EVOMIDFL: AGENTIC SELF-EVOLVING DECISION-FOCUSED LEARNING FOR MIXED-INTEGER PROGRAMMING

Anonymous authors
Paper under double-blind review

ABSTRACT

Decision-focused learning (DFL) embeds optimization problems into learning architectures and trains using downstream decision loss rather intermediate prediction errors alone. In mixed-integer DFL (MIDFL), however, the discontinuous prediction-to-decision map obstructs informative gradient propagation. Existing approaches typically rely on a fixed approximate backward mechanism with hand-tuned configurations, limiting the adaptation across problems, scales and leaving potentially useful gradient combinations unexplored. Recent advances in agentic evolution offer a promising way to search and refine mechanisms through successive evolution. To this end, we propose EVOMIDFL (Evolutionary Mixed-Integer Decision-Focused Learning), an agentic self-evolving MIDFL framework that adaptively discovers, configures, and composes surrogate backward-gradient policies across diverse settings. EVOMIDFL employs a validation-guided actor–critic workflow supported by a three-layer memory architecture: the Gradient Actor proposes gradient estimators, combinations, and configurations; the Critic Agent diagnoses failures and filters unsafe candidates; and the Memory Agent stores formulation-level experience, mechanism-level evidence, and transferable skills to guide subsequent evolution. Retrieved policies are further refined on the target task to mitigate negative transfer. Extensive experiments show that EVOMIDFL not only improves downstream decision quality over fixed MIDFL backward methods, but also discovers reusable policies across scales and scenarios.

1 INTRODUCTION

Decision-focused learning (DFL) has been widely used for decision-making under uncertainty. DFL integrates downstream optimization into the learning architecture and trains the neural network directly through task-specific errors to improve decision quality (Elmachtoub & Grigas, 2022; Schiffer et al., 2026). However, extending DFL to mixed-integer optimization remains challenging as the mapping from continuously predicted coefficients to mixed-integer decisions is generally discontinuous and non-differentiable. Consequently, the decision Jacobian required by standard backpropagation may be unavailable or uninformative, obstructing direct gradient propagation through the optimization layer. Existing studies address this difficulty by constructing approximate backward gradients through several families of surrogate mechanisms, including straight-through estimators (STE) (Bengio et al., 2013; Sahoo et al., 2023), black-box differentiation (Vlastelica Pogančić et al., 2020), and perturbation-based methods (Duchi et al., 2015; Berthet et al., 2020; Blondel et al., 2020). Nevertheless, the effectiveness of a backward mechanism can vary with the optimization structure, problem scale, decision sensitivity and etc. For example, STEs are computationally inexpensive but may ignore important optimizer structure, whereas perturbation-based methods are sensitive to their smoothing distributions, perturbation scales, and hyperparameter settings.

Recent open-ended agentic evolution systems demonstrate the potential of LLM-driven outer loops for proposing, evaluating, and refining executable strategies through feedback (Lu et al., 2024; Qu et al., 2026; He et al., 2026). Unlike a fixed, manually specified search procedure, an agentic system can diagnose observed failures, retrieve relevant experience, and adapt strategies and parameters over successive evolution rounds (Novikov et al., 2025). Despite this potential, the application of agentic evolution to DFL, particularly evolution of backward mechanisms, remains underexplored.

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

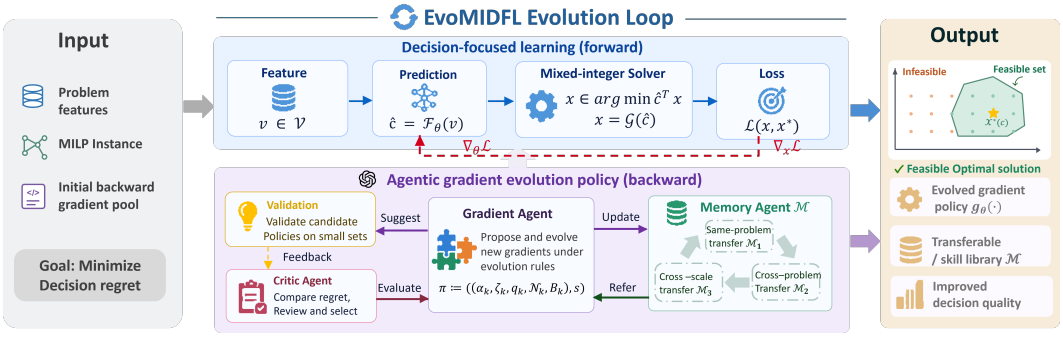

Figure 1: Conceptual architecture of EvoMIDFL. Actor–critic LLM agents iteratively propose, validate, and refine backward-gradient policies to improve downstream decision quality.

To this end, we propose EvoMIDFL (Evolutionary Mixed-Integer Decision-Focused Learning), an agentic self-evolution framework tailored for decision-focused learning with mixed-integer optimization (MIDFL). As illustrated in Figure 1, EvoMIDFL retains the standard forward path from prediction to solver-based optimization and decision loss, while introducing a validation-guided agentic loop into backward pass. Rather than relying on a fixed estimator, EvoMIDFL searches over a bounded gradient-policy space that controls the selection, configuration, stabilization, and composition of candidate backward mechanisms. A role-specialized actor–critic workflow navigates the search space: the Gradient Actor proposes candidate policies using diagnosed failures and retrieved experience, while the Critic Agent evaluates policies and filters unsafe candidates before validation. In addition, a three-layer memory architecture organizes evolution knowledge as formulation-level policy-trial experience, mechanism-level evidence, and transferable skills, allowing previously validated knowledge to guide subsequent evolution, while continued refinement on the target task helps revise retrieved policies and mitigate negative transfer. Through iterative diagnosis, proposal, validation, and promotion, EvoMIDFL progressively discovers and refines backward gradient policies across different problem settings. Our main contributions are threefold:

- We propose EvoMIDFL (Evolutionary Mixed-Integer Decision-Focused Learning), an agentic self-evolution framework for MILPs that adaptively generates backward gradients.
- EvoMIDFL automatically configures, stabilizes, and composes compatible backward mechanisms. The embedded three-layer memory architecture organizes formulation-level experience, mechanism-level evidence, and transferable skills for within- and cross-problem adaptation.
- Extensive experiments demonstrate that EvoMIDFL achieves superior final decision quality compared with fixed MIDFL backward methods across diverse scales and scenarios.

2 Related Work

Learning and Differentiation for Mixed-Integer Programming Machine learning for mixed-integer programming broadly follows two directions: learning-augmented optimization and end-to-end (E2E) decision learning. Learning-augmented methods improve components of conventional solvers; recent examples learn variable-selection policies for branch-and-bound (Strang et al., 2025), cut-generating functions for integer programming (Cheng & Basu, 2024), and neighborhood-selection strategies for large-scale MILPs (Ye et al., 2025). By contrast, E2E methods incorporate mixed-integer optimization, or a proxy for its decisions, into a trainable pipeline. Representative approaches include differentiable optimization layers, such as MIPaaL (Ferber et al., 2020), which differentiates through the optimality conditions of a cutting-plane-tightened continuous relaxation (Amos & Kolter, 2017); learning-to-optimize methods that predict solutions and apply integer-correction or feasibility-improvement procedures (Tang et al., 2025); and surrogate-backward methods that preserve the discrete forward optimization interface while approximating its backward gradients. The surrogate-backward methods include straight-through and identity-based estimators (Bengio et al., 2013; Sahoo et al., 2023), black-box differentiation (Vlastelica Pogančić et al., 2020), zeroth-order finite differences (Duchi et al., 2015), and stochastic perturb-and-optimize methods (Berthet et al., 2020; Blondel et al., 2020). EvoMIDFL advances the surrogate-backward

paradigm by automatically searching a bounded gradient-policy space over selection, configuration, stabilization, and composition of diverse mechanisms from multiple backward families.

Self-Evolving Agents Self-evolving agents iteratively propose, evaluate, and refine executable artifacts while retaining feedback to guide subsequent iterations. Early agent systems established foundations in tool use, reflection, evolution and persistent memory (Yao et al., 2022; Shinn et al., 2023; Wang et al., 2023). More recent systems use executable feedback to evolve programs, reward functions, heuristics, evaluation harnesses, and agent implementations (Romera-Paredes et al., 2024; Ma et al., 2024; Lu et al., 2024; Novikov et al., 2025; Zhang et al., 2026). Related work further emphasizes evaluator separation, persistent memory, and bounded evolution in parameter search (Qu et al., 2026; Nie et al., 2026). EvoMIDFL brings this paradigm to MIDFL through a validation-guided actor–critic loop that evolves gradient policies within a bounded search space: failure diagnoses and retrieved experience guide policy proposals, validation decision quality governs promotion, and a three-layer memory retains the resulting evidence for subsequent evolution.

3 EvoMIDFL: Evolutionary Mixed-Integer Decision-Focused Learning

3.1 Problem Formulation

EvoMIDFL is an self-evolution framework for discovering adaptive surrogate-backward gradient policies for mixed-integer decision-focused learning. EvoMIDFL retains the standard predictor-to-optimizer forward pass and evolves backward gradients at scheduled evolution rounds.

Definition 1 (Mixed-Integer Decision-Focused Learning)**.** *The MIDFL pipeline consists of a parametric prediction layer $\mathcal{F}_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^n$ with trainable parameters $\boldsymbol{\theta} \in \mathbb{R}^p$ and a downstream MILP that takes the predicted coefficients as parameters and returns a decision $\widehat{\mathbf{x}} \in \mathbb{R}^n$. Empirical loss function $\mathcal{L}$ minimizes the expected decision regret. The entire MIDFL framework is defined as:*

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & \mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(\mathbf{v},\mathbf{c})\sim\mathcal{D}} \left[\mathbf{c}^\top \mathcal{G}\big(\mathcal{F}_{\boldsymbol{\theta}}(\mathbf{v})\big) - \mathbf{c}^\top \mathbf{x}^\star(\mathbf{c}) \right], \\ \textit{s.t.} \quad & \widehat{\mathbf{c}} = \mathcal{F}_{\boldsymbol{\theta}}(\mathbf{v}), \qquad \widehat{\mathbf{x}} = \mathcal{G}(\widehat{\mathbf{c}}) \in \arg\min_{\mathbf{x}\in\mathcal{X}} \widehat{\mathbf{c}}^\top \mathbf{x}, \\ & \mathcal{X} = \{ \mathbf{x} \in \mathbb{Z}^{n_i} \times \mathbb{R}^{n_c} : \mathbf{A}\mathbf{x} \le \mathbf{b},\ \underline{\mathbf{x}} \le \mathbf{x} \le \overline{\mathbf{x}} \}, \end{aligned} \tag{1}$$

where $\mathbf{v} \in \mathbb{R}^d$ denotes the input features, $\mathbf{c} \in \mathbb{R}^n$ and $\widehat{\mathbf{c}} \in \mathbb{R}^n$ are the true coefficient and corresponding prediction. The dataset $\mathcal{D} = \{(\mathbf{v}_i, \mathbf{c}_i)\}_{i=1}^N$ contains N samples. The decision dimension is $n = n_i + n_c$, where n_i and n_c denote the numbers of integer and continuous variables, respectively.

Backward Pass To update the parameters $\boldsymbol{\theta}$ via gradient-based optimization, the chain rule gives:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L} = \mathcal{J}_{\mathcal{F}_{\boldsymbol{\theta}}}(\mathbf{v})^\top \underbrace{\mathcal{J}_{\mathcal{G}}(\widehat{\mathbf{c}})^\top \nabla_{\widehat{\mathbf{x}}} \mathcal{L}}_{\text{coefficient-space gradient}} = \mathcal{J}_{\mathcal{F}_{\boldsymbol{\theta}}}(\mathbf{v})^\top \mathbf{g}_{\mathbf{c}}, \tag{2}$$

where $\mathbf{g}_{\mathbf{c}} \in \mathbb{R}^n$ is the exact coefficient-space gradient, $\mathcal{J}_{\mathcal{G}}(\widehat{\mathbf{c}}) = \partial \mathcal{G}(\widehat{\mathbf{c}})/\partial \widehat{\mathbf{c}}$, $\mathcal{J}_{\mathcal{F}_{\boldsymbol{\theta}}}(\mathbf{v}) = \partial \mathcal{F}_{\boldsymbol{\theta}}(\mathbf{v})/\partial \boldsymbol{\theta}$. As the optimizer map of an MILP is generally nondifferentiable and cannot provide a direct decision Jacobian, existing methods construct approximate coefficient-space gradients from two categories: methods in $\mathcal{K}_{\mathrm{VJP}}$ approximate the vector–Jacobian product (VJP), and methods in $\mathcal{K}_{\mathrm{SUR}}$ differentiate a surrogate objective $\widetilde{\ell}^{(k)}$. The approximation $\widetilde{\mathbf{g}}_{\mathbf{c}}^{(k)}$ of exact gradient $\mathbf{g}_{\mathbf{c}}$ is defined as follows:

$$\widetilde{\mathbf{g}}_{\mathbf{c}}^{(k)} = \begin{cases} \mathrm{Approx}^{(k)} \left[\mathcal{J}_{\mathcal{G}}(\widehat{\mathbf{c}})^\top \nabla_{\widehat{\mathbf{x}}} \mathcal{L} \right], & k \in \mathcal{K}_{\mathrm{VJP}}, \\ \partial_{\widehat{\mathbf{c}}} \widetilde{\ell}^{(k)}(\widehat{\mathbf{c}}; \mathbf{c}), & k \in \mathcal{K}_{\mathrm{SUR}}. \end{cases} \tag{3}$$

EvoMIDFL Backward Pass Rather than relying on a single fixed backward gradient approach, EvoMIDFL defines a gradient policy π that searches over structured combinations of backward methods and method-specific hyperparameter configurations. In particular, EvoMIDFL constructs:

$$\widetilde{\mathbf{g}}_{\mathbf{c}}^{\pi} := \phi_\pi \left(\left\{ \mathbf{g}_{\mathbf{c}}^{(k)}(\boldsymbol{\eta}_k^\pi) \right\}_{k \in \mathcal{I}(\pi)} \right), \qquad \mathcal{I}(\pi) \subseteq \mathcal{K}_{\mathrm{VJP}} \cup \mathcal{K}_{\mathrm{SUR}}, \tag{4}$$

where ϕ_π denotes the policy-controlled composition of π, $\boldsymbol{\eta}_k^\pi$ denotes the configuration of mechanism k, $\mathcal{I}(\pi)$ is the active mechanism set. $\widetilde{\mathbf{g}}_{\mathbf{c}}^{\pi}$ is then mapped to $\boldsymbol{\theta}$ via $\widehat{\nabla_{\boldsymbol{\theta}} \mathcal{L}}^{\pi} := \mathcal{J}_{\mathcal{F}_{\boldsymbol{\theta}}}(\mathbf{v})^\top \widetilde{\mathbf{g}}_{\mathbf{c}}^{\pi} \in \mathbb{R}^p$.

3.2 AGENTIC EVOLUTION PIPELINE

This section introduces the training pipeline in EVOMIDFL, which uses $\widetilde{\mathbf{g}}_{\mathbf{c}}^{\pi}$ for backward. EVOMIDFL iteratively discovers and refines gradient policies π during training. We distinguish two time scales: the outer training epochs indexed by $t \in \mathcal{T} := \{1, \ldots, T\}$ and the inner evolution rounds indexed by $r \in \mathcal{R} := \{1, \ldots, M-1\}$. At training epoch E_t, model parameters are updated from $\boldsymbol{\theta}_t$ to $\boldsymbol{\theta}_{t+1}$ using the policy π_r. The evolution round R_r is triggered every τ epoch to update the policy from π_r to π_{r+1}. The parameter τ denotes the interval between consecutive evolution rounds.

Outer Training Epoch Pipeline At each outer training epoch E_t, EVOMIDFL uses the current gradient policy π_r to construct the backward gradient and update $\boldsymbol{\theta}_t$. The epoch-level pipeline is represented schematically as $\boldsymbol{\theta}_t \to \hat{c} \to x^*(\hat{c}) \to \mathcal{L} \to \tilde{g}_{\pi_r} \to \boldsymbol{\theta}_{t+1}$. Evolution round R_r updates π_r to π_{r+1}, which is then used during the next epoch block. The total number of epoch blocks is $M := \lceil T/\tau \rceil$. The overall epoch-level training pipeline follows the sequence as:

$$\underbrace{E_1, E_2, \ldots, E_\tau}_{\text{training with } \pi_1} \to \boxed{R_1} \to \underbrace{E_{\tau+1}, \ldots, E_{2\tau}}_{\text{training with } \pi_2} \to \boxed{R_2} \to \cdots \to \underbrace{E_{(M-1)\tau+1}, \ldots, E_T}_{\text{training with } \pi_M}. \tag{5}$$

Inner Gradient Policy Evolution Round At each scheduled evolution round R_r, $r \in \mathcal{R}$ is triggered after epoch E_t, EVOMIDFL freezes the current training state as checkpoint S_t and executes the agentic gradient-policy evolution workflow to obtain the next policy π_{r+1}. The workflow comprises two Critic Agents, C_1 and C_2, the Memory Agent M, the Gradient Actor A, a validation evaluator V, and a promotion operator U. Given the current policy π_r, memory $\mathcal{M}_r$, and backward-mechanism pool $\mathcal{H}$, the workflow updates the gradient policy, memory, and evaluation records during the backward pass. One evolution round proceeds schematically as:

$$(S_t, \pi_r, \mathcal{M}_r) \xrightarrow{\mathsf{C}_1} \mathcal{F}_r \xrightarrow{\mathsf{M}} \mathcal{K}_r \xrightarrow{\mathsf{A}} \Delta_r \xrightarrow{\mathsf{C}_2} \Pi_r \xrightarrow{\mathsf{V}} \mathcal{Z}_r \xrightarrow{\mathsf{U}} (\pi_{r+1}, \mathcal{M}_{r+1}), \tag{6}$$

where $\mathcal{F}_r$ contains diagnosed failures, $\mathcal{K}_r$ contains retrieved experience and skills, Δ_r is the set of proposed policy candidates, $\Pi_r \subseteq \Delta_r$ contains candidates satisfying safety conditions, and $\mathcal{Z}_r$ contains both accepted and rejected records. The promotion operator U promotes an accepted candidate or retains π_r when no candidate satisfies the acceptance criterion.

Evolution Workflow First, the pre-proposal Critic Agent diagnoses the active policy π_r using the validation evidence stored in $\mathcal{M}_r$ and the frozen training-state checkpoint S_t, producing a set of failure examples $\mathcal{F}_r$. The Memory Agent then retrieves relevant trials and skills $\mathcal{K}_r$ from the previous memory state $\mathcal{M}_r$. Conditioned on the diagnosed failures $\mathcal{F}_r$ and retrieved experience $\mathcal{K}_r$, the Gradient Actor proposes a set of candidate gradient policies Δ_r over the backward-mechanism pool $\mathcal{H}$. The post-proposal Critic Agent subsequently checkes the proposed policies against predefined conditions $\mathcal{P}_r^{\text{safe}}$, yielding $\Pi_r := \Delta_r \cap \mathcal{P}_r^{\text{safe}} = \{\pi_{r,1}, \ldots, \pi_{r,N_r}\}$, $N_r := |\Pi_r|$. Each candidate in Π_r is initialized from the same frozen checkpoint S_t and independently probe-trained for H_p epochs under identical evaluation settings. The resulting validation regret and gradient diagnostics are also collected in $\mathcal{Z}_r$ to support fair comparison. Then, the best-performing candidate is compared with parent π_r, and the policy with lower egret is selected as π_{r+1} in the subsequent training stage. The Memory Agent records the evolution trace $(\mathcal{F}_r, \mathcal{K}_r, \Delta_r, \Pi_r, \mathcal{Z}_r, \pi_{r+1})$ and updates memory $\mathcal{M}_{r+1}$.

3.3 GRADIENT POLICY REPRESENTATION

Gradient Policy Representation At evolution round r, a gradient policy π_r specifies how backward mechanisms are configured, stabilized, and composed to construct a coefficient-space backward gradient. Let $\mathcal{H} := \{h_k\}_{k=1}^K$ denote the backward mechanism pool containing K mechanisms. For each $k \in [K]$, π specifies the selection-and-mixing weight α_k, estimator-specific parameters $\boldsymbol{\zeta}_k$, allocation of backward-query budgets q_k, normalization operator $\mathcal{N}_k$, per-mechanism clipping threshold B_k, and global gradient-scaling factor s. We omit the round index r and write π as:

$$\pi := \Big(\big(\alpha_k, \boldsymbol{\zeta}_k, q_k, \mathcal{N}_k, B_k\big)_{k\in[K]}, s \Big). \tag{7}$$

Candidate Gradient Production Given a gradient policy π, the gradient $\widetilde{\mathbf{g}}^{\pi}_{\mathbf{c},i}$ is constructed in three stages: raw-gradient production, stabilization, and composition. For each instance i, each mechanism h_k first generates a raw gradient $\mathbf{u}^{\pi}_{\mathbf{c},k,i}$, which is then normalized and clipped to the stabilized gradient $\mathbf{g}^{\pi}_{\mathbf{c},k,i}$ and combined into $\widetilde{\mathbf{g}}^{\pi}_{\mathbf{c},i}$. The process $\mathbf{u}^{\pi}_{\mathbf{c},k,i} \to \mathbf{g}^{\pi}_{\mathbf{c},k,i} \to \widetilde{\mathbf{g}}^{\pi}_{\mathbf{c},i}$ is defined as:

$$\mathbf{u}^{\pi}_{\mathbf{c},k,i} := h_k(\mathbf{c}_i, \mathbf{x}_i, \widehat{\mathbf{c}}_i, \widehat{\mathbf{x}}_i, \nabla_{\widehat{\mathbf{x}}_i}\mathcal{L}_i; \boldsymbol{\zeta}_k, q_k) \in \mathbb{R}^n,\ k \in \mathcal{I}(\pi), \tag{8}$$

$$\mathbf{g}^{\pi}_{\mathbf{c},k,i} := \mathrm{Clip}_{B_k}\left[\mathcal{N}_k\big(\mathbf{u}^{\pi}_{\mathbf{c},k,i}\big)\right] \in \mathbb{R}^n,\ k \in \mathcal{I}(\pi), \tag{9}$$

$$\widetilde{\mathbf{g}}^{\pi}_{\mathbf{c},i} := s\sum_{k=1}^{K} \alpha_k \mathbf{g}^{\pi}_{\mathbf{c},k,i} \in \mathbb{R}^n. \tag{10}$$

Gradient Combination Conditions Unconstrained gradient combination may amplify gradient magnitudes or cause directional cancellation, leading to unstable training. We therefore impose four conditions for a gradient policy to be considered valid and eligible for validation.

Definition 2 (Compliant Gradient Policy)**.** *Define the active-mechanism set as*

$$\mathcal{I}(\pi) := \{k \in [K] : \alpha_k > 0\}. \tag{11}$$

For instance i, a policy π is condition-compliant if it satisfies candidate mechanism validity $\mathsf{V}(\pi)$, simplex-and-scale validity $\mathsf{S}(\pi)$, directional compatibility $\mathsf{C}_i(\pi)$, and numerical stability $\mathsf{T}_i(\pi)$. The safe set $\mathcal{P}_i^{\mathrm{safe}}$ is:

$$\mathcal{P}_i^{\mathrm{safe}} := \{\pi : \mathsf{V}(\pi) \wedge \mathsf{S}(\pi) \wedge \mathsf{C}_i(\pi) \wedge \mathsf{T}_i(\pi)\}. \tag{12}$$

Therefore, a candidate is eligible for validation only if all four conditions hold; otherwise, it is excluded and the parent policy remains the valid mechanism.

Candidate mechanism validity $\mathsf{V}(\pi)$***.*** *Every active mechanism must belong to the restricted candidate mechanism pool and its configuration must lie in the corresponding valid configuration domain:*

$$\mathsf{V}(\pi):\quad \{h_k : k \in \mathcal{I}(\pi)\} \subseteq \mathcal{H},\quad \boldsymbol{\eta}^{\pi}_k \in \Omega_k,\quad \forall k \in \mathcal{I}(\pi), \tag{13}$$

where $\boldsymbol{\eta}^{\pi}_k$ is the complete mechanism configuration and Ω_k is the valid configuration domain of h_k.

Simplex-and-scale validity $\mathsf{S}(\pi)$***.*** *The mixed weights are nonnegative and sum to one, and the global scaling factor lies in a bounded positive interval:*

$$\mathsf{S}(\pi):\quad \boldsymbol{\alpha} \in \Delta_K,\quad \Delta_K := \left\{\boldsymbol{\alpha} \in \mathbb{R}^K_+ : \sum_{k=1}^{K}\alpha_k = 1\right\},\quad 0 < s_{\min} \le s \le s_{\max}. \tag{14}$$

Directional compatibility $\mathsf{C}_i(\pi)$***.*** *Each pair of active stabilized gradients must satisfy:*

$$\mathsf{C}_i(\pi):\qquad \cos\big(\mathbf{g}^{\pi}_{\mathbf{c},k,i}, \mathbf{g}^{\pi}_{\mathbf{c},\ell,i}\big) \ge \rho,\quad \forall\, k,\ell \in \mathcal{I}(\pi),\ k \ne \ell,\quad \rho \in [0,1]. \tag{15}$$

Numerical stability $\mathsf{T}_i(\pi)$***.*** *Each active gradient must be nonzero, finite and norm-bounded:*

$$\mathsf{T}_i(\pi):\quad 0 < \varepsilon \le \left\|\mathbf{g}^{\pi}_{\mathbf{c},k,i}\right\|_2 \le G_{\max} < \infty,\quad \forall\, k \in \mathcal{I}(\pi). \tag{16}$$

3.4 Theoretical Analysis

The four gradient-policy conditions provide two guarantees for composed backward gradients: bounded composition and non-cancellation.

Proposition 1 (Bounded simplex gradient composition)**.** *Fix training epoch t and instance i. Suppose every active stabilized gradient $\mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i}$ satisfies $\|\mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i}\|_2 \le G_k < \infty\ \forall k \in \mathcal{I}(\pi_r)$. For $\boldsymbol{\alpha}_r \in \Delta_K$ and $0 < s_{\min} \le s_r \le s_{\max} < \infty$, define $\widetilde{\mathbf{g}}^{\pi_r}_{\mathbf{c},t,i} := s_r \sum_{k=1}^{K} \alpha_{r,k}\mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i}$, then the composed gradient is bounded by $s_{\max}$ times the largest candidate-gradient bound, as follows:*

$$\left\|\widetilde{\mathbf{g}}^{\pi_r}_{\mathbf{c},t,i}\right\|_2 \le s_r \sum_{k\in\mathcal{I}(\pi_r)} \alpha_{r,k}\left\|\mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i}\right\|_2 \le s_{\max}\sum_{k\in\mathcal{I}(\pi_r)} \alpha_{r,k}G_k \le s_{\max}\max_{k\in\mathcal{I}(\pi_r)} G_k. \tag{17}$$

Proposition 2 (Non-cancellation under cosine compatibility)**.** *Fix training epoch t and instance i, and let $\mathcal{I}(\pi_r) := \{k \in [K] : \alpha_{r,k} > 0\}$. Suppose every active stabilized gradient satisfies*

$$\left\| \mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i} \right\|_2 \geq m > 0, \qquad \forall k \in \mathcal{I}(\pi_r), \tag{18}$$

and every distinct active pair satisfies cosine directional compatibility:

$$\frac{\left\langle \mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i}, \mathbf{g}^{\pi_r}_{\mathbf{c},t,\ell,i} \right\rangle}{\left\| \mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i} \right\|_2 \left\| \mathbf{g}^{\pi_r}_{\mathbf{c},t,\ell,i} \right\|_2} \geq \rho, \quad \rho \in [0,1], \quad \forall k \neq \ell \in \mathcal{I}(\pi_r). \tag{19}$$

If $\boldsymbol{\alpha}_r \in \Delta_K$ and $0 < s_{\min} \leq s_r$, then the composed gradient $\widetilde{\mathbf{g}}^{\pi_r}_{\mathbf{c},t,i}$ satisfies:

$$\left\| \widetilde{\mathbf{g}}^{\pi_r}_{\mathbf{c},t,i} \right\|_2^2 \geq s_{\min}^2 m^2 \left[\rho + (1-\rho) \sum_{k \in \mathcal{I}(\pi_r)} \alpha_{r,k}^2 \right] > 0. \tag{20}$$

Remarks on Propositions Proposition 1 establishes a finite upper bound on the norm of the composed surrogate gradient, preventing the composition from arbitrarily amplifying gradient magnitudes. Proposition 2 provides a strictly positive lower bound under the directional-compatibility and numerical-stability conditions, thereby preventing destructive cancellation among gradient directions. Together, the two propositions ensure that combining individually controlled gradients produces a bounded, nonvanishing backward gradient. The Gradient Actor can therefore explore different estimators, configurations, mixture weights, and compositions subject to these conditions.

3.5 Policy Promotion and Transfer

Gradient Policy Promotion Let Π_r denote safe candidate policies in evolution round r, and define $\mathcal{Q}_r^{\text{safe}} := \{\pi_r\} \cup \Pi_r$ containing the parent policy and all safe candidates. For a fair comparison, each policy in $\mathcal{Q}_r^{\text{safe}}$ is evaluated from the same frozen checkpoint using the same validation instances, validation horizon, and settings. The validation regret and gradient diagnostics are recorded in $\mathcal{Z}_r$. Policies are ranked according to validation regret, and the best-performing policy is selected as:

$$\pi_r^\star \in \arg\min_{\pi \in \mathcal{Q}_r^{\text{safe}}} \text{Reg}_r(\pi), \tag{21}$$

where parent policy π_r remains if the results are equivalent. The selected candidate $\pi_r^\star$ is promoted only if it improves upon the parent policy by at least the acceptance margin $\delta \geq 0$. If no candidate satisfies this condition, the parent policy is retained. The outcomes of both promoted and rejected candidates are recorded in memory to guide subsequent evolution. The promotion is defined as:

$$\pi_{r+1} := \begin{cases} \pi_r^\star, & \text{Reg}_r(\pi_r^\star) \leq \text{Reg}_r(\pi_r) - \delta, \\ \pi_r, & \text{otherwise.} \end{cases} \tag{22}$$

Three-Layer Memory The Memory Agent maintains $\mathcal{M}_r = \{\mathcal{M}_r^{(1)}, \mathcal{M}_r^{(2)}, \mathcal{M}_r^{(3)}\}$, where $\mathcal{M}_r^{(\ell)}$ denotes records in Layer ℓ at the start of evolution round r. Layer 1 stores formulation-specific policy trials, including checkpoint S_t, diagnosed failures $\mathcal{F}_r$, policy changes Δ_r, validation evidence $\mathcal{Z}_r$, and promotion decisions. Layer 2 summarizes related Layer-1 trials into mechanism-level evidence about which policy changes succeed or fail under particular problem conditions. Layer 3 contains transferable skills distilled from Layer-2 and validated across problem scales or formulations. Each round appends a Layer-1 trial, which may update Layer 2 and, after successful transfer validation, Layer 3. Failed transfers are retained as negative evidence with reduced retrieval priority.

4 Experiments

To evaluate EvoMIDFL, we organize experiments around four research questions: *(1) Does the agentic policy evolution improve downstream decision quality over predict-then-optimize (PTO) and*

fixed backward MIDFL methods? (2) Can the Gradient Policy transfer across problem scales and formulations? (3) How does decision quality vary across different levels of prediction errors? (4) How does the Gradient Policy change across evolution rounds? We evaluate EvoMIDFL on two mixed-integer optimization problems: bipartite matching and knapsack. This section presents the aggregated comparisons and evolution evidence supporting our main findings. Detailed implementation settings and additional evaluation results are provided in Appendices D and E, respectively.

Proposed-Method Variants The proposed EvoMIDFL includes four variants distinguished by whether utilizing memory during training and whether enabling policy evolution during training:

- *-Fixed:* starts without memory, keeps the gradient policy fixed, and disables memory retrieval.
- *-Evo:* starts without memory and evolves the gradient policy using only evidence during training.
- *-Memory:* retrieves memory and then freezes the retrieved policy without further evolution.
- *-Full:* uses the same retrieved memory and continues policy evolution during training.

4.1 Benchmark Results

Small-Scale Benchmark Table 1 compares the four variants of EvoMIDFL against PTO and seven fixed-backward DFL baselines. On matching-225, EvoMIDFL-Evo achieves the lowest test regret of 19.602, outperforming PTO and all fixed DFL baselines with a 2.10% reduction relative to PTO. Similarly, on knapsack-200, EvoMIDFL-Evo remains competitive with fixed backward baselines at a test regret of 54.602, while -Full achieves the lowest at 54.294. Consequently, results on both datasets demonstrate the effectiveness of agentic backward-gradient evolution over fixed methods. Additionally, the comparison between EvoMIDFL-Memory and -Full on knapsack further highlights the role of policy evolution within training. Directly transferring and freezing memory in -Memory yields a regret of 55.650, which is 0.526 higher than -Fixed and indicates negative memory transfer. Enabling evolution allows EvoMIDFL-Full to adapt the transferred policy, reducing regret to 54.294, 0.60% below PTO. Overall, policy evolution is effective and essential for mitigating negative transfer of memory and adapting retrieved knowledge to a new problem family.

Table 1: Benchmark results at matching-225 and knapsack- 200. Change reports the relative regret change from PTO: $\Delta R = 100(R_{\text{PTO}} - R)/R_{\text{PTO}}$. Positive values indicate lower regret than PTO.

Method	Matching (scale = 225)‡			Knapsack (scale = 200)		
	Regret↓	RMSE($\mathbf{c}$) ↓	Change (ΔR)↑	Regret↓	RMSE($\mathbf{c}$) ↓	Change (ΔR)↑
EvoMIDFL–Fixed	20.122	6.901	-0.49%	55.124	4.901	-0.92%
EvoMIDFL–Evo	**19.602**	6.862	$\mathbf{+2.10\%}$	54.602	1.998	$+0.03\%$
EvoMIDFL–Memory	–	–	–	55.650	4.892	-1.89%
EvoMIDFL–Full	–	–	–	**54.294**	1.989	$\mathbf{+0.60\%}$
PTO (Elmachtoub & Grigas, 2022)	20.023	6.844	0.00%	54.619	**1.954**	0.00%
SPO+ (Elmachtoub & Grigas, 2022)	20.144	6.816	-0.60%	54.485	2.184	$+0.25\%$
DBB (Vlastelica Pogančić et al., 2020)	19.994	6.866	$+0.14\%$	57.638	2.000	-5.53%
Perturbation (Duchi et al., 2015)	21.024	**6.661**	-5.00%	60.957	4.321	-11.60%
Perturbed-FY (Blondel et al., 2020)	19.677	6.880	$+1.73\%$	55.640	4.871	-1.87%
STE (Bengio et al., 2013)	19.994	6.808	$+0.14\%$	54.608	2.064	$+0.02\%$
Identity (Sahoo et al., 2023)	23.592	9.982	-17.82%	92.503	4.171	-69.36%
Fixed-mixture†	20.019	6.859	$+0.02\%$	54.892	2.117	-0.50%

† Fixed-mixture combines perturbation and Perturbed-FY.

‡ Matching-225 is the first task in the evolution sequence without available memory and only -FIxed/Evo variant are applicable. Knapsack-200 retrieves memory learned from matching, enabling comparison of all variants.

Cross-Scale and Cross-Problem Transfer Table 2 extends the evaluation to matching at scales 900 and 1600 and knapsack at scales 1000 and 2000. The matching tasks retrieve formulation and mechanism-level memory, whereas the knapsack tasks use all three layers of memory transferred from matching. We draw three conclusions. (1) EvoMIDFL performs comparably on large-scale problems in both datasets and beats the baselines in most scenarios except matching-1600. These results support the stability of EvoMIDFL on large-scale problems and demonstrate robust agentic performance relative to fixed DFL backward methods. (2) Consistent with the small-scale results, memory provides useful transferable guidance, but direct transfer without adaptation is unreliable. The -Memory variant obtains 0.277 on knapsack-1000, which is 0.001 higher than -Fixed. Enabling agentic evolution in -Full reduces decision regret to 0.263. Similar patterns occur on knapsack-2000. (3) Evolution with all available memory in -Full achieves comparable performance on both datasets,

with 44.304 on matching-900 and 0.261 on knapsack-2000, highlighting the value of accumulating and exploiting formulation-specific trial experience during evolution.

Table 2: Large-scale benchmark results on matching and knapsack datasets.

Method	Matching				Knapsack			
	scale = 900		scale = 1600		scale = 1000		scale = 2000	
	Reg.↓	RMSE(**c**) ↓	Reg.↓	RMSE(**c**) ↓	Reg./item↓	RMSE(**c**) ↓	Reg./item↓	RMSE(**c**) ↓
EvoMIDFL–Fixed	46.831	7.251	65.972	7.230	0.276	2.287	0.267	5.251
EvoMIDFL–Evo	45.112	6.883	64.175	6.997	0.265	1.893	0.266	1.958
EvoMIDFL–Memory	46.753	7.233	64.183	7.037	0.277	2.291	0.267	5.251
EvoMIDFL–Full	**44.304**	7.204	64.011	**6.939**	**0.263**	1.873	**0.261**	2.025
PTO (Elmachtoub & Grigas, 2022)	45.113	**6.588**	64.138	7.055	0.270	**1.863**	0.266	**1.921**
SPO+ (Elmachtoub & Grigas, 2022)	45.105	7.379	**63.958**	7.021	0.265	1.887	0.265	2.270
DBB (Vlastelica Pogančić et al., 2020)	44.860	6.968	64.122	7.073	0.268	1.922	0.300	1.982
Perturbation (Duchi et al., 2015)	44.736	7.274	64.631	7.034	0.299	4.254	0.312	2.171
Perturbed-FY (Blondel et al., 2020)	44.345	6.688	64.413	7.048	0.277	2.288	0.267	5.251
STE (Bengio et al., 2013)	45.171	10.127	65.823	7.538	0.265	1.924	0.264	2.015
Identity (Sahoo et al., 2023)	46.003	10.793	69.078	9.097	0.345	2.868	0.404	5.101
Fixed-mixture†	45.451	7.193	64.135	7.162	0.265	1.894	0.265	2.245

Prediction–Decision Interaction Figure 2 examines the performance of EvoMIDFL across different levels of prediction difficulty. Overall, decision regret generally increases as prediction difficulty grows. In the hardest Q5, EvoMIDFL achieves lower decision regret than both PTO and fixed Perturbed-FY, demonstrating the advantage of agentic MIDFL when prediction errors are large. Results highlight the benefit of adaptive gradient-policy evolution under challenging prediction conditions, although EvoMIDFL performs marginally worse than Perturbed-FY in Q1, Q3, and Q4.

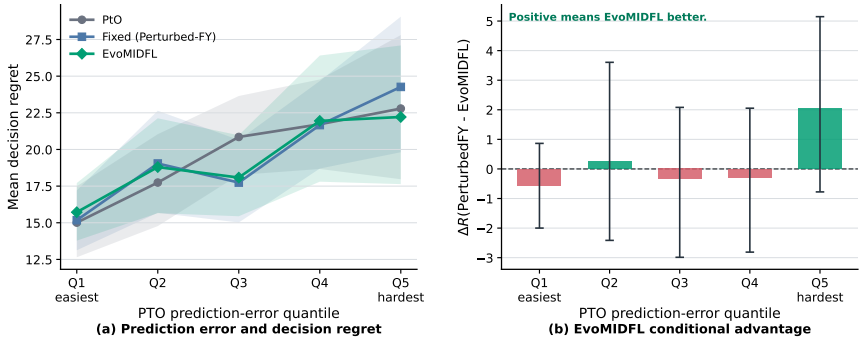

Figure 2: Prediction–decision difficulty map for matching-225. Figure (a) compares mean decision regret across quantiles of PTO prediction error, with Q1 the easiest and Q5 the hardest; the shaded regions denote 95% confidence intervals. Figure (b) reports the conditional regret difference ΔR between the best baseline, Perturbed-FY, and EvoMIDFL; positive values favor EvoMIDFL.

Memory Transfer Ablation Tables 1 and 2 report the performance of evolution using the complete available memory, whereas Table 3 isolates the contributions of different memory layers by restricting the memory available for retrieval. On both problems, using only a subset of the available memory results in higher regret than using the complete memory hierarchy. Relative to the best no-memory setting, the complete memory reduces regret by 0.26% on matching and 1.88% on knapsack. Across the two benchmarks, regret generally decreases as additional memory layers are incorporated, suggesting that the three layers provide complementary information. Layer 1, which retains formulation-specific trial records, yields the largest incremental improvement, while the higher-level memory layers organize this evidence into more general evolution strategies that support cross-scale and cross-problem transfer. Notably, the complete-memory setting achieves the lowest regret on knapsack despite a higher coefficient RMSE, indicating that optimizing for downstream decision quality may trade off coefficient-level prediction accuracy in some settings.

Table 3: Three-layer memory ablation on matching-1600 and knapsack-2000. The EvoMIDFL prefix is omitted. $\Delta R = 100(R_{\text{no-mem}} - R)/R_{\text{no-mem}}$.

Method	Matching (scale = 1600)			Knapsack (scale = 2000)		
	Regret↓	RMSE(**c**) ↓	Change (ΔR)↑	Reg./item↓	RMSE(**c**) ↓	Change (ΔR)↑
–Full (no memory)	64.175	6.997	0.00%	0.266	1.958	0.00%
–Full ($\mathcal{M}^{(1)}$)	64.083	7.142	+0.14%	0.263	2.010	+1.13%
–Full ($\mathcal{M}^{(1)} + \mathcal{M}^{(2)}$)	**64.011**	**6.939**	**+0.26%**	0.263	**1.994**	+1.13%
–Full ($\mathcal{M}^{(1)} + \mathcal{M}^{(2)} + \mathcal{M}^{(3)}$)	–	–	–	**0.261**	2.025	**+1.88%**

4.2 Evolution Analysis

Evolution Trajectory Figure 3 shows how the gradient policy changes over evolution on matching-1600. The upper panel shows all verified policy proposals and the rejected proposals. The lower panel contrasts validation regret with the running best and marks the rounds in which skills are promoted. Among the highlighted changes, the agent switches among SPO+, STE, DBB, and other estimators. The clipping parameter is also changed frequently. Across 49 evolution rounds, 44 of 588 proposals are promoted, while the best validation regret decreases from 81.57 to 64.49, dropping by 17.08. The trajectory further illustrates the intended roles of the critic and evaluator: useful changes are retained, whereas proposals without sufficient validation evidence are rejected. Appendix E.4 provides additional evolution traces.

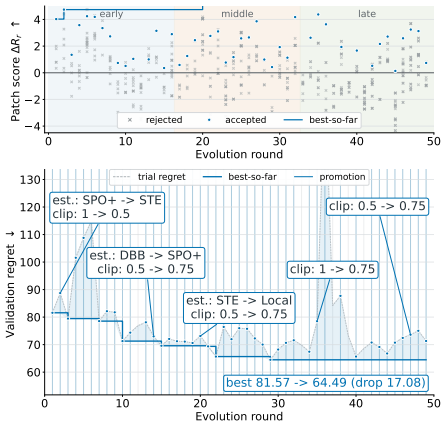

Figure 3: Policy evolution trajectory.

Policy Evolution Figures 4–5 characterize how EvoMIDFL searches and filters backward policies. Figure 4 shows that estimator utility is task dependent: DBB and SPO+ attain higher promotion rates on knapsack (30.0%, 21.3%) than on matching (6.3%, 9.2%), demonstrating that the evolution adaptively adjusts to different problem formulations. Figure 5 further shows that validation gating is selective: only 5.9–8.7% of matching and 13.4–27.0% of knapsack proposals are promoted.

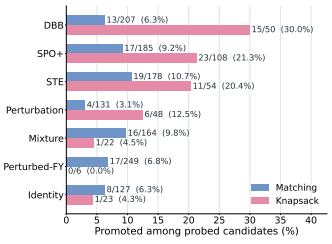

Figure 4: Estimator promotion rates on matching and knapsack datasets.

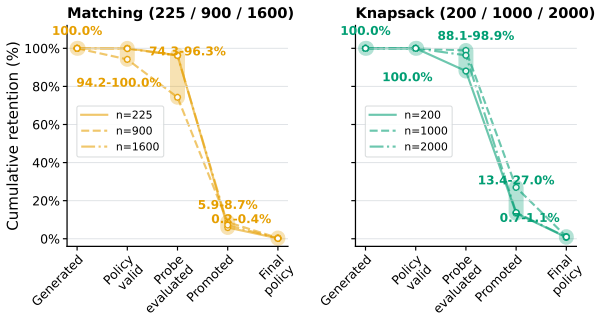

Figure 5: Proposal-to-promotion.

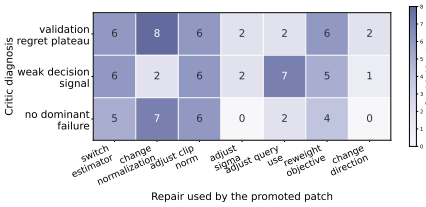

Figure 6: Failure-to-repair map.

Actor–Critic Repair Figure 6 examines how the Critic Agent reacts to validation-diagnosed failures. Each cell reports the repair action frequency by critic diagnosis. Validation-regret plateaus were most often addressed by changing normalization, while weak decision gradients primarily triggered adjustments to query usage. Overall, estimator switching, normalization, and clipping were the most frequently promoted repairs.

5 Conclusion

In this paper, we introduced EvoMIDFL, an agentic self-evolution framework for mixed-integer decision-focused learning. By leveraging a specialized actor–critic workflow and a three-layer memory architecture, EvoMIDFL iteratively refines its backward-gradient generation procedure and transfers validated knowledge across problem settings. Experiments on matching and knapsack problems demonstrate that EvoMIDFL improves final decision quality and outperforms fixed-backward DFL baselines. A current limitation is that the search space is restricted to a predefined pool of candidate gradient estimators. Extending the framework to discover and evaluate a broader range of gradient-generation mechanisms remains an important direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools to polish the writing, improve readability, check grammar, identify related literature, generate and debug code, check mathematical proof, draft the algorithm flow in the appendix. We did not use generative AI for research ideation, methodology design, experiment design, or dataset generation. All research concepts and analyses were developed by the authors. We reviewed all AI-assisted contents: references were checked manually, LLM-generated code was verified and tested. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

No human participants or animal experiments were involved. All datasets were used in ways that preserve privacy. No personally identifiable information was used, and no experiments were conducted that could raise privacy or security concerns.

REPRODUCIBILITY STATEMENT

We have made every effort to ensure that the results presented in this paper are reproducible. The code has been made publicly available to facilitate replication. The experimental setup, including training settings, model configurations, algorithm flow, hardware details and code link is described in the appendix.

REFERENCES

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 136–145, 2017.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis Bach. Learning with differentiable perturbed optimizers. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9508–9519, 2020.

Mathieu Blondel, André F. T. Martins, and Vlad Niculae. Learning with Fenchel–Young losses. *Journal of Machine Learning Research*, 21(35):1–69, 2020. URL `https://www.jmlr.org/papers/v21/19-021.html`.

Hongyu Cheng and Amitabh Basu. Learning cut generating functions for integer programming. In *Advances in Neural Information Processing Systems*, volume 37, pp. 61455–61480, 2024. doi: 10.52202/079017-1964.

Priya L. Donti, Brandon Amos, and J. Zico Kolter. Task-based end-to-end model learning in stochastic optimization. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

John C. Duchi, Michael I. Jordan, Martin J. Wainwright, and Andre Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015. doi: 10.1109/TIT.2015.2409256.

Adam N. Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Management Science*, 68 (1):9–26, 2022. doi: 10.1287/mnsc.2020.3922.

Aaron Ferber, Bryan Wilder, Bistra Dilkina, and Milind Tambe. MIPaaL: Mixed integer program as a layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 1504–1511, 2020. doi: 10.1609/aaai.v34i02.5509.

Junlin He, Yihong Tang, Tong Nie, Ao Qu, Yuebing Liang, Hamzeh Alizadeh, Bang Liu, Wei Ma, and Lijun Sun. MobEvolve: An agentic self-evolving heuristic system for interpretable human mobility generation. *arXiv preprint arXiv:2606.01640*, 2026.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Jim Fan, et al. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations*, 2024.

Jayanta Mandi, James Kotary, Senne Berden, Maxime Mulamba, Victor Bucarey, Tias Guns, and Ferdinando Fioretto. Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. *Journal of Artificial Intelligence Research*, 80:1623–1701, 2024. doi: 10.1613/JAIR.1.15320.

Tong Nie, Yuewen Mei, Yihong Tang, Junlin He, Jie Deng, Jian Sun, and Wei Ma. EvoDrive: Pareto evolution for safety-critical autonomous driving via self-improving LLM agents. *arXiv preprint arXiv:2606.03678*, 2026.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Ao Qu, Han Zheng, Zijian Zhou, Yihao Yan, Yihong Tang, Shao Yong Ong, Fenglu Hong, Kaichen Zhou, Chonghe Jiang, Minwei Kong, et al. CORAL: Towards autonomous multi-agent evolution for open-ended discovery. *arXiv preprint arXiv:2604.01658*, 2026.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Utsav Sadana, Abhilash Chenreddy, Erick Delage, Alexandre Forel, Emma Frejinger, and Thibaut Vidal. A survey of contextual optimization methods for decision-making under uncertainty. *European Journal of Operational Research*, 320(2):271–289, 2025. doi: 10.1016/j.ejor.2024.03.020.

Subham Sekhar Sahoo, Anselm Paulus, Marin Vlastelica, Vít Musil, Volodymyr Kuleshov, and Georg Martius. Backpropagation through combinatorial algorithms: Identity with projection works. In *International Conference on Learning Representations*, 2023.

Maximilian Schiffer, Heiko Hoppe, Yue Su, Louis Bouvier, and Axel Parmentier. Combinatorial optimization augmented machine learning. *arXiv preprint arXiv:2601.10583*, 2026.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Paul Strang, Zacharie Alès, Côme Bissuel, Olivier Juan, Safia Kedad-Sidhoum, and Emmanuel Rachelson. A markov decision process for variable selection in branch & bound. In *Advances in Neural Information Processing Systems*, volume 38, pp. 100886–100912, 2025. doi: 10.52202/085713-3377.

Bo Tang and Elias B. Khalil. PyEPO: A PyTorch-based end-to-end predict-then-optimize library with linear objective function. In *OPT 2022: Optimization for Machine Learning Workshop at NeurIPS 2022*, 2022.

Bo Tang, Elias B Khalil, and Ján Drgoňa. Learning to optimize for mixed-integer non-linear programming with feasibility guarantees. In *NeurIPS 2025 Workshop on GPU-Accelerated and Scalable Optimization*, 2025.

Marin Vlastelica Pogančić, Anselm Paulus, Vít Musil, Georg Martius, and Michal Rolínek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*, 2020.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Xinyu Wang, Jinxiao Du, Yiyang Peng, and Wei Ma. From sequential to recursive: Enhancing decision-focused learning with bidirectional feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 26579–26587, 2026. doi: 10.1609/aaai.v40i31.39866.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Huigen Ye, Hua Xu, An Yan, and Yaoyang Cheng. Large language model-driven large neighborhood search for large-scale MILP problems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 72131–72180. PMLR, 2025. URL https://proceedings.mlr.press/v267/ye25j.html.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Tjarko Lange, and Jeff Clune. Darwin Gödel machine: Open-ended evolution of self-improving agents. In *International Conference on Learning Representations*, 2026.

APPENDIX

TABLE OF CONTENTS

A Background and Preliminaries **14**

A.1 Detailed Related Work on Decision-Focused Learning 14

A.2 Preliminaries on Decision-Focused Learning 15

A.2.1 Decision Regret 15

A.2.2 Fine-tuning Objective 15

B Theoretical Proofs **15**

B.1 Proof of Bounded Simplex Gradient Composition 15

B.2 Proof of Cosine-Gated Non-Cancellation 16

C Extended Method **17**

C.1 Full Algorithm Flow for EVOMIDFL 17

C.2 Detailed Gradient Policy Search Space 17

D Implementation Details **20**

D.1 Benchmark Problems 20

D.1.1 Baselines 21

D.1.2 Training Objective 22

D.2 Evaluation Metrics 22

D.3 Basic Settings 23

D.3.1 Predictor Settings 23

D.3.2 Solver and LLM Backends 23

D.4 Evolution Protocol 23

E Supplementary Results **23**

E.1 Ablation Studies with Diverse Predictors 23

E.2 Ablation Study with Limited Training Data 25

E.3 Predictor Feature Ablation 25

E.4 Additional Evolution Traces 26

F Core Architectural Overview and Executable Prompt Contracts **28**

F.1 System Architecture 28

F.2 Prompt and Runtime Summary 28

A Background and Preliminaries

A.1 Detailed Related Work on Decision-Focused Learning

Decision-Focused Learning Real-world operational problems often require decisions to be made under uncertain parameters. The conventional predict-then-optimize (PTO) paradigm first estimates these parameters using a predictive model and subsequently solves the downstream optimization problem based on the predictions. Decision-focused learning (DFL), in contrast, incorporates downstream decision quality into model training and optimizes the predictive model with respect to decision-aware losses rather than prediction error alone (Donti et al., 2017; Elmachtoub & Grigas, 2022; Wang et al., 2026). Representative approaches include regret-aware surrogate objectives such as the Smart Predict-then-Optimize (SPO+) loss (Elmachtoub & Grigas, 2022). For continuous optimization problems, differentiable optimization layers have enabled gradients to propagate through optimization procedures, including OptNet for quadratic programs (Amos & Kolter, 2017) and CVXPYLayers for disciplined convex programs (Agrawal et al., 2019). Toolkits such as PyEPO further provide implementations of prediction-and-optimization methods for combinatorial optimization problems (Tang & Khalil, 2022). Extending DFL to discrete and mixed-integer optimization is more challenging because the optimizer induces a discontinuous and often locally constant mapping from predicted coefficients to decisions, making its exact Jacobian unavailable or uninformative. Existing approaches address this difficulty from different perspectives. MIPaaL differentiates through mixed-integer programs using a cutting-plane procedure that progressively tightens a continuous relaxation (Ferber et al., 2020), while other methods bypass explicit differentiation of the discrete optimizer through surrogate losses, perturbation-based smoothing, black-box differentiation, or approximate backward mappings (Mandi et al., 2024; Sadana et al., 2025; Schiffer et al., 2026). These developments provide several learning paradigms for MIDFL, including learning-to-branch methods and gradient-based approaches.

Surrogate Gradient Estimators for Mixed-Integer DFL Extending DFL to mixed-integer programming (MIDFL) remains challenging because the optimization map from continuous predicted coefficients to mixed-integer decisions can change discontinuously as the predicted coefficients vary. For purely discrete decisions, this map is typically locally constant away from solution-switching boundaries and discontinuous at such boundaries, yielding zero derivatives where locally constant and undefined derivatives at the boundaries. Existing approaches therefore construct alternative coefficient-space learning gradients, which can be broadly grouped into four families. First, identity-based or straight-through estimators replace the unavailable optimizer Jacobian action with a simple signed or scaled mapping. The straight-through estimator provides an early example of this principle (Bengio et al., 2013), while identity-based differentiation has subsequently been developed for combinatorial algorithms by exploiting the geometry of the discrete solution space (Sahoo et al., 2023). Second, solver-based black-box differentiation perturbs the coefficients according to the outer loss gradient, invokes the discrete optimizer again, and constructs a gradient substitute from the resulting decision difference (Vlastelica Pogančić et al., 2020). The solver-based black-box differentiation avoids explicitly forming or differentiating the internal Jacobian of the optimization procedure. Third, zeroth-order estimators use objective evaluations under paired coefficient perturbations to approximate gradients without differentiating through the solver (Duchi et al., 2015); their behavior depends on design choices such as the perturbation radius, sampling structure, and query budget. Fourth, surrogate-loss methods directly construct coefficient-space (sub)gradients from decision-aware surrogate objectives, with SPO+ and Fenchel–Young losses providing representative examples (Elmachtoub & Grigas, 2022; Blondel et al., 2020). Related perturb-and-optimize approaches instead smooth the discrete optimization map through stochastic perturbations and differentiate the resulting smoothed operator (Berthet et al., 2020). Existing surrogate gradient estimators provide complementary mechanisms for obtaining informative backward gradients without requiring the exact Jacobian of a mixed-integer optimizer. However, in most existing approaches, the differentiation mechanism and its associated configuration are human-determined before training. Consequently, how to adaptively select, configure, and combine heterogeneous gradient estimators according to observed training behavior remains insufficiently explored.

Summary Existing MIDFL methods provide several effective alternatives to exact differentiation, ranging from relaxation-based differentiation to gradient approximation. Nevertheless, these methods are generally deployed under a pre-specified differentiation strategy, although the effectiveness

and computational cost of different gradient estimators can vary across problem instances, training stages, and optimization structures. This motivates an adaptive formulation in which the backward differentiation strategy itself becomes an object of search and refinement, while the forward mixed-integer optimization problem and its feasibility requirements remain unchanged.

A.2 Preliminaries on Decision-Focused Learning

A.2.1 Decision Regret

For an instance $(\mathbf{v}, \mathbf{c})$ with feature vector $\mathbf{v}$, the prediction model produces the estimated objective coefficients $\widehat{\mathbf{c}} = \mathcal{F}_{\boldsymbol{\theta}}(\mathbf{v})$, and the optimizer $\mathcal{G}$ returns an optimal solution $\widehat{\mathbf{x}} = \mathcal{G}(\widehat{\mathbf{c}})$. Let $\mathbf{x}^\star(\mathbf{c})$ denote the ground-truth optimal decision. For a minimization problem, decision regret is defined as follows:

$$\mathcal{L}_{reg}(\boldsymbol{\theta}; \mathbf{v}, \mathbf{c}) = \mathbf{c}^\top \widehat{\mathbf{x}} - \mathbf{c}^\top \mathbf{x}^\star(\mathbf{c}) = \mathbf{c}^\top \mathcal{G}(\mathcal{F}_{\boldsymbol{\theta}}(\mathbf{v})) - \mathbf{c}^\top \mathbf{x}^\star(\mathbf{c}). \tag{23}$$

This regret loss measures the degradation in the realized objective value induced by the predicted coefficients, rather than the coefficient-wise prediction error.

A.2.2 Fine-tuning Objective

Given a training set $\mathcal{D} = \{(\mathbf{v}_i, \mathbf{c}_i)\}_{i=1}^N$ with N samples, the fine-tuning objective minimizes empirical decision regret together with an optional auxiliary loss, such as prediction error:

$$\min_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \big(\mathcal{L}_{\text{reg}}(\boldsymbol{\theta}; \mathbf{v}_i, \mathbf{c}_i) + \lambda_{\text{aux}} \mathcal{L}_{\text{aux}}(\boldsymbol{\theta}) \big), \tag{24}$$

where $\mathcal{L}_{\text{aux}}$ denotes optional auxiliary terms used during training and $\lambda_{\text{aux}} \geq 0$ controls their contribution. The decision-regret term therefore specifies the downstream learning target, whereas the auxiliary terms, when used, provide additional training signals.

B Theoretical Proofs

This section provides proofs of the two propositions. In EvoMIDFL, each candidate backward mechanism in the pool $\mathcal{H}$ produces a stabilized coefficient-space gradient $\mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i}$, without assuming that it coincides with the exact optimizer vector–Jacobian product (VJP). Consistent with Section 3.4, t indexes the training epoch, i the training instance, k the backward mechanism, and π_r the policy active during epoch t. Here, $r = r(t)$ denotes the active evolution round, whose policy has simplex weights $\boldsymbol{\alpha}_r$ and global scaling factor s_r. The propositions characterize the boundedness and directional compatibility of the composed gradient $\widetilde{\mathbf{g}}^{\pi_r}_{\mathbf{c},t,i}$.

B.1 Proof of Bounded Simplex Gradient Composition

Proposition 1 (Bounded simplex gradient composition). *Fix training epoch t and instance i. Suppose every active stabilized gradient $\mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i}$ satisfies $\|\mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i}\|_2 \leq G_k < \infty \ \forall k \in \mathcal{I}(\pi_r)$. For $\boldsymbol{\alpha}_r \in \Delta_K$ and $0 < s_{\min} \leq s_r \leq s_{\max} < \infty$, define $\widetilde{\mathbf{g}}^{\pi_r}_{\mathbf{c},t,i} := s_r \sum_{k=1}^K \alpha_{r,k} \mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i}$, then the composed gradient is bounded by $s_{\max}$ times the largest candidate-gradient bound, as follows:*

$$\left\| \widetilde{\mathbf{g}}^{\pi_r}_{\mathbf{c},t,i} \right\|_2 \leq s_r \sum_{k \in \mathcal{I}(\pi_r)} \alpha_{r,k} \left\| \mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i} \right\|_2 \leq s_{\max} \sum_{k \in \mathcal{I}(\pi_r)} \alpha_{r,k} G_k \leq s_{\max} \max_{k \in \mathcal{I}(\pi_r)} G_k. \tag{25}$$

Proof. Because $\boldsymbol{\alpha}_r \in \Delta_K$, every $\alpha_{r,k}$ is nonnegative and $\sum_{k=1}^K \alpha_{r,k} = 1$. The triangle inequality and the candidate-gradient bounds give

$$\begin{aligned} \left\| \widetilde{\mathbf{g}}^{\pi_r}_{\mathbf{c},t,i} \right\|_2 &\leq s_r \sum_{k=1}^{K} \alpha_{r,k} \left\| \mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i} \right\|_2 \\ &\leq s_{\max} \sum_{k=1}^{K} \alpha_{r,k} G_k \leq s_{\max} \max_{k \in [K]} G_k. \end{aligned} \tag{26}$$

The last inequality holds because $\sum_{k=1}^{K} \alpha_{r,k} G_k$ is a convex combination of the bounds G_k. □

B.2 Proof of Cosine-Gated Non-Cancellation

Proposition 2 (Non-cancellation under cosine compatibility). *Fix training epoch t and instance i, and let $\mathcal{I}(\pi_r) := \{k \in [K] : \alpha_{r,k} > 0\}$. Suppose every active stabilized gradient satisfies*

$$\left\| \mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i} \right\|_2 \geq m > 0, \qquad \forall k \in \mathcal{I}(\pi_r), \tag{27}$$

and every distinct active pair satisfies cosine directional compatibility:

$$\frac{\left\langle \mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i}, \mathbf{g}^{\pi_r}_{\mathbf{c},t,\ell,i} \right\rangle}{\left\| \mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i} \right\|_2 \left\| \mathbf{g}^{\pi_r}_{\mathbf{c},t,\ell,i} \right\|_2} \geq \rho, \quad \rho \in [0,1], \quad \forall k \neq \ell \in \mathcal{I}(\pi_r). \tag{28}$$

If $\boldsymbol{\alpha}_r \in \Delta_K$ and $0 < s_{\min} \leq s_r$, then the composed gradient $\widetilde{\mathbf{g}}^{\pi_r}_{\mathbf{c},t,i}$ satisfies:

$$\left\| \widetilde{\mathbf{g}}^{\pi_r}_{\mathbf{c},t,i} \right\|_2^2 \geq s_{\min}^2 m^2 \left[\rho + (1-\rho) \sum_{k \in \mathcal{I}(\pi_r)} \alpha_{r,k}^2 \right] > 0. \tag{29}$$

Proof. Since inactive weights are zero, the composition can be expanded over $\mathcal{I}(\pi_r)$ as

$$\begin{aligned} \left\| \widetilde{\mathbf{g}}^{\pi_r}_{\mathbf{c},t,i} \right\|_2^2 &= s_r^2 \sum_{k \in \mathcal{I}(\pi_r)} \alpha_{r,k}^2 \left\| \mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i} \right\|_2^2 \\ &\quad + 2 s_r^2 \sum_{\substack{k<\ell \\ k,\ell \in \mathcal{I}(\pi_r)}} \alpha_{r,k} \alpha_{r,\ell} \left\langle \mathbf{g}^{\pi_r}_{\mathbf{c},t,k,i}, \mathbf{g}^{\pi_r}_{\mathbf{c},t,\ell,i} \right\rangle \\ &\geq s_{\min}^2 m^2 \sum_{k \in \mathcal{I}(\pi_r)} \alpha_{r,k}^2 + 2\rho s_{\min}^2 m^2 \sum_{\substack{k<\ell \\ k,\ell \in \mathcal{I}(\pi_r)}} \alpha_{r,k} \alpha_{r,\ell}. \end{aligned} \tag{30}$$

Because $\boldsymbol{\alpha}_r \in \Delta_K$ and inactive weights are zero, $\sum_{k \in \mathcal{I}(\pi_r)} \alpha_{r,k} = 1$. Hence $2 \sum_{k<\ell} \alpha_{r,k} \alpha_{r,\ell} = 1 - \sum_k \alpha_{r,k}^2$, which yields

$$\left\| \widetilde{\mathbf{g}}^{\pi_r}_{\mathbf{c},t,i} \right\|_2^2 \geq s_{\min}^2 m^2 \left[\rho + (1-\rho) \sum_{k \in \mathcal{I}(\pi_r)} \alpha_{r,k}^2 \right]. \tag{31}$$

The active set is nonempty, so the squared active weights have positive sum. Together with $s_{\min} > 0$, $m > 0$, and $\rho \in [0,1]$, the lower bound is strictly positive.

Weight-Dependent Interpretation Because the mixture weights $\boldsymbol{\alpha}_r$ are explicitly specified by the active gradient policy, the lower bound in Proposition 2 can be evaluated directly for every policy. Define the weight-concentration measure

$$C(\boldsymbol{\alpha}_r) := \sum_{k \in \mathcal{I}(\pi_r)} \alpha_{r,k}^2, \qquad \frac{1}{|\mathcal{I}(\pi_r)|} \leq C(\boldsymbol{\alpha}_r) \leq 1. \tag{32}$$

The non-cancellation bound can then be written as

$$\left\| \widetilde{\mathbf{g}}^{\pi_r}_{\mathbf{c},t,i} \right\|_2^2 \geq s_{\min}^2 m^2 \left[\rho + (1-\rho) C(\boldsymbol{\alpha}_r) \right]. \tag{33}$$

For a fixed compatibility threshold $\rho < 1$, a more concentrated mixture has a larger value of $C(\boldsymbol{\alpha}_r)$ and therefore a stronger non-cancellation lower bound. A more balanced mixture has a smaller bound, but the gradient remains strictly nonzero under the stated conditions.

□

Scope of the Gradient-Composition Guarantees The bounds in the two propositions apply only to the composed surrogate gradient in coefficient space and should not be interpreted as descent or end-to-end training guarantees. In particular:

- Non-cancellation does not imply that the composed surrogate gradient is aligned with a descent direction of the target objective.
- A nonzero coefficient-space gradient need not yield a nonzero parameter-space gradient: it may lie in the null space of $\mathcal{J}_{\mathcal{F}_{\boldsymbol{\theta}}}(\mathbf{v})^\top$, so $\widetilde{\mathbf{g}}_{\mathbf{c}}^{\pi} \neq \mathbf{0}$ can still give $\mathcal{J}_{\mathcal{F}_{\boldsymbol{\theta}}}(\mathbf{v})^\top \widetilde{\mathbf{g}}_{\mathbf{c}}^{\pi} = \mathbf{0}$.
- A bound on the coefficient-space gradient norm does not by itself guarantee stability of the full training process. Such a guarantee would additionally require control of the predictor Jacobian, learning rates, optimizer state, stochastic variation, and their evolution over training.

C Extended Method

C.1 Full Algorithm Flow for EvoMIDFL

The framework follows a two-time-scale training process. The predictor is trained using the current fixed gradient policy π_r, while a policy-evolution round is triggered every τ training epochs. At each evolution round, the current training state is saved as a checkpoint S_t, and the parent and all candidate policies are compared from the same checkpoint using the same H_p-epoch probe setting. Probe training uses only the training set, policy selection uses the validation set, and the test set is evaluated only after model selection. During evolution, the agent can modify only the gradient policy π_r over the gradient candidate pool $\mathcal{H}$; the optimization problem and evaluation rules remain fixed.

Algorithms 1–4 describe the two-time-scale workflow in Eq. (5) and the evolution process in Eq. (6). Algorithm 1 controls the overall training process by training the predictor using the current gradient policy and starting a policy-evolution round after every τ training epochs. Algorithm 2 controls each evolution round: it analyzes the current policy, retrieves relevant memory, calls Algorithm 3 to generate candidate policies, and calls Algorithm 4 to evaluate them. Algorithm 3 generates and repairs candidates, screens them against the probe-batch safe set $\mathcal{P}_r^{\mathrm{safe}}$, and removes duplicates. Algorithm 4 compares the parent policy and all safe candidates from the same checkpoint S_t using the same short training schedule and selects policies according to the validation rule in the evolution round.

Table 4: Gradient policy search space for six registered estimators grouped into four mechanism families.

Method / component	Evolvable	Fixed
STE (Bengio et al., 2013; Sahoo et al., 2023)	Activation; temperature; mixture weight	Adaptive-cardinality threshold; unit step scale; STE gradient function
DBB (Vlastelica Pogančić et al., 2020)	Activation; mixture weight	DBB gradient function; perturbation size $\lambda = 10$; step scale 5
Perturbation / ZO (Duchi et al., 2015)	Activation; perturbation scale; number of query pairs; query budget; mixture weight	Gaussian perturbation; 2 samples per query pair; step scale 0.01; fixed random sampling rule
SPO+ (Elmachtoub & Grigas, 2022)	Activation; mixture weight	SPO+ gradient construction; step scale 0.75
Perturbed-FY / Local Surrogate (Blondel et al., 2020; Berthet et al., 2020)	Activation; perturbation scale and direction; query pairs; query budget; regret/FY weights; normalization; mixture weight	Definitions of regret and FY gradients; allowed perturbation directions
Composition and stabilization	Active methods; nonnegative mixture weights; normalization; clipping norm; query budget	Simplex check; finite-value check; budget check; fixed optimization operator and feasibility check

C.2 Detailed Gradient Policy Search Space

Candidate Backward Gradient Families The candidate backward mechanism pool contains four families of coefficient-space backward gradients: straight-through, black-box, zeroth-order, and surrogate-based mechanisms. These four families contain six registered estimators: STE, DBB, Perturbation/ZO, SPO+, Perturbed-FY, and Local Surrogate. Before policy-controlled stabilization

Algorithm 1 Two-time-scale MIDFL training and policy evolution

Input: Contexts $\mathcal{W}$; data splits $\mathcal{D}^{\omega}_{\mathrm{tr/val/test}}$; predictor $\mathcal{F}_{\boldsymbol{\theta}_0}$; fixed optimization operator $\mathcal{G}$; gradient pool $\mathcal{H}$; initial policy π_1 and memory $\mathcal{M}_1$; training epochs T; evolution interval τ; probe epochs H_p; proposals J; workers W; acceptance margin δ

Output: Trained predictor and selected gradient policy; test results and updated memory

for context $\omega \in \mathcal{W}$ **do**
  Initialize evolution round $r \leftarrow 1$ and predictor parameters $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$.
  for epoch $t = 1$ to T **do** ▷ *Train using the current fixed gradient policy π_r.*
    for batch $\mathcal{B} \subset \mathcal{D}^{\omega}_{\mathrm{tr}}$ **do**
      Initialize batch gradient $\widehat{\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{B}}}^{\pi_r} \leftarrow \mathbf{0}$.
      for instance $i \in \mathcal{B}$ **do**
        Predict coefficients $\widehat{\mathbf{c}}_i \leftarrow \mathcal{F}_{\boldsymbol{\theta}}(\mathbf{v}_i)$.
        Solve the fixed optimization problem $\widehat{\mathbf{x}}_i \leftarrow \mathcal{G}(\widehat{\mathbf{c}}_i)$.
        Compute the decision loss $\mathcal{L}_i = \mathcal{L}(\widehat{\mathbf{x}}_i, \mathbf{x}_i^{\star}; \mathbf{c}_i)$.
        Apply the current fixed policy π_r to construct the coefficient-space backward gradient $\widetilde{\mathbf{g}}^{\pi_r}_{\mathbf{c},i}$.
        Backpropagate through the predictor: $\widehat{\nabla_{\boldsymbol{\theta}} \mathcal{L}_i}^{\pi_r} \leftarrow \lambda_1 \mathcal{J}_{\mathcal{F}_{\boldsymbol{\theta}}}(\mathbf{v}_i)^{\top} \widetilde{\mathbf{g}}^{\pi_r}_{\mathbf{c},i} + \lambda_2 \nabla_{\theta} \mathcal{L}_{pred}$.
        Accumulate the parameter gradient: $\widehat{\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{B}}}^{\pi_r} \leftarrow \widehat{\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{B}}}^{\pi_r} + \widehat{\nabla_{\boldsymbol{\theta}} \mathcal{L}_i}^{\pi_r}$.
      end for
      Average the batch gradient: $\widehat{\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{B}}}^{\pi_r} \leftarrow |\mathcal{B}|^{-1} \widehat{\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{B}}}^{\pi_r}$.
      Update the predictor: $\boldsymbol{\theta} \leftarrow \textsc{OptimizerStep}(\boldsymbol{\theta}, \widehat{\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{B}}}^{\pi_r})$.
    end for
    Save the current training state as checkpoint S_t.
    if $t \bmod \tau = 0$ and $t < T$ **then** ▷ *Evolve the gradient policy while keeping the training state fixed.*
      Keep checkpoint S_t fixed and use π_r and $\mathcal{M}_r$ as the current policy and memory.
      $(\pi_{r+1}, \mathcal{M}_{r+1}) \leftarrow \textsc{EvolutionCoordinator}(S_t, \pi_r, \mathcal{M}_r, \omega, \mathcal{D}^{\omega}_{\mathrm{tr}}, \mathcal{D}^{\omega}_{\mathrm{val}}, \mathcal{G}, \mathcal{H}, H_p, J, W, \delta)$.
      Restore S_t and discard all parameters produced during policy probes.
      $r \leftarrow r + 1$.
    end if
  end for
  Select the final checkpoint using $\mathcal{D}^{\omega}_{\mathrm{val}}$ only.
  Evaluate the selected predictor and policy once on $\mathcal{D}^{\omega}_{\mathrm{test}}$.
end for
return trained predictors, selected policies, test results, and updated memory $\mathcal{M}_r$.

Algorithm 2 Evolution Coordinator: One Gradient Policy Evolution Round

Input: Frozen checkpoint S_t; current policy π_r; memory $\mathcal{M}_r$; context ω; training and validation sets $\mathcal{D}^{\omega}_{\mathrm{tr}}$ and $\mathcal{D}^{\omega}_{\mathrm{val}}$; fixed optimization operator $\mathcal{G}$; gradient pool $\mathcal{H}$; probe epochs H_p; proposals J; workers W; acceptance margin δ

Output: Accepted policy π_{r+1} and updated memory $\mathcal{M}_{r+1}$

$\mathcal{F}_r \leftarrow \textsc{PreProposalCritic}(\pi_r, \mathcal{M}_r, \omega)$ using the current training and validation evidence.
$\mathcal{K}_r \leftarrow \textsc{Retrieve}(\mathcal{M}_r, \mathcal{F}_r, \omega)$.
Remove retrieved records incompatible with the fixed gradient pool $\mathcal{H}$ or optimization operator $\mathcal{G}$.
Fix a screening probe batch $\mathcal{B}^{\mathrm{probe}}_r \subset \mathcal{D}^{\omega}_{\mathrm{tr}}$ at checkpoint S_t.
$(\Delta_r, \Pi_r, \mathcal{Z}^{\mathrm{scr}}_r) \leftarrow \textsc{GenerateCandidates}(S_t, \pi_r, \mathcal{F}_r, \mathcal{K}_r, \mathcal{G}, \mathcal{H}, \mathcal{B}^{\mathrm{probe}}_r, H_p, J)$. ▷ *Propose, check, repair, and safely screen candidate policies*
if $\Pi_r = \emptyset$ **then**
  $\pi_{r+1} \leftarrow \pi_r$ and $\mathcal{Z}^{\mathrm{probe}}_r \leftarrow \emptyset$. ▷ *No safe candidate; retain the parent policy*
else
  $(\pi_{r+1}, \mathcal{Z}^{\mathrm{probe}}_r) \leftarrow \textsc{CompareCandidates}(S_t, \pi_r, \Pi_r, \mathcal{D}^{\omega}_{\mathrm{tr}}, \mathcal{D}^{\omega}_{\mathrm{val}}, \mathcal{G}, H_p, W, \delta)$. ▷ *Same-checkpoint probe*
end if
$\mathcal{Z}_r \leftarrow \mathcal{Z}^{\mathrm{scr}}_r \cup \mathcal{Z}^{\mathrm{probe}}_r$.
$\mathcal{M}_{r+1} \leftarrow \mathcal{M}_r$ with candidate configurations, safety-screening records, validation results, and promotion/rejection outcomes.
Store candidate-derived evidence as a transferable skill only after it shows positive validation evidence.
return $(\pi_{r+1}, \mathcal{M}_{r+1})$.

Algorithm 3 Generate and safely screen candidate gradient policies

Input: Frozen checkpoint S_t; current policy π_r; diagnosis $\mathcal{F}_r$; retrieved memory $\mathcal{K}_r$; fixed optimization operator $\mathcal{G}$; gradient pool $\mathcal{H}$; fixed screening probe batch $\mathcal{B}_r^{\text{probe}}$; probe horizon H_p; number of proposals J

Output: Proposed policies Δ_r; safe candidate policies Π_r; safety-screening records $\mathcal{Z}_r^{\text{scr}}$

Initialize $\Delta_r \leftarrow \emptyset$, $\Pi_r \leftarrow \emptyset$, and $\mathcal{Z}_r^{\text{scr}} \leftarrow \emptyset$.
Define the probe-batch safe set $\mathcal{P}_r^{\text{safe}} \leftarrow \bigcap_{i \in \mathcal{B}_r^{\text{probe}}} \{\pi : \mathsf{V}(\pi) \wedge \mathsf{S}(\pi) \wedge \mathsf{C}_i(\pi) \wedge \mathsf{T}_i(\pi)\}$.
for $j = 1$ to J **do**
  $\Delta\pi_{r,j} \leftarrow \textsc{ActorPropose}(\pi_r, \mathcal{F}_r, \mathcal{K}_r, \mathcal{H})$. ▷ *Propose one change to the current policy*
  $\pi_{r,j} \leftarrow \pi_r \oplus \Delta\pi_{r,j}$.
  $\chi_{r,j} \leftarrow \textsc{PostProposalCritic}(\pi_r, \pi_{r,j}, \Delta\pi_{r,j}, \mathcal{H}, H_p)$. ▷ *Review structure, configuration, protected fields, and rollout budget*
  if $\chi_{r,j}$ requests a repair **then**
    $\pi_{r,j} \leftarrow \textsc{ActorRepair}(\pi_{r,j}, \chi_{r,j})$.
    Recheck $\pi_{r,j}$ once using the same post-proposal Critic.
  end if
  if $\chi_{r,j}$ returns `pass` **then**
    Deterministically evaluate $\mathsf{V}(\pi_{r,j})$, $\mathsf{S}(\pi_{r,j})$, $\mathsf{C}_i(\pi_{r,j})$, and $\mathsf{T}_i(\pi_{r,j})$ for every $i \in \mathcal{B}_r^{\text{probe}}$ at S_t using $\mathcal{G}$.
  end if
  $\Delta_r \leftarrow \Delta_r \cup \{\pi_{r,j}\}$.
  if $\chi_{r,j}$ returns `pass` and $\pi_{r,j} \in \mathcal{P}_r^{\text{safe}}$ **then**
    $\Pi_r \leftarrow \Pi_r \cup \{\pi_{r,j}\}$.
  else
    Reject $\pi_{r,j}$ and record the violated safety conditions.
  end if
  Append the screening outcome for $\pi_{r,j}$ to $\mathcal{Z}_r^{\text{scr}}$.
end for
Remove duplicate policies from Π_r and enforce $\Pi_r \leftarrow \Delta_r \cap \mathcal{P}_r^{\text{safe}}$.
return $(\Delta_r, \Pi_r, \mathcal{Z}_r^{\text{scr}})$.

Algorithm 4 Same-checkpoint candidate evaluation and policy promotion

Input: Frozen checkpoint S_t; parent policy π_r; candidate policies Π_r; training and validation sets $\mathcal{D}_{\text{tr}}^{\omega}$ and $\mathcal{D}_{\text{val}}^{\omega}$; fixed optimization operator $\mathcal{G}$; probe epochs H_p; workers W; acceptance margin δ

Output: Accepted policy π_{r+1} and probe-evaluation records $\mathcal{Z}_r^{\text{probe}}$

Form the comparison set $\mathcal{Q}_r^{\text{safe}} \leftarrow \{\pi_r\} \cup \Pi_r$.
for policy $\pi \in \mathcal{Q}_r^{\text{safe}}$ in parallel **do**
  Restore the same checkpoint S_t for policy π.
  for probe epoch $e = 1$ to H_p **do**
    Train the predictor on $\mathcal{D}_{\text{tr}}^{\omega}$ using the fixed policy π.
  end for
  Evaluate the probe-trained predictor on $\mathcal{D}_{\text{val}}^{\omega}$.
  Record validation regret $\text{Reg}_r(\pi)$ and the remaining diagnostics in $\mathcal{Z}_r^{\text{probe}}(\pi)$.
end for
Discard failed candidate runs and keep all valid evaluation results.
Select the valid candidate with the lowest validation regret; use computational cost to break ties.
Let $\pi_r^{\star}$ denote the selected candidate.
if $\pi_r^{\star} \neq \pi_r$ and $\text{Reg}_r(\pi_r^{\star}) \leq \text{Reg}_r(\pi_r) - \delta$ **then**
  $\pi_{r+1} \leftarrow \pi_r^{\star}$. ▷ *Promote the candidate*
else
  $\pi_{r+1} \leftarrow \pi_r$. ▷ *Keep the parent policy*
end if
Discard all probe-trained parameters and restore the original checkpoint S_t.
return $(\pi_{r+1}, \mathcal{Z}_r^{\text{probe}})$.

and composition, their raw gradients are defined as follows:

$$\mathbf{u}_{\mathbf{c}}^{\mathrm{STE}} := \gamma_{\mathrm{STE}} \nabla_{\widehat{\mathbf{x}}} \mathcal{L} \approx \mathcal{J}_{\mathcal{G}}(\widehat{\mathbf{c}})^{\top} \nabla_{\widehat{\mathbf{x}}} \mathcal{L}, \tag{34}$$

$$\mathbf{u}_{\mathbf{c}}^{\mathrm{DBB}} := \frac{\sigma_{\mathrm{DBB}}}{\lambda} \left[\mathcal{G}(\widehat{\mathbf{c}} + \lambda \nabla_{\widehat{\mathbf{x}}} \mathcal{L}) - \mathcal{G}(\widehat{\mathbf{c}}) \right] \approx \mathcal{J}_{\mathcal{G}}(\widehat{\mathbf{c}})^{\top} \nabla_{\widehat{\mathbf{x}}} \mathcal{L}, \tag{35}$$

$$\mathbf{u}_{\mathbf{c}}^{\mathrm{ZO}} := \frac{1}{Q_{\mathrm{ZO}}} \sum_{q=1}^{Q_{\mathrm{ZO}}} \frac{\Phi(\widehat{\mathbf{c}} + \mu \mathbf{z}_q) - \Phi(\widehat{\mathbf{c}} - \mu \mathbf{z}_q)}{2\mu} \mathbf{z}_q, \qquad \mathbf{z}_q \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n), \tag{36}$$

$$\mathbf{u}_{\mathbf{c}}^{\mathrm{SUR}} \in \partial_{\widehat{\mathbf{c}}} \widetilde{\ell}^{(\mathrm{SUR})}(\widehat{\mathbf{c}}; \mathbf{c}), \quad \text{or} \quad \mathbf{u}_{\mathbf{c}}^{\mathrm{SUR}} = \mathcal{J}_{\mathcal{G}_\epsilon}(\widehat{\mathbf{c}})^{\top} \nabla_{\widehat{\mathbf{x}}} \mathcal{L}. \tag{37}$$

For the zeroth-order estimator,

$$\mathbb{E}_{\mathbf{z}} \left[\mathbf{u}_{\mathbf{c}}^{\mathrm{ZO}} \right] = \nabla_{\widehat{\mathbf{c}}} \Phi_{\mu}(\widehat{\mathbf{c}}), \tag{38}$$

where $\Phi(\widehat{\mathbf{c}})$ denotes the scalar decision loss induced by solving with coefficients $\widehat{\mathbf{c}}$, and Φ_μ denotes its Gaussian-smoothed counterpart.

In Eq. (34), $\gamma_{\mathrm{STE}} > 0$ controls the straight-through scale (Bengio et al., 2013; Sahoo et al., 2023). In Eq. (35), $\sigma_{\mathrm{DBB}} \in \{-1, +1\}$ accounts for the solver-sign convention, and $\lambda > 0$ is the black-box perturbation scale (Vlastelica Pogančić et al., 2020). In Eq. (36), $\mu > 0$ is the perturbation radius, Q_{ZO} is the query budget, and $\mathbf{z}_q \in \mathbb{R}^n$ is a sampled perturbation direction (Duchi et al., 2015). In Eq. (37), $\widetilde{\ell}^{(\mathrm{SUR})}$ denotes a registered coefficient-space surrogate, such as SPO+, while $\mathcal{G}_\epsilon$ denotes a smoothed decision map, such as one induced by a perturbed Fenchel–Young construction (Elmachtoub & Grigas, 2022; Blondel et al., 2020; Berthet et al., 2020).

Thus, each mechanism $h_k \in \mathcal{H}$ returns a raw coefficient-space gradient $\mathbf{u}_{\mathbf{c}}^{(k)} \in \mathbb{R}^n$ using its family-specific construction. The gradient policy subsequently normalizes, clips, and composes these gradients according to Eq. (9)–Eq. (10).

Evolvable Search Space Table 4 summarizes evolvable hyperparameters in the candidate gradient families. The *Evolvable* column includes the hyperparameters in each estimator that can be revised during policy evolution. The *Fixed* column includes the unchanged hyperparameters.

Gradient Composition We distinguish between composition across backward estimators and composition within an individual estimator. At the estimator level, the policy combines stabilized gradients from different backward estimators using the weights $\boldsymbol{\alpha}$. For example,

$$\widetilde{\mathbf{g}}_{\mathbf{c},i}^{\pi} = s \left(\alpha_{\mathrm{DBB}} \mathbf{g}_{\mathbf{c},\mathrm{DBB},i}^{\pi} + \alpha_{\mathrm{SPO+}} \mathbf{g}_{\mathbf{c},\mathrm{SPO+},i}^{\pi} \right). \tag{39}$$

Within an individual estimator, multiple component gradients may be combined to construct its raw output. For example, the Local Surrogate combines regret-based and Fenchel–Young gradients as

$$\mathbf{u}_{\mathbf{c},\mathrm{Local},i}^{\pi} = \lambda_R \mathbf{u}_{\mathbf{c},R,i} + \lambda_{\mathrm{FY}} \mathbf{u}_{\mathbf{c},\mathrm{FY},i}, \tag{40}$$

where the internal weights λ_R and λ_{FY} are specified by $\boldsymbol{\zeta}_{\mathrm{Local}}$. These component gradients belong to the same estimator and are therefore not treated as independent backward mechanisms.

The two composition levels can be applied jointly:

$$\begin{aligned} \mathbf{u}_{\mathbf{c},\mathrm{Local},i}^{\pi} &= \lambda_R \mathbf{u}_{\mathbf{c},R,i}^{\pi} + \lambda_{\mathrm{FY}} \mathbf{u}_{\mathbf{c},\mathrm{FY},i}^{\pi}, \\ \mathbf{g}_{\mathbf{c},\mathrm{Local},i}^{\pi} &= \mathrm{Clip}_{B_{\mathrm{Local}}} \left[\mathcal{N}_{\mathrm{Local}} \left(\mathbf{u}_{\mathbf{c},\mathrm{Local},i}^{\pi} \right) \right], \\ \widetilde{\mathbf{g}}_{\mathbf{c},i}^{\pi} &= s \left(\alpha_{\mathrm{DBB}} \mathbf{g}_{\mathbf{c},\mathrm{DBB},i}^{\pi} + \alpha_{\mathrm{Local}} \mathbf{g}_{\mathbf{c},\mathrm{Local},i}^{\pi} \right). \end{aligned} \tag{41}$$

Note that the simplex, directional-compatibility, and numerical-stability conditions are imposed on the active stabilized mechanism-level gradients $\mathbf{g}_{\mathbf{c},k,i}^{\pi}$, rather than on the internal component gradients of the individual mechanism.

D Implementation Details

D.1 Benchmark Problems

Side-Constrained Bipartite Matching For drivers $i \in \mathcal{I}$ and riders $j \in \mathcal{J}$, let $x_{ij} \in \{0, 1\}$ indicate a selected pair, with predicted cost $\widehat{c}_{ij}$, priority p_{ij}, and resource use r_{ij}. The forward

optimization problem is:

$$\min_{\mathbf{x}} \quad \sum_i \sum_j \widehat{c}_{ij} x_{ij} \tag{42}$$

$$\text{s.t.} \quad \sum_j x_{ij} \le 1 \quad \forall i \in \mathcal{I}, \quad \sum_i x_{ij} \le 1 \quad \forall j \in \mathcal{J}, \tag{43}$$

$$K_{\min} \le \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} x_{ij} \le K_{\max}, \tag{44}$$

$$\sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} p_{ij} x_{ij} \ge P_{\min}, \tag{45}$$

$$\sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} r_{ij} x_{ij} \le B, \tag{46}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in \mathcal{I} \quad \forall j \in \mathcal{J}. \tag{47}$$

The objective in Eq. (42) minimizes the total predicted cost of the selected driver–rider pairs. The one-to-one assignment constraints in Eq. (43) ensure that each driver is matched with at most one rider and each rider with at most one driver. The cardinality constraint Eq. (44) requires the number of selected matches to lie between $K_{\min}$ and $K_{\max}$. The priority constraint Eq. (45) imposes a minimum total priority score $P_{\min}$, whereas the resource constraint Eq. (46) limits total resource consumption to the available budget B. Finally, Eq. (47) enforces the binary decisions.

0–1 Knapsack with Continuous Capacity Expansion For items $i \in \{1, \dots, n\}$, the predictor outputs the profit coefficients $\widehat{\mathbf{c}} = (\widehat{c}_1, \dots, \widehat{c}_n) \in \mathbb{R}^n$. The binary variable x_i indicates whether item i is selected, and the continuous variable y purchases additional capacity. Given item weights w_i, base capacity B, expansion limit $\bar{y}$, and per-unit expansion cost $\kappa > 0$, the forward MILP is

$$\max_{\mathbf{x}, y} \quad \sum_{i=1}^{n} \widehat{c}_i x_i - \kappa y, \tag{48}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} w_i x_i \le B + y, \tag{49}$$

$$x_i \in \{0,1\}, \quad i \in \{1, \dots, n\}, \tag{50}$$

$$0 \le y \le \bar{y}. \tag{51}$$

The objective in Eq. (48) maximizes the predicted profit of the selected items minus the capacity-expansion cost. The capacity constraint Eq. (49) permits the selected items to exceed the base capacity B only by purchasing y units of additional capacity. Constraint Eq. (50) enforces the n binary item-selection decisions, while Eq. (51) makes y a bounded continuous variable. Hence the model contains n binary variables and one continuous variable and is a 0–1 knapsack MILP with continuous capacity expansion.

For consistency with the minimization convention in Definition 1, the same problem can be written using the augmented decision and coefficient vectors

$$\mathbf{z} := (\mathbf{x}, y), \qquad \widehat{\mathbf{c}}^{\text{aug}} := (-\widehat{c}_1, \dots, -\widehat{c}_n, \kappa), \tag{52}$$

so that,

$$\min_{\mathbf{z} \in \mathcal{X}_{\text{KS}}} (\widehat{\mathbf{c}}^{\text{aug}})^\top \mathbf{z}, \tag{53}$$

is equivalent to Eq. (48). The last coefficient κ is fixed rather than predicted and therefore does not contribute to the predictor-parameter gradient.

D.1.1 BASELINES

We compare EVOMIDFL with the conventional predict-then-optimize baseline and seven fixed-backward DFL baselines. The baseline names and abbreviations are kept consistent with those used in the main result tables. All experiments are repeated five times with different random seeds, and we report the average results. Their backward constructions and their corresponding roles in the gradient policy search space are summarized in Table 4.

- *PTO:* conventional two-stage predict-then-optimize (Elmachtoub & Grigas, 2022).
- *SPO+:* the Smart Predict-then-Optimize surrogate (Elmachtoub & Grigas, 2022).
- *DBB:* differentiation of black-box combinatorial solvers (Vlastelica Pogančić et al., 2020).
- *Perturbation:* zeroth-order finite-difference estimation based on random coefficient perturbations (Duchi et al., 2015).
- *Perturbed-FY:* a perturbed Fenchel–Young loss with stochastic smoothing (Blondel et al., 2020).
- *STE:* the straight-through estimator (Bengio et al., 2013).
- *Identity:* identity-based backpropagation through the optimization layer (Sahoo et al., 2023).
- *Fixed-mixture:* a non-evolving convex combination of backward mechanisms with prespecified mixture weights.

D.1.2 Training Objective

All EvoMIDFL variants and fixed-backward DFL baselines use the same mini-batch hybrid objective within each benchmark. For a training mini-batch $\mathcal{B} \subseteq \mathcal{D}_{\mathrm{tr}}$, the objective at epoch t is

$$\mathcal{L}_{\mathrm{train}}(\boldsymbol{\theta}_t; \mathcal{B}) := \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \left[\lambda_{d,t} \mathcal{L}_{\mathrm{dec},i} + \lambda_{p,t} \mathcal{L}_{\mathrm{pred},i} \right], \tag{54}$$

where $\lambda_{d,t}, \lambda_{p,t} \geq 0$ control the contributions of the decision and prediction losses and are adjusted according to the same training schedule for all DFL methods. PTO is the prediction-only special case, with $(\lambda_{d,t}, \lambda_{p,t}) = (0, 1)$.

The auxiliary prediction loss is task-specific:

$$\mathcal{L}_{\mathrm{pred},i} = \begin{cases} \sqrt{\dfrac{1}{d_c} \left\| \widehat{\bar{\mathbf{c}}}_i - \bar{\mathbf{c}}_i \right\|_2^2 + 10^{-8}}, & \text{matching}, \\ \dfrac{1}{n} \left\| \dfrac{\widehat{\mathbf{c}}_i - \mathbf{c}_i}{s_{\mathcal{B}}} \right\|_2^2, & \text{knapsack}. \end{cases} \tag{55}$$

Here, d_c is the number of matching cost coefficients, and $\bar{\mathbf{c}}_i$ and $\widehat{\bar{\mathbf{c}}}_i$ are the true and predicted coefficients transformed by a normalizer fitted only on the matching training set. For knapsack, $s_{\mathcal{B}} := \max\left\{1, \frac{1}{|\mathcal{B}|n} \sum_{i \in \mathcal{B}} \sum_{j=1}^{n} |c_{i,j}|\right\}$ is the coefficient scale of the current mini-batch.

Matching uses RMSE in the normalized coefficient space, whereas knapsack uses scale-normalized MSE to account for variation in profit magnitudes. These task-specific losses affect only the scaling of the auxiliary prediction term. Within each benchmark, every compared method uses the same prediction loss, mini-batches, and loss-weight schedule.

D.2 Evaluation Metrics

We evaluate downstream decision quality using mean test regret and prediction accuracy using coefficient RMSE. Let $\mathcal{D}_{\mathrm{te}}$ denote the test set and d_c the coefficient dimension. The two metrics are

$$\overline{R}_{\mathrm{test}} := \frac{1}{|\mathcal{D}_{\mathrm{te}}|} \sum_{i \in \mathcal{D}_{\mathrm{te}}} \left[\mathbf{c}_i^\top \widehat{\mathbf{x}}_i - \mathbf{c}_i^\top \mathbf{x}_i^\star \right]_+, \tag{56}$$

$$\mathrm{RMSE}_c := \left[\frac{1}{|\mathcal{D}_{\mathrm{te}}| d_c} \sum_{i \in \mathcal{D}_{\mathrm{te}}} \left\| \widehat{\mathbf{c}}_i - \mathbf{c}_i \right\|_2^2 \right]^{1/2}. \tag{57}$$

Here, $[a]_+ := \max\{a, 0\}$. For maximization problems, the objective is converted to the equivalent minimization convention before regret is computed.

D.3 Basic Settings

D.3.1 Predictor Settings

Both benchmark families use MLP predictors, but their output structures follow the coefficient parameterization of the corresponding optimization problem. The architectures and training settings are summarized in Table 5. All normalization statistics are fitted on the predictor-training split only and then frozen for validation and test.

Table 5: MLP predictor settings used in the reported experiments. Here n denotes the knapsack item count at the selected scale.

Setting	Matching	Knapsack
Input	Urgency, fatigue, and trip time	First three global source features
Output	Predicted cost for each candidate edge	n predicted item-profit coefficients
Architecture	MLP 3–32–32–1	MLP 3–64–64–n
Activation	ReLU	ReLU
Dropout	0.1	0
Optimizer	Adam	Adam
Learning rate	10^{-3}	10^{-3}
Batch size	16	16
Maximum epochs	2,000	1,000
Early stopping	Patience 100 after a 20-epoch warm-up	Patience 100

D.3.2 Solver and LLM Backends

All solver-based forward passes use Gurobi 12.0.2, with one solver thread and a requested MIP gap of zero. The adaptive matching runs impose a two-second limit on each MILP solve, whereas the independently executed fixed-baseline runs allow up to 900 seconds per solve. No timeout or non-optimal test solve is included in the reported results. Knapsack uses the same single-thread, zero-gap configuration without a per-solve time limit; all reported validation and test instances terminate with optimal status. The framework uses the Codex CLI with a ChatGPT-5.6 backend for policy proposal and critique. Fixed baselines and numerical optimization do not require an external LLM. In the reported adaptive -Evo, Memory, and Full runs, Codex CLI was enabled as a test-blind, validation-summary-guided research advisor. Codex outputs have no direct promotion authority: every proposal is parsed through the same typed policy schema, checked by the candidate validator, and evaluated using same-checkpoint validation rollouts. Promotion is determined exclusively from validation data, and the held-out test set is not exposed to the proposal, critique, or promotion process. All experiments were conducted on a desktop equipped with an Intel Core i7-13700K CPU at 3.40 GHz, 32 GB of RAM, a 500 GB SSD, and an NVIDIA GeForce RTX 3090 Ti GPU. The code is available at `https://anonymous.4open.science/r/EvoMIDFL_iclr_2027-1F7E/`.

D.4 Evolution Protocol

The general evolution mechanism follows Algorithms 1–4 and uses the promotion rule in Eq. (22). All proposals are restricted to the fields marked as evolvable in Table 4. The forward MILP, data partitions, predictor optimizer, feasibility checks, and test set remain unchanged throughout evolution, and the test set is masked during both training and validation. Other settings, including the evolution trigger, probe horizon, proposal count, worker count, coefficient-RMSE tolerance, and query budget, are fixed before evolution. Additionally, each candidate probe is limited to 128 recorded solver queries and MILP calls. Regret values within one percent of the parent's absolute regret, with a 10^{-9} floor, are treated as equivalent when policies are compared.

E Supplementary Results

E.1 Ablation Studies with Diverse Predictors

We additionally evaluate LSTM and RNN predictors as shown in Tables 6 and 7. The RNN and LSTM predictors each consist of one recurrent layer with a hidden dimension of 32, followed by

a shared linear output layer that maps each hidden state to one predicted cost coefficient. Across these predictor settings, EvoMIDFL remains competitive and attains the lowest regret in both RNN experiments and in the knapsack-200 LSTM experiment.

Table 6: LSTM-predictor ablation at Matching-225 and Knapsack-200. All variants use an LSTM predictor. $\Delta R = 100(R_{\mathrm{PTO}} - R)/R_{\mathrm{PTO}}$, where positive values indicate lower regret than PTO.

Method	Matching (scale = 225)			Knapsack (scale = 200)		
	Regret↓	RMSE($\mathbf{c}$) ↓	Change (ΔR)↑	Regret↓	RMSE($\mathbf{c}$) ↓	Change (ΔR)↑
EvoMIDFL–Fixed	21.646	7.892	−3.97%	70.990	2.415	+0.03%
EvoMIDFL–Evo	20.560	7.560	+1.25%	70.989	4.562	+0.03%
EvoMIDFL–Memory	21.103	7.391	−1.36%	70.789	2.551	+0.31%
EvoMIDFL–Full	20.624	6.953	+0.94%	**70.190**	2.431	**+1.16%**
PTO	20.820	7.742	0.00%	71.012	**2.211**	0.00%
SPO+	**20.083**	7.568	**+3.54%**	70.932	2.576	+0.11%
DBB	20.563	**6.907**	+1.23%	70.926	2.386	+0.12%
Perturbation	20.816	7.266	+0.02%	71.181	2.382	−0.24%
Perturbed-FY	22.061	7.359	−5.96%	71.023	2.695	−0.02%
STE	22.410	7.616	−7.64%	79.481	8.952	−11.93%
Identity	25.828	9.792	−24.05%	79.354	9.967	−11.75%
Fixed-mixture	24.943	9.610	−19.80%	76.412	9.393	−7.60%

Table 6 reports the LSTM-predictor results. On matching-225, SPO+ attains the lowest regret (20.083) and the largest improvement over PTO (3.54%), while DBB yields the lowest coefficient RMSE (6.907); among the proposed variants, EvoMIDFL–Evo achieves the lowest regret (20.560), improving over PTO by 1.25%. On Knapsack-200, EvoMIDFL–Full attains the lowest regret (70.190) and the largest improvement over PTO (1.16%), whereas PTO retains the lowest coefficient RMSE (2.211). The disagreement between the regret and RMSE rankings shows that lower coefficient error does not necessarily yield better decision quality and that the preferred backward mechanism is problem-dependent but the evolution and memory combination structure shows advantage in the final decision quality improvement.

Table 7: RNN-predictor ablation at Matching-225 and Matching-900. All variants and baselines use an RNN predictor. Change follows Table 1: $\Delta R = 100(R_{\mathrm{PTO}} - R)/R_{\mathrm{PTO}}$, where positive values indicate lower regret than PTO.

Method	Matching (scale = 225)			Matching (scale = 900)		
	Regret↓	RMSE($\mathbf{c}$) ↓	Change (ΔR)↑	Regret↓	RMSE($\mathbf{c}$) ↓	Change (ΔR)↑
EvoMIDFL–Evo	22.236	7.567	−2.79%	49.623	7.831	0.00%
EvoMIDFL–Full	**21.211**	7.122	**+1.95%**	**48.985**	7.515	**+1.28%**
PTO	21.632	7.241	0.00%	49.622	7.851	0.00%
SPO+	21.329	6.993	+1.40%	48.994	7.846	+1.27%
DBB	21.271	**6.908**	+1.67%	49.552	**6.913**	+0.14%
Perturbation	23.160	7.626	−7.06%	49.845	7.867	−0.45%
Perturbed-FY	21.576	7.262	+0.26%	50.011	7.934	−0.78%
STE	21.656	7.505	−0.11%	50.047	7.930	−0.86%
Identity	24.677	8.527	−14.08%	59.445	9.998	−19.80%
Fixed-mixture	25.572	8.768	−18.21%	55.731	9.463	−12.31%

Table 7 reports the corresponding RNN-predictor results. On matching-225, EvoMIDFL–Full attains the lowest regret (21.211) and the largest improvement over PTO (1.95%), while DBB yields the lowest coefficient RMSE (6.908). The same pattern holds on Matching-900: EvoMIDFL–Full achieves the lowest regret (48.985) and the largest improvement over PTO (1.28%), narrowly ahead of SPO+ (1.27%), whereas DBB again yields the lowest coefficient RMSE (6.913).

E.2 Ablation Study with Limited Training Data

We next examine how EvoMIDFL performs when only a limited number of training samples is available. To this end, we vary the available training-data fraction while keeping the matching scale fixed at 225. Table 8 compares EvoMIDFL with standard DFL baselines and PTO on D25 and D75. Table 8 shows that EvoMIDFL–Full remains robust across both data fractions, reducing regret relative to PTO by 4.56% on D25 and 4.72% on D75, whereas EvoMIDFL–Evo does not improve over PTO in either setting. On D25, Perturbed-FY achieves the lowest overall regret (19.633) and the largest improvement (6.45%), while DBB gives the lowest coefficient RMSE (6.640). On D75, EvoMIDFL–Full attains the lowest regret (19.035) and the largest improvement (4.72%), whereas STE yields the lowest coefficient RMSE (6.594). The differing regret and RMSE rankings further indicate that prediction accuracy alone does not determine downstream decision quality.

Table 8: Data fraction ablation at scale 225. D25 and D75 retain 25% and 75% of the training data, respectively. Change is computed relative to PTO within each dataset.

Method / predictor	Matching D25			Matching D75		
	Regret↓	RMSE($\mathbf{c}$) ↓	Change (ΔR)↑	Regret↓	RMSE($\mathbf{c}$) ↓	Change (ΔR)↑
EvoMIDFL–Evo	21.307	9.682	-1.52%	20.965	7.314	-4.94%
EvoMIDFL–Full	20.031	6.748	$+4.56\%$	**19.035**	6.996	$+\mathbf{4.72}\%$
PTO	20.987	6.671	0.00%	19.979	6.729	0.00%
SPO+	20.894	6.818	$+0.44\%$	19.696	7.217	$+1.42\%$
DBB	20.489	**6.640**	$+2.37\%$	20.364	7.003	-1.93%
Perturbation	20.566	6.796	$+2.01\%$	19.941	6.768	$+0.19\%$
Perturbed-FY	**19.633**	6.708	$+\mathbf{6.45}\%$	20.136	7.011	-0.79%
STE	20.023	7.381	$+4.59\%$	19.706	**6.594**	$+1.37\%$
Identity	23.473	7.941	-11.85%	23.106	22.009	-15.65%
Fixed-mixture	22.418	13.641	-6.82%	20.215	6.724	-1.18%

E.3 Predictor Feature Ablation

We further report the results for an MLP predictor using seven input features in Table 9. On matching-225, EvoMIDFL–Full attains the lowest regret (13.511), whereas EvoMIDFL–Evo yields the lowest coefficient RMSE (4.438). At the larger scales, a proposed variant obtains the lowest regret: EvoMIDFL–Full reaches 31.234 on matching-900, and EvoMIDFL–Evo reaches 47.336 on matching-1600. EvoMIDFL–Full also yields the lowest RMSE at these two scales, with 4.220 and 4.285, respectively. Overall, the adaptive framework remains competitive with the richer seven-feature representation and is strongest at the larger problem scales, although the preferred variant depends on scale.

Table 9: Matching results for the MLP predictor with seven input features on the D50 dataset variants.

Method	Matching 225		Matching 900		Matching 1600	
	Regret↓	RMSE($\mathbf{c}$) ↓	Regret↓	RMSE($\mathbf{c}$) ↓	Regret↓	RMSE($\mathbf{c}$) ↓
EvoMIDFL–Evo	13.517	**4.438**	31.768	4.315	**47.336**	4.322
EvoMIDFL–Full	**13.511**	4.497	**31.234**	**4.220**	47.393	**4.285**
PTO	13.900	4.442	32.178	4.252	47.695	4.307
SPO+	13.822	4.441	31.295	4.271	47.363	4.310
DBB	13.749	4.452	31.451	4.248	48.259	4.308
Perturbation	14.011	4.551	31.645	4.245	48.487	4.378
Perturbed-FY	13.751	4.542	31.609	4.252	47.961	4.370
STE	13.684	4.542	31.241	6.109	47.798	10.693
Identity	21.267	7.011	39.125	8.634	78.410	9.027
Fixed-mixture	14.030	4.473	31.517	4.404	47.950	4.448

E.4 Additional Evolution Traces

Figures 7, 8, and 9 report the matching evolution trajectories under D25, D50, and D75, respectively, at scales 225, 900, and 1600, while Figure 10 reports the D50 knapsack trajectories at scales 200, 1000, and 2000. Across the matching settings, validation regret decreases along the promoted lineages at all three scales, with the larger scales generally exhibiting larger absolute reductions, showing that validation-gated evolution remains effective as data availability and problem size change. The knapsack trajectories display the same pattern of retaining beneficial gradient-policy changes while rejecting weaker proposals, indicating that the evolution mechanism also adapts across a distinct MILP family and substantially larger problem scales.



Figure 7: Evolution trajectories and learned gradient skills for matching under D25.

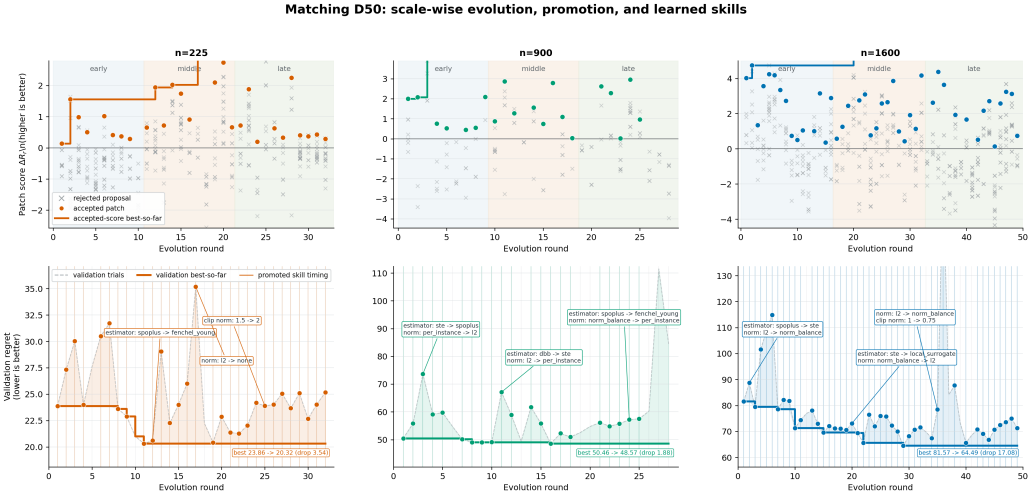

Figure 8: Evolution trajectories and learned gradient skills for matching under D50.

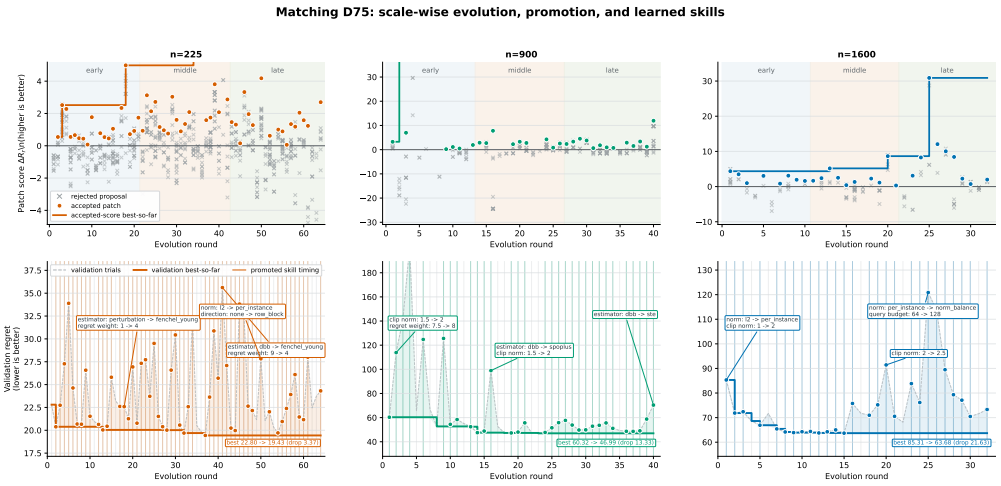

Figure 9: Evolution trajectories and learned gradient skills for matching under D75.

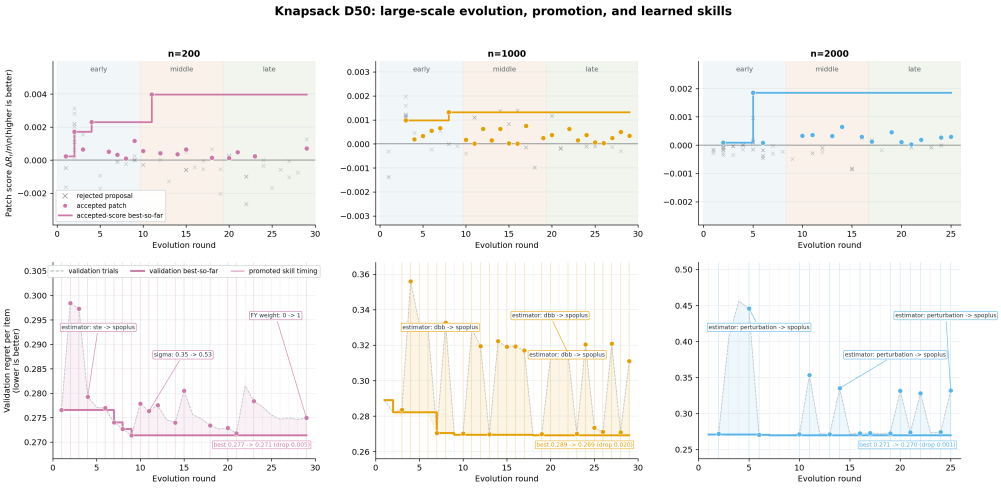

Figure 10: Evolution trajectories and learned gradient skills for large-scale knapsack under D50.

F Core Architectural Overview and Executable Prompt Contracts

F.1 System Architecture

Figure 11 summarizes the system architecture of EvoMIDFL, including data processing, the protected forward path, gradient policies, and the evolution controller.

System Architecture of EvoMIDFL

```
EvoMIDFL/
  data_and_tasks/
    split methods              fixed train, validation, and held-out test partitions
    assignment MILP             side-constrained bipartite matching instances
    knapsack MILP               binary selection with continuous capacity expansion
  protected_forward/
    cost predictor              maps observed features v to predicted costs c_hat
    generalized function G      task-registered exact mixed-integer decision interface
    solver adapters             assignment and knapsack MILP backends
    status + feasibility        accepts qualified states and checks every integer decision
    decision loss               evaluates the feasible decision under realized costs
  gradient_policy/
    policy schema               estimator family, parameters, mixture, budget, stabilization
    candidate family            ST, BB, ZO, and surrogate-family cost-space signals
    surrogate family            SPO+ and perturbed Fenchel--Young/local-surrogate signals
    policy runtime              executes only valid backward constructions
    composition + stability     mixtures, norm balancing, cosine gates, and clipping
    gradient audit              gradient norms, query counts, and failure diagnostics
  baseline_suite/
    prediction baseline         predict-then-optimize (PtO)
    fixed gradient policies     SPO+, DBB, ZO, PFYL, STE, identity, and fixed mixtures
    solver-layer baseline       single-seed reproduction under shared evaluation
  evolution_controller/
    stage trigger               freezes predictor, optimizer, and solver accounting
    pre-proposal critic         summarizes validation failures of the parent policy
    memory retrieval            returns effective trials, mechanisms, and transferable skills
    structured actor            proposes bounded GradientPolicy patches only
    review + validation         schema, lineage, edit surface, ranges, mixture, and budget
    same-checkpoint probes      trains parent and children on identical probe subsets
    acceptor + rollback         promotes by fixed gates or restores the parent policy
  three_layer_memory/
    trial records               parent, patch, probe metrics, rejection, and promotion outcome
    mechanism records           recurring estimator behavior under matched problem contexts
    transferable skills         validation-approved evidence for compatible target tasks
    retrieval + transfer gate   matches context and records positive or negative transfer
  experiment_protocol/
    fixed variants              Scratch-Fixed (SF) and Warm-Frozen (WF)
    evolving variants           Scratch-Evolving (SE) and Warm-Evolving (WE)
    validation selection        chooses checkpoints and policies without test-set feedback
    final test + accounting     one held-out evaluation with quality and compute metrics
  evidence_and_provenance/
    protocol + stage logs       configuration, policy states, metrics, and solver accounting
    trials                      probes, diagnostics, rejection reasons, and decisions
    policy record               parent/child identifiers and reproducible policy transition
```

Figure 11: Implementation architecture of EvoMIDFL. The diagram shows the functional modules used during training, validation, and transfer.

F.2 Prompt and Runtime Summary

At each scheduled evolution round, the system constructs a role-specific, validation-safe payload. The Gradient Actor, Pre-Proposal Critic, and Post-Proposal Critic boxes below summarize the prompts used by the language-model components in the formal experiments. The memory box instead documents the deterministic runtime contract used to store and retrieve trials and skills; completed formal runs do not invoke a language model for memory updates. Long run-specific records and retrieved examples are omitted for clarity, while the actual role, inputs, editable fields, checks, and outputs are retained.

Shared Runtime I/O Contract

```
Prompted roles receive only role-specific, training/validation-safe payloads.  The
Pre-Proposal Critic receives a typed diagnostic report and retrieved research excerpts.
The Actor receives the current policy JSON, one typed failure signature, retrieved
research hits, validation-backed skills, past trials, the allowed edit surface, field
constraints, and the remaining query budget.  The Post-Proposal Critic receives the
parent policy, proposed child and typed patch, validation context, edit and protection
```

constraints, predicted rollout cost, and probe horizon. The deterministic memory runtime is not sent a language-model prompt in the completed formal runs.

The Actor and both Critic instances must each return exactly one JSON object and no explanatory prose outside JSON. Every referenced trial or skill must use its archived stable identifier.

Shared protected components: train/validation/test partitions, true labels, forward MILP formulation, integrality and constraints, solver-status and feasibility checks, predictor optimizer, probe schedule, evaluator, and promotion rule. The held-out test partition is never included in a prompt.

Failure handling: malformed JSON, unknown fields, parent-lineage mismatch, unsupported estimator values, out-of-range parameters, protected-component edits, or over-budget requests are rejected before a full same-checkpoint rollout.

Gradient Actor Prompt

Role: EvoMIDFL structured proposal agent; runtime role = research_learning_policy_actor. Workflow: validation-grounded gradient policy evolution.

Goal: propose diverse, bounded patches against the supplied parent gradient policy. Continue the recorded lineage; do not return a replacement training pipeline or modify the forward optimization problem.

Context provided at runtime: stage id, problem signature, current policy JSON, parent policy id, one typed failure signature, retrieved research hits, validation-backed skills, causal past trials, allowed edit surface, field-specific bounds, protected components, and remaining query budget.

Editable surface: active_estimator_set, estimator-level_mixture weights, STE_temperature, perturbation_scale and_direction, estimator_family, perturbation_sigma, direction, query_pairs, regret_weight, fenchel_young_weight, normalization, clipping_norm, and query_budget.

Patch design rules: derive each proposal from the supplied failure evidence or retrieved experience; include a mechanism-level hypothesis; generate alternatives rather than repeatedly changing one scalar; use only registered families and bounds; keep the requested queries within the remaining budget; and never output accept, approve, decision, or promote fields.

Current registered bounds: perturbation sigma in $[10^{-4},2]$; direction in {none, gaussian, row_block, column_block, coordinate}; integer query pairs in [1,16]; regret and Fenchel--Young weights in [0,10]; normalization in {none, l2, per_instance, norm_balance}; clipping norm in (0, 100]; and schema-level query budget in [0,1024], further capped by the round budget (128 in the current formal engine).

Output schema: { "proposals": [{ "changes": { "allowed_field": "value" }, "hypothesis": "mechanistic rationale" }] }. No other top-level field is permitted.

Pre-Proposal Critic Agent Prompt

Role: Pre-proposal Critic (C_1), a read-only diagnostic agent.

Task: diagnose weaknesses of the current parent gradient policy from the supplied training and validation evidence, and identify mechanism-level directions for the Actor to explore.

Inputs: a typed base diagnostic report and retrieved research excerpts. The report contains available validation-regret trends, prediction-error metrics, feasibility, gradient-stability diagnostics, and query or solver accounting.

Rules: use only supplied evidence. Distinguish observed failures from hypotheses. Do not inspect held-out test data, modify policies, review child candidates, or make promotion decisions.

Output: return exactly one JSON object: { "hypotheses": ["Evidence-grounded

explanations"], "recommended_directions": ["Mechanism-level changes worth probing"], "risk_tags": ["Identified risks"] }.

Post-Proposal Critic Agent Prompt

Role: Post-proposal Critic (C_2), a candidate-review agent operating after Actor proposal and before probe training.

Task: review a proposed child policy for admission to an independent same-checkpoint probe. Examine estimator compatibility, parameter configuration, protected fields, and the full training-plus-validation solve budget for the H_p-epoch rollout.

Inputs: the parent policy, proposed child policy and typed patch, validation context, allowed edit surface, protected components, predicted probe solve count, and H_p.

Rules: use only supplied evidence. Do not inspect held-out test data, modify the policy directly, select a validation winner, or promote a candidate. A pass recommendation cannot override deterministic validation. Do not certify nonzero gradients or cosine compatibility without realized probe evidence; these conditions remain subject to runtime checks.

Repair protocol: when structural checks identify an eligible bounded repair, return the candidate to the Actor for one repair attempt, then recheck the repaired candidate using the same Post-Proposal Critic. Unresolved violations lead to rejection.

Output: return exactly one JSON object: { "decision": "pass", "reasons": ["Evidence supporting admission to the probe"], "risk_tags": [] }.

The decision field must be "pass" or "reject". Passing this review authorizes only probe admission; policy promotion remains the responsibility of the independent validation and promotion procedure.

Deterministic Memory Runtime Contract

Role: deterministic validation-grounded memory runtime; implementation = ResearchEvolutionMemoryAgent backed by EvolutionMemoryAgent.
Workflow: post-probe trial recording, skill update, and guarded retrieval.

Update input: one typed TrialRecord containing the immutable trial id, problem signature, training state, failure signature, retrieved skill ids, actor hypothesis, typed policy patch, pre- and post-proposal critic reports, same-checkpoint ProbeResult, deterministic decision, and recorded lesson.

Update operation: record_trial appends the TrialRecord, updates its local SkillCard, and creates family- or cross-problem SkillCards only when their deterministic evidence-count conditions are met. No memory_proposal JSON is generated in the completed formal runs.

Retrieval input: target ProblemSignature, TrainingState, FailureSignature, mechanism prerequisites, target-validation availability, and top-k limits. The runtime reads validation-safe TrialRecords, promoted SkillCards, mechanism knowledge, and transfer-health records; held-out test artifacts are excluded.

Retrieval operation: retain only compatible promoted skills that are not quarantined, rank them by problem match, failure match, stage similarity, evidence quality, and transfer health, and return a scratch fallback when no compatible trial or skill is available. Retrieved evidence is advisory and must be revalidated by a same-checkpoint probe.

Stored records: TrialRecord {trial_id, problem, training_state, failure, retrieved_skill_ids, actor_hypothesis, patch, critic_reports, probe_result, decision, lesson}; SkillCard {skill_id, level, lineage_id, mechanism, prerequisites, evidence_trial_ids, positive_observations, negative_observations, status}, where level is local, family, or cross_problem.