. Existing black-box optimization methods improve efficiency by directing evaluations toward promising candidates (Wang et al., 2023; Malherbe et al., 2022), while multi-fidelity methods reduce evaluation cost by varying the amount of computational effort (Li et al., 2021; Poiani et al., 2024). These approaches correspond to two coupled decisions: *where to search* and *how much evaluation effort to spend*, which existing methods typically make in isolation. For sequential stochastic simulators, partial rollouts naturally connect the two decisions because they provide both lower-cost estimates of candidate quality and intermediate information that can guide subsequent search.

We study budgeted combinatorial black-box optimization with such partial simulation trajectories and propose **LAMF-MCTS**, a learning-augmented multi-fidelity Monte Carlo tree search framework. The key idea is to exploit the information that becomes available before a simulation finishes, rather than discarding it until the full rollout completes. MCTS organizes sequential design modifications, with action factorization and progressive widening controlling large branching spaces. The framework combines observed partial returns with learned predictions of the remaining return, while an uncertainty penalty discounts unreliable estimates. Candidates that remain competitive through repeated selection are progressively promoted to higher fidelities, while simulator side information is reused to guide tree expansion.

Our main contributions are threefold:

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

- We formulate a budgeted combinatorial black-box optimization setting in which candidate designs are evaluated by a stochastic sequential simulator, with fidelity levels defined by different trajectory truncation points.
- We develop **LAMF-MCTS**, a learning-augmented multi-fidelity Monte Carlo tree search framework that integrates structured tree search with adaptive simulation allocation. Its key idea is to reuse the same partial simulation trajectory for both candidate evaluation and search guidance. Observed partial returns are combined with learned predictions of the remaining return, while an uncertainty penalty discounts unreliable estimates. Side information from the same trajectories provides action priors that guide tree expansion, and repeatedly selected candidates are progressively promoted to higher fidelities.
- We evaluate LAMF-MCTS on an application-derived combinatorial black-box optimization problem under a common full-fidelity evaluation protocol. Under the same number of simulator calls, LAMF-MCTS reduces the cumulative simulation cost by 35.2% relative to full-fidelity MCTS while reaching a higher running-best objective, and attains the highest final solution quality among application-specific planning baselines. Controlled ablations further demonstrate the complementary roles of multi-fidelity evaluation, side-information guidance, and action factorization.

2 Related Work

2.1 Expensive Combinatorial Black-Box Optimization

Black-box optimization (BBO) considers objectives that are unavailable in analytical form and can only be accessed through function evaluations, making evaluation efficiency critical when queries are expensive (Wang et al., 2023). Bayesian optimization (BO) addresses this setting with surrogate models and acquisition functions, but extending BO to combinatorial and mixed-variable spaces remains challenging because of their rapidly growing search spaces and lack of continuous geometry. Recent methods therefore rely on specialized representations, embeddings, trust regions, or learned proposal mechanisms (Papenmeier et al., 2023; Dreczkowski et al., 2023; Lee et al., 2024). Recent work also shows that intermediate outputs exposed during expensive evaluations can provide useful optimization signals (Maus et al., 2023). Our setting follows this direction but focuses on stochastic sequential simulators, where partial trajectories arise naturally at different computational costs. Optimization must therefore determine both which designs to explore and how much simulation effort to allocate to each evaluation.

2.2 Tree-Based Search for Black-Box Optimization

Tree-based methods organize the search space hierarchically and allocate evaluations toward promising branches. LA-MCTS combines learned space partitioning with Monte Carlo tree search for black-box optimization (Wang et al., 2020), while subsequent methods use optimistic tree search, local stochastic search, variable-subset selection, or transferable search subspaces to improve exploration (Malherbe et al., 2022; Zhai & Gao, 2022; Song et al., 2022; Wang et al., 2024). Tree structure can also be modified directly: Kowalski et al. (2022) decompose compound moves into sequences of semimoves to reduce branching width. These methods primarily address *where to search*. In contrast, our method couples structured MCTS with partial-trajectory evaluation so that the tree search also determines *how much simulation effort* is allocated to promising candidates.

2.3 Multi-Fidelity and Partial-Evaluation Optimization

Multi-fidelity optimization reduces evaluation cost by combining expensive high-fidelity observations with cheaper approximations. In BO, candidate and fidelity selection are often optimized jointly using cross-fidelity surrogate models, while related resource-allocation and best-arm formulations address the same cost–accuracy trade-off (Li et al., 2021; Wistuba et al., 2022; Poiani et al., 2024). More recent methods consider partial evaluations of structured computations rather than fixed low- and high-fidelity models. For example, Buathong et al. (2023) perform cost-aware partial evaluation over function networks, while XING et al. (2026) model how observations evolve with increasing fidelity. These studies emphasize that low-fidelity evaluations are useful only when their reduced cost is balanced against their reliability.

3 PROBLEM FORMULATION

3.1 EXPENSIVE COMBINATORIAL BLACK-BOX OPTIMIZATION

We consider a discrete design vector $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{X}$, where $\mathbb{X}$ is a finite but potentially very large combinatorial design space. Each design incurs a deterministic cost $C(\boldsymbol{x})$ and must satisfy a prescribed design budget B_{d}. The resulting black-box optimization problem is

$$\boldsymbol{x}^\star \in \arg\max_{\boldsymbol{x}\in\mathbb{X}} f(\boldsymbol{x}) \qquad \text{s.t.} \qquad C(\boldsymbol{x}) \le B_{\mathrm{d}}, \qquad f(\boldsymbol{x}) = \mathbb{E}_\xi[R(\boldsymbol{x};\xi)] - \lambda C(\boldsymbol{x}). \tag{1}$$

Here, $R(\boldsymbol{x};\xi)$ denotes the stochastic return of design $\boldsymbol{x}$ under a random environment realization ξ, $C(\boldsymbol{x})$ is a deterministic design cost, and λ controls the trade-off between expected operational performance and design expenditure. The budget constraint determines the feasible design space, whereas the cost term distinguishes among feasible designs with different expenditure levels. We assume that f has no tractable analytical form and can only be estimated by querying an expensive stochastic simulator, which produces a realization $R(\boldsymbol{x};\xi)$ for a given ξ.

We further assume that a feasible design can be constructed sequentially from an initial design $\boldsymbol{x}_0$ through a sequence of discrete modifications,

$$\boldsymbol{x}_0 \xrightarrow{a_1} \boldsymbol{x}_1 \xrightarrow{a_2} \cdots \xrightarrow{a_J} \boldsymbol{x}_J, \qquad a_j \in \mathbb{A}(\boldsymbol{x}_{j-1}), \tag{2}$$

where $\mathbb{A}(\boldsymbol{x})$ denotes the set of feasible actions at design $\boldsymbol{x}$. Each action modifies one or more components of the current design while preserving feasibility. This sequential representation provides a structured construction of the combinatorial design space without requiring an explicit analytical model of the objective.

3.2 BUDGETED MULTI-FIDELITY EVALUATION

We consider a multi-fidelity evaluation setting in which lower-cost evaluations are obtained by truncating a sequential simulator at different stages of execution. Let $\mathbb{F} = \{m_1, m_2, \ldots, m_L, T\}$ with $m_1 < \cdots < m_L < T$ denote the set of available fidelity levels, where T corresponds to a complete simulation and each $m < T$ corresponds to an earlier truncation point. The associated evaluation costs satisfy $c_{m_1} < c_{m_2} < \cdots < c_{m_L} < c_T$. A fidelity-$m$ evaluation executes the simulator up to stage m and returns

$$\mathcal{E}_m(\boldsymbol{x};\xi) = \big(R_{1:m}(\boldsymbol{x};\xi),\, \boldsymbol{s}_m(\boldsymbol{x}),\, \boldsymbol{h}_m(\boldsymbol{x})\big), \tag{3}$$

where $R_{1:m}(\boldsymbol{x};\xi)$ denotes the accumulated partial return up to stage m, $\boldsymbol{s}_m(\boldsymbol{x})$ is the intermediate simulator state, and $\boldsymbol{h}_m(\boldsymbol{x})$ is auxiliary trajectory information extracted from the partial simulation. Optimization is performed under a finite evaluation budget. At each simulator call t, the optimizer selects both a candidate design $\boldsymbol{x}_t$ and an evaluation fidelity $m_t \in \mathbb{F}$. The budget is specified as a maximum number of simulator calls, and the objective is

$$\max f(\boldsymbol{x}) \qquad \text{s.t.} \qquad N_{\mathrm{s}} \le B_{\mathrm{e}}, \tag{4}$$

where N_{s} is the total number of simulator calls performed and $\boldsymbol{x}$ is the final candidate returned by the optimization procedure. Because different fidelity levels incur different simulation costs, we additionally report the cumulative evaluation cost $\sum_t c_{m_t}$ as an efficiency measure. This quantity is not constrained a priori; it follows directly from the fidelity-allocation policy, so two methods that consume the same number of simulator calls may incur different cumulative costs.

Lower-fidelity evaluations are less costly but provide incomplete information about full-fidelity performance, so the optimizer must jointly decide **where to search** and **how much simulation effort to allocate**. Section 4 develops LAMF-MCTS, which addresses these coupled decisions through multi-fidelity evaluation, action factorization, and side-information-guided search.

4 LEARNING-AUGMENTED MULTI-FIDELITY MONTE CARLO TREE SEARCH

We propose LAMF-MCTS, a learning-augmented multi-fidelity Monte Carlo tree search framework for the budgeted optimization problem in Eq. 4. The framework organizes the combinatorial design space as a structured search tree over sequential design modifications and integrates three

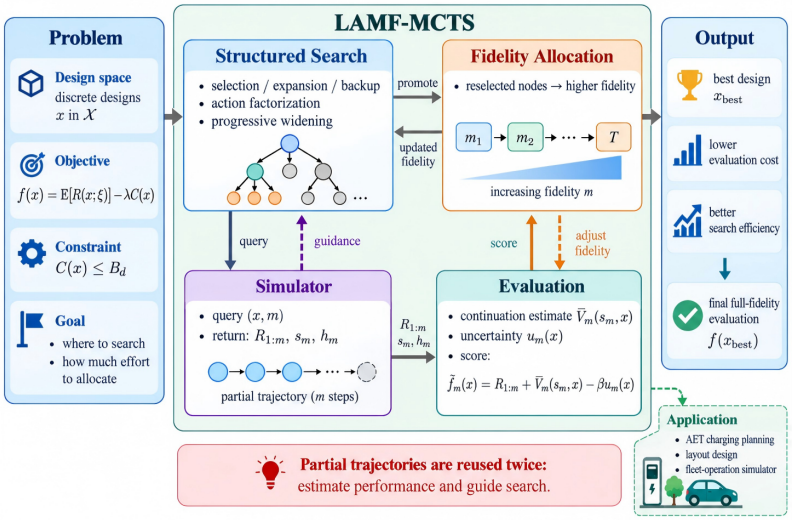

Figure 1: Overview of the proposed LAMF-MCTS framework.

key mechanisms: multi-fidelity evaluation of partially completed trajectories, action factorization of compound edits, and side-information-guided tree expansion. Figure 1 provides an overview of the complete framework. The following subsections introduce the search structure, value estimation, fidelity allocation, and search-guidance mechanisms before presenting the complete algorithm.

4.1 Structured Monte Carlo Tree Search

We use Monte Carlo tree search (MCTS) as the search backbone for the sequential design formulation in Eq. 2. Starting from the initial design $\boldsymbol{x}_0$, each iteration proceeds through the standard stages of selection, expansion, evaluation, and backup. In the resulting search tree $\mathcal{T}$, each real node corresponds to a feasible design $\boldsymbol{x}$, each edge represents an admissible edit $a \in \mathbb{A}(\boldsymbol{x})$, and a path from the root therefore corresponds to a sequence of design modifications $\boldsymbol{x}_0 \xrightarrow{a_1} \boldsymbol{x}_1 \xrightarrow{a_2} \cdots \xrightarrow{a_j} \boldsymbol{x}_j$. When a new node is reached, its value is obtained through the multi-fidelity evaluation procedure introduced in Section 4.2 and backed up along the selected path.

Selection. Let $N(\boldsymbol{x})$ denote the visit count of node $\boldsymbol{x}$, $N(\boldsymbol{x}, a)$ the visit count of action a, and $Q(\boldsymbol{x}, a)$ its running mean value. Among the currently expanded actions $\mathbb{A}_{\text{exp}}(\boldsymbol{x}) \subseteq \mathbb{A}(\boldsymbol{x})$, selection follows a prior-guided PUCT rule (Silver et al., 2016),

$$a^* = \arg \max_{a \in \mathbb{A}_{\text{exp}}(\boldsymbol{x})} \left[Q(\boldsymbol{x}, a) + c_{\text{puct}} P(a \mid \boldsymbol{x}, \boldsymbol{h}_m(\boldsymbol{x})) \frac{\sqrt{N(\boldsymbol{x})}}{1 + N(\boldsymbol{x}, a)} \right], \tag{5}$$

where $P(a \mid \boldsymbol{x}, \boldsymbol{h}_m(\boldsymbol{x}))$ is an action prior derived from simulator-side information available at node $\boldsymbol{x}$. The construction of this prior is described in Section 4.4. The prior affects the exploration preference but does not replace the value statistics accumulated by MCTS.

Structured expansion. In many combinatorial design problems, a feasible edit is a compound action $a = (i, \delta)$, where i selects a design component and $\delta \in \Delta_i(\boldsymbol{x})$ specifies an admissible modification. Treating every (i, δ) pair as an independent branch can create a large branching factor. We therefore apply *action factorization*, following the split-move principle of Kowalski et al. (2022), and decompose each compound action into successive semimoves:

$$\text{REAL}(\boldsymbol{x}) \to \text{COMPONENT}(i) \to \text{MODIFY}(\delta) \to \text{REAL}(\boldsymbol{x}'). \tag{6}$$

Intermediate nodes represent incomplete actions and are not evaluated by the simulator; only real nodes correspond to feasible designs. Visit and value statistics are nevertheless backed up through both real and semimove nodes, allowing shared prefixes to aggregate evidence across downstream modifications. This trades branching width for additional search depth.

We further control expansion using progressive widening (Couëtoux et al., 2011). At node $\boldsymbol{x}$, the number of instantiated actions satisfies

$$|\mathbb{A}_{\text{exp}}(\boldsymbol{x})| \leq k_{\text{pw}} N(\boldsymbol{x})^{\alpha}, \qquad 0 < \alpha < 1, \tag{7}$$

so additional actions are introduced only as the node receives more visits. Together, action factorization and progressive widening reduce early branching while gradually expanding the set of explored alternatives.

4.2 Multi-fidelity Value Estimation

A newly reached design does not need to be evaluated immediately at full fidelity. For a fidelity-m evaluation with $m < T$, the simulator directly observes the accumulated prefix return $R_{1:m}(\boldsymbol{x}; \xi)$, while the remaining return $R_{m+1:T}(\boldsymbol{x}; \xi)$ is unobserved, so the eventual performance must be inferred from the partial trajectory. We define the continuation value at fidelity m as

$$V_m(\boldsymbol{s}_m(\boldsymbol{x}), \boldsymbol{x}) = \mathbb{E}\left[R_{m+1:T}(\boldsymbol{x}; \xi) \mid \boldsymbol{s}_m(\boldsymbol{x}), \boldsymbol{x}\right], \tag{8}$$

which represents the expected return remaining after the simulator reaches the truncation state $\boldsymbol{s}_m(\boldsymbol{x})$ under design $\boldsymbol{x}$. Since V_m is not directly available at a truncated evaluation, we approximate it with an ensemble of K independently trained estimators $\hat{V}_m^{(k)}(\boldsymbol{s}_m(\boldsymbol{x}), \boldsymbol{x}; \boldsymbol{\theta}_V^{(k)})$, $k = 1, \ldots, K$. Each estimator is trained offline from completed trajectories: each full trajectory provides, for every truncation point m, an input $(\boldsymbol{x}, \boldsymbol{s}_m(\boldsymbol{x}))$ together with the realized remaining return $R_{m+1:T}$, and the k-th estimator is obtained by minimizing the squared error over its training data $\mathcal{D}_V^{(k)}$,

$$\boldsymbol{\theta}_V^{(k)} = \arg\min_{\boldsymbol{\theta}} \frac{1}{|\mathcal{D}_V^{(k)}|} \sum_{(\boldsymbol{x}, \boldsymbol{s}_m(\boldsymbol{x}), R_{m+1:T}) \in \mathcal{D}_V^{(k)}} \left[\hat{V}_m^{(k)}(\boldsymbol{s}_m(\boldsymbol{x}), \boldsymbol{x}; \boldsymbol{\theta}) - R_{m+1:T}\right]^2, \qquad k = 1, \ldots, K. \tag{9}$$

Their mean prediction is used as the continuation estimate,

$$\bar{V}_m(\boldsymbol{s}_m(\boldsymbol{x}), \boldsymbol{x}) = \frac{1}{K} \sum_{k=1}^{K} \hat{V}_m^{(k)}\left(\boldsymbol{s}_m(\boldsymbol{x}), \boldsymbol{x}; \boldsymbol{\theta}_V^{(k)}\right). \tag{10}$$

This formulation is agnostic to the particular predictive model used for the continuation estimators; the optimization framework only requires them to map a partial simulation state and candidate design to an estimate of the remaining return. Because the continuation term is inferred rather than directly observed, it may be inaccurate or overly optimistic for trajectories that are difficult to predict. We therefore associate each low-fidelity evaluation with a nonnegative uncertainty score $u_m(\boldsymbol{x}; \xi)$ and define the conservative performance estimate as

$$\tilde{f}_m(\boldsymbol{x}; \xi) = R_{1:m}(\boldsymbol{x}; \xi) + \bar{V}_m(\boldsymbol{s}_m(\boldsymbol{x}), \boldsymbol{x}) - \beta u_m(\boldsymbol{x}; \xi) - \lambda C(\boldsymbol{x}), \tag{11}$$

where $\beta \geq 0$ controls the strength of the uncertainty penalty. At full fidelity, $m = T$, the continuation term is no longer required and $\tilde{f}_T(\boldsymbol{x}; \xi) = R_{1:T}(\boldsymbol{x}; \xi) - \lambda C(\boldsymbol{x})$. The construction of u_m and its role in fidelity allocation are introduced next.

4.3 Uncertainty-Aware Value Estimation and Progressive Fidelity Allocation

The continuation estimate is learned from previously observed trajectories and can therefore become unreliable when the partial trajectory is difficult to predict or the queried design lies outside the training distribution. To quantify this reliability, we construct the uncertainty score $u_m(\boldsymbol{x}; \xi)$ from three complementary signals.

First, *predictive disagreement* measures dispersion among the continuation-value estimators:

$$d_m^{\text{pred}}(\boldsymbol{x}) = \sqrt{\frac{1}{K-1} \sum_{k=1}^{K} \left[\hat{V}_m^{(k)}\left(\boldsymbol{s}_m(\boldsymbol{x}), \boldsymbol{x}; \boldsymbol{\theta}_V^{(k)}\right) - \bar{V}_m(\boldsymbol{s}_m(\boldsymbol{x}), \boldsymbol{x})\right]^2}. \tag{12}$$

Second, *temporal-consistency residual* measures disagreement between predicted continuation values and the observed partial trajectory:

$$d_m^{\text{res}}(\boldsymbol{x}) = \frac{1}{m-1} \sum_{t=1}^{m-1} \left| r_{t+1} + \bar{V}_{t+1}(\boldsymbol{s}_{t+1}(\boldsymbol{x}), \boldsymbol{x}) - \bar{V}_t(\boldsymbol{s}_t(\boldsymbol{x}), \boldsymbol{x}) \right|, \tag{13}$$

where r_{t+1} denotes the stage return observed between simulator stages t and $t+1$.

Third, *design-distribution distance* measures extrapolation from the training designs:

$$d^{\mathrm{ood}}(\boldsymbol{x}) = \min_{\boldsymbol{x}' \in \mathcal{D}_V^x} \|\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')\|_2 , \tag{14}$$

where $\phi(\boldsymbol{x})$ is a normalized design representation and $\mathcal{D}_V^x$ is the set of training designs.

Because these signals have different scales, we normalize them using a held-out calibration set and combine them as

$$u_m(\boldsymbol{x}; \xi) = w_{\mathrm{pred}} \bar{d}_m^{\mathrm{pred}} + w_{\mathrm{res}} \bar{d}_m^{\mathrm{res}} + w_{\mathrm{ood}} \bar{d}^{\mathrm{ood}}, \tag{15}$$

where $w_{\mathrm{pred}}, w_{\mathrm{res}}, w_{\mathrm{ood}} \geq 0$. Larger values indicate less reliable continuation estimates and are penalized in Eq. 11.

Progressive fidelity allocation. Rather than evaluating every candidate at full fidelity, we progressively increase the fidelity of a design as it is revisited by MCTS. Let $N_{\mathrm{e}}(\boldsymbol{x})$ denote the number of previous simulator evaluations of node $\boldsymbol{x}$, and let $m_{L+1} \equiv T$ for notational convenience. Its next evaluation uses

$$m(\boldsymbol{x}) = m_{\min\{N_{\mathrm{e}}(\boldsymbol{x})+1, L+1\}}. \tag{16}$$

A newly expanded node is therefore evaluated at m_1, while repeatedly selected nodes progress toward T. This allocates expensive simulation effort to candidates that remain competitive under the current search policy. Once full fidelity is reached, the result is cached and reused.

4.4 Side-Information-Guided Search

Besides the partial return, a simulator trajectory exposes auxiliary information about the evaluated design and its intermediate behavior. We reuse this *side information* to guide subsequent tree expansion. Let $H(a \mid \boldsymbol{x}, \boldsymbol{h}_m(\boldsymbol{x}))$ denote a heuristic score for feasible action $a \in \mathbb{A}(\boldsymbol{x})$, with larger values indicating stronger support from the observed trajectory. We convert these scores into a normalized action prior,

$$P(a \mid \boldsymbol{x}, \boldsymbol{h}_m(\boldsymbol{x})) = \frac{\exp\left(H(a \mid \boldsymbol{x}, \boldsymbol{h}_m(\boldsymbol{x}))/\tau\right)}{\sum_{a' \in \mathbb{A}_{\exp}(\boldsymbol{x})} \exp\left(H(a' \mid \boldsymbol{x}, \boldsymbol{h}_m(\boldsymbol{x}))/\tau\right)}, \tag{17}$$

where $\tau > 0$ controls the concentration of the prior. The prior is used in the PUCT rule in Eq. 5, biasing expansion toward actions supported by simulator evidence. The application-specific construction of H is described in Appendix A.3.

Thus, partial trajectories are reused in two complementary ways: they support low-fidelity value estimation and uncertainty assessment, while their side information guides where the tree expands next.

4.5 Overall Algorithm

Algorithm 1 summarizes the complete procedure. Each iteration traverses the tree via the prior-guided PUCT rule, expands it under progressive widening with factorized compound actions, and evaluates the reached real node at the fidelity determined by its evaluation count. Low-fidelity outcomes are converted into the uncertainty-adjusted score $\tilde{f}_m$ and backed up through all real and semimove nodes on the path, and the newly obtained side information updates the action prior for subsequent expansion. Full-fidelity scores are cached and reused without additional simulation.

5 Experiments

The experiments are organized to evaluate optimization efficiency under a limited simulation budget, assess performance on an application-derived instance, and identify the contributions of the main algorithmic components.

Algorithm 1 Learning-Augmented Multi-Fidelity Monte Carlo Tree Search

Require: Initial design $\boldsymbol{x}_0$; simulation budget B_{e} (number of simulator calls); fidelity set $\mathbb{F} = \{m_1, \ldots, m_L, T\}$ with $m_{L+1} \equiv T$; continuation model $\bar{V}$;
Ensure: Best full-fidelity design $\boldsymbol{x}^\star$

```
Initialize search tree 𝒯 with root x_0
Initialize evaluation counts N_e(x) = 0 and simulator calls N_s = 0
while N_s < B_e do
    Select a path in 𝒯 using the prior-guided PUCT rule
    Expand the tree when allowed by progressive widening, using action factorization for compound actions
    Let x be the reached real design node
    if x has a cached full-fidelity evaluation then
        y ← f̃_T(x)
    else
        m ← m_{min{N_e(x)+1, L+1}}
        Run the simulator at fidelity m to obtain (R_{1:m}, s_m(x), h_m(x))
        if m < T then
            Estimate the continuation value V̄_m(s_m(x), x)
            Compute uncertainty score u_m(x; ξ)
            y ← R_{1:m} + V̄_m(s_m, x) − β u_m(x; ξ) − λ C(x)
        else
            y ← R_{1:T} − λ C(x)
            Cache the full-fidelity score of x
        end if
        Store h_m(x) and update the side-information-guided action prior
        N_e(x) ← N_e(x) + 1;   N_s ← N_s + 1
    end if
    Back up y along the complete selected path, updating visit counts and value statistics
end while
Let 𝒳_T be the set of designs evaluated at full fidelity
x* ← arg max_{x ∈ 𝒳_T} f̃_T(x)
return x*
```

5.1 Experimental Setup

Application setting. We evaluate LAMF-MCTS on an automated electric taxi charging-infrastructure planning problem, where each feasible infrastructure design is assessed by a stochastic sequential fleet-operation simulator. Truncating the simulator provides the accumulated return, intermediate state, and trajectory side information, directly instantiating the multi-fidelity setting in Section 3.

Multi-fidelity evaluation and search budget. The full simulation horizon contains $T = 32$ steps, with fidelity levels $\mathbb{F} = \{8, 16, 24, 32\}$. We measure evaluation cost relative to the lowest fidelity:

$$\bar{c}_m = \frac{c_m}{c_{m_1}} = \frac{m}{m_1}, \tag{18}$$

so that the four fidelities have normalized costs 1, 2, 3, and 4, respectively. All search-based methods use the same budget of 10,000 MCTS simulation calls. Their cumulative evaluation costs may nevertheless differ because the fidelity assigned to each evaluation varies across methods.

Evaluation protocol and metrics. The final design returned by each method is independently re-evaluated under 20 full-fidelity stochastic demand scenarios. We report the resulting mean objective and empirical distribution as measures of final solution quality. Search efficiency is evaluated using the running-best search-time objective and cumulative normalized evaluation cost. The post-optimization evaluations are therefore kept separate from the search-time estimates and provide a common full-fidelity basis for comparing the returned designs.

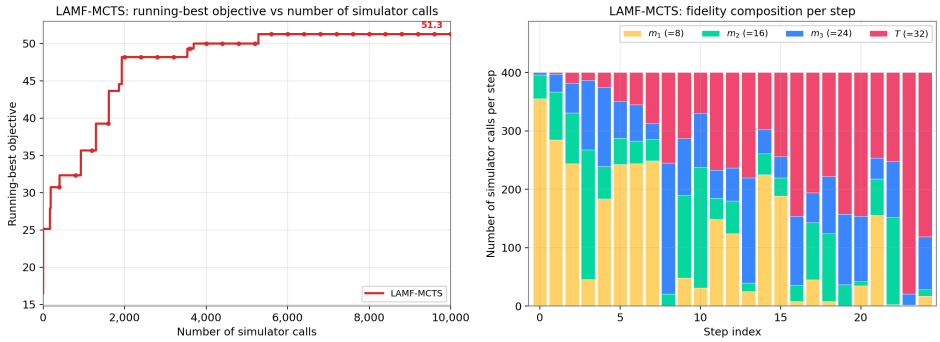

Figure 2: Search behavior of the complete LAMF-MCTS. The left panel reports the running-best objective, and the right panel the composition of simulator calls across fidelity levels.

5.2 Main Optimization Performance

Search behavior. Figure 2 shows the search trajectory of LAMF-MCTS and its fidelity allocation. The running-best objective improves rapidly and reaches 51.3 by the end of the search. Low-fidelity evaluations are used extensively during exploration, while higher fidelities become more prominent as candidates are repeatedly selected. This pattern is consistent with the progressive fidelity-allocation mechanism in Section 4.3, which concentrates additional simulation effort on competitive candidates.

Comparison methods. We compare LAMF-MCTS with representative charging-infrastructure planning baselines from the transportation literature. C-MILP follows a clustering-based location-and-sizing paradigm (Ameer et al., 2025), while FC-MILP adopts a flow-capturing formulation based on origin–destination flows (Upchurch et al., 2009). RO applies scenario-based robust optimization to taxi charging-network planning (Godbersen et al., 2024); its two variants, RO(SA-0.9) and RO(VA-0.9), respectively require 90% scenario-level feasibility and 90% vehicle-level feasibility within each scenario. SO uses a two-stage stochastic integer program for joint infrastructure planning and fleet operations (Chen & Liu, 2023), whereas ACPM combines an agent-based charging-demand simulator with random-embedding Bayesian optimization (Liu et al., 2024). Together, these baselines cover deterministic, robust, stochastic, and simulation-based planning paradigms.

Final solution quality. We evaluate the final designs using the common post-optimization protocol in Section 5.1. These results are distinct from the search-time running-best values: for example, 51.3 is the best objective observed during LAMF-MCTS search, whereas final solution quality is obtained by re-evaluating the returned design over 20 independent full-fidelity stochastic demand scenarios. Figure 3 reports the resulting distributions. LAMF-MCTS achieves the highest mean objective, 48.13 ± 2.19, followed by RO(SA-0.9) with 42.33 ± 2.89. The remaining baselines obtain lower mean objectives. These results show that LAMF-MCTS can identify high-quality infrastructure designs without an explicit analytical representation of the operational objective.

5.3 Analysis and Ablations

We conduct controlled ablations to isolate the main components of LAMF-MCTS. All variants use the same search horizon of 10,000 simulator calls, organized as 25 search steps with 400 calls per step. Because these calls may use different fidelities, cumulative normalized evaluation costs can differ. Figure 4 therefore compares both the running-best objective and cumulative cost $\sum_t \bar{c}_{m_t}$ defined in Eq. 18.

We compare LAMF-MCTS with **full-fidelity MCTS**, **w/o side-information guidance**, and **w/o action factorization**. After 10,000 calls, LAMF-MCTS reaches a running-best objective of 51.3 versus 50.2 for full-fidelity MCTS, while reducing cumulative evaluation cost from 40,000 to 25,924, a 35.2% reduction. Removing side-information guidance or action factorization further reduces the cost to 21,947 and 17,840, respectively, but lowers the running-best objective to 49.1 and 48.1. These

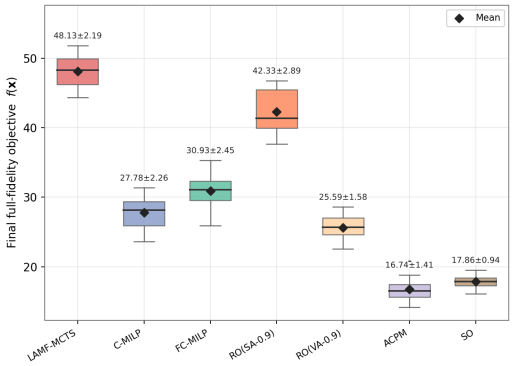

Figure 3: Distribution of final full-fidelity objective values over independent stochastic demand scenarios. Each box reports the mean ± standard deviation, and diamonds indicate the means.

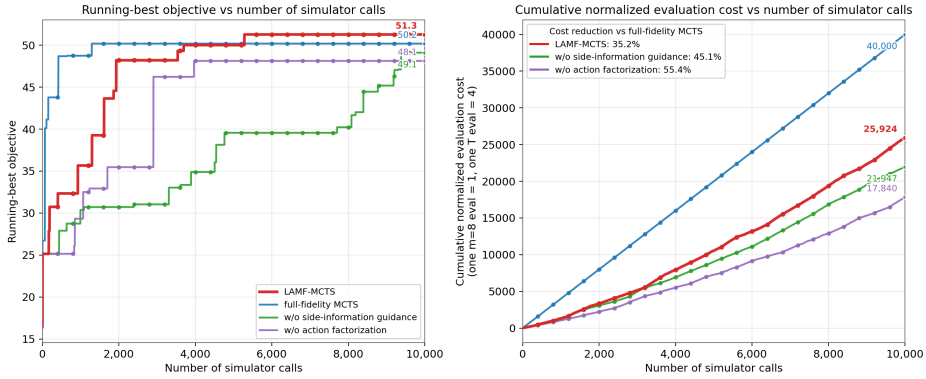

Figure 4: Controlled ablation study under the same simulator-call budget. The left panel reports the running-best objective and the right panel the cumulative normalized evaluation cost, both as functions of the number of simulator calls.

results indicate complementary roles: multi-fidelity evaluation reduces candidate-assessment cost, while side-information guidance and action factorization improve how the available search effort is directed and structured.

6 Conclusion and Limitations

We studied expensive combinatorial black-box optimization with stochastic sequential simulation and proposed **LAMF-MCTS**, which jointly coordinates structured tree search and simulation-effort allocation. The method uses partial trajectories for low-fidelity value estimation, progressively promotes promising candidates to higher fidelities, and incorporates side information to guide expansion. On an application-derived combinatorial design problem, LAMF-MCTS outperforms application-specific planning baselines in final solution quality and reduces cumulative evaluation cost by 35.2% relative to full-fidelity MCTS while achieving a higher running-best objective. Ablations confirm the complementary roles of multi-fidelity evaluation, side-information guidance, and action factorization.

The effectiveness of LAMF-MCTS depends on partial trajectories being sufficiently informative about full-fidelity performance. Weak fidelity correlation, unreliable continuation estimates, or strong interactions among multiple design variables may reduce the benefit of early evaluation and sequential search. Future work may therefore consider more robust continuation estimation and fidelity allocation, as well as broader classes of simulation-based combinatorial optimization problems.

AI USE STATEMENT

In this work, we used generative AI tools to assist with research ideation and execution, including brainstorming and refining methodological ideas, providing feedback on experimental design, and assisting with code development and implementation. We also used generative AI tools for literature retrieval and discovery, including identifying potentially relevant work and summarizing literature during the research process. In addition, generative AI tools were used to aid and polish the writing of the manuscript, including improving clarity, wording, organization, and presentation.

All AI-assisted research content was independently reviewed and verified by the authors. Relevant literature suggested by AI tools was checked against the original sources before citation; methodological and mathematical content was manually examined by the authors; and AI-assisted code was executed, tested, and validated before being used in the reported experiments. Experimental results and scientific conclusions were obtained and assessed by the authors rather than accepted directly from AI-generated outputs. We take responsibility for the final content of this work, including text, claims, code, and other artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have taken several steps to facilitate reproducibility. The problem formulation, objective, multi-fidelity evaluation setting, and the complete LAMF-MCTS procedure are described in Sections 3 and 4, including the search, value-estimation, uncertainty, fidelity-allocation, and side-information mechanisms. The application-specific design representation, action space, budget definition, stochastic fleet-operation simulator, and multi-fidelity simulation interface are detailed in Appendix A. The main algorithmic hyperparameters, search protocol, evaluation procedure, and computational environment are reported in Appendix B. These materials are intended to make the experimental setup and reported results reproducible without relying on undocumented implementation choices.

REFERENCES

Hamza Ameer, Yujie Wang, and Zonghai Chen. A density-based spatial clustering and linear programming method for electric vehicle charging station location and price optimization. *Energy*, 317:134581, 2025.

Poompol Buathong, Jiayue Wan, Raul Astudillo, Samuel Daulton, Maximilian Balandat, and Peter I Frazier. Bayesian optimization of function networks with partial evaluations. *arXiv preprint arXiv:2311.02146*, 2023.

Yao Chen and Yang Liu. Integrated optimization of planning and operations for shared autonomous electric vehicle systems. *Transportation Science*, 57(1):106–134, 2023.

Adrien Couëtoux, Jean-Baptiste Hoock, Nataliya Sokolovska, Olivier Teytaud, and Nicolas Bonnard. Continuous upper confidence trees. In *International conference on learning and intelligent optimization*, pp. 433–445. Springer, 2011.

Kamil Dreczkowski, Antoine Grosnit, and Haitham Bou Ammar. Framework and benchmarks for combinatorial and mixed-variable bayesian optimization. *Advances in Neural Information Processing Systems*, 36:69464–69489, 2023.

Gregor Godbersen, Rainer Kolisch, and Maximilian Schiffer. Robust charging network planning for metropolitan taxi fleets. *Transportation Science*, 58(2):295–314, 2024.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Jakub Kowalski, Maksymilian Mika, Wojciech Pawlik, Jakub Sutowicz, Marek Szykuła, and Mark HM Winands. Split moves for monte-carlo tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 10247–10255, 2022.

Deokjae Lee, Hyun Oh Song, and Kyunghyun Cho. Training greedy policy for proposal batch selection in expensive multi-objective combinatorial optimization. *arXiv preprint arXiv:2406.14876*, 2024.

Shibo Li, Robert Kirby, and Shandian Zhe. Batch multi-fidelity bayesian optimization with deep auto-regressive networks. *Advances in Neural Information Processing Systems*, 34:25463–25475, 2021.

Xiaozhuang Li, Xindi Tang, and Fang He. Few-shot learning for dynamic operations of automated electric taxi fleets under evolving charging infrastructure: A meta-deep reinforcement learning approach. *arXiv preprint arXiv:2601.21312*, 2026.

Yuechen Sophia Liu, Mohammad Tayarani, Fengqi You, and H Oliver Gao. Bayesian optimization for battery electric vehicle charging station placement by agent-based demand simulation. *Applied Energy*, 375:123975, 2024.

Cedric Malherbe, Antoine Grosnit, Rasul Tutunov, Haitham Bou Ammar, and Jun Wang. Optimistic tree searches for combinatorial black-box optimization. *Advances in Neural Information Processing Systems*, 35:33080–33092, 2022.

Natalie Maus, Zhiyuan Jerry Lin, Maximilian Balandat, and Eytan Bakshy. Joint composite latent space bayesian optimization. *arXiv preprint arXiv:2311.02213*, 2023.

Leonard Papenmeier, Luigi Nardi, and Matthias Poloczek. Bounce: Reliable high-dimensional bayesian optimization for combinatorial and mixed spaces. *Advances in Neural Information Processing Systems*, 36:1764–1793, 2023.

Riccardo Poiani, Rémy Degenne, Emilie Kaufmann, Alberto Maria Metelli, and Marcello Restelli. Optimal multi-fidelity best-arm identification. *Advances in Neural Information Processing Systems*, 37:121882–121927, 2024.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Lei Song, Ke Xue, Xiaobin Huang, and Chao Qian. Monte carlo tree search based variable selection for high dimensional bayesian optimization. *Advances in Neural Information Processing Systems*, 35:28488–28501, 2022.

Christopher Upchurch, Michael Kuby, and Seow Lim. A model for location of capacitated alternative-fuel stations. *Geographical Analysis*, 41(1):85–106, 2009.

Linnan Wang, Rodrigo Fonseca, and Yuandong Tian. Learning search space partition for black-box optimization using monte carlo tree search. *Advances in Neural Information Processing Systems*, 33:19511–19522, 2020.

Shukuan Wang, Ke Xue, Lei Song, Xiaobin Huang, and Chao Qian. Monte carlo tree search based space transfer for black box optimization. *Advances in Neural Information Processing Systems*, 37:49591–49624, 2024.

Xilu Wang, Yaochu Jin, Sebastian Schmitt, and Markus Olhofer. Recent advances in bayesian optimization. *ACM computing surveys*, 55(13s):1–36, 2023.

Martin Wistuba, Arlind Kadra, and Josif Grabocka. Supervising the multi-fidelity race of hyperparameter configurations. *Advances in Neural Information Processing Systems*, 35:13470–13484, 2022.

WEI XING, Zhenjie Lu, and Akeel Shah. Camo: Convergence-aware multi-fidelity bayesian optimization. *Advances in Neural Information Processing Systems*, 38:143047–143088, 2026.

Yaoguang Zhai and Sicun Gao. Monte carlo tree descent for black-box optimization. *Advances in Neural Information Processing Systems*, 35:12581–12593, 2022.

A Application and Simulator Details

This appendix describes how the generic combinatorial black-box optimization problem in Section 3 is instantiated for automated electric taxi (AET) charging-infrastructure planning. The upper-level design specifies the charging capacity deployed across candidate regions, while its operational value is evaluated through a stochastic fleet-operation simulator controlled by a learned meta-reinforcement learning policy.

A.1 Charging-Infrastructure Design

Let $\mathcal{I}$ denote the set of candidate regions in the service area. A charging-infrastructure design is represented by

$$\boldsymbol{x} = (x_i)_{i \in \mathcal{I}}, \tag{19}$$

where x_i denotes the number of charging piles deployed in region i. The feasible capacity levels are discretized as

$$x_i \in \{0, 10, 20, 30, 40, 50\}. \tag{20}$$

Hence, $x_i = 0$ indicates that no charging station is deployed in region i, whereas a positive value specifies both the presence of a station and its charging capacity. Although the physical capacity can be interpreted as an integer number of charging piles between 0 and 50, the planning problem uses increments of 10 to obtain a structured discrete design space suitable for tree search.

The infrastructure design is subject to a real monetary investment budget,

$$C(\boldsymbol{x}) \le B_{\mathrm{d}}, \tag{21}$$

where B_{d} denotes the available charging-infrastructure budget. The design cost $C(\boldsymbol{x})$ includes the annualized capital cost of constructing the charging infrastructure together with the corresponding operating cost. This budget is distinct from the simulation-call budget B_{e} used by LAMF-MCTS during optimization.

A.2 Design Actions

LAMF-MCTS constructs candidate layouts through local modifications of the current design. In the application, the primitive operations are Add, Remove, Inc, and Dec. These operations correspond to opening a charging station at a previously inactive candidate region, removing an existing station, increasing its number of charging piles, and decreasing its number of charging piles, respectively. More specifically, Add activates an empty candidate region at the minimum positive capacity level, Remove closes an existing station, and Inc/Dec change its capacity by one discrete level.

Under the discretization above, capacity modifications are made in increments of 10. Thus, the compound action can be represented as

$$a = (i, \delta), \tag{22}$$

where i specifies the candidate region and δ specifies the type of infrastructure modification. This provides the application-specific instantiation of the action-factorization mechanism described in Section 4.

A.3 Heuristic Score for Action Guidance

This subsection describes the rule-based construction of the heuristic score $H(a \mid \boldsymbol{x}, \boldsymbol{h}_m(\boldsymbol{x}))$ used by the action prior in Section 4.4. In the AET application, the score is computed from the demand realized in the partial trajectory $\boldsymbol{h}_m(\boldsymbol{x})$, the current design $\boldsymbol{x}$, and the static travel distances of the problem instance. The first ingredient is the normalized demand heat of region i,

$$g_i = \frac{\mathrm{pickup}_i + \mathrm{dropoff}_i}{2N_{\mathrm{ord}}}, \tag{23}$$

where pickup_i and $\mathrm{dropoff}_i$ denote the numbers of orders whose origins and destinations, respectively, fall in region i along the partial trajectory realized so far, and N_{ord} is the total number of such orders. Based on g_i, the utilization pressure of candidate site j is defined as

$$\rho_j = \frac{g_j}{\max(\varepsilon,\, x_j)}, \tag{24}$$

that is, the demand heat of the region containing site j relative to the currently deployed capacity x_j, with $\varepsilon = 10^{-3}$ preventing division by zero; ρ_j characterizes how underserved the site is. The second ingredient is the travel distance d_{ij} between region i and candidate site j; when this distance matrix is not provided, the corresponding bonus is set to zero. The third ingredient is the spatial coverage score of candidate site j, which aggregates demand over all regions with a distance-decayed kernel,

$$S(j) = \sum_{i \in \mathcal{I}} g_i \exp(-d_{ij}/\sigma), \tag{25}$$

where $\sigma = 1.0$ controls the effective radiation radius: nearby high-demand regions contribute strongly to $S(j)$, whereas distant ones decay exponentially.

Table 1 lists the action-dependent bonus $H(a \mid \boldsymbol{x}, \boldsymbol{h}_m(\boldsymbol{x}))$ for each action type, where ΔC denotes the change in design cost $C(\boldsymbol{x})$ induced by the action and $x_{\min}$ is the minimum positive capacity level.

Table 1: Rule-based action bonus $H(a \mid \boldsymbol{x}, \boldsymbol{h}_m(\boldsymbol{x}))$ used in the heuristic score.

Action	Bonus $H(a \mid \boldsymbol{x}, \boldsymbol{h}_m(\boldsymbol{x}))$
ADD$(j, x_{\min})$ (open a new station)	$x_{\min}\, S(j) - \kappa_{\text{add}} \max(0, \Delta C)$
INC(j, δ) (add piles)	$\delta\, S(j) + \eta\, \rho_j - \kappa_{\text{inc}} \max(0, \Delta C)$
DEC(j, δ) (remove piles)	$-\delta\, S(j) - \eta\, \rho_j + \kappa_{\text{save}} \max(0, -\Delta C)$
REMOVE(j) (close a station)	$-x_j\, S(j) - \eta\, \rho_j + \kappa_{\text{save}} \max(0, -\Delta C)$
STOP (terminate the construction)	0

The bonus follows an intuitive symmetry. ADD and INC are rewarded for increasing capacity at locations with high demand coverage, and INC is additionally rewarded at high-pressure sites so that capacity is added where it is most needed. Conversely, DEC and REMOVE are penalized for reducing coverage or pressure relief at valuable sites but are rewarded for saving expenditure through κ_{save}. The cost regularizers $\kappa_{\text{add}}, \kappa_{\text{inc}}, \kappa_{\text{save}}$ are all set to 10^{-4}; their magnitude is deliberately small so that the cost term never dominates the geometric coverage and pressure terms.

A.4 FLEET-OPERATION SIMULATOR

Each candidate infrastructure design $\boldsymbol{x}$ is evaluated through a stochastic AET fleet-operation simulator. The operating horizon covers eight hours, from 14:00 to 22:00, and is divided into 15-minute decision periods, resulting in 32 strategic periods per simulation episode. Within each episode, vehicle movements, passenger-service decisions, relocation, and charging operations evolve dynamically subject to the charging infrastructure specified by $\boldsymbol{x}$.

The fleet size and passenger-request process are constructed from real-world mobility data. Passenger requests are generated according to empirically estimated origin–destination–time (ODT) distributions, which constitute the main source of stochasticity in the simulator. Let ξ denote one realization of this stochastic request process. For a fixed design $\boldsymbol{x}$ and realization ξ, the simulator generates a complete operational trajectory and its associated cumulative return $R(\boldsymbol{x}; \xi)$.

Fleet operations are controlled by the GAT-PEARL meta-reinforcement learning controller proposed by Li et al. (2026). Given a charging-infrastructure design $\boldsymbol{x}$, the controller adapts its operational decisions to the corresponding infrastructure configuration and produces a high-quality fleet-operation policy. The resulting operational return is used as the simulation-based evaluation of the infrastructure design. Accordingly, the upper-level objective takes the form

$$f(\boldsymbol{x}) = \mathbb{E}_\xi[R(\boldsymbol{x}; \xi)] - \lambda C(\boldsymbol{x}), \tag{26}$$

where the expectation is taken over stochastic passenger-demand realizations and $C(\boldsymbol{x})$ accounts for infrastructure expenditure.

A.5 MULTI-FIDELITY SIMULATION INTERFACE

The sequential structure of the fleet simulator enables partial evaluations without modifying the underlying operational model. For a fidelity level $m < T$, the simulation is terminated after the first

m steps and returns

$$\mathcal{E}_m(\boldsymbol{x};\xi) = \left(R_{1:m}(\boldsymbol{x};\xi), \boldsymbol{s}_m, \boldsymbol{h}_m\right), \tag{27}$$

where $R_{1:m}$ is the accumulated operational return up to the truncation point, $\boldsymbol{s}_m$ is the corresponding fleet-system state, and $\boldsymbol{h}_m$ contains auxiliary trajectory information exposed by the simulator. A full-fidelity evaluation executes the complete eight-hour episode.

These truncated trajectories constitute the multi-fidelity information used by LAMF-MCTS. The partial return and simulator state are used for continuation-value estimation, while the associated side information is reused to guide subsequent tree expansion. Hence, different fidelity levels correspond to different truncation points of the same operational simulator, rather than to separate low- and high-fidelity models.

B Hyperparameters and Reproducibility Details

This appendix summarizes the main hyperparameters and implementation settings used in the experiments. Application-specific simulator details are provided in Appendix A. Unless otherwise stated, the same configuration is used for the complete LAMF-MCTS method and its ablated variants, with only the component under investigation being modified.

B.1 Search and Multi-Fidelity Configuration

Table 2 summarizes the main search hyperparameters. The search starts from an empty infrastructure design, represented by the all-zero design vector. Each optimization run consists of 25 search steps, with 400 MCTS simulations performed per step, resulting in 10,000 MCTS simulations in total. Each MCTS simulation corresponds to at most one simulator call, so the same simulator-call budget $B_{\mathrm{e}} = 10{,}000$ is used for all methods and all ablated variants. The search terminates when this call budget is exhausted.

Table 2: Search and multi-fidelity hyperparameters used by LAMF-MCTS.

Parameter	Description	Value
c_{puct}	PUCT exploration coefficient	30
k_{pw}	Progressive-widening coefficient	2
α	Progressive-widening exponent	0.5
τ	Side-information prior temperature	1
β	Uncertainty-penalty coefficient	0.05
w_{pred}	Predictive-disagreement weight	2
w_{res}	Temporal-residual weight	1
w_{ood}	Design-OOD weight	10^{-3}
$\mathbb{F}$	Fidelity levels	$\{8, 16, 24, 32\}$
T	Full-fidelity horizon	32 periods
—	MCTS simulations per search step	400
—	Number of search steps	25
—	Total MCTS simulations	10,000
—	Initial design	$\mathbf{0}$

The full-fidelity horizon contains $T = 32$ periods, corresponding to the eight-hour operating window from 14:00 to 22:00 with 15-minute intervals. The four fidelity levels therefore correspond to truncating the same simulation after 2, 4, 6, and 8 hours, respectively. A newly evaluated node is first queried at $m = 8$ and is progressively promoted through $m = 16$, $m = 24$, and $m = 32$ when it continues to be selected by MCTS. Once a full-fidelity result has been obtained, it is cached and reused in subsequent visits.

B.2 Continuation-Value and Uncertainty Implementation

In the application, the learned continuation estimator is instantiated using the pretrained critic networks from the GAT-PEARL controller. The fleet-operation controller follows a hierarchical multi-agent architecture, in which area-level agents are trained using Soft Actor-Critic (SAC) (Haarnoja

et al., 2018). The trained critics are reused during infrastructure search as continuation-value estimators, avoiding additional online training during LAMF-MCTS optimization.

The number of continuation estimators is set to the number of area-level agents, which equals the number of candidate regions,

$$K = |\mathcal{I}|, \tag{28}$$

so that each area-level SAC critic contributes one continuation estimate. Their predictions are aggregated according to the continuation-value construction in Section 4.2.

B.3 Evaluation Protocol

All methods are evaluated under the same application setting and design budget. The design budget B_{d} corresponds to the actual monetary budget available for charging-infrastructure deployment, while the search is terminated after 10,000 MCTS simulations.

Because passenger demand is stochastic, each returned infrastructure design is independently evaluated under 20 stochastic demand scenarios. Each scenario corresponds to a complete full-fidelity simulation with an independent realization of the origin–destination–time demand process. The reported objective is the mean over these 20 realizations, and the corresponding distributions are used to construct the box plots in Section 5.2. The same set of evaluation scenarios is used across methods to ensure a common post-optimization comparison.

For the ablation experiments, all variants use the same 10,000 MCTS simulations. Their cumulative normalized evaluation costs may differ because the variants allocate these simulations across different fidelity levels. Accordingly, search quality and cumulative simulation cost are reported separately rather than assuming that one MCTS simulation has the same computational cost at every fidelity.

B.4 Computational Environment

All experiments are implemented in Python 3.10.18 and executed on an Ubuntu 22.04 server equipped with dual Intel Xeon Platinum 8581C CPUs and two NVIDIA RTX PRO 5000 Blackwell GPUs. The learning-based components are implemented with PyTorch 2.7.1, while Gurobi 13.0.0 is used to solve the mathematical-programming benchmark models.