WHAT TO TRY, WHERE TO TRY IT: ONLINE ADAPTIVE SEARCH POLICIES FOR MACHINE LEARNING AGENTS.

Anonymous authors
Paper under double-blind review

ABSTRACT

When should a machine learning agent start over, repair a failed program, or refine an existing solution? And which of its own past solutions should it work on? These two decisions determine how an agent spends its experimental budget, yet common search controllers fix both with hand-designed rules that do not learn from the outcomes of previous expansions. We propose Factorized Adaptive Search (FAS), an online controller that separates *operator allocation* (what to try) from *parent selection* (where to try it). A fitness-rate-rank bandit chooses among Draft, Debug, and Improve, while operator-specific LinUCB models select eligible parents from validation and search-history features. Both components learn from a shared validation-improvement reward normalized by action cost, start afresh for every run, and never observe held-out test feedback. We evaluate FAS as a controlled intervention on top of AIRA-DOJO across 11 MLE-bench Lite tasks: we hold the language model, prompts, and operators fixed while varying the two policies independently and together. Learning the operator policy alone (FAS-OP) raises anytime percentile-rank AUC from 0.781 to 0.820 over the fixed AIRA baseline, using 26% fewer tokens, producing more search-valid candidates (42.7% to 52.8%), more medal-worthy solutions among them (36.8% to 48.8%), and the best mean final rank across the four conditions (1.59 vs. 2.14). Adding online parent selection, alone or together with the operator policy, instead reduces performance below the fixed-rule baseline. This asymmetry is informative: under the sparse, noisy validation rewards available online, learning *what to try* is the more valuable and more reliably learnable decision, while learning *where to try it* requires more care than a direct application of contextual bandits provides.

1 INTRODUCTION

Language-model agents are increasingly capable of implementing complete machine learning pipelines, executing experiments, and revising solutions from feedback. MLE-bench made this progress measurable on realistic Kaggle competitions (Chan et al., 2025); AIDE, MLE-STAR, SELA, MARS, and AIRA show that iterative refinement and structured search can substantially improve over single attempts (Jiang et al., 2025; Nam et al., 2025; Chi et al., 2024; Chen et al., 2026; Toledo et al., 2025). The central resource in these systems is not only the language model. It is the finite budget of candidate programs that can be generated, executed, diagnosed, and compared.

Most agent scaffolds allocate that budget with a fixed policy. Greedy search repeatedly improves the current validation incumbent; Monte Carlo tree search (MCTS) uses a hand-designed exploration bonus; evolutionary search samples parents and operators from fixed schedules. AIRA's controlled study shows that the interaction between operators and search policy matters substantially (Toledo et al., 2025), but each policy still hard-codes how node selection, operator choice, and stopping interact. This is brittle: Draft can be valuable before a viable solution exists, Debug after execution failures, and Improve after a useful direction emerges. Which action is useful depends on the task, the observed graph, and the remaining budget, and it can change within a single run.

A separate line of work strengthens the case that allocation matters without answering whether it can be learned online with no prior data. FML-Bench finds that a stagnation-triggered heuristic can benefit from switching between focused and broad search (Zou et al., 2026); MARS introduces cost-constrained MCTS and reflective memory (Chen et al., 2026); AI Research Preference Models

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.

(RPMs) use frozen LLM judges or pilot experiments to rank *unexecuted sibling proposals* before committing execution compute, using an offline subtree-comparison corpus (Foster et al., 2026). These methods motivate a narrower question that they do not themselves answer: within one run, with no offline training data and no access to held-out feedback, can an agent learn *which already-evaluated state to expand* and *which operator to apply*, and does adapting one of these decisions matter more than the other?

We answer this question with FAS, a factorized online controller built on top of AIRA's Greedy search mechanics. At each step, a bandit chooses among currently feasible operators (Draft, Debug, Improve). A separate operator-specific linear model then scores each compatible, already-evaluated parent and selects one. After execution, both policies receive the same validation-improvement reward and update their own statistics; all learning starts afresh within each run. The generated code, execution environment, analyzer, and evaluator are unchanged throughout. This separation supports controlled operator-only, parent-only, and combined comparisons against the fixed AIRA rule, isolating the search policy from everything else an agent study can otherwise confound.

Two design choices keep this comparison clean. First, action feasibility is state-dependent: Debug requires a debuggable failure and Improve requires a valid parent, so the policy receives a feasibility mask rather than being penalized for unavailable actions. Second, both policies read only a normalized, allowlisted view of the search graph and never the held-out test score; `use_test_score=true` is rejected by construction, so the online decisions cannot be influenced by information unavailable at deployment.

In our evaluation, we find a clear asymmetry. Learning the operator policy (FAS-OP) improves anytime search quality, token efficiency, candidate validity, and medal yield over the fixed AIRA rule. Learning the parent policy (FAS-PARENT), whether alone or together with the operator policy (FAS-OP+PARENT), instead underperforms the simple rule of always expanding the current best-validation node. It identifies, with a controlled 2×2 design, which of the two decisions in an AIRA-style agent is the more valuable and more reliably learnable target for online adaptation, and it points to why the other one is harder (sparse per-parent reward and validation gains that do not always transfer to held-out performance).

Our contributions are:

- We factor the online control of an LLM-driven ML engineering search into two independently testable decisions – an operator policy π_{op} over feasible actions and a parent-selection policy π_{sel} over compatible evaluated nodes – while freezing the language model, prompts, operators, and evaluation pipeline.
- We adapt a rank-based multi-armed bandit (fitness-rate-rank credit, phase-aware and feasibility-masked) to operator selection, and a linear contextual bandit to parent selection from decision-time search features. Both learn online from one shared, cost-normalized validation-improvement reward, with no offline training data and no access to held-out feedback.
- FAS-OP raises anytime percentile-rank AUC from 0.781 to 0.820 over AIRA while using 26% fewer tokens, and improves candidate validity, medal yield, and mean final rank. FAS-PARENT and FAS-OP+PARENT instead perform below the fixed AIRA rule on the same metrics.
- We give process-level evidence for why: operator usage shifts from a debugging-heavy allocation under the fixed rule toward substantially more Drafting under FAS-OP without sacrificing Debug or Improve yield, while the parent bandit receives few positive-reward updates and can be misled on tasks where validation improvement does not track held-out performance.

2 RELATED WORK

Machine learning engineering and research agents. MLE-bench evaluates end-to-end agents on 75 Kaggle competitions and supplies a human-referenced ranking protocol (Chan et al., 2025). AIDE casts MLE as tree-structured code optimization (Jiang et al., 2025); SELA applies MCTS to AutoML pipelines (Chi et al., 2024); MLE-STAR combines web retrieval, targeted component refinement,

and ensembling (Nam et al., 2025); and MARS adds cost-aware planning and comparative reflective memory (Chen et al., 2026). MLE-Dojo supplies an interactive environment for training MLE agents (Qiang et al., 2025), while SandMLE uses small synthetic environments to make on-policy training affordable (Zhou et al., 2026). FAS does not train the base LLM; it learns the search scaffold around a frozen operator stack, so any gain is attributable to the search policy alone.

Search policy for research agents. AIRA factorizes research agents conceptually into fitness, selection, operators, operator policy, and termination, and compares Greedy, rollout-free MCTS, and evolutionary search, showing that the choice of search policy and operator set jointly determine performance (Toledo et al., 2025). AIRA-DOJO$_2$ adds asynchronous execution, more consistent evaluation, and interactive operators (Hambardzumyan et al., 2026). We take the same factorization and ask a question orthogonal to comparing fixed policies: given one fixed operator set, can the operator choice and the parent choice each be learned online from the run's own validation feedback, and does one of them matter more than the other. FML-Bench's stagnation rule (Zou et al., 2026) and RPM's offline sibling ranking (Foster et al., 2026) address related but distinct questions – switching exploration breadth on a trigger, and comparing unexecuted children before spending execution budget on them, respectively – rather than online value estimation over already-evaluated states.

Adaptive operator selection and non-stationary bandits. Upper-confidence-bound methods balance exploitation and exploration using optimism under uncertainty (Auer et al., 2002), and sliding-window variants address changing reward distributions (Garivier & Moulines, 2011). In evolutionary computation, the fitness-rate-rank multi-armed bandit (FRRMAB) ranks recent improvement rates and applies a decaying credit before bandit selection (Li et al., 2014). We adapt this credit rule to costly, state-dependent language-model operators: rewards are normalized by action cost (which varies with generation, execution, and analysis time), arms are masked by run-time feasibility rather than always available, and credit is computed separately within each budget phase.

Contextual bandits and learned search value. Learned policy and value estimates can guide tree expansion (Silver et al., 2017). LinUCB combines a linear reward estimate with an uncertainty bonus for contextual action selection (Li et al., 2010). We adapt LinUCB's linear reward model and exploration bonus to score already-evaluated search nodes – rather than unexecuted candidate actions – using only features available at decision time, sharing one parameter vector per operator across all eligible parents so the model requires no offline parent-model training corpus.

3 Problem Formulation

An agent run produces a directed acyclic search graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$. A node $v \in \mathcal{V}_t$ contains a candidate program, its public execution output, an online proxy score $\mathcal{F}(v)$, and provenance. Applying operator $o \in \mathcal{O}$ to compatible parent nodes yields a child, incurs observed cost c_t, and reveals a new proxy score. The private test score $s_{\text{test}}(v)$ is available to the offline evaluator but never to the online policy.

At step t, the controller receives an online state S_t containing an allowlisted view of $\mathcal{G}_t$, budget used b_t/B, and a feasibility mask $\mathcal{A}_t \subseteq \mathcal{O}$. It first chooses an operator

$$o_t \sim \pi_{\text{op}}(o \mid S_t, \mathcal{A}_t), \tag{1}$$

then selects compatible parent nodes

$$P_t \sim \pi_{\text{sel}}(P \mid S_t, o_t), \qquad P_t = \varnothing \text{ for Draft}, \quad P_t = \{v_t\} \text{ otherwise.} \tag{2}$$

The unchanged operator generates a child, which is executed, analyzed, and appended to the graph. Thus a research agent is $(\mathcal{F}, \pi_{\text{sel}}, \mathcal{O}, \pi_{\text{op}}, \tau)$ with a fixed termination rule τ enforcing the common wall-clock budget, but only π_{sel} and π_{op} vary across our four conditions.

At budget fraction q, the deployed agent returns the direction-corrected validation incumbent $\widehat{v}_\pi(q) = \arg\max_{v: c(v) \le qB,\ v \text{ valid}} \mathcal{F}(v)$, with deterministic tie-breaking. Our primary evaluation is the normalized area under the private percentile rank of this *online-selected* incumbent,

$$\text{AUC}_{\text{sel}}(\pi) = \int_0^1 \text{PR}(\widehat{v}_\pi(q))\ dq, \tag{3}$$

where $c(v)$ is cumulative wall-clock time within one four-hour run. An oracle-best-in-graph AUC, obtained by replacing $\widehat{v}_\pi(q)$ with the private best node generated by q, is reported alongside it as a search-coverage upper bound: it separates how good the generated pool of candidates is from how well the policy selects within that pool. Endpoint selected-incumbent percentile rank and medal rate are secondary. The online policy sees only the proxy; private percentile rank is joined after a run closes and is never exposed during search.

Controlled action space. The action set is $\mathcal{O} = \{\text{DRAFT}, \text{IMPROVE}, \text{DEBUG}\}$, matching AIRA's stock Greedy operator set. Draft is always feasible; Improve requires at least one valid node; Debug requires a buggy leaf within a fixed maximum debug depth (20). Backtracking is realized by selecting an older parent for Improve rather than by a separate operator.

4 FACTORIZED ADAPTIVE SEARCH

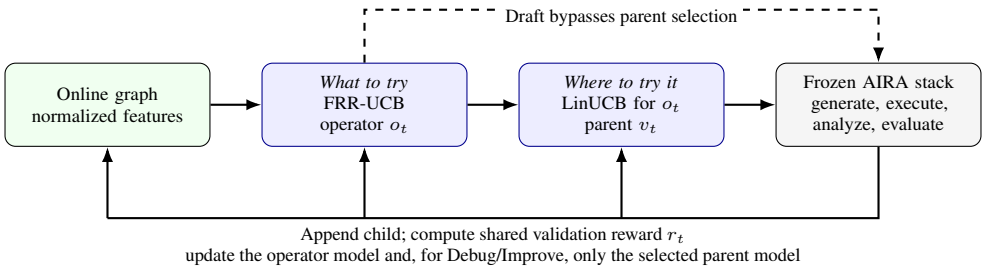


Figure 1: **Factorized Adaptive Search.** FRR-UCB chooses the operator (*what to try*); LinUCB chooses an eligible parent for Debug or Improve (*where to try it*). Both adapt online from the same validation reward. Draft updates only the operator model. Private test reports are isolated from features, rewards, and both model states.

4.1 COMMON WARM START AND FEASIBILITY MASK

Every run begins with the same number ($K = 5$) of warm-start Draft evaluations before adaptive decisions begin, under the same seed across the four conditions. Their generation, analysis, and execution costs are charged to the run's budget, but they are not treated as policy-selected evidence for learning. Thereafter, one shared function computes $\mathcal{A}_t$ and the compatible parent pool for each operator. If an action becomes infeasible it is masked rather than assigned zero reward, and analysis is mandatory post-execution overhead rather than a bandit arm. Warm-start Draft rewards populate the operator bandit's history identically across conditions; the parent model starts from its prior after the warm start, since Draft supplies no parent observation.

4.2 OPERATOR SELECTION: FRR-UCB

Let $f_t^\star$ be the best observable validation proxy after step t. For higher-is-better tasks, an action receives cost-normalized incumbent progress

$$r_t = \frac{\max(0, f_t^\star - f_{t-1}^\star)}{\max(c_t, \epsilon)}, \tag{4}$$

after direction-correcting the proxy so larger is better, with a task-specific floor before the first valid candidate ($\epsilon = 10^{-12}$). Both components receive this same scalar r_t. One completed candidate evaluation defines the cost unit $c_t = 1$, so the charged interval already reflects operator generation, mandatory analysis, execution, evaluation, policy inference, failures, and retries relative to that unit.

Budget fraction maps each step to phase $p_t \in \{\text{early}, \text{middle}, \text{late}\}$ at the boundaries $1/3$ and $2/3$ of the wall-clock budget. A *common calendar window* $\mathcal{W}_p(t)$ contains the last $W = 8$ action–reward pairs observed in phase p. Its window-local counts and fitness rates are

$$N_{p,o} = \sum_{i \in \mathcal{W}_p(t)} \mathbb{1}[o_i = o], \qquad R_{p,o} = \frac{\sum_{i \in \mathcal{W}_p(t)} r_i \mathbb{1}[o_i = o]}{\max(1, N_{p,o})}, \quad N_p = |\mathcal{W}_p(t)|. \tag{5}$$

Observed arms are ranked by $R_{p,o}$, with average ranks for ties and rank one best, and receive decayed rank credit

$$C_{p,o} = \frac{D^{\mathrm{rank}_{p,o}-1} R_{p,o}}{\sum_{a \in \mathcal{A}_t} D^{\mathrm{rank}_{p,a}-1} R_{p,a} + \epsilon}, \qquad D = 0.5. \tag{6}$$

Each arm is forced once per phase when it first becomes feasible. Thereafter, the selected feasible arm maximizes

$$U_{p,o} = C_{p,o} + \alpha \sqrt{\frac{2 \log(1 + N_p)}{1 + N_{p,o}}}, \qquad \alpha = 0.25. \tag{7}$$

When all recent rewards are zero, all credits are set to zero; seeded uniform tie-breaking is used throughout. We call this FRRMAB-inspired, windowed-credit UCB rule *FRR-UCB*. It uses counts from the same window as its rewards, so stale pulls do not suppress exploration after drift.

4.3 Parent selection: LinUCB over evaluated nodes

Draft requires no parent. For the selected operator o_t, the shared eligibility function returns

$$\mathcal{V}_t^{o_t} = \begin{cases} \{\text{eligible buggy leaves}\}, & o_t = \textsc{Debug}, \\ \{\text{valid nodes}\}, & o_t = \textsc{Improve}. \end{cases} \tag{8}$$

Improve can revisit internal nodes; Debug obeys the common maximum debug depth. Empty pools make their operator infeasible. Each eligible node has a decision-time feature vector

$$\begin{aligned} x_t(v) = [\,1,\ &\text{validation/ancestor rank},\ \text{depth},\ \text{age},\ \text{children count}, \\ &\text{expansion count},\ \text{mean child reward},\ \text{last child reward},\ \text{origin operator}\,]^\top. \end{aligned} \tag{9}$$

All features are normalized using only information available before this decision. Depth, age, materialized-children count, and expansion count are each divided by a shared reference cap $N_{\mathrm{ref}} = 500$ (the run's candidate-evaluation limit) and clipped to $[0, 1]$; child-reward features use the saturating map $z \mapsto z/(1+z)$ rather than a hard cap. Origin operator is one-hot encoded; the intercept and seven numeric features plus three operator indicators give $d = 11$. A valid node uses its own validation rank; a buggy node uses its nearest valid ancestor's rank, or zero if none exists. Child rewards summarize previously completed expansions, with zero for empty histories. Appendix B fixes normalization and missing-value conventions.

Maintain separate linear models for $o \in \{\textsc{Debug}, \textsc{Improve}\}$:

$$M_o = \lambda I_d, \qquad b_o = 0, \qquad \widehat{\theta}_o = M_o^{-1} b_o, \qquad \lambda = 1. \tag{10}$$

Each model shares parameters across all eligible parents for its operator; there is no separate fitted model for every node. Its LinUCB score is

$$U_t^o(v) = \widehat{\theta}_o^\top x_t(v) + \beta \sqrt{x_t(v)^\top M_o^{-1} x_t(v)}, \qquad v_t = \arg\max_{v \in \mathcal{V}_t^{o_t}} U_t^{o_t}(v), \qquad \beta = 0.5. \tag{11}$$

The first term estimates immediate validation reward; the second encourages exploration of less-observed feature directions. Ties use a seeded uniform draw from a stable candidate ordering. After receiving r_t, update only the selected operator's model, using the *saved pre-decision* vector:

$$M_{o_t} \leftarrow M_{o_t} + x_t(v_t) x_t(v_t)^\top, \qquad b_{o_t} \leftarrow b_{o_t} + r_t x_t(v_t). \tag{12}$$

A zero reward still updates M_{o_t}; Draft updates neither parent model. Both models reset for every task run and retain cumulative statistics within that run, with no phase resets or forgetting. The parent target is the raw shared r_t, whereas the operator bandit converts its recent observations into rank credit.

4.4 Combined controller and information boundary

The adaptive decision is

$$(o_t, v_t) = \big(\text{FRR-UCB operator selection}, \text{LinUCB parent selection}\big), \tag{13}$$

with $v_t = \varnothing$ for Draft. FAS-OP (operator bandit only) retains AIRA's validation-best valid parent for Improve and a seeded-uniform eligible buggy leaf for Debug. FAS-PARENT (parent bandit only) retains AIRA's stock operator rule: after K drafts, attempt Debug with fixed probability when eligible, otherwise Draft if no valid node exists, otherwise Improve. FAS-OP+PARENT applies both learned decisions. All four conditions share identical eligible pools, and the same r_t updates the operator model and, for parented actions, the selected parent model; it is charged and logged once per attempted expansion.

LinUCB estimates the reward of the current expansion, not an unobserved subtree return, and the two learners interact because the parent policy changes the reward distribution seen by the operator bandit. An explicit feature schema prevents either policy from accessing private reports, and `use_test_score=true` is rejected. Traces store the decision-time vector, prediction, bonus, r_t, and both policies' states; private outcomes are joined only after the run closes. Algorithm 1 specifies the complete loop.

5 EXPERIMENTAL PROTOCOL

Tasks and evaluation signal. We evaluate on 11 tasks from MLE-Bench Lite spanning tabular, vision, and text problems: Spooky Author Identification, Leaf Classification, Aerial Cactus Identification, Nomad2018 Predict Transparent Conductors, Plant Pathology 2020, Dogs vs. Cats Redux, Detecting Insults in Social Commentary, MLSP 2013 Birds, Right Whale Redux, Random Acts of Pizza, and Jigsaw Toxic Comment Classification. We randomly selected benchmarks from MLE-Bench Lite to ensure a reasonable coverage across tasks, data modalities, and prediction problems, enabling effective assessment of our methods' robustness across diverse settings. Each condition runs for a four-hour wall-clock budget per task, with a single seed per task and snapshots every 30 minutes. All statistics reported in the rest of this paper come from this one seed for each of the 11 tasks. Rates and percentages are task means, and count totals sum completed work across tasks. The online policy is scored on the agent-reported validation proxy $\mathcal{F}$, and the held-out test score is joined only after each run closes.

Frozen agent stack. All four conditions use the same language model, Draft/Improve/Debug prompts, memory renderer, analyzer, execution container, package set, and dataset mounts, with `use_test_score=false`. Wall-clock time is the single budget that defines both stopping and the horizontal axis in our figures. If a run stops early, its last online-selected incumbent is carried forward; a run with no valid incumbent receives the preregistered floor.

Compute. Each run executes on a single NVIDIA GeForce RTX 4090 GPU with 24 GB of VRAM; the host system has an AMD EPYC 7513 32-core CPU (64 threads) and approximately 516 GB of system RAM. With 11 tasks, 4 methods, 1 seed, and a 4-hour budget per run, the full study totals 176 GPU-hours.

Methods. We compare four conditions in a 2×2 design that turns operator adaptation and parent adaptation on or off independently: AIRA (both fixed, matching the stock Greedy solver), FAS-OP (operator bandit only), FAS-PARENT (parent bandit only), and FAS-OP+PARENT (both adaptive). AIRA is the single comparator because it is the frozen agent stack our controller is built directly on top of, so any difference is attributable to the search policy alone rather than to prompts, tools, or model changes; it is also the strongest policy in AIRA's own controlled study, where Greedy earns the most medals and swapping in more advanced search policies such as MCTS or evolutionary search "gain[s] no advantage, underscoring the operator set as the bottleneck" (Toledo et al., 2025).

Metrics. We report the anytime selected-incumbent PR-AUC (Equation 3) and its oracle-coverage upper bound, endpoint selected-incumbent percentile rank, endpoint medal rate, pooled and task-mean candidate validity, medal rate among valid candidates, mean final rank across all four conditions, and total recorded tokens, each computed following standard aggregate-evaluation practice for a small task suite (Agarwal et al., 2021) and computed on the single seed as defined above.

6 RESULTS

Table 1: **Matched-compute headline results.** Higher is better for all columns. FAS-OP attains the best selected and oracle AUC and the best endpoint percentile rank, while using the fewest tokens of any condition. Medal-rate detail is in Appendix D.

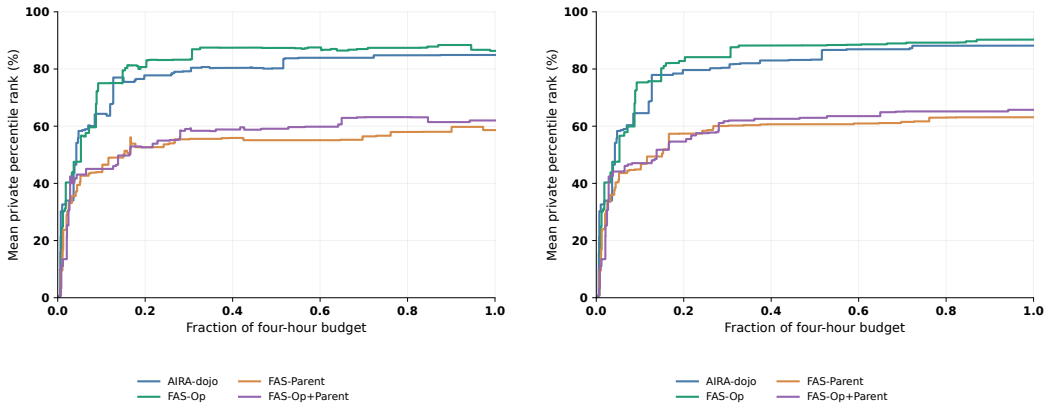
Method	Adaptive operator	Adaptive parent	Selected AUC	Oracle AUC	Endpoint selected PR
AIRA-DOJO	no	no	0.781	0.805	0.849
FAS-OP	yes	no	**0.820**	**0.833**	**0.863**
FAS-PARENT	no	yes	0.534	0.577	0.586
FAS-OP+PARENT	yes	yes	0.565	0.594	0.620

Table 2: **Candidate-level efficiency.** Valid% is the fraction of completed candidates that are search-valid; Medal% is the fraction of valid candidates that earn a held-out medal. FAS-OP leads on every column.

Method	Valid% (pooled)	Valid% (task mean)	Medal% (of valid)	Medals / M tokens	Mean final rank
AIRA-DOJO	42.7	55.5	36.8	7.4	2.14
FAS-OP	**52.8**	**58.0**	**48.8**	**13.0**	**1.59**
FAS-PARENT	45.4	53.6	20.8	4.7	3.41
FAS-OP+PARENT	40.9	52.5	17.0	3.6	2.86

6.1 LEARNING THE OPERATOR POLICY IMPROVES ANYTIME PERFORMANCE AND EFFICIENCY

Tables 1 and 2 and Figure 2 contain the primary comparison. FAS-OP raises selected PR-AUC from 0.781 to 0.820. FAS-OP also attains the highest oracle-coverage AUC (0.833 vs. 0.805 for AIRA) with the smallest oracle gap of any condition (0.013 vs. 0.023 for AIRA), so its gain reflects both a better pool of generated candidates and more efficient use of that pool, not only better selection among a fixed pool. Head-to-head against AIRA specifically on final held-out test score, FAS-OP wins on 7 of the 11 tasks, ties on 1, and loses on 3 (Appendix D gives the full outcome summary across all four methods). FAS-OP reaches this quality using 11.1M recorded tokens against 15.0M for AIRA, a 26% reduction, and its pooled search-valid rate is 52.8% against 42.7% for AIRA (Table 2, Figure 3a). Among valid candidates, 48.8% earn a held-out medal under FAS-OP versus 36.8% under AIRA, and FAS-OP attains the best mean final rank of the four conditions (1.59 vs. 2.14, Figure 6).



(a) Validation-selected incumbent.

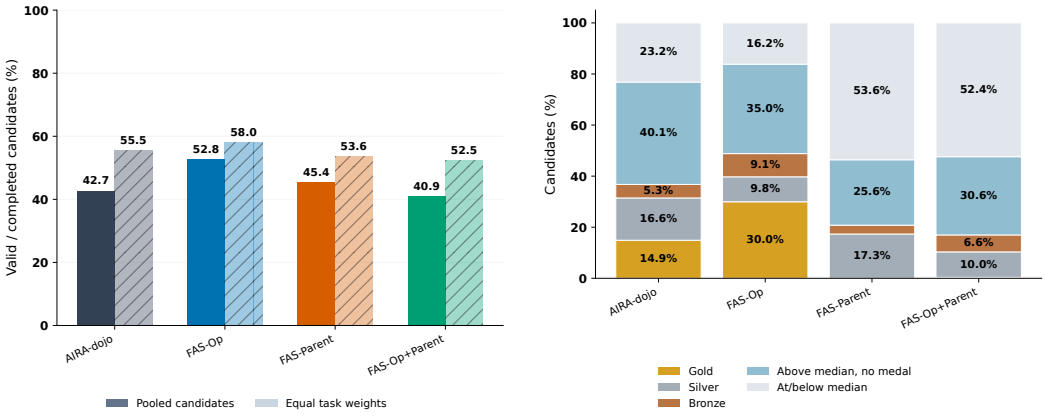
(b) Private oracle (coverage upper bound).

Figure 2: **Anytime search quality** across the 11 tasks. (a) Percentile rank of the online-selected incumbent against normalized wall-clock budget for all four conditions. (b) The corresponding oracle-coverage upper bound, obtained by replacing the selected incumbent with the best node generated so far. FAS-OP leads on both panels throughout most of the budget.

6.2 LEARNING THE PARENT POLICY UNDERPERFORMS A FIXED SELECTION RULE

The same 2×2 design shows the opposite pattern for parent selection. FAS-PARENT lowers selected PR-AUC to 0.534 and FAS-OP+PARENT to 0.565, both below the fixed AIRA rule (Table 1). The degradation is consistent across candidate-level metrics: pooled medal rate among valid candidates falls to 20.8% for FAS-PARENT and 17.0% for FAS-OP+PARENT, versus 36.8% for AIRA (Figure 3b), and mean final rank worsens to 3.41 and 2.86 respectively, the two worst of the four conditions (Figure 6). Adding the operator bandit on top of the parent bandit (FAS-OP+PARENT vs. FAS-PARENT) recovers some quality (selected AUC 0.565 vs. 0.534) but does not close the gap to AIRA, let alone to FAS-OP: FAS-OP+PARENT earns a medal on 36.4% tasks and is the outright best-scoring method on 27.3%, below both FAS-OP's 54.5% and 45.5% (Appendix D). This asymmetry – one adaptive component helps, the other hurts, and combining them does not recover the loss – is the central empirical finding of the 2×2 design, and we analyze its source next.

7 WHY DOES OPERATOR ADAPTATION HELP MORE THAN PARENT ADAPTATION?

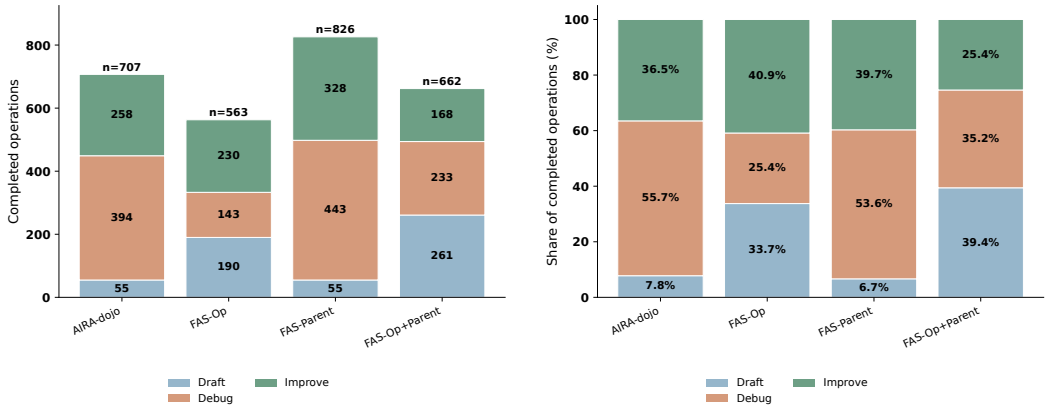


(a) Candidates accepted by search.

(b) Held-out medal tier among valid candidates.

Figure 3: **Search-valid rate and medal yield**, single seed. FAS-OP produces the largest share of search-valid candidates and converts them into medals at the highest rate; FAS-PARENT and FAS-OP+PARENT trail AIRA-DOJO on medal yield despite FAS-PARENT matching or exceeding it on raw validity.

FAS-Op drafts more, debugs less, and improves more wisely – and that combination is what drives the efficiency gain. Figure 4 shows that AIRA's fixed rule spends 55.7% of its completed operations on Debug and only 7.8% on Draft; FAS-OP instead allocates 33.7% to Draft, 25.4% to Debug, and 40.9% to Improve. This is not simply diversification: Table 3 shows that every operator converts into a medal at a higher rate per attempt under FAS-OP than under AIRA's fixed rule (Draft 12.7%→16.3%, Debug 11.4%→25.2%, Improve 22.9%→33.9%), so FAS-OP's gain is not simply a matter of trying more things. The mechanism is instead about how a fixed wall-clock budget gets spent: AIRA's fixed rule might repeatedly re-debugs the same failing candidates, and each such attempt tends to fail quickly (its average Debug attempt takes 155 seconds), so the budget is consumed by a large raw count of mostly unproductive repair attempts. FAS-OP instead explores more and re-debugs less, and its remaining Debug and Improve attempts run longer on average (244 and 277 seconds, respectively) because they are more likely to carry a candidate through to a valid, evaluated result rather than failing early. Within the same four-hour budget this means FAS-OP completes fewer operations in total (563 vs. 707) but a higher percentage of them are search-valid: the pooled search-valid rate rises from 42.7% under AIRA to 52.8% under FAS-OP (Figure 3a). It is this shift in the *proportion* of the budget spent on productive work, not a larger volume of attempts, that makes FAS-OP more token-efficient overall.



(a) Operation counts, single seed.

(b) Operation shares, single seed.

Figure 4: **Share of Draft, Debug, and Improve** among completed operations. FAS-OP moves budget away from AIRA-DOJO's debug-heavy default toward a more even split, with the largest share on Improve (40.9%) and Draft (33.7%).

Table 3: **Per-operator yield**, single seed. Medal% of attempts is medals divided by completed operations of that type. FAS-OP's Debug attempts are less than half as frequent as AIRA-DOJO's but more than twice as likely to end in a medal.

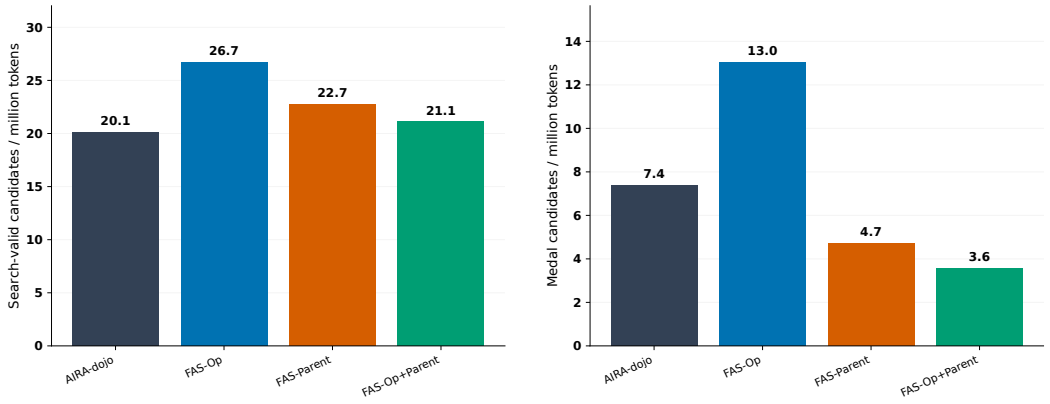
Method	Operator	Completed	Valid%	Medals	Medal% of attempts
AIRA-DOJO	Debug	394	33.2	45	11.4
AIRA-DOJO	Improve	258	57.8	59	22.9
AIRA-DOJO	Draft	55	40.0	7	12.7
FAS-OP	Debug	143	58.7	36	25.2
FAS-OP	Improve	230	61.7	78	33.9
FAS-OP	Draft	190	37.4	31	16.3

Operator share alone does not explain the outcome. Two further comparisons isolate operator allocation from parent selection directly. First, FAS-PARENT retains AIRA's exact fixed operator rule, and its resulting operator mix is close to AIRA's (Draft 6.7% vs. 7.8%, Debug 53.6% vs. 55.7%, Improve 39.7% vs. 36.5%, all within about 3 points), yet its selected PR-AUC is far lower (0.534 vs. 0.781). With the operator distribution held essentially fixed, the entire gap between FAS-PARENT and AIRA is attributable to parent choice alone. Second, FAS-OP+PARENT runs the identical FRR-UCB operator bandit as FAS-OP, but ends up with a visibly different mix – more Debug (35.2% vs. 25.4%) and less Improve (25.4% vs. 40.9%) – and a much lower selected PR-AUC (0.565 vs. 0.820), below even AIRA's fixed rule. Because the two conditions share the same operator-selection algorithm, this shift can only come from the reward the operator bandit observes: FAS-PARENT's parent choices change the realized reward of each operator, and once expansions are on average less productive, FRR-UCB's own credit assignment drifts toward a different, worse allocation. Together these two comparisons show that operator adaptation's benefit is not simply a property of which operator-mix percentages a run lands on; it also depends on the quality of the parent choices that generate the rewards the operator bandit learns from.

Parent-selection rewards are sparse, and validation gains do not always transfer. The parent bandit updates only on Debug and Improve steps, and most of those updates receive zero reward: across the pooled runs, only 5.8% of FAS-PARENT's completed candidates and 7.3% of FAS-OP+PARENT's carry a positive reward (48 out of 826 for FAS-PARENT, 48 out of 662 for FAS-OP+PARENT). With this little signal, the linear model's exploration bonus can pull it away from the node that a simple validation-best heuristic would have picked. Validation improvement also does not always correspond to a real held-out gain, so the parent model – whose update is tied to the single

node it just chose – can be trained on a misleading target more easily than the operator bandit, whose reward is aggregated over many nodes rather than one.

Cost. Figure 5 and Table 2 summarize token use. FAS-OP both spends fewer tokens in total (11.1M vs. 15.0M for AIRA, 16.5M for FAS-PARENT, and 12.8M for FAS-OP+PARENT) and converts each million tokens into more valid candidates (26.7 vs. 20.1 for AIRA) and nearly twice as many medals (13.0 vs. 7.4 for AIRA). The operator bandit's own computation is closed-form rank arithmetic over an $O(W)$ window and requires no additional LLM calls; the LinUCB parent score requires an $O(|\mathcal{V}_t^{o_t}|d^2)$ solve with $d = 11$, also negligible next to a single LLM call. The efficiency gain is therefore a consequence of better decisions, not of the policy computation itself, and Appendix E shows that the four conditions spend shares of their wall-clock budget on generation, execution, and analysis that are close to each other – the gap comes from which decisions are made with that time, not from a different time allocation.



(a) Valid candidates per million tokens.

(b) Medals per million tokens.

Figure 5: **Token efficiency.** FAS-OP converts recorded tokens into valid candidates and medals at the highest rate of the four conditions.

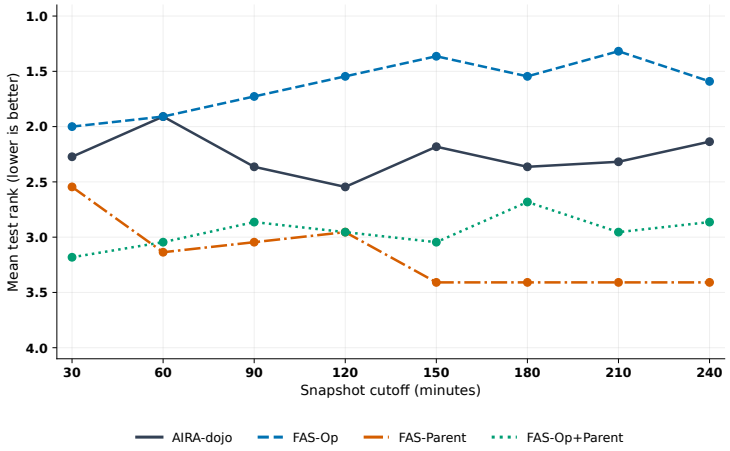


Figure 6: **Task-balanced held-out rank over time.** Mean rank (1 = best) of the validation-selected incumbent across the four conditions, with equal weight on each of the 11 tasks; failed runs tie for worst rank. FAS-OP reaches and holds the best mean rank earliest and most consistently.

8 LIMITATIONS

The linear parent model may miss nonlinear relationships between features and reward, and it observes only selected expansions, so its prediction quality on those actions does not establish correct ranking of unchosen parents. Both policies learn from sparse immediate incumbent gains rather than delayed credit: a useful repair may earn zero reward until a later Improve succeeds. The operator bandit accumulates statistics within a phase-local sliding window, while the parent model has no forgetting, so reward drift within a run can leave old linear estimates stale. Our study uses a single frozen language model and a single fixed comparator (AIRA's stock Greedy rule); performance may depend on the underlying model and may not transfer to task families with qualitative, non-numeric feedback. Although no private outcomes enter policy learning, repeated use of a validation signal can still induce selection bias on tasks where validation and held-out performance diverge, which is precisely where parent-selection learning struggles most in our results.

9 CONCLUSION

We ask a narrow question with broad practical consequences for LLM-driven research agents: given one fixed operator set, does it pay to learn which operator to apply, which node to expand, or both, online and with no prior data? FAS separates these two decisions and learns each from the same validation reward. In our study, learning the operator policy improves anytime search quality, token efficiency, candidate validity, and medal yield over AIRA's fixed rule, while learning the parent policy does not. This asymmetry is itself the contribution: it shows, with a controlled decomposition rather than a single end-to-end comparison, which source of online adaptivity is currently the more valuable and more reliably learnable one for this class of agents, and it motivates future parent-selection designs that address sparse reward and validation–test divergence directly rather than applying a contextual bandit unmodified.

Prior search-policy comparisons, including AIRA's own, vary operator choice and node choice together as part of one policy (Greedy, MCTS, evolutionary search), so whether parent selection specifically contributes was not previously isolated. Our factorization separates the two and shows they are not interchangeable: parent selection can move performance substantially, here for the worse under a direct online contextual bandit trained from scratch within a single run. That result is itself useful – it marks parent selection as a decision worth targeting rather than one that is inherently unimportant – and it opens a path for future work: richer credit assignment, offline pretraining of the parent value function, or explicit correction for validation–test divergence may recover a positive effect where our from-scratch online approach does not.

AI USE STATEMENT

Generative AI tools, including OpenAI Codex, were used to help organize related literature, critique the experimental design, edit LaTeX text, review software, and suggest reproducibility checks. All reported numerical results were computed from the recorded benchmark traces and analysis tables; the authors verified every citation, mathematical statement, and experimental claim before submission and take responsibility for the final content.

ETHICS STATEMENT

This study uses existing machine learning benchmarks and does not involve human subjects. Automating ML experimentation may lower barriers to beneficial research, but it can also increase compute consumption, facilitate unsafe model development, and amplify benchmark or dataset biases. We mitigate immediate risks through sandboxed execution, read-only data mounts, network restrictions, complete action/cost logging, and reporting of failed as well as successful runs. Any released traces will exclude secrets, private test labels, and license-restricted data. We report token usage and do not claim that benchmark performance alone measures safe or socially beneficial research.

Reproducibility statement

The method is specified in Section 4 and Algorithm 1. Appendix B defines feature normalization and update timing; Appendix C specifies the online information boundary and learning checks; and Appendix D summarizes outcomes across the full task suite. The harness, resolved configs, trace schema, analysis scripts, and per-run policy checkpoints are available in our anonymous repository: https://anonymous.4open.science/r/FAS-EB9C/.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, volume 34, pp. 29304–29320, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002. doi: 10.1023/A:1013689704352.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. MLE-Bench: Evaluating Machine Learning Agents on Machine Learning Engineering. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=6s5uXNWGIh.

Jiefeng Chen, Bhavana Dalvi Mishra, Jaehyun Nam, Rui Meng, Tomas Pfister, and Jinsung Yoon. MARS: Modular Agent with Reflective Search for Automated AI Research. *arXiv preprint arXiv:2602.02660*, 2026. doi: 10.48550/arXiv.2602.02660. URL https://arxiv.org/abs/2602.02660. Accepted at ICML 2026 according to the arXiv record.

Yizhou Chi, Yizhang Lin, Sirui Hong, Duyi Pan, Yaying Fei, Guanghao Mei, Bangbang Liu, Tianqi Pang, Jacky Kwok, Ceyao Zhang, Bang Liu, and Chenglin Wu. SELA: Tree-Search Enhanced LLM Agents for Automated Machine Learning. *arXiv preprint arXiv:2410.17238*, 2024. doi: 10.48550/arXiv.2410.17238. URL https://arxiv.org/abs/2410.17238.

Thomas Simon Foster, Bassel Al Omari, Tingchen Fu, Thomas Mann, Carl Domond, Lucia Cipolina-Kun, Bhavul Gauri, Muna Aghamelu, Alexander D. Goldie, Eryk Helenowski, Jean-Christophe Gagnon-Audet, Alberto Pepe, Saba Nazir, Daniel Izcovich, Noam Levi, Rishi Hazra, Karen Hambardzumyan, Nicolas Baldwin, Xian Li, Martin Josifoski, Paris Giampouras, Masoud Jalili Sabet, Anya Sims, Hela Momand, Tatiana Shavrina, Despoina Magka, Jason Weston, Yulin Wang, Anirudh Goyal, João Henriques, Yoram Bachrach, Emily McMilin, and Jakob Nicolaus Foerster. AI Research Preference Models. *arXiv preprint arXiv:2608.13940*, 2026. doi: 10.48550/arXiv.2608.13940. URL https://arxiv.org/abs/2608.13940.

Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In *Algorithmic Learning Theory*, volume 6925 of *Lecture Notes in Computer Science*, pp. 174–188, 2011. doi: 10.1007/978-3-642-24412-4_16.

Karen Hambardzumyan, Nicolas Baldwin, Edan Toledo, Rishi Hazra, Michael Kuchnik, Bassel Al Omari, Thomas Simon Foster, Anton Protopopov, Jean-Christophe Gagnon-Audet, Ishita Mediratta, Kelvin Niu, Michael Shvartsman, Alisia Lupidi, Alexis Audran-Reiss, Parth Pathak, Tatiana Shavrina, Despoina Magka, Hela Momand, Derek Dunfield, Nicola Cancedda, Pontus Stenetorp, Carole-Jean Wu, Jakob Nicolaus Foerster, Yoram Bachrach, and Martin Josifoski. AIRA_2: Overcoming Bottlenecks in AI Research Agents. *arXiv preprint arXiv:2603.26499*, 2026. doi: 10.48550/arXiv.2603.26499. URL https://arxiv.org/abs/2603.26499.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. AIDE: AI-Driven Exploration in the Space of Code. *arXiv preprint arXiv:2502.13138*, 2025. doi: 10.48550/arXiv.2502.13138. URL https://arxiv.org/abs/2502.13138.

Ke Li, Álvaro Fialho, Sam Kwong, and Qingfu Zhang. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 18(1):114–130, 2014. doi: 10.1109/TEVC.2013.2239648.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, 2010. doi: 10.1145/1772690.1772758. URL https://arxiv.org/abs/1003.0146.

Jaehyun Nam, Jinsung Yoon, Jiefeng Chen, Jinwoo Shin, Sercan O. Arik, and Tomas Pfister. MLE-STAR: Machine Learning Engineering Agent via Search and Targeted Refinement. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a9619dd0f0d54a5cf7734add1dc38cd1-Abstract-Conference.html.

Rushi Qiang, Yuchen Zhuang, Yinghao Li, Dingu Sagar V K, Rongzhi Zhang, ChangHao Li, Ian Wong, Sherry Yang, Percy Liang, Chao Zhang, and Bo Dai. MLE-Dojo: Interactive Environments for Empowering LLM Agents in Machine Learning Engineering. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0139. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/0603c69125ad4b964bc9c4832f7b9f8f-Abstract-Datasets_and_Benchmarks_Track.html.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017. doi: 10.1038/nature24270. URL https://doi.org/10.1038/nature24270.

Edan Toledo, Karen Hambardzumyan, Martin Josifoski, Rishi Hazra, Nicolas Baldwin, Alexis Audran-Reiss, Michael Kuchnik, Despoina Magka, Minqi Jiang, Alisia Lupidi, Andrei Lupu, Roberta Raileanu, Tatiana Shavrina, Kelvin Niu, Jean-Christophe Gagnon-Audet, Michael Shvartsman, Shagun Sodhani, Alexander Miller, Abhishek Charnalia, Derek Dunfield, Carole-Jean Wu, Pontus Lars Erik Saito Stenetorp, Nicola Cancedda, Jakob Foerster, and Yoram Bachrach. AI Research Agents for Machine Learning: Search, Exploration, and Generalization in MLE-bench. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1186. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/328b81881da145412f2bc56c998dfb6a-Abstract-Conference.html.

Yuhang Zhou, Lizhu Zhang, Yifan Wu, Jiayi Liu, Xiangjun Fan, Zhuokai Zhao, and Hong Yan. Synthetic Sandbox for Training Machine Learning Engineering Agents. *arXiv preprint arXiv:2604.04872*, 2026. doi: 10.48550/arXiv.2604.04872. URL https://arxiv.org/abs/2604.04872.

Qiran Zou, Hou Hei Lam, Wenhao Zhao, Tingting Chen, Yiming Tang, Samson Yu, Yingtao Zhu, Srinivas Anumasa, Zufeng Zhang, Tianyi Zhang, Chang Liu, Zhengyao Jiang, Anirudh Goyal, and Dianbo Liu. FML-bench: A Controlled Study of AI Research Agent Strategies from the Perspective of Search Dynamics. *arXiv preprint arXiv:2605.17373*, 2026. doi: 10.48550/arXiv.2605.17373. URL https://arxiv.org/abs/2605.17373.

A CONTROLLER ALGORITHM

Algorithm 1 Factorized Adaptive Search: FRR-UCB + LinUCB (online)

Require: frozen stack, wall-clock budget B, $K = 5$ warm-start Drafts, $\lambda = 1$, $\beta = 0.5$
reset graph, operator history, feature statistics, and seeded policy RNGs
generate and charge warm-start Drafts; update the operator bandit with their validation rewards
$M_o \leftarrow \lambda I_d,\ b_o \leftarrow 0$ for $o \in \{\text{DEBUG}, \text{IMPROVE}\}$
while budget remains **do**
    build feasible operator mask $\mathcal{A}_t$ from the current graph
    $o_t \leftarrow$ FRR-UCB($\mathcal{A}_t$, operator history, budget phase)
    if $o_t = \text{DRAFT}$ **then**
        $v_t \leftarrow \varnothing$; no parent feature vector is constructed
    else
        build $\mathcal{V}_t^{o_t}$ and normalized $x_t(v)$ for each eligible node
        solve $M_{o_t}\widehat{\theta}_{o_t} = b_{o_t}$ and score Eq. 11
        choose v_t; save $x \leftarrow x_t(v_t)$, prediction, bonus, and parent ID
    end if
    generate, execute, and analyze the child using the frozen stack
    charge cost; obtain r_t from Eq. 4, including failures
    update the operator bandit's history with the single (o_t, r_t) observation
    if $o_t \neq \text{DRAFT}$ **then**
        $M_{o_t} \leftarrow M_{o_t} + xx^\top$; $b_{o_t} \leftarrow b_{o_t} + r_t x$
    end if
    append the child if available; update expansion counts and reward histories
    log transition and checkpoint policy states and RNGs
end while
return validation-selected incumbent; join private outcomes after run closure

A terminal failed attempt consumes budget and receives zero incumbent gain, but its saved parent vector is still used for the update. Retries are charged within the same attempted expansion, and a completed expansion yields one shared reward. No feature is recomputed after seeing the current child for use in that child's learning update.

B DECISION-TIME FEATURES AND NORMALIZATION

The following definitions make Equation 9 reproducible. Only completed observations available before the action are used. The intercept remains one; origin operator uses three binary indicators in the fixed order Draft, Debug, Improve, giving $d = 11$.

Table 4: The numeric part of $x_t(v)$. Empty reward histories are zero. $N_{\text{ref}} = 500$ is the run's candidate-evaluation limit; no future trace statistics are used.

Feature	Definition available at decision time t
Validation/ancestor rank	Rank of the node's valid score, or its nearest valid ancestor's score for a buggy node, among all current valid nodes. For $n > 1$, use $(k-1)/(n-1)$ with average tied rank k, worst first; use 1 for the sole valid node and 0 if no valid ancestor exists.
Depth	$\min(\text{graph depth}/N_{\text{ref}}, 1)$; Draft depth is zero.
Age	$\min(\text{completed decision steps since node creation}/N_{\text{ref}}, 1)$.
Children count	$\min(\text{materialized direct children}/N_{\text{ref}}, 1)$, including invalid children.
Expansion count	$\min(\text{all completed attempted expansions from this parent}/N_{\text{ref}}, 1)$, including failures with no materialized child.
Mean child reward	$z/(1+z)$ of the mean shared reward from prior expansions of this parent, including zero-reward failures.
Last child reward	$z/(1+z)$ of the most recent such reward.

The common parent-history convention is used for every eligible node; origin refers to the operator that created the node, not the current action. Counts distinguish generated children from attempted expansions. The saved vector travels with each transition so the linear update can be replayed exactly. For numerical computation, linear systems are solved rather than explicitly forming an inverse; a checkpoint includes both M_o, b_o pairs and feature/reward histories as a within-run resume point, not a prior for another task.

C Online Information Boundary and Learning Checks

Table 5: Information access. The parent model consumes only the normalized feature vector; the frozen AIRA stack retains its ordinary execution inputs.

Field	Operator policy	Parent model	Final analysis
Validation reward and past rewards	yes	yes	yes
Eligible pools and operator history	yes	features	yes
Budget, phase, and audited costs	yes	via reward	yes
Normalized decision-time node features	mask only	yes	yes
Raw generated code and execution text	no	no	yes
Private score / medal / grader report	no	no	yes
Future children or future rewards	no	no	post-run

Required checks are: (i) policy inputs contain no private-report fields; (ii) randomizing private fields leaves features, r_t, decisions, and both model updates identical; (iii) `use_test_score=true` raises before a run; (iv) each run starts with $M_o = \lambda I_d, b_o = 0$; (v) Draft updates no parent model, and a Debug/Improve transition updates only its operator's model; (vi) updates use the saved pre-decision vector; (vii) a zero reward changes the selected M_o by $xx^\top$ but not b_o; (viii) a checkpoint round trip reproduces decisions and states; and (ix) private evaluation joins occur only after the run closes.

D Outcome Summary Across the Full Task Suite

Table 6 summarizes, for the final validation-selected candidate at the four-hour cutoff and aggregated across all tasks (single seed), how often each method wins on held-out test score, wins on the online validation signal, earns any held-out medal, and the tier of medal earned. Outright test wins counts only tasks where a method wins without a tie; test wins including ties additionally credits every method tied for the best held-out score on a task.

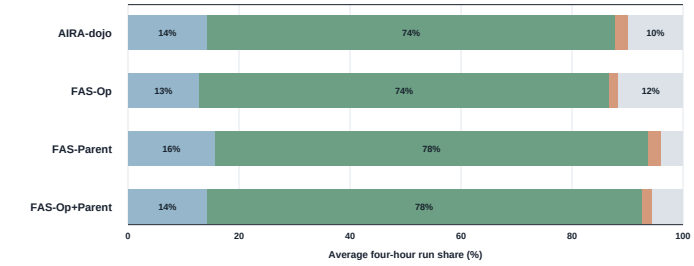
Table 6: **Outcome summary**, aggregated across 11 tasks with a single seed each, final validation-selected candidate at the four-hour cutoff. FAS-Op leads on every column.

Method	Outright test wins	Test wins, incl. ties	Validation wins	Tasks earning a medal	Gold / Silver / Bronze (%)
FAS-Op	**45.5%**	**54.5%**	**45.5%**	54.5%	54.5 / 0 / 0
AIRA-dojo	18.2%	27.3%	27.3%	54.5%	36.4 / 18.2 / 0
FAS-Op+Parent	27.3%	27.3%	27.3%	36.4%	9.1 / 18.2 / 9.1
FAS-Parent	0%	0%	0%	18.2%	0 / 18.2 / 0

FAS-Op has the highest outright test-win rate of any method (45.5%, versus 27.3% for the next-best FAS-Op+Parent and 18.2% for AIRA), and including ties it is tied or outright best on 54.5% of tasks. FAS-Op also has the highest gold-medal rate of any method (54.5%), and every medal it earns is gold, with no silver or bronze. AIRA matches FAS-Op's share of tasks earning a medal (54.5%) but with a lower-value mix (36.4% gold, 18.2% silver, versus FAS-Op's 54.5% gold, 0% silver, 0% bronze). FAS-Op+Parent earns a medal on fewer tasks (36.4%) and only one gold; FAS-Parent never wins a task outright on either signal and earns a medal on 18.2% of tasks.

E WALL-CLOCK TIME BREAKDOWN

Figure 7 breaks down each condition's four-hour wall-clock budget into generation, execution and grading, analysis, and unfinished/other time, averaged over runs. The breakdown is close across all four conditions: execution and grading occupies 73.5% of the budget under AIRA, 74.0% under FAS-OP, 78.1% under FAS-PARENT, and 78.4% under FAS-OP+PARENT, with generation at 12.8%–15.7% in every condition. The four conditions therefore spend their wall-clock budget on similar activities in similar proportions; what differs is which operator and which parent that time is spent on. This argues against a simple timing confound for the results in Sections 6 and 7: FAS-OP's gain is not explained by a markedly different allocation of wall-clock time, but by which decisions it makes with that time.



Method	Generation	Execution + grading	Analysis	Unfinished / other
AIRA-dojo	14.3%	73.5%	2.3%	9.9%
FAS-Op	12.8%	74.0%	1.7%	11.6%
FAS-Parent	15.7%	78.1%	2.4%	3.9%
FAS-Op+Parent	14.2%	78.4%	2.0%	5.5%

Average across 11 primary task runs per method; each run has a four-hour budget.

Figure 7: **Average wall-clock time breakdown**, averaged over runs within each condition's four-hour budget.

F TRACE SCHEMA AND ARTIFACT CHECKLIST

Each generated child has one policy record keyed by run ID, step, and node ID, with fields for: feasible operators; selected operator; compatible and selected parent IDs; policy scores; the saved decision-time $x_t(v_t)$ and normalization constants; the selected operator's pre-update linear mean and bonus; r_t; M_o, b_o update identifiers; budget phase; validation incumbent before/after; generation and analysis tokens/latency; execution/evaluation time; cumulative budget; and failure/retry events. The released artifact will include schema validation, a data dictionary, license information, and scripts that regenerate every table and figure in this paper from the recorded traces.