AUTOMATED FEATURE SELECTION ALGORITHM DESIGN AS SEQUENTIAL DECISION MAKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Automated feature selection algorithm design aims to generate algorithms tailored to individual datasets. However, how to construct and iteratively refine executable feature selection algorithms using dataset analysis and evaluation feedback remains underexplored. Fixed algorithm pools restrict the available design decisions, while difficulties in obtaining reliable evaluation feedback hinder iterative refinement. To address this problem, we propose FSDESIGN, which decouples algorithm design into an environment that supports design and an LLM-based designer agent that makes design decisions. Their interaction constitutes the algorithm design process, which we formulate as sequential decision making. The environment provides an action space in which algorithms are constructed by composing atomic operators, enabling adaptation beyond a predefined pool of complete algorithms. It also evaluates candidates by jointly considering result profiles and downstream predictive metrics, providing comprehensive feedback for iterative refinement. Based on this feedback and the accumulated interaction history, the designer agent repeatedly constructs and refines candidate algorithms. We conduct a series of experiments, demonstrating the effectiveness of FSDESIGN.

1 INTRODUCTION

Feature selection is a fundamental step in high-dimensional data analysis (Guyon & Elisseeff, 2003), yet the effectiveness of a feature selection algorithm often depends on the characteristics of the target dataset (Parmezan et al., 2021). Different data may exhibit different characteristics, such as feature redundancy (Peng et al., 2005; Han et al., 2024), nonlinear feature–label dependencies (Yamada et al., 2014) or label imbalance (Charte et al., 2015), making a predefined selection mechanism unlikely to be universally suitable (Wang et al., 2013). This has motivated increasing interest in automated feature selection algorithm construction for individual datasets, where dataset characteristics are analyzed and used to construct an algorithm adapted to them (Li et al., 2026).

Although dataset analysis can provide useful theoretical guidance for algorithm design, the resulting algorithms may not perform as expected on complex real-world data. In such datasets, feature redundancy, irrelevant features (Zhang et al., 2021), and label imbalance may coexist. While dataset analysis can help identify these issues and guide the design of algorithmic components tailored to each, these insights alone may not reliably predict whether the resulting algorithm will deliver the intended benefits in practice. Therefore, a central question is **how to enable fine-grained construction and iterative refinement of feature selection algorithms, with dataset characteristics and empirical evidence from candidate evaluations jointly guiding subsequent design decisions** (Liu et al., 2024; Ye et al., 2024).

Some approaches construct algorithms from dataset diagnostics without further refinement from evaluation feedback (Li et al., 2026). Other methods repeatedly evaluate component configurations or generated programs (Ibrahim et al., 2021; Ding et al., 2025), but face different constraints in supporting a sequential design process for feature selection. (1) Some methods select a suitable algorithm from a fixed pool of predefined candidates (Wang et al., 2013; Neutatz et al., 2021) (Figure 1(a)), but their performance is inherently constrained by the coverage and quality of the pool itself. Such decisions provide coarse-grained matching, while adapting individual mechanisms requires manually extending the pool. (2) General-purpose algorithm design methods are often evaluated on tasks with explicit objectives, such as minimizing tour length or maximizing knapsack

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.

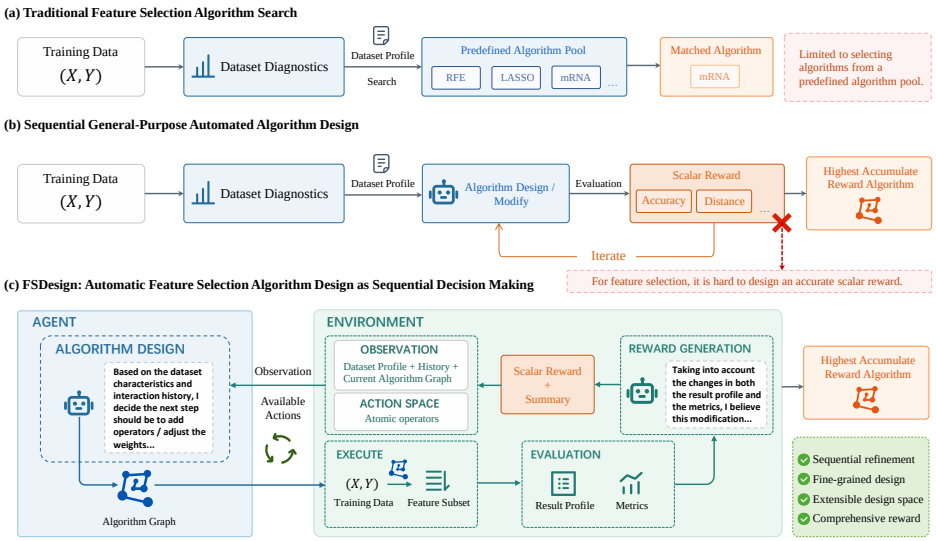


Figure 1: **Conceptual comparison of automated algorithm design paradigms.** (a) Dataset-guided selection from a predefined algorithm pool. (b) Sequential refinement using explicit task-specific rewards. (c) Our agent–environment framework sequentially constructs and refines algorithm graphs using composable atomic operators and comprehensive rewards derived from result profiles and predictive metrics.

value, which provide direct feedback for iterative refinement (Ye et al., 2024) (Figure 1(b)). However, feature selection is often assessed through multiple predictive criteria with different implications and trade-offs (Wu & Zhou, 2017), making it difficult to derive a single definitive signal for judging revisions and guiding subsequent decisions. As a result, how to model an effective sequential algorithm-design process tailored to feature selection remains underexplored.

To address these challenges, we propose FSDESIGN, an environment–designer framework for constructing and iteratively refining executable feature-ranking algorithms (Figure 1(c)). We decouple the design environment from the designer agent and formulate their interaction as sequential decision making: the environment defines a compositional action space and provides evaluation feedback, while an LLM-based designer uses this feedback, dataset characteristics, and interaction history to construct and revise candidate algorithms. Rather than restricting design to a manually curated pool of complete algorithms, the environment provides an action space for composing atomic operators derived from common feature selection considerations. This representation enables fine-grained adaptation and is naturally extensible: adding a new operator introduces a combinatorial family of new candidate designs through its compositions with existing operators. To provide informative feedback beyond a single predictive metric, candidate evaluation combines result profiles, which characterize selected-subset properties such as redundancy, with predictive metrics that measure downstream performance. An LLM integrates these complementary signals into a comprehensive reward. Using dataset characteristics, interaction history, and reward feedback, the designer agent selects subsequent design actions. This repeated interaction enables sequential exploration and refinement, ultimately returning the candidate with the highest accumulated reward as the final feature selection algorithm. We select multi-label feature selection as the specific feature selection task and conduct a series of experiments to validate the effectiveness of FSDESIGN.

In summary, our contributions are:

- We formulate automated feature selection algorithm design as sequential decision making, explicitly decoupling the environment that supports design from the agent that makes design decisions.
- We develop an environment to support automated feature selection algorithm design. It provides an atomic action space for fine-grained algorithm design and offers comprehensive evaluation to provide more reliable guidance for iterative improvement.
- We develop a designer agent tailored to the environment to carry out feature selection algorithm design.

- We select multi-label feature selection as a representative research setting and design a series of experiments to validate the effectiveness of our method.

2 Related Work

2.1 Automated Feature Selection Algorithm Design

Automated feature selection ranges from recommending existing algorithms to constructing new procedures. Dataset-based recommendation selects methods from a predefined pool (Wang et al., 2013; Neutatz et al., 2021). Recent LLM-based methods extend adaptation to algorithm construction: FeatureHospital derives optimization strategies from dataset diagnostics (Li et al., 2026), and EvoVLMA evolves executable feature selection programs through repeated evaluation for vision–language adaptation (Ding et al., 2025). These methods progressively increase the flexibility of automated design, but existing feature-selection-specific approaches either remain dependent on predefined design spaces or do not explicitly organize feature selection design around a general sequential process that repeatedly uses task-specific empirical feedback for refinement.

2.2 Iterative Automatic Algorithm Design

General automatic algorithm design increasingly uses evaluation to guide successive revisions. FunSearch evolves executable programs using evaluator scores (Romera-Paredes et al., 2024), and EoH jointly evolves algorithmic ideas and implementations (Liu et al., 2024). ReEvo incorporates reflection over evaluated candidates (Ye et al., 2024), while MCTS-AHD organizes exploration through a search tree (Zheng et al., 2025). Together, these methods establish empirical feedback and structured exploration as useful ingredients for iterative design. Their common benchmarks often provide direct optimization objectives, whereas feature selection requires interpreting complementary predictive criteria (Wu & Zhou, 2017) alongside properties of the selected features.

3 Preliminaries

3.1 Multi-Label Feature Selection

Let $\mathcal{D} = (X, Y)$ denote a dataset with n instances, d input features, and L labels, where $X \in \mathbb{R}^{n \times d}$ is the feature matrix and $Y \in \{0, 1\}^{n \times L}$ is the binary label matrix. Let $\mathcal{F} = \{f_1, \ldots, f_d\}$ denote the set of features. Given a feature budget $k < d$, multi-label feature selection aims to identify a subset

$$S \subseteq \mathcal{F}, \qquad |S| = k, \tag{1}$$

that preserves informative structure for predicting the multiple labels while removing irrelevant or redundant features. The quality of S is typically assessed through downstream multi-label prediction according to task-specific evaluation criteria.

3.2 Feature-Ranking Algorithm as the Design Target

Instead of directly optimizing a feature subset for a particular budget, we take a *feature-ranking algorithm* as the design target. Let G denote such an algorithm. Given the training data (X, Y), G produces a complete ordering of all d features,

$$\sigma_G = G(X, Y) = \big(f_{(1)}, f_{(2)}, \ldots, f_{(d)}\big), \tag{2}$$

where σ_G denotes the ranking produced by G, and $f_{(i)}$ is the feature ranked at position i. For any required feature budget k, the corresponding subset is obtained from the first k ranked features:

$$S_k(G) = \big\{f_{(1)}, \ldots, f_{(k)}\big\}. \tag{3}$$

4 Method

Figure 2 illustrates the overall workflow of FSDesign.

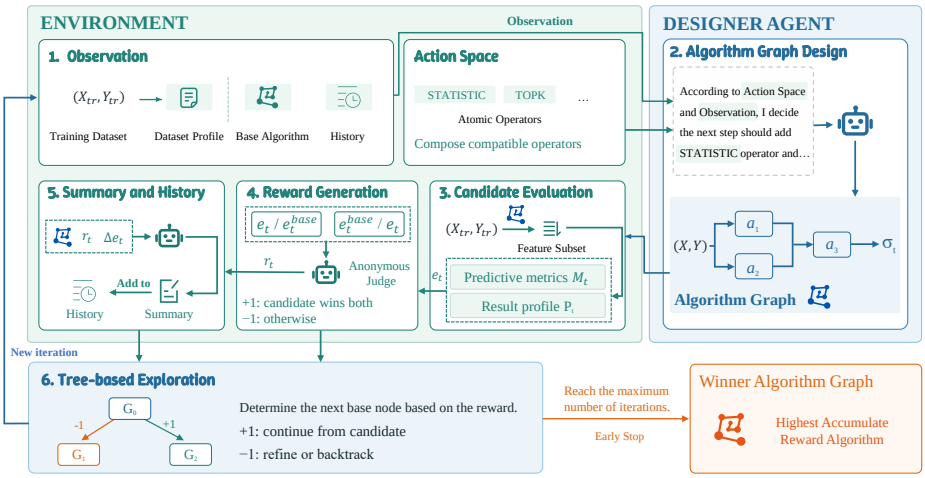


Figure 2: Workflow of FSDESIGN. The environment provides training-data analysis, design history, and an action space. Guided by this information, the agent designs a candidate algorithm. Then, the environment evaluates the candidate and generates a reward indicating whether it is judged to improve upon the base algorithm. The round is then summarized, and tree-based exploration guides the next round. If an early-stopping condition is met or the maximum number of rounds is reached, an algorithm with the highest accumulated reward is returned.

4.1 FEATURE SELECTION ALGORITHM DESIGN ENVIRONMENT

4.1.1 OBSERVATION

Let $\mathcal{D}_{\text{tr}} = (X_{\text{tr}}, Y_{\text{tr}})$ denote the training partition available for algorithm design. The dataset profile is computed solely from $\mathcal{D}_{\text{tr}}$; held-out test data are excluded from profile construction and design decisions. At design step t, the designer observes this training-data profile $\phi(\mathcal{D}_{\text{tr}})$, the base algorithm G_t^{base} and its evidence e_t^{base}, and the accumulated interaction history $\mathcal{H}_t$:

$$o_t = \left[\phi(\mathcal{D}_{\text{tr}}), G_t^{\text{base}}, e_t^{\text{base}}, \mathcal{H}_t\right]. \tag{4}$$

The dataset profile summarizes intrinsic properties of the data, including data scale, label prevalence and cardinality, sparsity, and feature correlation. The detailed construction of the dataset profile is provided in Appendix B.1. The base is a previously evaluated graph selected by the exploration strategy and provides the reference for graph revision and reward generation.

4.1.2 ACTION SPACE

Rather than selecting a complete algorithm from a manually curated pool, the environment constructs candidate algorithms by composing atomic operators. Each candidate is represented as a typed directed acyclic graph $G = (V, E)$. Each computational node in V is instantiated by an atomic operator from a predefined operator set $\mathcal{A}$, and edges in E encode data dependencies between nodes with compatible output and input types.

The operator set is built from common operations in feature selection, including feature statistics, feature–label dependence, feature–feature redundancy, reduction, normalization, score combination, and ranking. A valid graph takes (X, Y) as input and produces a complete feature ranking,

$$\sigma_G = G(X, Y). \tag{5}$$

By selecting atomic operators, the designer can construct algorithms with different computational structures at a finer granularity than choosing among complete predefined methods. The complete definition of the action space and the atomic operator set is provided in Appendix B.2.

4.1.3 CANDIDATE EVALUATION

To provide a more comprehensive assessment of each candidate algorithm G_t, the environment evaluates it from two complementary perspectives: downstream predictive performance and the properties of the selected features. Accordingly, the evaluation evidence is defined as

$$e_t = (M_t, P_t), \tag{6}$$

where M_t denotes the downstream performance metrics and P_t denotes the result profile.

For the downstream performance component M_t, we employ repeated cross-validation to reduce the sensitivity of candidate assessment to a particular train–validation split and obtain a more stable evaluation signal (Krstajic et al., 2014). Let R denote the number of repeats, K the number of folds in each repeat, and $\mathcal{B} = \{\rho_1, \ldots, \rho_m\}$ the set of feature-selection rates used for evaluation. For repeat r and fold f, the training partition $\mathcal{D}_{\text{tr}}$ is divided into an internal training fold $\mathcal{D}_{\text{tr}}^{(r,f)}$ and a validation fold $\mathcal{D}_{\text{val}}^{(r,f)}$.

To compute each feature ranking using only the corresponding internal training fold, G_t is executed independently on that fold:

$$\sigma_t^{(r,f)} = G_t\left(X_{\text{tr}}^{(r,f)}, Y_{\text{tr}}^{(r,f)}\right). \tag{7}$$

For each selection rate $\rho \in \mathcal{B}$, the number of retained features is

$$k_\rho = \max(1, \lceil \rho d \rceil), \tag{8}$$

where d is the total number of input features. The first k_ρ features in $\sigma_t^{(r,f)}$ are selected, and a downstream multi-label predictor is trained on the corresponding training-fold features and evaluated on the validation fold.

Let $M_j^{(r,f,\rho)}(G_t)$ denote the value of the j-th predefined downstream evaluation metric obtained at repeat r, fold f, and selection rate ρ. Each metric is independently aggregated by equally averaging over all evaluation cells:

$$\overline{M}_j(G_t) = \frac{1}{RK|\mathcal{B}|} \sum_{r=1}^{R} \sum_{f=1}^{K} \sum_{\rho \in \mathcal{B}} M_j^{(r,f,\rho)}(G_t). \tag{9}$$

The downstream performance component is then

$$M_t = \left[\overline{M}_1(G_t), \ldots, \overline{M}_q(G_t)\right], \tag{10}$$

where q denotes the number of predefined evaluation metrics. The metric set is manually specified before algorithm design and kept consistent with the final evaluation criteria. In addition, these metrics are retained separately rather than manually aggregated into a weighted scalar score. They are jointly interpreted by the reward mechanism described in the following section.

Complementary to downstream performance, the result profile P_t characterizes the properties of the feature subsets induced by G_t. It summarizes information such as feature–label relationships, label coverage, feature redundancy, and ranking-related structure. Unlike M_t, which measures predictive performance on validation data, P_t provides diagnostic evidence about how the candidate changes the selected features themselves. The detailed construction of the result profile is provided in Appendix B.3.

4.1.4 REWARD GENERATION

An LLM judge compares the candidate G_t with its base G_t^{base} using only their anonymized evaluation evidence e_t and e_t^{base}. It jointly interprets predictive metrics and result profiles without a manually weighted scalar objective. Let $\mathcal{J}(e_a, e_b) \in \{G_a, G_b\}$ denote its preference. To reduce positional bias (Zheng et al., 2023), we reverse the presentation order in a second comparison and assign a positive reward only if both comparisons prefer the candidate:

$$r_t = \begin{cases} +1, & \mathcal{J}\left(e_t, e_t^{\text{base}}\right) = G_t \ \wedge \ \mathcal{J}\left(e_t^{\text{base}}, e_t\right) = G_t, \\ -1, & \text{otherwise.} \end{cases} \tag{11}$$

The reward expresses a comparison with the current base. A negative reward includes disagreement between the mirrored judgments and therefore does not imply that the candidate is worse on every metric. Such candidates remain informative and may still be refined by the navigation policy.

4.1.5 SUMMARY AND HISTORY

After evaluation, the environment summarizes the design intention, metric and profile changes relative to the base ($\Delta M_t, \Delta P_t$), and the frozen reward. Let η_t contain the proposal rationale and frozen comparison records. The summary is stored together with the candidate graph:

$$\xi_t = \Gamma(\eta_t, \Delta M_t, \Delta P_t, r_t), \qquad \mathcal{H}_{t+1} = \mathcal{H}_t \,\|\, [(G_t, \xi_t)]. \tag{12}$$

Here Γ combines an LLM-written purpose sentence with rule-based formatting of measured changes and the frozen reward, and $\|$ denotes chronological concatenation. This procedure preserves the evaluation and reward; Appendix B.4 details its construction. Subsequent observations expose this history as in-context experience.

4.2 DESIGNER AGENT

The Designer Agent performs the actual feature-ranking algorithm design within the environment. At design step t, the agent receives the current observation o_t, together with the operator set $\mathcal{A}$, and generates a candidate algorithm graph

$$G_t = \mu(o_t, \mathcal{A}), \tag{13}$$

where $\mu(\cdot)$ denotes the LLM-based design policy. The observation provides the dataset profile, the base algorithm and its evaluation evidence, and the accumulated interaction history, while $\mathcal{A}$ and the composition constraints specify the available operations for algorithm construction.

Based on this information, the agent analyzes the current design state, identifies a potential direction for improvement, and constructs a new algorithm graph by selecting and composing atomic operators. In this way, the agent performs decision making over the environment-defined design space rather than generating an unconstrained algorithm from scratch.

4.3 TREE-BASED EXPLORATION

To support efficient exploration of the compositional algorithm design space, we organize evaluated algorithm graphs into a search tree

$$\mathcal{T}_t = (\mathcal{N}_t, \mathcal{L}_t)\,, \tag{14}$$

where $\mathcal{N}_t$ contains the algorithm graphs evaluated up to design step t, and $\mathcal{L}_t$ contains the corresponding design transitions. Each node therefore represents an algorithm graph. When the Designer Agent iterates on an existing algorithm to generate a new candidate, the existing algorithm becomes the parent node and the newly generated algorithm becomes its child node. The initial algorithm G_0 serves as the root of the tree, and each edge is associated with the signed reward obtained for the corresponding design transition.

The reward determines how the search proceeds after each iteration. When a newly generated candidate G_t receives a positive reward, it becomes the base algorithm for the next iteration. When $r_t = -1$, an LLM-based navigation decision determines whether to continue refining G_t or roll back to its parent and explore an alternative direction. The decision is made according to the evaluation outcome of the current transition and the existing search context. Therefore, a negatively rewarded candidate is not necessarily discarded, since its observed failure may still motivate a targeted subsequent modification.

To prevent excessive exploration of an unproductive region, let $c^-(G)$ denote the number of negatively rewarded children generated from node G, and let τ_{fail} denote a predefined failure threshold. A node is marked as exhausted when

$$c^-(G) \geq \tau_{\text{fail}}. \tag{15}$$

An exhausted node is no longer used as a base for further exploration, and the search backtracks to its nearest non-exhausted ancestor. If no such ancestor remains, the search tree is considered exhausted and the design process terminates early. We additionally define T as the maximum number of evaluated design iterations; hence, exploration terminates when either the tree is exhausted or the design budget T is reached. Let $t_{\text{end}} \leq T$ denote the actual design step at which exploration terminates.

Table 1: Comparison on MacroAP and Jaccard. Dataset results are mean ± standard deviation over five seeds. Average is the unweighted mean of dataset means. Higher is better. The best and second-best means, ranked before rounding, are shown in **bold** and underlined, respectively.

Dataset	Ours	Feature Hospital	DFS	MMFS	Deep-Lasso	RLBMLFS	GRROOR	SSFS	Sequential-Attention
					MacroAP ↑				
Bibtex	**0.2221 ± 0.0092**	0.1401 ± 0.0319	0.1037 ± 0.0138	0.1043 ± 0.0035	0.2087 ± 0.0022	0.1082 ± 0.0086	0.0889 ± 0.0144	0.1205 ± 0.0031	0.1478 ± 0.0056
Birds	**0.2954 ± 0.0324**	0.2763 ± 0.0104	0.2508 ± 0.0229	0.2872 ± 0.0111	0.2414 ± 0.0123	0.2666 ± 0.0134	0.2427 ± 0.0272	0.2707 ± 0.0278	0.2603 ± 0.0251
Emotions	0.6206 ± 0.0161	0.6106 ± 0.0209	0.5857 ± 0.0228	**0.6391 ± 0.0077**	0.6160 ± 0.0250	0.5729 ± 0.0445	0.5556 ± 0.0240	0.5269 ± 0.0190	0.5548 ± 0.0267
Genbase	0.8917 ± 0.0514	**0.8955 ± 0.0587**	0.7573 ± 0.1321	0.6034 ± 0.0431	0.8445 ± 0.0443	0.8939 ± 0.0565	0.8289 ± 0.0577	0.8901 ± 0.0545	0.8641 ± 0.0488
Medical	**0.4673 ± 0.0404**	0.3702 ± 0.0483	0.3922 ± 0.0435	0.2627 ± 0.0177	0.1181 ± 0.0185	0.3782 ± 0.0285	0.2299 ± 0.0424	0.3297 ± 0.0202	0.3456 ± 0.0422
Scene	0.6562 ± 0.0313	0.3608 ± 0.0404	0.5184 ± 0.0352	**0.7169 ± 0.0086**	0.5820 ± 0.0263	0.5464 ± 0.0257	0.6447 ± 0.0283	0.4802 ± 0.0102	0.7022 ± 0.0205
Yeast	0.4207 ± 0.0051	0.4127 ± 0.0080	0.4021 ± 0.0053	**0.4215 ± 0.0047**	0.4197 ± 0.0034	0.4060 ± 0.0111	0.3877 ± 0.0154	0.3684 ± 0.0070	0.3865 ± 0.0074
Average	**0.5106**	0.4380	0.4300	0.4336	0.4329	0.4532	0.4255	0.4266	0.4659
					Jaccard ↑				
Bibtex	**0.2206 ± 0.0111**	0.1086 ± 0.0420	0.1000 ± 0.0135	0.0907 ± 0.0023	0.1889 ± 0.0101	0.0675 ± 0.0068	0.0583 ± 0.0201	0.1254 ± 0.0068	0.1591 ± 0.0067
Birds	**0.1632 ± 0.0174**	0.1483 ± 0.0191	0.0952 ± 0.0131	0.1313 ± 0.0190	0.1021 ± 0.0255	0.1118 ± 0.0158	0.1101 ± 0.0188	0.1105 ± 0.0089	0.1094 ± 0.0124
Emotions	0.4450 ± 0.0167	0.4400 ± 0.0163	0.3989 ± 0.0122	**0.4670 ± 0.0119**	0.4428 ± 0.0101	0.3950 ± 0.0431	0.3498 ± 0.0330	0.3146 ± 0.0187	0.3655 ± 0.0305
Genbase	**0.9300 ± 0.0120**	0.9020 ± 0.0387	0.8585 ± 0.0835	0.6849 ± 0.0401	0.7811 ± 0.0258	0.9059 ± 0.0222	0.8593 ± 0.0286	0.9262 ± 0.0166	0.8687 ± 0.0367
Medical	**0.5814 ± 0.0517**	0.3713 ± 0.1227	0.5577 ± 0.0377	0.4071 ± 0.0307	0.1598 ± 0.0605	0.4539 ± 0.0300	0.3665 ± 0.0477	0.3841 ± 0.0068	0.3781 ± 0.0401
Scene	0.4681 ± 0.0301	0.1462 ± 0.0324	0.3241 ± 0.0389	**0.5221 ± 0.0058**	0.3762 ± 0.0283	0.3150 ± 0.0307	0.4206 ± 0.0269	0.2588 ± 0.0093	0.5090 ± 0.0159
Yeast	0.4440 ± 0.0043	0.4401 ± 0.0058	0.4224 ± 0.0065	**0.4501 ± 0.0061**	0.4459 ± 0.0081	0.4311 ± 0.0059	0.4184 ± 0.0155	0.4059 ± 0.0030	0.4149 ± 0.0168
Average	**0.4646**	0.3652	0.3938	0.3933	0.3567	0.3829	0.3690	0.3608	0.4007

To evaluate the overall quality of each explored design path, we define the accumulated reward of an algorithm graph G as the sum of the edge rewards along the path from the root G_0 to G:

$$Q(G) = \sum_{\ell \in \mathrm{Path}(G_0, G)} r(\ell), \tag{16}$$

where $r(\ell)$ denotes the signed reward associated with edge ℓ. After exploration terminates, the candidate set with the highest accumulated reward is

$$\mathcal{G}^* = \arg \max_{G \in \mathcal{N}_{t_{\mathrm{end}}}} Q(G). \tag{17}$$

If $\mathcal{G}^*$ contains a single candidate, it is directly returned as the final feature-ranking algorithm. If multiple candidates share the same maximum accumulated reward, an LLM-based tie breaker compares only the candidates in $\mathcal{G}^*$ using their algorithm graphs and evaluation evidence, and selects one of them as the final algorithm G^*. The tie-breaking procedure does not modify the rewards or the explored search tree.

5 Experiments

5.1 Experimental Setup

Evaluation Protocol. We evaluate all methods on seven multi-label datasets: Bibtex, Birds, Emotions, Genbase, Medical, Scene, and Yeast. Each method is evaluated at selection rates from 2% to 20% in increments of 2%, using the same downstream predictor, data splits, and evaluation procedure. For FSDesign, we use gpt-5-mini (OpenAI, 2025) as the LLM for both the designer and the judge. More details about the experimental setup and hyperparameters are provided in Appendix C.1.

Evaluation Metrics. We report Macro Average Precision (MacroAP), Jaccard score, Ranking Loss (RL), and Hamming Loss (HL) as evaluation metrics. The detail of metrics are illstrated in Appendix C.2.

Comparison Methods. We compare FSDesign with two kinds of methods: **(1) automated feature selection algorithm construction methods**: FeatureHospital (Li et al., 2026) and DFS (Neutatz et al., 2021); and **(2) MLFS baselines**: RLBMLFS (Faraji et al., 2026), GRROOR (Xu et al., 2025), SSFS (Gao et al., 2023), Deep-Lasso (Cherepanova et al., 2023), Sequential-Attention (Yasuda et al., 2023), and MMFS (Dong et al., 2020). MMFS provides a comparison with explicit multi-objective optimization, as FSDesign also considers multiple evaluation criteria. The implement detail of these methods are provided in Appendix C.3.

Table 2: Comparison on Ranking Loss (RL) and Hamming Loss (HL). Dataset results are mean ± standard deviation over five seeds. Average is the unweighted mean of dataset means. Lower is better. The best and second-best means, ranked before rounding, are shown in **bold** and underlined, respectively.

Dataset	Ours	Feature Hospital	DFS	MMFS	Deep-Lasso	RLBMLFS	GRROOR	SSFS	Sequential-Attention
					RL ↓				
Bibtex	**0.3604 ± 0.0103**	0.4529 ± 0.0562	0.4897 ± 0.0252	0.5256 ± 0.0060	0.3834 ± 0.0060	0.5288 ± 0.0070	0.5407 ± 0.0314	0.4664 ± 0.0085	0.4409 ± 0.0099
Birds	0.3456 ± 0.0328	0.3639 ± 0.0300	0.3619 ± 0.0238	0.3349 ± 0.0293	0.3856 ± 0.0289	0.3709 ± 0.0235	0.3742 ± 0.0245	**0.3180 ± 0.0251**	0.3350 ± 0.0317
Emotions	0.2303 ± 0.0135	0.2394 ± 0.0154	0.2622 ± 0.0049	**0.2264 ± 0.0067**	0.2390 ± 0.0143	0.2781 ± 0.0315	0.2952 ± 0.0165	0.3037 ± 0.0154	0.2907 ± 0.0216
Genbase	0.0134 ± 0.0038	0.0160 ± 0.0120	0.0260 ± 0.0222	0.0841 ± 0.0248	0.0526 ± 0.0061	0.0153 ± 0.0085	0.0269 ± 0.0112	**0.0122 ± 0.0061**	0.0234 ± 0.0094
Medical	**0.0977 ± 0.0097**	0.1441 ± 0.0354	0.1263 ± 0.0124	0.2030 ± 0.0097	0.3990 ± 0.0655	0.1201 ± 0.0196	0.1950 ± 0.0338	0.1475 ± 0.0203	0.1321 ± 0.0273
Scene	0.1669 ± 0.0170	0.3641 ± 0.0198	0.2444 ± 0.0241	**0.1446 ± 0.0057**	0.2240 ± 0.0169	0.2427 ± 0.0205	0.1821 ± 0.0171	0.2843 ± 0.0094	0.1525 ± 0.0157
Yeast	0.2363 ± 0.0031	0.2432 ± 0.0028	0.2504 ± 0.0042	0.2350 ± 0.0045	**0.2345 ± 0.0049**	0.2435 ± 0.0047	0.2503 ± 0.0099	0.2627 ± 0.0045	0.2519 ± 0.0086
Average	**0.2072**	0.2605	0.2516	0.2505	0.2740	0.2571	0.2663	0.2564	0.2324
					HL ↓				
Bibtex	**0.0128 ± 0.0001**	0.0140 ± 0.0006	0.0139 ± 0.0002	0.0142 ± 0.0001	0.0131 ± 0.0002	0.0147 ± 0.0002	0.0147 ± 0.0003	0.0136 ± 0.0001	0.0134 ± 0.0001
Birds	**0.0463 ± 0.0028**	0.0474 ± 0.0016	0.0479 ± 0.0052	0.0484 ± 0.0024	0.0499 ± 0.0020	0.0484 ± 0.0026	0.0497 ± 0.0024	0.0500 ± 0.0021	0.0494 ± 0.0026
Emotions	0.2248 ± 0.0085	0.2277 ± 0.0111	0.2401 ± 0.0032	**0.2184 ± 0.0074**	0.2271 ± 0.0095	0.2444 ± 0.0206	0.2561 ± 0.0122	0.2673 ± 0.0061	0.2525 ± 0.0069
Genbase	**0.0061 ± 0.0009**	0.0075 ± 0.0021	0.0092 ± 0.0040	0.0193 ± 0.0013	0.0119 ± 0.0013	0.0076 ± 0.0015	0.0104 ± 0.0023	0.0063 ± 0.0013	0.0097 ± 0.0017
Medical	**0.0153 ± 0.0016**	0.0209 ± 0.0035	0.0160 ± 0.0011	0.0192 ± 0.0012	0.0241 ± 0.0014	0.0180 ± 0.0008	0.0211 ± 0.0017	0.0198 ± 0.0005	0.0202 ± 0.0013
Scene	0.1235 ± 0.0065	0.1730 ± 0.0041	0.1520 ± 0.0064	**0.1122 ± 0.0013**	0.1374 ± 0.0051	0.1464 ± 0.0067	0.1284 ± 0.0052	0.1534 ± 0.0015	0.1144 ± 0.0044
Yeast	0.2143 ± 0.0017	0.2149 ± 0.0023	0.2238 ± 0.0022	**0.2137 ± 0.0023**	0.2148 ± 0.0027	0.2198 ± 0.0007	0.2237 ± 0.0042	0.2271 ± 0.0025	0.2231 ± 0.0049
Average	**0.0919**	0.1008	0.1004	0.0922	0.0969	0.0999	0.1006	0.1054	0.0975

5.2 Comparison Experiment

Overall Effectiveness. Tables 1 and 2 show that FSDesign achieves the highest average MacroAP and Jaccard and the lowest average RL and HL, supporting its overall effectiveness across the evaluated datasets. It also improves on both automated design baselines in these averages. Relative to DFS, which selects strategies from a predefined pool, the results suggest benefits from composing atomic operators into candidate algorithms. The comparison with the one-shot customization of FeatureHospital is consistent with the value of refining designs through repeated iteration.

Dataset-specific Suitability. The relative performance of existing methods varies across datasets. For example, Deep-Lasso attains the best RL on Yeast and nearly matches the best MacroAP and Jaccard, yet is less competitive on Medical. Similarly, MMFS achieves strong performance on several datasets, but exhibits marked performance degradation on Bibtex and Medical. This contrast suggests that an algorithm well suited to one dataset may be less suitable for another. By conditioning graph revisions on the training-data profile and evaluation feedback, FSDesign generally achieves best or near-best performance across datasets, without the pronounced performance drops observed for some baselines. Together with its average performance, this provides indirect evidence for the value of adapting algorithms to individual datasets.

5.3 Ablation Study

We evaluate six variants on Bibtex, Scene, and Birds. The evaluation result is illustrated in Table 3.

Ablation Settings. Full denotes the complete FSDesign. **w/o Tree** replaces tree-based exploration with a linear trajectory. **w/o Profile** removes both dataset and result profiles, retaining basic dataset-scale information and predictive metrics for reward generation. **w/o Metrics** removes predictive metrics from reward generation and judges candidates only from result profiles. **w/o History** hides interaction history from the designer, preserving exploration, stopping, and final-selection rules. **Fixed** uses a manually specified graph from the same action space, computing Pearson feature–label dependence, averaging over labels, and ranking with TOPK. **One-shot** generates an algorithm once from the dataset profile and action space, without iterative evaluation or refinement. Appendix C.6 gives the complete settings.

Analysis. (1) **w/o Tree**: Full generally performs better, suggesting that more effective exploration improves algorithm design; Birds RL is an exception. (2) **w/o Profile**: The performance drop supports statistical analysis of the dataset and selected features as useful guidance for algorithm design. (3) **w/o Metrics**: The performance decline highlight the importance of downstream feedback alongside statistical diagnostics. (4) **w/o History**: The substantial degradation is consistent with the importance of accumulating experience during iterative design. (5) **Fixed**: It is relatively competitive on Birds

Table 3: Ablation results on Bibtex, Scene, and Birds. Dataset results are mean ± standard deviation over five seeds. ↑ and ↓ indicate that higher and lower values are better, respectively. The best and second-best results for each dataset and metric are shown in **bold** and underlined, respectively.

Dataset	Method	MacroAP ↑	Jaccard ↑	RL ↓	HL ↓
Bibtex	**Full**	**0.2221 ± 0.0092**	**0.2206 ± 0.0111**	**0.3604 ± 0.0103**	**0.0128 ± 0.0001**
	w/o Tree	0.2047 ± 0.0144	0.1924 ± 0.0253	0.3655 ± 0.0087	0.0131 ± 0.0002
	w/o Profile	0.2104 ± 0.0154	0.2002 ± 0.0220	0.3634 ± 0.0095	0.0130 ± 0.0002
	w/o Metrics	0.1980 ± 0.0069	0.1830 ± 0.0193	0.3697 ± 0.0080	0.0131 ± 0.0002
	w/o History	0.1313 ± 0.0797	0.1090 ± 0.0914	0.5035 ± 0.1331	0.0139 ± 0.0009
	Fixed	0.1943 ± 0.0026	0.1725 ± 0.0048	0.3728 ± 0.0034	0.0132 ± 0.0002
	One-shot	0.0962 ± 0.0521	0.0703 ± 0.0628	0.5565 ± 0.0983	0.0143 ± 0.0007
Scene	**Full**	**0.6562 ± 0.0313**	**0.4681 ± 0.0301**	**0.1669 ± 0.0170**	**0.1235 ± 0.0065**
	w/o Tree	0.5373 ± 0.0630	0.3270 ± 0.0825	0.2303 ± 0.0349	0.1487 ± 0.0127
	w/o Profile	0.5249 ± 0.0539	0.3081 ± 0.0634	0.2387 ± 0.0304	0.1515 ± 0.0102
	w/o Metrics	0.5018 ± 0.0226	0.2814 ± 0.0271	0.2514 ± 0.0167	0.1555 ± 0.0036
	w/o History	0.4421 ± 0.1283	0.2240 ± 0.1447	0.3059 ± 0.0950	0.1622 ± 0.0196
	Fixed	0.4933 ± 0.0080	0.2702 ± 0.0057	0.2565 ± 0.0072	0.1568 ± 0.0015
	One-shot	0.3968 ± 0.0857	0.1654 ± 0.0918	0.3385 ± 0.0635	0.1684 ± 0.0125
Birds	Full	**0.2954 ± 0.0324**	**0.1632 ± 0.0174**	0.3456 ± 0.0328	**0.0463 ± 0.0028**
	w/o Tree	0.2810 ± 0.0270	0.1087 ± 0.0430	0.3321 ± 0.0293	0.0491 ± 0.0030
	w/o Profile	0.2794 ± 0.0138	0.1553 ± 0.0214	0.3602 ± 0.0286	0.0473 ± 0.0017
	w/o Metrics	0.2615 ± 0.0420	0.0889 ± 0.0388	**0.3184 ± 0.0149**	0.0500 ± 0.0024
	w/o History	0.2385 ± 0.0484	0.0787 ± 0.0353	0.3773 ± 0.1079	0.0502 ± 0.0011
	Fixed	0.2824 ± 0.0147	0.1553 ± 0.0214	0.3565 ± 0.0271	0.0472 ± 0.0017
	One-shot	0.2478 ± 0.0422	0.0951 ± 0.0305	0.3602 ± 0.0469	0.0498 ± 0.0014

but less competitive on Bibtex and Scene, suggesting that fixed algorithms can work well on suitable datasets yet deteriorate when poorly matched. (6) **One-shot**: Full consistently performs better, supporting the value of iterative refinement over one-shot algorithm design.

5.4 Case Study

We manually inspect several cases and present three examples that provide qualitative evidence for the effectiveness of FSDesign.

Case 1: Generated algorithm graph. The final algorithm graph illustrates that FSDesign can compose atomic operators into a coherent and interpretable feature-ranking procedure.

Case 2: Metrics-only versus comprehensive judgment. In a case where a simple metric-only aggregation heuristic favors a candidate, comprehensive judgment and retrospective inspection support rejecting it. This illustrates the value of result profiles in interpreting mixed predictive outcomes.

Case 3: Profile-only versus comprehensive judgment. Several profile changes could favor a candidate under a profile-only reading, whereas comprehensive judgment and retrospective inspection favor its parent. This highlights the importance of relating structural diagnostics to downstream predictive performance.

The complete analyses of all three cases are provided in Appendix D.4.

We provide additional experiments, including LLM cost analysis, statistical significance analysis, and performance across feature selection rates, in Appendix D.

6 Conclusion

We formulate feature selection algorithm process as sequential interaction between an explicit environment and a history-conditioned agent. FSDesign combines executable candidate programs, comparable evaluation evidence, inspectable rewards, and persistent exploration decisions. Our framework supports fine-grained algorithm construction through atomic operator composition and guides iterative refinement with comprehensive evaluation feedback.

AI USE STATEMENT

We used generative AI tools for language polishing, code development, and assistance in constructing the atomic operator set. Specifically, GPT-6-Astra (OpenAI, 2026b) was used to improve the clarity and readability of the manuscript and to survey common feature selection problems and candidate atomic operations. The resulting operator candidates were manually reviewed, modified, and finalized by the authors. GPT-5.6-Sol (OpenAI, 2026a) with Codex (OpenAI, 2025) was used to assist code development. The authors manually inspected and verified the generated code, including its inputs and outputs, prompts, configurations, and implementation behavior.

The core research contributions, including the formulation and decomposition of automated feature selection algorithm design process, the environment framework and implementation, the agent design, and the exploration strategy, were developed by the human authors. We reviewed all AI-assisted work and take full responsibility for the final content, implementation, and results of this work.

REFERENCES

Francisco Charte, Antonio J. Rivera, María J. del Jesus, and Francisco Herrera. Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing*, 163: 3–16, 2015. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2014.08.091. URL https://www.sciencedirect.com/science/article/pii/S0925231215004269. Recent Advancements in Hybrid Artificial Intelligence Systems and its Application to Real-World Problems Progress in Intelligent Systems Mining Humanistic Data.

Valeriia Cherepanova, Roman Levin, Gowthami Somepalli, Jonas Geiping, C. Bayan Bruss, Andrew G Wilson, Tom Goldstein, and Micah Goldblum. A performance-driven benchmark for feature selection in tabular deep learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 41956–41979. Curran Associates, Inc., 2023. doi: 10.52202/075280-1818. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/82f39c7409155b74d15d73b048f06771-Paper-Datasets_and_Benchmarks.pdf.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7: 1–30, December 2006. ISSN 1532-4435.

Kun Ding, Ying Wang, and Shiming Xiang. Evovlma: Evolutionary vision-language model adaptation. In *Proceedings of the 33rd ACM International Conference on Multimedia*, MM '25, pp. 4619–4628, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400720352. doi: 10.1145/3746027.3755520. URL https://doi.org/10.1145/3746027.3755520.

Hongbin Dong, Jing Sun, Xiaohang Sun, and Rui Ding. A many-objective feature selection for multi-label classification. *Knowledge-Based Systems*, 208:106456, 2020. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2020.106456. URL https://www.sciencedirect.com/science/article/pii/S0950705120305852.

Mohammad Faraji, Amjad Seyedi, and Fardin Akhlaghian Tab. Robust log-based multi-label feature selection with dynamic label correlation and relevance-redundancy optimization. *Knowledge-Based Systems*, 342:115825, 2026. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2026.115825. URL https://www.sciencedirect.com/science/article/pii/S0950705126005514.

Wanfu Gao, Yonghao Li, and Liang Hu. Multilabel feature selection with constrained latent structure shared term. *IEEE Transactions on Neural Networks and Learning Systems*, 34(3):1253–1262, 2023. doi: 10.1109/TNNLS.2021.3105142.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, March 2003. ISSN 1532-4435.

Qingqi Han, Liang Hu, and Wanfu Gao. Feature relevance and redundancy coefficients for multi-view multi-label feature selection. *Information Sciences*, 652:119747, 2024. ISSN 0020-0255.

doi: https://doi.org/10.1016/j.ins.2023.119747. URL https://www.sciencedirect.com/science/article/pii/S0020025523013324.

Rehab Ali Ibrahim, Mohamed Abd Elaziz, Ahmed A. Ewees, Mohammed El-Abd, and Songfeng Lu. New feature selection paradigm based on hyper-heuristic technique. *Applied Mathematical Modelling*, 98:14–37, 2021. ISSN 0307-904X. doi: https://doi.org/10.1016/j.apm.2021.04.018. URL https://www.sciencedirect.com/science/article/pii/S0307904X21002183.

Damjan Krstajic, Ljubomir J Buturovic, David E Leahy, and Simon Thomas. Cross-validation pitfalls when selecting and assessing regression and classification models. *Journal of cheminformatics*, 6 (1):10, 2014.

Junxuan Li, Zhiqi Chen, Yuzhou Liu, Peng Zhang, and Huaxiao Liu. Featurehospital: A skill-driven multi-agent framework for automated algorithm customization in multi-view multi-label feature selection. *arXiv preprint arXiv:2608.16148*, 2026.

Fei Liu, Tong Xialiang, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32201–32223. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/liu24bs.html.

Felix Neutatz, Felix Biessmann, and Ziawasch Abedjan. Enforcing constraints for machine learning systems via declarative feature selection: an experimental study. In *Proceedings of the 2021 International Conference on Management of Data*, pp. 1345–1358, 2021.

OpenAI. Introducing codex. https://openai.com/index/introducing-codex/, May 2025.

OpenAI. Gpt-5 system card, August 2025. URL https://openai.com/index/gpt-5-system-card/.

OpenAI. Gpt-5.6 system card. Technical report, OpenAI, 2026a. URL https://deploymentsafety.openai.com/gpt-5-6.

OpenAI. Gpt-6 astra system card. Technical report, OpenAI, 2026b. URL https://deploymentsafety.openai.com/gpt-6-astra.

Antonio Rafael Sabino Parmezan, Huei Diana Lee, Newton Spolaôr, and Feng Chung Wu. Automatic recommendation of feature selection algorithms based on dataset characteristics. *Expert Systems with Applications*, 185:115589, 2021. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2021.115589. URL https://www.sciencedirect.com/science/article/pii/S0957417421009908.

Hanchuan Peng, Fuhui Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005. doi: 10.1109/TPAMI.2005.159.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR 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.

Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. On the stratification of multi-label data. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 145–158. Springer, 2011.

Guangtao Wang, Qinbao Song, Heli Sun, Xueying Zhang, Baowen Xu, and Yuming Zhou. A feature subset selection algorithm automatic recommendation method. *Journal of Artificial Intelligence Research*, 47:1–34, 2013.

Xi-Zhu Wu and Zhi-Hua Zhou. A unified view of multi-label performance measures. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3780–3788. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/wu17a.html.

Xueyuan Xu, Fulin Wei, Tianze Yu, Jinxin Lu, Aomei Liu, Li Zhuo, Feiping Nie, and Xia Wu. Embedded multi-label feature selection via orthogonal regression. *Pattern Recognition*, 163: 111477, 2025. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2025.111477. URL https://www.sciencedirect.com/science/article/pii/S0031320325001372.

Makoto Yamada, Wittawat Jitkrittum, Leonid Sigal, Eric P. Xing, and Masashi Sugiyama. High-dimensional feature selection by feature-wise kernelized lasso. *Neural Computation*, 26(1): 185–207, 2014. doi: 10.1162/NECO_a_00537.

Taisuke Yasuda, Mohammadhossein Bateni, Lin Chen, Matthew Fahrbach, Gang Fu, and Vahab Mirrokni. Sequential attention for feature selection. In *ICML 2023 Workshop: Sampling and Optimization in Discrete Space*, 2023. URL https://openreview.net/forum?id=Pn7CPyHmrr.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. Reevo: Large language models as hyper-heuristics with reflective evolution. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 43571–43608. Curran Associates, Inc., 2024. doi: 10.52202/079017-1381. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/4ced59d480e07d290b6f29fc8798f195-Paper-Conference.pdf.

Jia Zhang, Yidong Lin, Min Jiang, Shaozi Li, Yong Tang, and Kay Chen Tan. Multi-label feature selection via global relevance and redundancy optimization. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20, 2021. ISBN 9780999241165.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623. Curran Associates, Inc., 2023. doi: 10.52202/075280-2020. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf.

Zhi Zheng, Zhuoliang Xie, Zhenkun Wang, and Bryan Hooi. Monte Carlo tree search for comprehensive exploration in LLM-based automatic heuristic design. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78338–78373. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/zheng25o.html.

A Discussion and Limitations

A.1 Discussion

Practical goals of automated feature selection algorithm design. The primary goal of automated feature selection algorithm design is to produce executable algorithms that suit the target dataset and are useful in practice. This goal does not require every generated algorithm to introduce a fundamentally new mechanism or outperform all alternatives. A method that performs strongly on one dataset may be poorly matched to another, so practical usefulness depends on adapting to dataset characteristics and reducing the risk of severe performance deterioration. An algorithm assembled from familiar components can therefore be a successful design if it provides reliable performance for the intended application.

Our comparisons illustrate this distinction. In Table 1, Deep-Lasso and FSDesign achieve similar MacroAP on Yeast (0.4197 and 0.4207), whereas their results differ substantially on Medical (0.1181 and 0.4673). This change in relative performance illustrates the importance of assessing algorithm suitability for each dataset. Across the evaluated datasets, FSDesign achieves the best average performance on all four metrics: the highest MacroAP and Jaccard and the lowest RL and HL (Tables 1 and 2). Individual methods still perform better in some settings, such as MMFS on Scene. These results support the value of dataset-specific design without implying that the generated algorithms are uniformly superior.

We therefore emphasize strong average performance together with consistently competitive results on individual datasets, aiming to remain close to the best available methods while avoiding severe mismatches. Average scores alone cannot establish this property, since they can conceal poor performance on particular datasets. The practical objective is dependable adaptation and usability, with algorithmic novelty as a possible outcome rather than a requirement. This objective is assessed empirically; the present results do not guarantee optimality or freedom from performance deterioration on unseen datasets.

One-shot vs. iterative generation. Theoretical analysis provides a principled basis for feature selection algorithm design. For example, evidence of redundant features can motivate redundancy control, while nonlinear feature–label relationships can motivate a nonlinear dependence measure. However, the justification for a mechanism does not by itself establish that the resulting algorithm will perform well on a particular dataset. Theoretical guarantees depend on assumptions, whereas finite data may combine several interacting characteristics that are only partially captured by dataset diagnostics. The suitability of feature selection methods consequently varies across datasets (Wang et al., 2013; Parmezan et al., 2021). A one-shot design uses the available analysis to propose an algorithm, but has no opportunity to revise that design in response to its observed performance. Iterative generation connects these design hypotheses to empirical evidence: candidate evaluation tests whether an intended change produces the expected benefit, and subsequent revisions can retain useful mechanisms or reconsider unsuccessful ones. In FSDesign, this evidence is accumulated in the interaction history, allowing later decisions to build on both successful and unsuccessful trials. Iteration is therefore important for adapting theoretical considerations to the target dataset, although it does not guarantee improvement at every step.

Fixed algorithm pools vs. fixed atomic operator sets. A predefined algorithm pool supports adaptation by choosing among complete methods. This is useful when an appropriate method is already available, but the granularity of adaptation is determined by the pool entries: changing one internal mechanism generally requires adding a corresponding algorithm variant. In contrast, a fixed set of atomic operators allows the designer to modify how individual computations are selected, combined, and connected. The primitive operations remain fixed during a design run, while their compositions define a substantially richer space of candidate structures. This distinction also affects extensibility. Expanding the coverage of an algorithm pool requires implementing or integrating additional complete methods. Once a new atomic operator has been implemented and validated, it can participate in many type-compatible compositions with existing operators, introducing a family of candidate algorithms without requiring each to be manually specified. The benefit is thus a finer design granularity and greater reuse of new functionality. It remains necessary to search this expanded space effectively; additional compositions are opportunities for adaptation, not guarantees of useful or behaviorally distinct algorithms.

Evaluating feature selection algorithms. Some optimization problems specify a direct criterion for comparing feasible solutions: the traveling salesman problem minimizes total tour length, while the standard knapsack problem maximizes total item value subject to a weight-capacity constraint. Feature selection can also be assigned a precise objective once the downstream learner, feature budget, and evaluation criterion are fixed. However, it has no equally universal scalar criterion that determines algorithm quality across datasets and predictive requirements. In multi-label learning, different predictive measures capture different aspects of performance and can favor different algorithms (Wu & Zhou, 2017). A single chosen metric therefore specifies one evaluation preference rather than a generally accepted gold standard for feature selection.

We address this issue by retaining two complementary sources of evidence, as described in Section 4.1.3. The result profile characterizes the statistical properties of the selected features, while downstream metrics measure the predictive consequences under the chosen evaluation protocol. These sources answer related but distinct questions: a profile can help explain what a design changes, whereas predictive evaluation tests whether those changes are useful for the downstream task. For example, reduced feature redundancy is diagnostically informative but does not by itself establish an improvement in prediction. Neither source alone fully captures the intended assessment.

Our use of an LLM judge is motivated by two difficulties in combining this evidence. First, profile coordinates and predictive metrics differ in origin, scale, and meaning, so there is no natural common unit in which their changes can be added. Second, even after numerical normalization, a fixed aggregation rule must specify how conflicting signals should be resolved across different comparisons. The significance of a profile change may depend on which predictive effects accompany it, making a broadly suitable set of weights or thresholds difficult to specify in advance. An LLM provides a way to interpret these relationships jointly and return a preference between a candidate and its base. This choice concerns evidence interpretation: the mirrored comparison procedure and the mapping from preferences to rewards in Section 4.1.4 remain explicitly defined. We present this as one possible evaluation approach, rather than a definitive solution. Whether alternative fixed, learned, or multi-objective comparison mechanisms can provide more reliable assessments, and how best to align such assessments with held-out predictive quality, remain open questions.

Decoupling the environment and designer agent. A central contribution of FSDESIGN is to decouple automated feature selection algorithm design into an environment that supports design and a designer agent that makes design decisions. The environment provides the action space, observations, candidate execution, evaluation, and rewards. Executing a candidate produces a feature ranking from which feature subsets are selected; the environment then evaluates these subsets, generates a reward relative to the base algorithm, and updates the history available in subsequent observations. The designer agent uses this interface to construct and revise algorithms, with an exploration mechanism guiding which design directions to pursue. This interaction casts algorithm design as sequential decision making: each decision is conditioned on the current observation, and its execution and feedback inform subsequent decisions. The resulting sequence of design, evaluation, and refinement seeks an algorithm suited to the target dataset.

This separation allows the environment and agent to be improved independently through a shared interaction interface. On the environment side, future work can develop more accurate candidate assessments and reward signals, as well as more effective and comprehensive action spaces. On the agent side, the current implementation uses in-context learning (ICL), drawing on dataset information and accumulated interaction history to guide proposals and tree-based exploration. A natural extension is to train the designer agent with reinforcement learning (RL) through interactions with the environment, using the resulting rewards to improve algorithm construction and exploration. The sequential decision-making formulation thus supports investigating both stronger environments and learned design policies within the same framework.

A.2 LIMITATIONS

Scope of feature selection tasks. The current formulation and empirical study focus on multi-label feature selection. We have not yet extended the framework to other feature selection settings, and its applicability beyond the multi-label task remains to be established.

Coverage of the action space. Our action space is built from a finite set of atomic operators and specified composition rules. Although their combinations define a large family of candidate algorithms, we have not established that this family covers the open-ended diversity of feature selection problems. Atomic composition allows finer modifications and exploration beyond a fixed pool of complete algorithms, relaxing the design restrictions imposed by that pool. However, the available operators and their composition rules still bound what the environment can express and, consequently, what algorithm design can achieve within it. Expanding the space of compositions does not remove this representational limit. Developing more flexible and broadly expressive action spaces is therefore an important direction for future work.

Evaluation and reward design. We combine result profiles and downstream predictive metrics to provide a more comprehensive assessment of candidate algorithms, and map their joint interpretation to a reward for iterative design. The ablations in Section 5.3 support the usefulness of both profile information and predictive feedback. However, these findings do not establish that our evaluation and reward mechanism is the most accurate choice. It is one possible way to assess algorithms and turn evaluation evidence into design guidance. Alternative combinations of evidence, comparison mechanisms, and reward mappings remain to be explored systematically. Determining which approaches assess algorithm quality more accurately and provide more useful guidance is another direction for future work.

B Method Details

B.1 Dataset Profile Construction

The fixed dataset profile $\phi(\mathcal{D}_{\mathrm{tr}})$ combines descriptive statistics of the design partition with diagnostics of a fixed reference ranking. Write $\mathcal{D}_{\mathrm{tr}} = (X, Y)$ with n examples, d features, and L binary labels. Its descriptive component records (n, d, L), the fraction $(nd)^{-1}\sum_{i,j}\mathbf{1}[X_{ij} = 0]$, and the mean, population standard deviation, minimum, and maximum of both label prevalence $p_\ell = n^{-1}\sum_i Y_{i\ell}$ and example cardinality $c_i = \sum_\ell Y_{i\ell}$. It also records the mean, 90th percentile, and maximum of $|\operatorname{corr}(X_{:j}, X_{:h})|$ over $j < h$. These statistics use the entire design partition; in particular, zero fractions are measured before fold-specific standardization. They do not use the held-out test partition.

Fixed reference diagnostics. For each training portion T of the fixed cross-validation views, features are standardized using means and scales fitted on T. Define

$$A_{j\ell} = |\operatorname{corr}(X_{T:j}, Y_{T:\ell})|, \qquad b_j = L^{-1}\sum_{\ell=1}^{L} A_{j\ell}. \tag{18}$$

The reference ranking orders features by decreasing b_j, resolving ties by feature index, and retains its first k_0 features as S^0. The reference rate is fixed independently of the candidate. An additional cross-validation procedure within T obtains reference probabilities: every nested training portion fits its own standardizer, recomputes the reference ranking, and fits the neighbor-average predictor on its reference prefix. The predictions for the corresponding nested held-out rows give residuals $E_{i\ell} = Y_{i\ell} - \widehat{p}^{\,0}_{i\ell}$ for all $i \in T$. Thus the reference feature set is recomputed inside each nested split. The resulting residuals remain fixed throughout candidate search on that view.

Let $R_{j\ell} = |\operatorname{corr}(X_{T:j}, E_{:\ell})|$, $a_j = L^{-1}\sum_\ell R_{j\ell}$, and $e_\ell = |T|^{-1}\sum_{i\in T} E_{i\ell}^2$. Define $C_{jh} = |\operatorname{corr}(X_{T:j}, X_{T:h})|$ for $j \neq h$ and $C_{jj} = 0$. Each view contributes the following reference diagnostics:

- the largest a_j among excluded features, the mean a_j over S^0, and their ratio, with the denominator lower-bounded by ε;
- reference redundancy $k_0^{-2}\sum_{j,h\in S^0} C_{jh}$;
- the normalized entropy of label errors and the ratio $\max_\ell e_\ell / \max(L^{-1}\sum_\ell e_\ell, \varepsilon)$.

For the entropy, set $q_\ell = e_\ell / \max(\sum_h e_h, \varepsilon)$ and compute $-\sum_\ell q_\ell \log\max(q_\ell, \varepsilon)/\log\max(2, L)$. The excluded-feature maximum is zero when the excluded set is empty. Fold diagnostics are averaged

equally within each repeat and then equally across repeats; the reported residual-association ratio is an average of foldwise ratios. The full-partition descriptive statistics are identical across repeats. These reference diagnostics provide additional context for the dataset profile without prescribing an operator or a direction of improvement.

B.2 Action Space Implementation and Operator Set

The implemented design action is a complete typed directed acyclic graph, represented by a list of computational nodes and an `output_node`. Each node specifies an identifier, an operator, an ordered list of input references, and a parameter object. References point to another node or to the input sources `X` and `Y`. A proposal therefore determines both the computation and its connectivity; the planner does not issue individual graph-edit commands.

For a training view with n examples, d features, and L labels, the source types are $\mathsf{D} = \mathbb{R}^{n\times d}$ for X and $\mathsf{Y} = \{0,1\}^{n\times L}$ for Y. The reachable intermediate types are a feature vector $\mathsf{F} = \mathbb{R}^d$, a label vector $\mathsf{L} = \mathbb{R}^L$, a feature–label relation $\mathsf{R} = \mathbb{R}^{d\times L}$, a feature–feature relation $\mathsf{Q} = \mathbb{R}^{d\times d}$, and a scalar $\mathsf{S} = \mathbb{R}$. The terminal type I is a vector of feature indices. Table 4 specifies the seven operators exposed to the planner. The implementation contains other library components, but they are not additional actions in this instantiation. Redundancy-aware scores are constructed by composing the feature–feature relation with reduction and score-combination operators; a complete greedy redundancy-penalized ranking is not exposed as a separate operator.

Table 4: Complete exposed operator set. Parameter names and categorical values are literal identifiers in the graph representation. All listed parameter fields are required by the planner schema. The admissible type combinations for `REDUCE` and `BINARY` are specified below.

Operator	Type signature	Parameters and allowed values
`STATISTIC`	$\mathsf{D} \to \mathsf{F}$	`kind`: `variance`, `std`, `mean_abs`, `max_abs`, `nonzero_fraction`.
`DEPENDENCE`	$(\mathsf{D}, \mathsf{Y}) \to \mathsf{R}$	`metric`: `pearson` or `mutual_info`; `absolute`: Boolean; `n_neighbors`: positive integer; `random_state`: null or an integer in $[0, 2^{32}-1]$.
`REDUNDANCY`	$\mathsf{D} \to \mathsf{Q}$	`metric`: `abs_pearson`. Returns absolute feature–feature correlations with a zero diagonal.
`REDUCE`	$\mathsf{R} \to \mathsf{F}$ or L; $\mathsf{Q} \to \mathsf{F}$; $\mathsf{F}, \mathsf{L} \to \mathsf{S}$	`axis`: `labels`, `features`, `rows`, `columns`, or `all`, subject to input type; `reducer`: `mean`, `sum`, `max`, `min`, or `l2`; `absolute`: Boolean.
`NORMALIZE`	$\mathsf{F} \to \mathsf{F}$; $\mathsf{L} \to \mathsf{L}$	`method`: `minmax`, `zscore`, `l1`, `l2`, `rank`, or `none`; `eps`: finite positive number.
`BINARY`	Compatible pairs of $\mathsf{F}, \mathsf{L}, \mathsf{R}, \mathsf{Q}, \mathsf{S}$	`op`: `add`, `subtract`, `multiply`, `divide`, `maximum`, or `minimum`; `left_weight`, `right_weight`: finite numbers; `eps`: finite positive number.
`TOPK`	$\mathsf{F} \to \mathsf{I}$	`largest`: Boolean. The planner must omit `k`; the execution environment supplies $k = d$.

Feature statistics and label dependence. `STATISTIC` operates independently on each feature column. Its five choices return the population variance $n^{-1}\sum_i (X_{ij} - \bar{X}_j)^2$, its square root, $n^{-1}\sum_i |X_{ij}|$, $\max_i |X_{ij}|$, and $n^{-1}\sum_i \mathbf{1}\{X_{ij} \neq 0\}$, respectively.

For `DEPENDENCE` with `metric=pearson`, the entry for feature j and label ℓ is

$$R_{j\ell} = \frac{\sum_i (X_{ij} - \bar{X}_j)(Y_{i\ell} - \bar{Y}_\ell)}{\sqrt{\sum_i (X_{ij} - \bar{X}_j)^2 \sum_i (Y_{i\ell} - \bar{Y}_\ell)^2}}. \tag{19}$$

The entry is set to zero when the denominator does not exceed machine epsilon for the computation dtype. This includes constant features and constant labels. Setting `absolute=true` replaces R by $|R|$. The CUDA path computes the same expression in its configured floating-point dtype; the CPU path uses NumPy floating-point arithmetic.

With `metric=mutual_info`, the operator calls `mutual_info_classif` independently for each label, treating every feature as continuous (`discrete_features=false`). For label ℓ, the effective neighbor count is the minimum of the requested count and the smallest observed class count minus one. A constant label or an effective count below one produces an all-zero column. A null `random_state` selects the execution seed, and the zero-based label index is added to that seed for each call. The `absolute` parameter has no effect in this branch; `n_neighbors` and `random_state` have no effect in the Pearson branch. Mutual information retains the CPU estimator when CUDA acceleration is enabled.

Feature–feature redundancy. `REDUNDANCY` takes only X and returns the symmetric relation

$$Q_{jh} = \begin{cases} |\operatorname{corr}(X_{:j}, X_{:h})|, & j \neq h, \\ 0, & j = h. \end{cases} \tag{20}$$

The Pearson calculation uses centered columns as in Equation 19, with both arguments taken from X. Entries with a denominator at or below the machine epsilon of the computation dtype are zero, so constant features introduce no undefined correlations. Values are clipped to $[0, 1]$ before the diagonal is set to zero. A reduction can turn this relation into per-feature redundancy scores, which can be normalized and combined with relevance scores through `BINARY`. The operator provides the relation without prescribing a particular relevance–redundancy tradeoff.

Reduction and normalization. For a feature–label relation, `axis=labels` aggregates each row into a feature vector, while `axis=features` aggregates each column into a label vector. For a feature–feature relation, `columns` aggregates across columns, while `rows` and `features` aggregate across rows; each produces a feature vector. The `mean` reducer includes the zero diagonal, dividing the sum by d. For either vector type, `all`, `features`, and `labels` all reduce the complete vector to a scalar. `absolute=true` takes elementwise absolute values before applying the reduction. The `l2` reducer returns $\sqrt{\sum_i v_i^2}$; the other reducers implement their usual arithmetic meanings.

For a vector v, the four scale-based choices of `NORMALIZE` compute

$$\begin{aligned} \texttt{minmax}: \ & \frac{v - \min(v)}{\max(v) - \min(v)}, & \texttt{zscore}: \ & \frac{v - \operatorname{mean}(v)}{\operatorname{std}(v)}, \\ \texttt{l1}: \ & \frac{v}{\sum_i |v_i|}, & \texttt{l2}: \ & \frac{v}{\sqrt{\sum_i v_i^2}}. \end{aligned} \tag{21}$$

The standard deviation is the population standard deviation. Each choice returns an all-zero vector if its denominator is at most `eps`. The `rank` choice assigns ascending zero-based ranks, gives equal values their average rank, and divides by $\max(1, |v| - 1)$. The `none` choice returns a copy of the input. Normalization never changes the semantic vector type.

Score combination. `BINARY` accepts two values of the same type from $\{\mathsf{F}, \mathsf{L}, \mathsf{R}, \mathsf{Q}, \mathsf{S}\}$ and returns that type. It also permits a scalar with a feature or label vector, in either order, and returns the vector type. No feature-vector/label-vector combination, vector/relation broadcast, scalar/relation broadcast, or direct arithmetic on the source matrices is allowed. With $a = w_{\text{left}} u$ and $b = w_{\text{right}} v$, its six operations are the elementwise values $a + b$, $a - b$, ab, $a/\widetilde{b}$, $\max(a, b)$, and $\min(a, b)$. The protected denominator is

$$\widetilde{b}_i = \begin{cases} -\varepsilon, & -\varepsilon < b_i < 0, \\ \varepsilon, & 0 \leq b_i < \varepsilon, \\ b_i, & |b_i| \geq \varepsilon, \end{cases} \qquad \varepsilon = \texttt{eps} > 0. \tag{22}$$

Thus the magnitude floor preserves the sign of a nonzero denominator and treats zero as positive. Weights may be negative or zero; finite values are required.

Terminal ranking and graph validity. `TOPK` sorts a feature-score vector in descending order when `largest=true` and ascending order otherwise. Equal scores are ordered by ascending original feature index. Although the library operator supports a general selection budget, the atomic runner supplies d and the proposal schema excludes an explicit `k`. Every candidate therefore returns one

complete permutation $\pi \in \mathfrak{S}_d$. Evaluation at a smaller feature budget uses a prefix of this permutation rather than a separately generated graph.

The planner schema requires between two and M computational nodes, where M is the configured graph-size limit. The graph library additionally imposes a hard ceiling of 128 nodes. Node identifiers must start with a letter, contain only letters, digits, underscores, periods, or hyphens, and have at most 64 characters. Identifiers must be unique, references must exist, and every node must contribute to the specified output. Static validation checks operator membership, exact input arity, allowed parameters, compatible types, acyclicity, and a terminal feature-index output. With the exposed types, these conditions require a single terminal `TOPK` node. The input node list need not already be topologically ordered; execution obtains a topological order from its references.

At execution time, X and Y must be aligned two-dimensional arrays. Numeric operator results must be finite, and each intermediate result is limited to $50{,}000{,}000$ elements. The terminal result is checked for the required length, duplicate indices, and indices outside the feature range; the atomic wrapper additionally checks that it is a complete permutation. Invalid proposals and failed computations enter the planner repair mechanism rather than receiving predictive evaluation. These checks enforce executable compositions; they do not guarantee predictive quality or make arbitrary feature-selection mechanisms expressible.

B.3 Result Profile Construction

The result profile P_t measures a candidate ranking against the fixed references above, separately on every training view. For a diagnostic rate r, let S_r contain the first $k_r = \max(1, \lceil rd/100 \rceil)$ features. For any nonnegative feature scores v, define retained mass by

$$\mathcal{C}(v, S) = \begin{cases} \dfrac{\sum_{j \in S} v_j}{\sum_{j \in \mathrm{Top}_{|S|}(v)} v_j}, & \displaystyle\sum_{j \in \mathrm{Top}_{|S|}(v)} v_j > \varepsilon, \\ 1, & \text{otherwise.} \end{cases} \tag{23}$$

The denominator is the largest mass attainable by a feature set of the same size, rather than the mass of all features.

Label allocation. The retained association of label ℓ is

$$\mathrm{cov}_\ell(S_r) = \mathcal{C}(A_{:\ell}, S_r). \tag{24}$$

At each prefix, the profile measures its mean, 10th percentile, and minimum over labels, together with the weighted mean $\sum_\ell w_\ell \, \mathrm{cov}_\ell(S_r)$, where $w_\ell = \max(e_\ell, \varepsilon) / \sum_h \max(e_h, \varepsilon)$. To describe allocation, assign each feature to $\ell^*(j) = \arg\max_\ell A_{j\ell}$, with the smallest label index breaking ties. Let $u_\ell = k_r^{-1} \sum_{j \in S_r} \mathbf{1}[\ell^*(j) = \ell]$. The represented-label fraction is $L^{-1} |\{\ell : u_\ell > 0\}|$, and the normalized allocation entropy is $-\sum_\ell u_\ell \log \max(u_\ell, \varepsilon) / \log \max(2, L)$. The seventh allocation statistic measures specialization:

$$\frac{1}{k_r} \sum_{j \in S_r} \frac{\max_\ell A_{j\ell}}{\max(L^{-1} \sum_\ell A_{j\ell}, \varepsilon)}. \tag{25}$$

Each of these seven quantities is averaged over the diagnostic prefixes. The winning-label rule also applies to features with zero association to every label, so representation and entropy should be read together with coverage.

Ranking geometry. At each prefix, relevance coverage is $\mathcal{C}(b, S_r)$. Nonredundancy and normalized participation ratio are

$$\mathrm{NR}(S_r) = 1 - \frac{1}{k_r^2} \sum_{j,h \in S_r} C_{jh}, \qquad \mathrm{PR}(S_r) = \frac{\mathrm{tr}(K_r)^2}{\max(k_r \|K_r\|_F^2, \varepsilon)}, \quad K_r = C_{S_r, S_r} + I_{k_r}. \tag{26}$$

The redundancy mean includes the zero diagonal and therefore uses k_r^2 entries. The participation ratio is a matrix-norm diagnostic; it does not require an eigendecomposition. The profile reports these three quantities at the reference prefix and their means across diagnostic prefixes.

A fourth geometry diagnostic measures local label prediction on a deterministic sample from T. A reference set J is sampled without replacement, followed by a probe subset $Q \subseteq J$. In the selected standardized coordinates, each probe is predicted by averaging the labels of its nearest reference neighbors, excluding the probe itself. With $\bar{y}_J = |J|^{-1} \sum_{i \in J} Y_i$, define

$$\mathrm{NS}(S_r) = 1 - \frac{\frac{1}{|Q|L} \sum_{i \in Q} \|Y_i - \widehat{y}_i^{\mathrm{local}}\|_2^2}{\max\left(\frac{1}{|Q|L} \sum_{i \in Q} \|Y_i - \bar{y}_J\|_2^2, \varepsilon\right)}. \tag{27}$$

The same reference and probe samples are used for every candidate and prefix in a training view. The profile records the mean and minimum of this score over prefixes. This is a training-side neighborhood diagnostic: the prevalence reference includes the probes, and the score is not a validation metric.

Reference-residual coverage. For each prefix, the profile records $\mathcal{C}(Rw, S_r)$ under uniform weights $w_\ell = 1/L$ and under the error weights defined above. It also records the minimum and mean of $\mathcal{C}(R_{:\ell}, S_r)$ over labels. Each weighting produces six fields: these three quantities at the reference prefix and their three prefix averages. The four labelwise fields are identical under both weightings, since weighting changes only the association aggregated across labels. Reference-residual coverage describes association with errors of the fixed reference; it does not fit a predictor for the candidate or guarantee a predictive improvement.

The complete P_t contains seven allocation fields, eight geometry fields, and two groups of six residual fields. Each scalar is averaged equally across the fixed training views. A minimum over labels or prefixes is taken within a view before this averaging. Profile differences are computed coordinatewise as candidate minus base, without assigning a universal favorable direction. Numeric settings and aggregation conventions are given in Appendix C.4.

B.4 Structured Model Calls and Reward Generation

The implementation separates graph design, evidence comparison, memory construction, and search navigation into structured model calls. Table 5 summarizes their information boundaries. Each response is parsed as JSON and checked against its role-specific schema; invalid responses receive bounded repair requests containing the validation error. Exhausted repairs or transport failures interrupt the run rather than being assigned a negative scientific reward. Appendix E provides the system instructions used by these calls.

Table 5: Model-call roles and information boundaries in the implemented loop.

Role	Receives	Returns
Planner	Dataset profile; operator registry; current base graph and evidence; tree; chronological memory.	Diagnosis; prospective hypothesis; change summary; complete graph.
Reward judge	Anonymous A/B result profiles and predictive evidence; dataset measurements; evidence identifiers.	Preference; signed A-reward; evidence references; interpretation and advice.
History summarizer	Graph purpose; observed changes; frozen mirrored decisions and reward.	A purpose sentence and exact copies of the fixed evidence-summary fields.
Navigation advisor	Parent and negative child graphs; frozen interpretation and reward; tree context.	Continue from the child or return to the parent, with a rationale.
Tie breaker	Dataset profile; eligible candidates under anonymous labels; their graphs, profiles, and validation evidence.	Exactly one eligible label, with a rationale.

Graph proposal. The planner receives the current base graph together with its result profile and compact validation measurements. Its chronological memory contains earlier graphs and their summaries, while a separate tree view reports parent links, depths, path scores, failed-child counts, and exhausted nodes. When behavior novelty is enabled, the request additionally includes the evaluated graph registry and duplicate-behavior constraints. The returned hypothesis states the intended change, expected validation and profile effects, and what failure would mean. The instructions favor a focused,

testable revision but permit a larger structural change when justified. The planner cannot change the fixed data splits, retained-feature percentages, metrics, or downstream predictor.

Blinded comparison. The judge receives the two result profiles, raw profile differences, metric means, and direction-normalized metric differences summarized across repeats, folds, and feature percentages. Dataset measurements supply context, but graph structure, design hypotheses, chronological history, and candidate/base identities are withheld. The two requests exchange the A/B measurements and reverse the comparative differences. A runtime check verifies this symmetry, and private role mappings are removed before either request is sent.

The response must jointly interpret statistical and predictive evidence, cite supplied evidence identifiers, and return one of A, B, indistinguishable, or tradeoff. Its structured attribution records the validation effect, profile interpretation, relationship between the two evidence sources, profile-pattern assessment, plausible explanation, and next-experiment advice. The anonymous reward is $+1$ exactly when A is preferred and -1 otherwise. The runtime separately maps each preference to the candidate, then applies Equation 11: the candidate receives $+1$ only when both presentation orders prefer it. A reference preference, indistinguishable comparison, unresolved tradeoff, or mirror disagreement yields -1. These instructions and checks constrain evidence use without guaranteeing a correct judgment.

Compact chronological memory. Each model-visible history entry contains exactly the complete `algorithm` and a `summary`. The summary has four fields: `purpose`, `result_profile_change`, `metric_change`, and `reward`. The construction Γ combines an LLM-generated description of the design intention with deterministic records of the observed outcome.

The LLM writes only `purpose`, using the change summary and hypothesis statement produced when proposing the candidate. It describes the intended computation in one complete sentence, targeting at most 160 characters with a hard limit of 220. This sentence does not reinterpret the reward or comparison outcome; the complete executable graph is retained separately in the history entry.

Before the model call, the program constructs the remaining fields. For the result profile, it flattens the coordinatewise candidate-minus-base differences, removes exact zeros, and retains at most three coordinates with the largest absolute changes. Equal magnitudes are ordered lexically by the displayed coordinate name. These differences are not normalized across coordinates and are descriptive rather than intrinsically favorable or unfavorable; an exactly unchanged profile receives a fixed zero-change statement. The metric field reports every exposed improvement-oriented delta, with positive values indicating improvement and negative values deterioration. Metric differences with absolute magnitude at most 10^{-12} are displayed as zero; other metric differences and the retained profile differences use six significant digits.

The reward field stores the already assigned value, whether candidate improvement was established, and a deterministic explanation of the mirrored-fusion outcome. The explanation distinguishes mutual preference for the candidate or base, mirror disagreement, mutual indistinguishability, mutual tradeoff, and mixed non-improvement. A strict response schema fixes the profile sentence, metric sentence, and every reward field as constants that the model must copy exactly. Validation checks these constants, sentence completion, length limits, and anonymous-role leakage. Invalid responses trigger bounded repairs; exhausted repairs or transport failures interrupt the run without a fallback summary or a change to the reward.

All completed trials remain in chronological memory, including negative trials and branches subsequently left by backtracking. Full evaluations, hypotheses, judge responses, and navigation records are retained separately in the run artifacts. Consequently, compact planner memory and the complete experimental record serve different purposes. The history summary is constructed after reward fusion and before the tree commits the new trial.

B.5 Complete Search Procedure

The initial graph G_0 computes absolute Pearson feature–label dependence, averages over labels, and sorts the resulting feature scores in descending order. It is evaluated before any model-generated

candidate and initializes the root with path score zero. Algorithm 1 gives the formal search procedure; numerical budgets are specified in Appendix C.5.

A negative child does not automatically terminate its branch: the advisor may continue from it when the frozen interpretation supports a focused corrective experiment. However, closing a parent overrides this choice and forces a return to the nearest open ancestor. Closing a node only prevents further expansion from that node; it does not remove the node from final selection. Final selection uses accumulated edge rewards over all evaluated nodes, with an additional model call only for an exact tie. Neither navigation nor tie breaking revises an earlier reward.

Algorithm 1 FSDESIGN with the implemented signed-reward tree

Require: Training data $\mathcal{D}_{\mathrm{tr}}$, fixed evaluation protocol, initial graph G_0
Construct the dataset profile and fixed fold references; evaluate G_0
Initialize root v_0 with $C(v_0) = 0$, base $b \leftarrow v_0$, and empty history
for $t = 1, \ldots, T$ **do**
    Propose a complete graph and hypothesis from the current observation
    Validate and execute the graph on the fixed training views, with bounded repair
    If novelty is enabled, repair duplicate ranking behavior before predictive evaluation
    Evaluate the candidate against G_b to obtain predictive and profile evidence
    Judge the anonymized evidence in both A/B orders; fuse and freeze r_t
    Construct the compact history summary from the frozen evidence and reward
    Attach child v_t to b with $C(v_t) \leftarrow C(b) + r_t$
    if $r_t = -1$ **then**
        Increment the number of direct negative children of b
    end if
    if b has reached the failed-child limit **then**
        Close b; set the next base to its nearest open ancestor
    else if $r_t = +1$ **then**
        Set the next base to v_t
    else
        Ask the advisor to continue from v_t or return to b
    end if
    Commit the graph and summary to chronological memory; update the base
    if no base remains **then**
        break
    end if
end for
Find all maximum-path-score nodes, including the root and closed nodes
If tied, ask the tie breaker to choose one eligible node
return the selected executable graph

C EXPERIMENTAL DETAILS

The main comparisons and ablations use the HL4 predictive panel: MacroAP, Jaccard, ranking loss, and Hamming loss. These four measurements are exposed separately to the designer and reward judge, except in the **w/o Metrics** ablation. The metric panel is fixed before search and reused for every candidate.

C.1 DATA SPLITS AND DOWNSTREAM PREDICTION

The seven benchmark datasets are Bibtex, Birds, Emotions, Genbase, Medical, Scene, and Yeast. Their domains and sizes are summarized in Table 6.

For each environment seed, a fixed random split reserves 70% of instances for algorithm design and 30% for outer evaluation. The outer split is stratified by the number of positive labels per instance when every stratum contains at least two instances and both partitions can accommodate all strata; otherwise, the split is unstratified. Splitting precedes preprocessing. The search reads a separate archive containing only the design instances and their original row identifiers.

Table 6: Characteristics of the seven benchmark datasets. Counts refer to the full input matrices used in our experiments, before train–test splitting. The protein identifier is excluded from the Genbase feature count.

Dataset	Domain	Samples	Features	Labels
Bibtex	Text	7,395	1,836	159
Birds	Audio	645	260	19
Emotions	Music	593	72	6
Genbase	Biology	662	1,185	27
Medical	Text	978	1,449	45
Scene	Image	2,407	294	6
Yeast	Biology	2,417	103	14

Within the design partition, two fixed repetitions of three-fold multi-label iterative stratification (Sechidis et al., 2011) give six training–validation views. Each repetition partitions all design instances into three disjoint validation blocks. The partitions are determined by the environment seed and repetition index before search and are reused for every candidate. For each view, a standard scaler is fitted only on its training instances and applied to both its training and validation matrices. The candidate graph receives this standardized training matrix and the corresponding training labels, and produces a complete ranking of the original features.

At each percentage $r \in \{2, 4, \ldots, 20\}$, the evaluator retains the first $k_r = \max(1, \lceil rd/100 \rceil)$ features of that ranking. The downstream predictor uses uniform Euclidean 10-nearest-neighbor probabilities:

$$\widehat{p}_\ell(x; S) = \frac{1}{10} \sum_{i \in N_{10}(x;S)} Y_{i\ell}, \tag{28}$$

where $N_{10}(x; S)$ indexes the ten nearest training instances using only the selected coordinates S. A training block containing fewer than ten instances is rejected. Constant training labels produce probabilities zero or one. All predictive metrics for a given view and prefix use the same probability matrix.

After winner selection, the outer evaluator fits a scaler on the entire design partition, re-executes each recorded graph on that partition, and freezes its complete ranking before opening the outer data. The already selected winner remains fixed. One predictor is fitted for each of the ten ranking prefixes and evaluated on the same outer partition; two inner CV repetitions do not introduce additional outer test splits. The outer score for an environment seed is the unweighted mean over the ten percentages. The comparison tables report the mean and standard deviation of these seed-level scores over five seeds; their final Average row gives equal weight to each dataset.

The CUDA implementation computes squared Euclidean distances in batches and selects ten neighbors by partial partitioning. The supplied launch configuration uses float32 arithmetic and batches of 1,024 validation instances. It does not impose a secondary ordering for equal distances at the tenth-neighbor boundary, so CPU and GPU results need not be identical in the presence of distance ties or floating-point differences.

C.2 Evaluation Metrics and Unavailable Measurements

For an evaluation set containing N instances and L labels, let $y_{i\ell} \in \{0, 1\}$ and $p_{i\ell} \in [0, 1]$ denote the target and predicted probability. Binary predictions use the strict threshold $\widehat{y}_{i\ell} = \mathbf{1}[p_{i\ell} > 0.5]$; an exact probability of 0.5 is therefore negative. Write $Y_i = \{\ell : y_{i\ell} = 1\}$ and $\widehat{Y}_i = \{\ell : \widehat{y}_{i\ell} = 1\}$.

- **MacroAP** averages labelwise average precision over labels containing both positive and negative evaluation instances. Average precision is computed from the precision–recall steps at distinct scores, with equal scores grouped together. Single-class labels are excluded.
- **Jaccard** averages $|Y_i \cap \widehat{Y}_i| / |Y_i \cup \widehat{Y}_i|$ over instances for which $|Y_i| > 0$. Instances with no positive target label are excluded even when the prediction is nonempty.

- **Ranking loss (RL)** averages $\big(|Y_i|(L-|Y_i|)\big)^{-1} \sum_{\ell \in Y_i} \sum_{m \notin Y_i} \mathbf{1}[p_{i\ell} \leq p_{im}]$ over instances with both positive and negative labels. Score ties count as errors; instances lacking either class are excluded.
- **Hamming loss (HL)** is $(NL)^{-1} \sum_{i,\ell} \mathbf{1}[\widehat{y}_{i\ell} \neq y_{i\ell}]$, including all instance–label pairs.

Higher MacroAP and Jaccard are better; lower RL and HL are better.

Internal validation weights the $2 \times 3 \times 10 = 60$ repetition–fold–percentage cells equally for each metric. It reports the cell mean and sample standard deviation (`ddof=1`), together with summaries by repetition, fold, and percentage. These standard deviations describe variation across the evaluation surface, not uncertainty estimated from 60 independent trials. They are distinct from the across-seed standard deviations in the comparison tables.

The evaluator records the number of valid instances or labels used by each metric. A metric is undefined when its valid set is empty. If any constituent cell is undefined, the corresponding complete aggregate mean and standard deviation are unavailable; available cells are not silently reweighted. The inherited evaluator additionally computes compatibility measurements and a training-prevalence Brier reference. Undefined required compatibility measurements or a zero Brier reference can raise an evaluation error even when only HL4 is exposed. Such failures are not replaced by zero scores or negative scientific rewards.

C.3 Baseline Implementation and Evaluation

Evaluation protocol. For the eight baselines detailed below, evaluation uses seeds 1–5 and feature percentages $2, 4, \ldots, 20$ on the seven datasets in Appendix C.1. Final performance is measured with the shared outer-test predictor and the four metrics in Appendix C.2. Scores are first averaged equally over the ten percentages within each seed; the tables then report the mean and sample standard deviation across the five seeds. Any internal validation or diagnostic subsets used by the selectors are confined to the outer-training partition. Their use is method-specific, as detailed below; outer-test data are reserved for final evaluation.

Ranking-based baselines. Table 7 lists the configurations used for the five ranking-based baselines. Features constant on the outer-training partition are removed before selector fitting and appended after the ranking of nonconstant features. The feature-budget denominator remains the original dimensionality d. Each fitted ranking is reused across the ten feature percentages. Selector-specific preprocessing is fitted on the appropriate training partition; the downstream standard scaler is fitted separately as specified in Appendix C.1.

Table 7: Configurations of the ranking-based baselines. Internal validation fractions are relative to the outer-training partition.

Method	Configuration
SSFS	Latent dimension 23; 30 iterations; 5-neighbor graph; $\sigma = 30$, $\alpha = 0.01$, $\beta = 0.7$, $\gamma = 0.9$; training-only MinMax scaling.
RLBMLFS	2 iterations; $\lambda_1 = 0.7$, $\lambda_2 = 0.7$, $\lambda_3 = 2.5$; one fit per seed; training-only MinMax scaling.
Sequential-Attention	Hidden layer [67]; nominally 250 epochs; batch size 256; learning rate 0.0002; internal validation fraction 0.125; one feature per selection event; multi-label BCE.
Deep-Lasso	MLP [100, 100]; dropout 0.1; at most 200 epochs; batch size 512; AdamW with learning rate 0.001 and weight decay 0.0002; regularization weight 0.1; internal validation fraction 0.1875; patience 20.
GRROOR	$\lambda = 0.6$, $\alpha = 1$, $\beta = 0.8$, latent dimension $c = 2$; at most 10 optimization cycles, with the original stopping conditions.

SSFS and RLBMLFS. SSFS (Gao et al., 2023) uses a transcription of a pinned public Python reproduction, preserving its neighborhood graph, multiplicative updates, and ranking by the row norms of the learned feature matrix. This source is an independent reproduction rather than an author implementation. RLBMLFS (Faraji et al., 2026) uses a compatibility implementation of

the multiplicative updates from the public author code and ranks features by the row ℓ_2 norms of the learned weights. Its two-iteration budget is fixed in our experiments. The setting `repeats=1` denotes one selector fit per seed; experimental repetition is supplied by the five external seeds.

Sequential-Attention and Deep-Lasso. Both methods are adapted to multi-label supervision using one output logit per label and binary cross-entropy (BCE). Sequential-Attention (Yasuda et al., 2023) uses a PyTorch port of the official TensorFlow selection mechanism. MinMax scaling is fitted on its internal selection-training partition; the reserved validation instances do not participate in optimization or model selection. One run targets the largest feature budget, and smaller budgets use the chronological prefixes of its selection events. The nominal epoch count is extended only if necessary to complete that selection schedule. Deep-Lasso (Cherepanova et al., 2023) directly uses the official MLP, gradient regularizer, and feature-importance routine. Standardization is fitted on its internal training split; the internal validation split supplies the BCE checkpoint criterion and gradient-based importance estimates. Batch sizes are capped by the available training instances when needed. These methods use fixed cross-dataset configurations rather than dataset-specific hyperparameter searches.

GRROOR. We call the public MATLAB implementation of GRROOR (Xu et al., 2025) at commit `0d8c6e4` without modifying its optimization core. The adapter supplies only raw outer-training instances in the original feature coordinates, converts double-precision labels from $\{0, 1\}$ to $\{-1, +1\}$, and fixes MATLAB initialization with the experiment seed. The PCA preprocessing in the upstream demonstration is omitted. The returned feature ranking is mapped back to the original indices and passed to the common downstream evaluator.

DFS. We use the official algorithm pool and search implementation of DFS (Neutatz et al., 2021). Our integration changes only the dataset input format; the core algorithmic logic and implementations remain unchanged. The retained evaluation records identify strategies from the TPE, simulated annealing (SA), and NSGA-II families. In these records, a feature percentage r specifies a budget upper bound $b_r = \lfloor rd/100 \rfloor$, and the returned subset may contain fewer than b_r features. Predictive metrics are computed on the returned subset. The aggregates include both constraint-satisfying outputs and the recorded budget-admissible fallback outputs when the AUC constraint is unmet. Only the seven benchmark datasets are included in our comparison, with ten percentages for each of seeds 1–5.

FeatureHospital. We run the supplied multi-agent implementation of FeatureHospital (Li et al., 2026) with GPT-5-mini for dataset diagnosis and triage, specialist loss selection, and pharmacist composition. Each outer-training partition undergoes one design pass, which fixes the loss terms and their coefficients without subsequent automatic tuning. The pharmacist receives training-only objective-mechanics evidence from a 30-step primary probe and 20-step counterfactual probes; these diagnostics use the training side of a single internal $70/30$ split with seed 0 and do not evaluate a downstream classifier. The final loss is optimized on the full outer-training partition for 300 Adam steps with learning rate 0.03 and sigmoid temperature 1, targeting $\lceil 0.2d \rceil$ features; early stopping is disabled. These are numerical optimization steps, not repeated LLM algorithm-design rounds. Features are ranked once by the optimized mask scores, and the ten prefixes are fixed before outer-test evaluation. Emotions and Yeast use three training-fitted equal-width bins only during specialist consultation. Feature-quality flags remain diagnostic and do not mechanically exclude features. Evaluation retains all training instances and labels and follows the shared predictor and metric protocol above.

MMFS. We independently reimplemented MMFS (Dong et al., 2020) from the paper with assistance from Codex, as author-provided code was unavailable to us. The implementation retains two archives, label-frequency-difference (LFD) guidance, convergence-information selection for the convergence archive (CA), and NSGA-III selection for the diversity archive (DA). To align evaluation with our predictive panel, we replace the original objective vector with $(1 - \mathrm{MacroAP}, 1 - \mathrm{Jaccard}, \mathrm{RL}, \mathrm{HL})$, computed with the common 10-NN predictor and fixed 2×3 internal cross-validation. Each feature percentage is searched independently under an exact budget $k = \max(1, \lceil \rho d \rceil)$, enforced by random cardinality repair after threshold decoding. The selected subsets therefore need not be nested.

Both archives contain 20 individuals and evolve for 20 generations, with crossover probability 0.7, mutation probability 0.3, a fixed DA threshold of 0.6, and Das–Dennis reference partitions of 2 (10 directions for four objectives). LFD is computed from the first internal training fold, using nonzero presence when that training matrix is binary and above-median presence otherwise. The implementation uses the prose-based crossover condition for differing selection bits, normalized rank guidance, and CA-to-DA elite exchange. From the nondominated candidates in the final CA–DA union, one subset is selected by minimizing the sum of the four validation losses after coordinate-wise min–max normalization; constant coordinates contribute zero and ties favor the earlier candidate identifier. All percentages are finalized before outer-test evaluation, and all four reported metrics use the same selected subset at each percentage. The objective set, exact feature budgets, predictor, final compromise rule, and fixed generation budget are explicit adaptations for this comparison; we do not claim reproduction of the numerical results of the original paper or convergence after 20 generations.

C.4 Profile Evaluation Settings

Table 8 gives the fixed settings used to construct the profiles. The diagnostic rates $\{5, 10, 15, 20\}\%$ are separate from the predictive evaluation rates $\{2, 4, \ldots, 20\}\%$. A field with suffix `_auc` is an unweighted arithmetic mean over the four diagnostic rates, not a trapezoidal integral or a ROC area. Rates retain equal weight even if rounding produces the same feature count. The reference-prefix fields use $k_0 = \max(1, \lceil 0.20d \rceil)$.

Table 8: Fixed profile-construction settings. All sampling and fitting use the design partition.

Setting	Value or convention
Training views	Two fixed repeats of three-fold cross-validation; six training views.
Reference prefix	20% of features, rounded upward with a minimum of one.
Reference residuals	Three nested folds within each training view; uniform 10-neighbor probabilities.
Diagnostic prefixes	$5\%, 10\%, 15\%, 20\%$, each rounded upward with a minimum of one.
Neighborhood reference sample	$\min(512, \lvert T \rvert)$ training rows, sampled without replacement.
Neighborhood probes	$\min(64, \lvert J \rvert)$ rows sampled without replacement from the reference sample.
Neighborhood size	$\min(10, \max(1, \lvert J \rvert - 1))$; self-distances are set to infinity.
Denominator floor	$\varepsilon = 10^{-12}$ for retained mass, error weighting, entropy, ratios, and neighborhood skill.
Profile aggregation	Equal means over folds and repeats; population standard deviations for dataset prevalence and cardinality.

The standardizer is fitted separately within each training view and again within each nested training split. It centers each feature and divides by its population standard deviation; constant features use unit scale. Residual-reference fitting keeps ten neighbors and rejects a nested training portion with fewer than ten rows. This differs from the sampled neighborhood diagnostic, which adjusts its neighbor count to the reference sample size.

For each view, neighborhood sampling uses a NumPy generator seeded by the fixed residual-profile seed plus 9157; sampled indices are sorted. Consequently, the sample is shared across rankings and is not resampled during search. Profile denominators at or below ε yield retained coverage one. All-zero label errors yield uniform weights in the result profile, while the dataset error-entropy statistic is zero. An all-zero feature–label association row has specialization zero and is assigned to the first label by the winning-label rule. A single-label allocation entropy is zero. A single-feature dataset has zero reported off-diagonal feature-correlation summaries on the GPU path.

The supplied CUDA configuration uses GPU `float32` arithmetic for Pearson associations, feature correlations, and reference neighbor prediction. Its correlation kernels set a relation to zero when the product of centered-vector norms is at or below machine epsilon of the selected GPU datatype. The CPU feature–label correlation kernel instead uses `float64` machine epsilon, and undefined feature–feature correlations are replaced by zero. These numerical thresholds are separate from ε. Result-profile scalar leaves must be finite; the profile aggregator rejects nonfinite or mismatched fields. Profiles do not report bootstrap intervals or a standard deviation across training views.

Table 9: Configuration of the supplied HL4 reproduction launcher. Output-token limits are maxima per request; repair budgets count additional attempts.

Setting	Value
Model / reasoning effort	`gpt-5-mini` / medium
Candidate trials / graph-node limit	10 / 12
Internal evaluation	Two repetitions of three folds
Selected-feature percentages	$2, 4, \ldots, 20$
Designer / judge / summary token limits	16,384 / 8,192 / 4,096
Format or execution / summary repairs	2 / 4
Behavior novelty / duplicate repairs	Enabled / 20
Provider transport retry limit	5
API / response format	Chat Completions / JSON schema
Compute backend / precision	CUDA / float32
CUDA device index / prediction batch size	0 / 1,024

C.5 Search Budgets and Repair Settings

The formal search budget is ten completed candidate trials after the initial graph evaluation, with at most 12 operator nodes per candidate graph. A parent closes after three direct negative children; search may therefore terminate before exhausting ten trials. An ordinary completed trial requires one designer call, two mirrored reward-judge calls, and one history-summary call. A negative edge whose parent remains open requires a navigation call, and a terminal path-score tie requires a final selection call. Repair calls do not consume completed-trial slots. Computational cost includes these calls, candidate execution, reference-profile construction, and the initial graph evaluation.

The shared runner uses `gpt-5-mini` with medium reasoning effort. Its defaults allow two format or execution repairs per structured decision, four repairs for a history summary, and 20 behavior-duplicate repairs when novelty enforcement is enabled. Novelty enforcement is disabled by default. When enabled, it rejects a proposal whose complete feature rankings match a previously evaluated graph on the training portion of every fixed cross-validation view; this check is separate from graph syntax validation. The chronological history supplied to the designer contains each previous algorithm and its compact summary, while complete evaluation and decision records remain in the run outputs.

The supplied HL4 reproduction launcher explicitly requests ten trials, enables behavior novelty with a 20-repair limit, and uses the settings in Table 9. The launcher specifies a small set of reproduction jobs; its job list is separate from the full benchmark collection. The standalone runner instead defaults to a two-trial diagnostic cutoff; a cutoff before formal termination does not freeze a winner.

C.6 Ablation Settings

Full denotes the complete FSDesign. **w/o Tree** replaces tree-based exploration with a linear trajectory: each iteration continues from the previous candidate, and the search stops after three consecutive negative rewards. **w/o Profile** removes both the dataset profile and the result profile, retaining only basic dataset-scale information in the observation and predictive metrics for reward generation. **w/o Metrics** excludes predictive metrics from design feedback and judges candidates only from their result profiles. The HL4 metrics are still used for final performance reporting, but are not supplied to the designer or reward judge to guide this variant.

w/o History removes direct access to interaction history during candidate design while preserving the exploration mechanism. The designer receives only the fixed training-data profile, the current base graph selected by exploration, and the action space. It receives no chronological summaries, other candidate graphs, past metrics or result profiles (including the recorded evaluation evidence of the base), rewards, tree statistics, or navigation rationales. These restrictions also apply to proposal-repair prompts and any reused conversation context. If novelty screening is enabled, the evaluated-graph registry and duplicate-match information remain private to exploration.

Cross-round search records are exposed only to the exploration component, including its navigation and final-selection stages. It retains the same positive-edge continuation, negative-edge navigation, three-negative-child threshold, and tree-exhaustion stopping condition as Full, with at most ten

completed candidate trials after the initial graph evaluation. Final selection maximizes accumulated path reward over all evaluated nodes, including the root and closed nodes, with the same tie-breaking procedure. Candidate evaluation and mirrored reward generation are unchanged: the judge receives anonymized evidence for the current candidate–base comparison and fixed dataset context, without access to other rounds or branches. This comparison targets the value of history available for candidate generation under unchanged exploration, stopping, and final-selection rules.

Fixed uses a manually specified graph from the same action space, computing Pearson feature–label dependence, taking the mean over labels, and ranking features with TOPK, without automated design. **One-shot** generates an algorithm once from the dataset profile and action space, without iterative evaluation or refinement.

D Additional Experiments

D.1 LLM Token Usage and Cost

Table 10 reports GPT-5-mini usage for the complete method in the main experiments, averaged over seeds 1–5 for each dataset. Each entry covers the LLM calls in an entire algorithm-design run. Input tokens include reported cached input, and output tokens include reasoning tokens.

Let N_{in} denote total input, N_{cache} the subset reported as cached, and N_{out} total output. Using the official per-million-token prices for GPT-5-mini,[1] we estimate the cost in US dollars as

$$\widehat{c} = \frac{0.25(N_{\text{in}} - N_{\text{cache}}) + 0.025N_{\text{cache}} + 2N_{\text{out}}}{10^6}. \tag{29}$$

Table 10: Mean GPT-5-mini token usage and estimated API cost per complete algorithm-design run. Each dataset averages seeds 1–5; the final row averages the seven datasets.

Dataset	Input tokens	Reported cached input tokens	Output tokens	Estimated cost (USD)
Bibtex	395,728	102,963	125,620	0.3270
Birds	298,557	65,254	92,809	0.2456
Emotions	375,969	92,595	108,724	0.2906
Genbase	350,536	62,310	88,196	0.2500
Medical	290,942	52,966	84,555	0.2299
Scene	379,149	105,651	118,738	0.3085
Yeast	422,191	109,491	111,918	0.3047
Average	359,010	84,461	104,366	0.2795

The dataset-average cost ranges from US$0.2299 to US$0.3270 per complete run, with an overall average of approximately US$0.28. This indicates a modest and acceptable LLM expense for iterative algorithm design in the evaluated setting. The selected graph subsequently executes without further LLM calls. These list-price estimates cover LLM usage only and exclude local computation for graph execution, profile construction, and downstream prediction.

[1] Official GPT-5-mini pricing: https://developers.openai.com/api/docs/models/gpt-5-mini.

D.2 Statistical Significance Analysis

We compare nine methods across the seven datasets, ranking each metric independently with lower rank indicating better performance. Each dataset contributes one rank per method, based on its score aggregated over feature percentages and seeds. Thus, datasets are the units of the rank analysis; percentages and seeds are not treated as additional independent datasets. Table 11 reports the mean ranks and the Iman–Davenport omnibus p-values. At the per-metric 0.05 level, the equal-performance null is rejected for MacroAP, Jaccard, and HL, but not for RL. The corresponding Nemenyi post-hoc analysis (Demšar, 2006) uses a critical difference of 4.540 at $\alpha = 0.05$. Figure 3 visualizes these mean ranks and the corresponding critical-difference groups. Although FSDesign has the lowest mean rank on all four metrics, this ordering does not establish a significant advantage over every baseline.

Table 11: Mean ranks over seven datasets for nine methods; lower is better. The last row reports Iman–Davenport omnibus p-values, assessed separately for each metric.

Method	MacroAP	Jaccard	RL	HL
FSDesign	**1.857**	**1.714**	**2.286**	**1.571**
FeatureHospital	4.143	5.000	5.143	5.000
DFS	6.143	5.857	5.571	5.000
MMFS	<u>4.000</u>	<u>3.714</u>	<u>4.286</u>	<u>4.000</u>
Deep-Lasso	5.286	5.286	5.286	5.429
RLBMLFS	4.714	5.143	5.286	5.143
GRROOR	7.143	6.857	7.000	7.143
SSFS	6.429	6.000	5.571	6.571
Sequential-Attention	5.286	5.429	4.571	5.143
Omnibus p-value	0.00906	0.02091	0.13761	0.00808

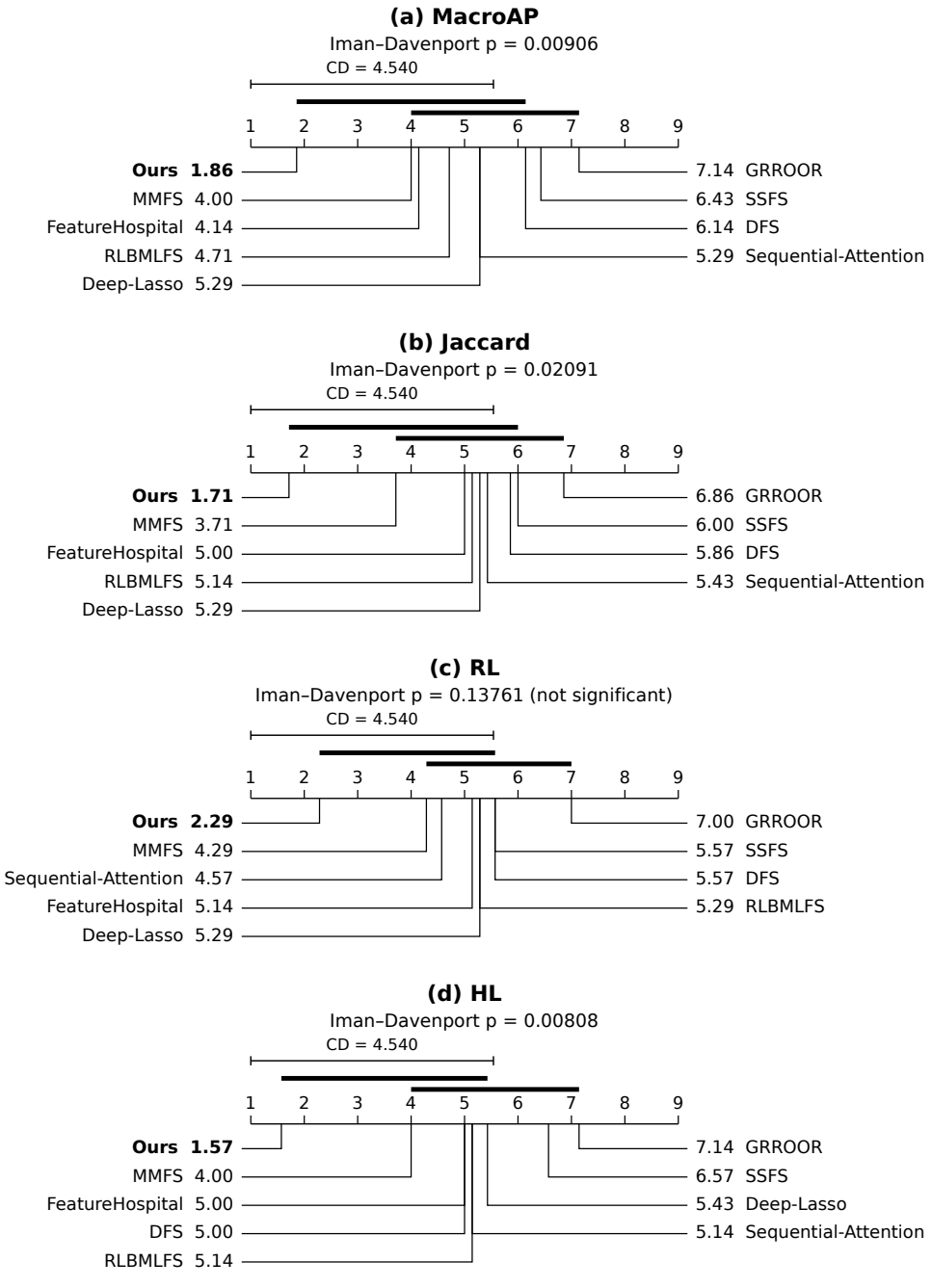


Figure 3: Critical-difference diagrams for nine methods over seven datasets. Lower mean ranks indicate better performance. Thick bars connect groups whose pairwise mean-rank differences do not exceed the Nemenyi critical difference ($\mathrm{CD} = 4.540$, $\alpha = 0.05$). Each panel reports the Iman–Davenport omnibus p-value; the RL panel is descriptive because its omnibus test is not significant.

D.3 Performance across Feature Selection Ratios

Figures 4–10 show how MacroAP changes over the ten feature-selection percentages. Each point averages seeds 1–5. FSDesign leads throughout Bibtex and remains competitive on Birds, where MMFS is stronger at some rates. On Medical, FSDesign leads at lower rates while DFS performs better at larger budgets. MMFS and Sequential-Attention are strong on Scene, while FSDesign, MMFS, and Deep-Lasso have similar results on Yeast. On Emotions, MMFS leads at most rates, whereas on Genbase several methods approach high performance as more features are retained. These curves reveal differences across feature budgets that are hidden by an across-percentage mean.

FeatureHospital is plotted using the rerun under the shared evaluation protocol, and MMFS uses independently selected subsets at each percentage. For DFS, the horizontal axis denotes the nominal feature budget; the actual subset may be smaller. The legend label Ours denotes FSDesign. The curves show mean performance without confidence intervals.

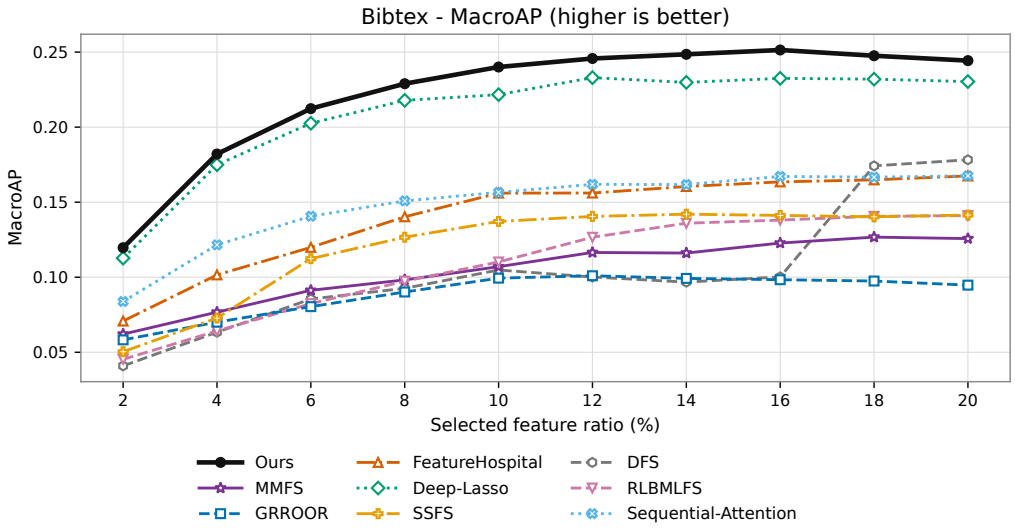


Figure 4: MacroAP across retained-feature percentages on Bibtex. Higher values are better; each point is a mean over five seeds.

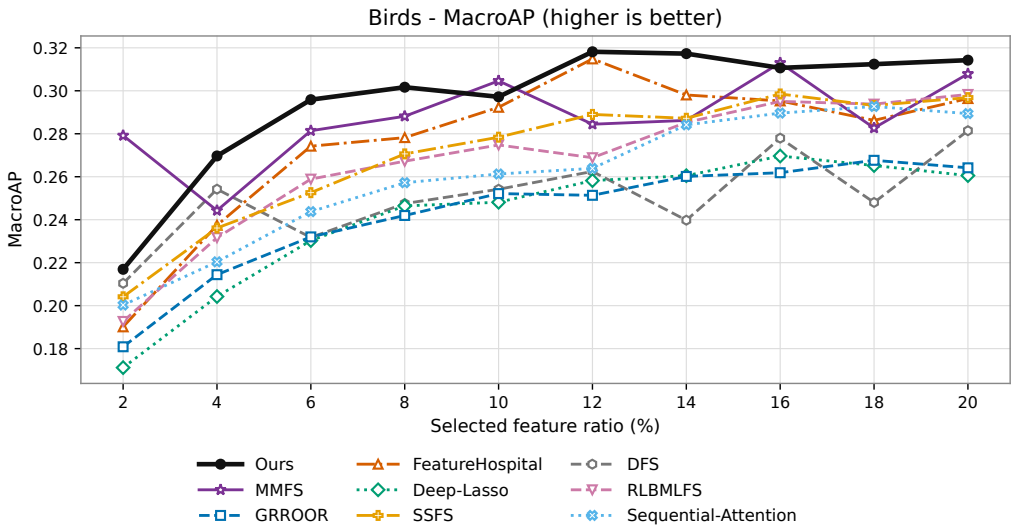


Figure 5: MacroAP across retained-feature percentages on Birds. Higher values are better; each point is a mean over five seeds.

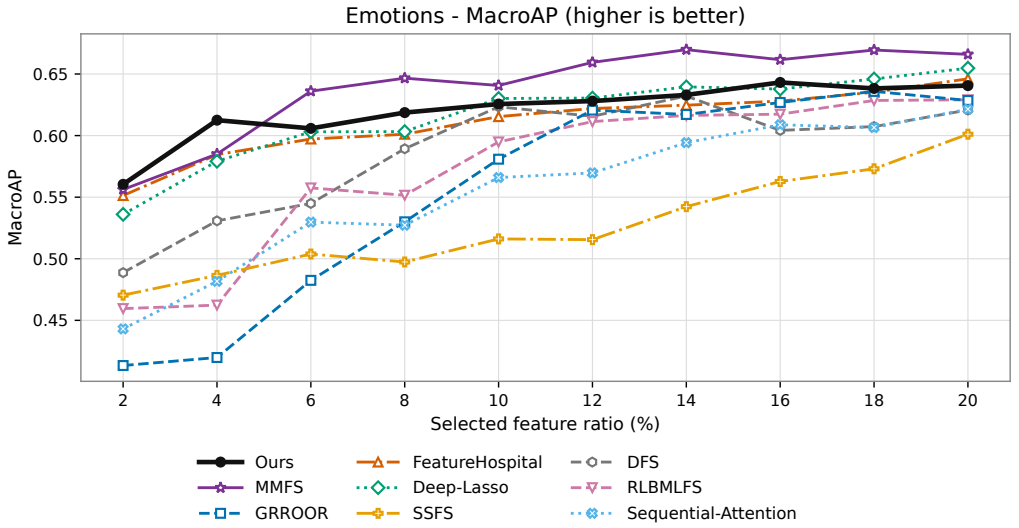


Figure 6: MacroAP across retained-feature percentages on Emotions. Higher values are better; each point is a mean over five seeds.

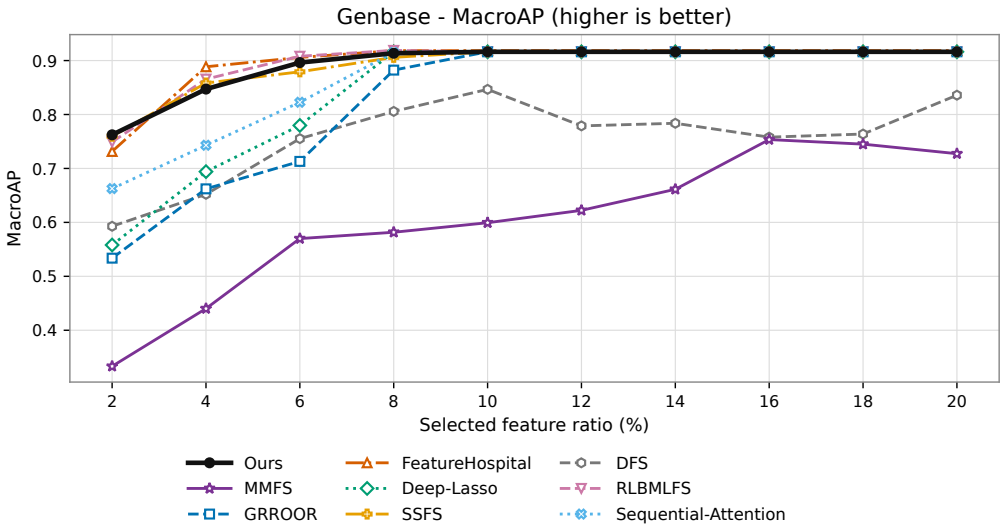


Figure 7: MacroAP across retained-feature percentages on Genbase. Higher values are better; each point is a mean over five seeds.

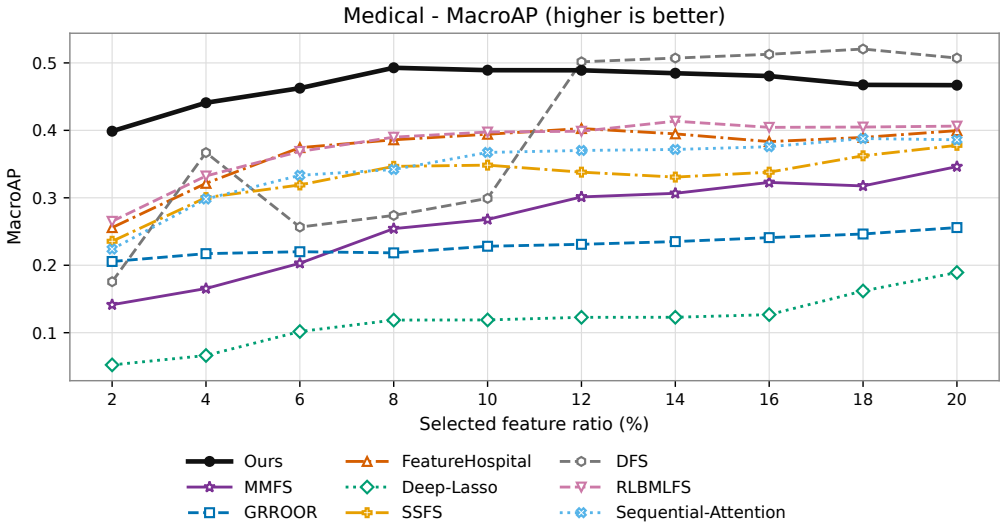


Figure 8: MacroAP across retained-feature percentages on Medical. Higher values are better; each point is a mean over five seeds.

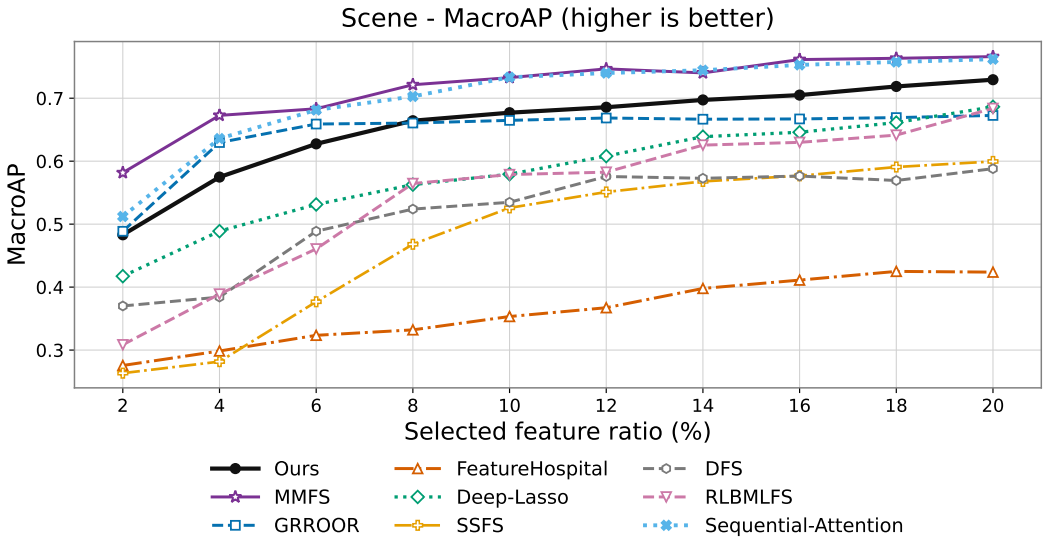


Figure 9: MacroAP across retained-feature percentages on Scene. Higher values are better; each point is a mean over five seeds.

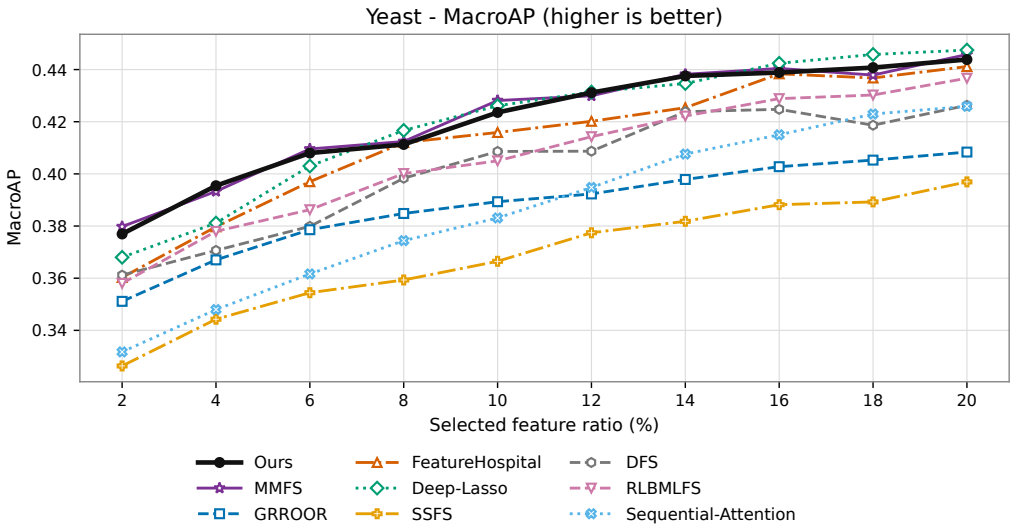


Figure 10: MacroAP across retained-feature percentages on Yeast. Higher values are better; each point is a mean over five seeds.

D.4 CASE STUDY

We examine three GPT-5-mini trajectories to illustrate algorithm construction and the complementary roles of validation metrics and result profiles in reward assessment. Throughout this subsection, a metric vector is ordered as $(\text{MacroAP} \uparrow, \text{Jaccard} \uparrow, \text{RL} \downarrow, \text{HL} \downarrow)$. Both the inner validation and held-out outer evaluation average performance over feature-selection ratios from 2% to 20% in steps of 2%. The outer results are consulted only after algorithm design and final selection, and are used here for retrospective assessment. Profile statistics follow Appendix B.3; prefix-averaged quantities use the diagnostic rates $\{5, 10, 15, 20\}\%$ specified in Appendix C.4. The metric-only heuristic and profile-only reading below are retrospective comparisons, rather than additional runs of the reward judge with one evidence source removed.

Case 1: Interpreting a selected algorithm graph. The Scene dataset contains 2,407 instances, 294 features, and six labels. In the seed=8 design partition, about 1.15% of feature values are zero and the mean label cardinality is 1.08, describing a largely dense dataset with few positive labels per instance. The final selected graph, N008, computes feature–label mutual information $I_{j\ell} = \text{MI}(X_j, Y_\ell)$, reduces it across labels using both the maximum and the mean, and rank-normalizes the resulting feature scores. Its two mutual-information branches are

$$u_j = \text{Rank}\left(\max_{\ell} I_{j\ell}\right), \qquad v_j = \text{Rank}\left(L^{-1}\sum_{\ell=1}^{L} I_{j\ell}\right), \tag{30}$$

where L is the number of labels and Rank is the normalized ascending rank transform defined in Appendix B.2. The graph combines information from these branches and also includes ratio and nonzero-frequency computations. Its terminal `TOPK` produces one complete feature ranking, whose prefixes are evaluated at the specified selection budgets.

The two dependence branches admit a "specialist-or-generalist" interpretation: the maximum emphasizes a strong association with an individual label, while the mean summarizes relevance across labels. Their combination illustrates an interpretable use of atomic operators to consider complementary feature–label relationships. The selected graph improves all four held-out outer metrics over the root graph: $(0.4887, 0.2705, 0.2582, 0.1549) \rightarrow (0.5661, 0.3614, 0.2036, 0.1452)$. This observation supports the usefulness of the selected graph on this run, without attributing the improvement to an individual branch or inferring its suitability from dataset statistics alone.

Case 2: Result profiles contextualize a positive metric average. In Emotions (seed 4, round 5), the candidate N005 and its parent N004 have inner-validation metric vectors $(0.60645, 0.42879, 0.24958, 0.23083)$ and $(0.61160, 0.41975, 0.24966, 0.23235)$, respectively. The candidate gains 0.00904 in Jaccard but loses 0.00515 in MacroAP; its RL and HL gains are much smaller. Simply averaging the four improvement-oriented differences gives approximately $+0.00137$, which would favor the candidate under an equal-weight metric-only heuristic. This heuristic is used only for the retrospective comparison and is not the reward rule of FSDESIGN.

The result profile reveals another aspect of the change: compared with the parent, the candidate reduces represented-label fraction from 0.3681 to 0.3056, label-allocation entropy from 0.2804 to 0.1775, and selected-feature nonredundancy from 0.5766 to 0.5282. Base-relevance coverage increases slightly ($0.9552 \rightarrow 0.9672$). These changes describe a tradeoff between retained relevance and label breadth/diversity, rather than a uniformly favorable or unfavorable profile shift.

The reward model compared anonymized profiles and validation measurements twice, swapping the positions of the alternatives. Both judgments identified the MacroAP–Jaccard tension and the reduction in label breadth/diversity, judged that the candidate was not a clear multi-metric improvement, and yielded a fused reward of -1. Held-out outer evaluation is consistent with this conservative decision: the parent and candidate score $(0.64167, 0.45300, 0.22436, 0.21816)$ and $(0.58464, 0.42860, 0.25925, 0.23745)$, respectively, so the candidate is worse on all four metrics. The profile supplies structural context for interpreting mixed metric changes; the outer result corroborates the decision without having influenced it.

Case 3: Metrics qualify an apparently favorable profile. In Yeast (seed 1, round 5), the candidate N005 shows several profile changes relative to its parent N003 that could appear favorable under

a profile-only reading: base-relevance coverage increases from 0.8402 to 0.8760, prefix-averaged normalized participation ratio from 0.6135 to 0.6630, and prefix-averaged nonredundancy from 0.8380 to 0.8591. These are descriptive changes, not coordinates with a universally favorable direction. Other statistics reveal a cost in label breadth: represented-label fraction falls from 0.2887 to 0.2708 and label-allocation entropy from 0.4756 to 0.4670.

The corresponding inner-validation vectors are $(0.41888, 0.42909, 0.24148, 0.22050)$ for the parent and $(0.41518, 0.42802, 0.24339, 0.22028)$ for the candidate. Thus MacroAP, Jaccard, and RL worsen, while HL improves only slightly. Both position-swapped reward judgments acknowledged the changes in ranking geometry, but preferred the parent because the apparent structural benefits were not accompanied by predictive improvements and the parent retained broader label representation; the fused reward is -1. Held-out outer evaluation supports that conclusion: parent $(0.42490, 0.45599, 0.22941, 0.21009)$ versus candidate $(0.41128, 0.44699, 0.23577, 0.21335)$, with the candidate worse on all four metrics. Here, metric evidence helps assess whether apparently favorable profile changes correspond to predictive benefits, while the profile supplies additional context for the comparison.

Together, Cases 2 and 3 illustrate complementary diagnostic roles: result profiles can expose coverage and diversity costs obscured by a positive metric aggregate, whereas validation metrics can qualify profile changes that appear favorable in isolation. These cases illustrate the value of considering both forms of evidence; they do not constitute a separate causal ablation of either input.

E PROMPTS

The following system instructions are extracted from the active implementation. Each call also receives a structured JSON payload and a response schema with the role-specific fields described in Appendix B.4. Numerical measurements, graphs, eligible candidate labels, and fixed summary constants are inserted at runtime. Repair requests add the validation error and request a complete corrected response; they do not relax the evaluation protocol.

E.1 GRAPH PLANNER

> You are the algorithm-design planner for ONE multi-label feature-selection dataset. Design the next executable feature-ranking algorithm as a typed atomic computation graph. Use ONLY the supplied operators and observed train-side evidence. Do not use outer/test data. Use only the validation metrics declared in validation_metric_protocol; treat every other predictive metric as unavailable. The previous graph is a starting point, not a fixed template: you may add, remove, or reconnect primitive operations. The graph must end in TOPK; TOPK.k must be omitted because the environment supplies the full-ranking budget. Use X and Y as source references. Every other input must reference an earlier valid graph node. Prefer a focused, testable structural modification unless existing evidence supports a larger jump. A Result Profile movement has no intrinsically good direction. Validation feedback is observed predictive evidence. Treat reward-judge interpretations and prior attribution as hypotheses, not causal ground truth; check them against the numeric evidence. Each chronological_history item contains exactly algorithm and summary. In summary, purpose is concise model-generated memory; Result Profile movement, validation-metric movement, reward, and reward rationale are evidence-bound deterministic constants. Do not interpret residual_coverage as residual-error magnitude: it measures coverage of association with a fixed reference OOF residual. Keep the proposed algorithm and anticipated follow-up experiments within the supplied operator registry and fixed evaluation protocol. Do not propose changing CV repeats, folds, split seeds, feature-selection rates, validation metrics, the downstream predictor, or outer/test data. Do not claim an untested configuration is better. State what a failure of the proposed experiment would teach. Return exactly one JSON object matching response_contract.

E.2 REWARD JUDGE

> You are the reward judgement stage for a feature-ranking search. Use only the supplied dataset measurements, Result Profiles, and fixed repeated-CV validation measurements. Judge their combined evidence; neither source automatically overrides the other. No metric weights or aggregate reward formula are supplied. Explain conflicts, uncertainty, and what the comparison teaches the next iteration. Do not infer unseen predictive performance.

Set reward_for_A to +1 if and only if preferred_alternative is A; set it to -1 for B, indistinguishable, or tradeoff. The program privately maps the anonymous preference to the candidate edge. Do not infer algorithm identity. The structured_attribution object is mandatory and must contain every field declared by response_contract, including validation_effect, profile_change_interpretation, profile_validation_relationship, profile_pattern_assessment, most_plausible_explanation, and next_experiment_advice. Do not omit a required field even when its content is uncertain; use an allowed uncertainty enum and explain it. Result Profile coordinates are descriptive train-side measurements, not causal effects. In particular, residual_coverage and related residual-coverage coordinates measure how much feature association with a fixed reference OOF residual is represented; they are not the magnitude of predictive residual error, and neither a higher nor a lower value is intrinsically better. Do not call a profile shift a mechanism or mediator unless the supplied evidence actually identifies one. When validation and profile movements disagree, state the disagreement rather than forcing a causal story. Keep next-experiment advice within the fixed search and evaluation protocol: do not propose changing CV repeats, folds, split seeds, feature-selection rates, validation metrics, the downstream predictor, or using outer/test data. Do not prescribe manual feature swaps or unavailable training procedures; express the advice as an algorithmic property or hypothesis that a planner can test with its available operators. Return exactly one complete JSON object matching response_contract.

E.3 History Summarizer

You return one compact completed-round memory object for a future feature-ranking planner. Only purpose is LLM-authored prose: write one short, grammatically complete sentence that describes the supplied graph's goal, without mentioning reward, judges, comparison outcome, IDs, hashes, protocols, tree navigation, folds, or rates. result_profile_change, metric_change, and every reward field are evidence-bound constants in response_contract. Copy all of them exactly; never recompute, rewrite, expand, interpret, or translate them. The Result Profile sentence reports raw candidate-minus-reference descriptive deltas and does not assert an inherent good/bad direction. The metric sentence reports direction-normalized improvement deltas. The reward reason reports the frozen mirrored-judge fusion outcome. Target at most 160 characters for purpose; its hard rejection limit is 220 characters. Every text field must end with sentence punctuation. Return exactly one JSON object matching response_contract.

E.4 Negative-Edge Navigation

Act only as the search-tree navigation stage. The reward and shared post-decision attribution are already frozen and cannot be revised. Choose continue_from_failed_child only when the supplied attribution supports a focused corrective experiment on that exact child; otherwise choose rollback_to_parent. Do not infer unseen performance or redo the scientific attribution. Return only JSON matching response_contract.

E.5 Final Tie Breaking

Resolve a final tie between eligible feature-ranking candidates. Compare their executable algorithm descriptions, objective dataset and Result Profile measurements, and fixed repeated-CV validation results. No aggregate metric weights are supplied; discuss conflicts and robustness. Do not infer or request unseen evaluation information. Select exactly one eligible node and return only JSON matching the contract.