FROM SEARCH STATE TO CONTROL ACTIONS: A MODULAR INTERFACE FOR BUDGETED BLACK-BOX OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In expensive black-box optimization, the central bottleneck is often how the optimizer spends its scarce oracle budget, rather than how it generates candidate solutions. When each candidate requires a costly rollout, a control rule that mismatches the current search state can consume evaluations without improving the decision. Existing parameter-control methods usually bind such decisions to a particular optimizer or a predefined schedule, leaving the control semantics difficult to separate from their implementation. We introduce Search-State Assisted Policy Orchestration (SAPO), an optimizer-side interface that maps online search states to search-step scale, relative acceptance slack, and conditional perturbation, then executes them through backbone-specific adapters. On a nine-dimensional delivery-mechanism task, the hand-crafted Expert-SAPO-DE instance significantly improves the final objective over DE, jDE, and SHADE under matched oracle, budget, and domain conditions. Its lower median than a deterministic schedule is not statistically significant, and channel ablations do not establish a benefit from joint use of all channels. Backbone-specific SAPO implementations improve on their corresponding baselines in a Standard environment, whereas CEC2020 and BBOB reveal backbone-dependent outcomes. SAPO separates search-control semantics from backbone execution, while the observed benefit remains conditional on the policy, backbone, and task.

1 INTRODUCTION

Expensive simulation-based optimization seeks good parameter configurations from a limited number of black-box evaluations. In delivery-mechanism design, evaluating a candidate may require simulating several days of operation under stochastic demand and dispatching. Such evaluations provide objective feedback without an analytical model or readily available gradients, making the use of a finite evaluation budget a central concern (Amaran et al., 2016; Jin, 2011; Garnett, 2023). Population-based optimizers can search these spaces using objective values alone (Storn & Price, 1997; Kennedy & Eberhart, 1995; Beyer & Schwefel, 2002). Their search behavior depends on how candidate solutions are generated and on control decisions governing movement scale, acceptance, and perturbation. The resulting question is how these controls should respond to search state under a fixed evaluation budget.

Parameter control offers established approaches to regulating search behavior. Fixed settings retain the same control values, deterministic schedules change them with budget progress, and adaptive methods update them using search feedback (Eiben et al., 1999; Karafotias et al., 2015). In differential evolution (DE), jDE adapts parameters associated with individuals, while SHADE uses memories of successful parameter settings (Brest et al., 2006; Tanabe & Fukunaga, 2013). Hyperheuristics also study the selection and configuration of search procedures (Burke et al., 2013; Drake et al., 2020). The control variables differ across backbones, raising a question about which aspects of their regulation can share an explicit representation.

We study a control interface that separates intervention semantics from backbone-specific execution. Movement scale is expressed through mutation scaling in DE, inertia and velocity controls in particle swarm optimization (PSO), and mutation scale in evolution strategies (ES). Acceptance and pertur-

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.

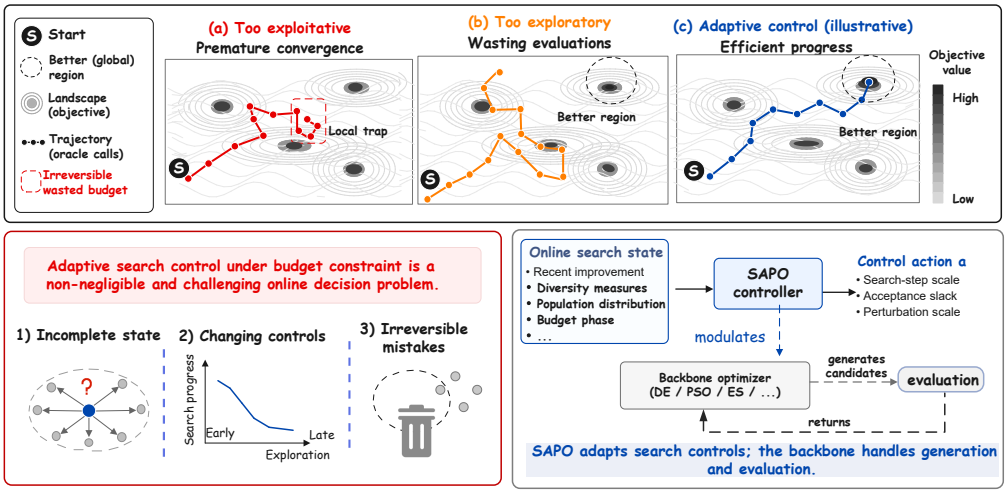


Figure 1: Motivation of search-state-assisted policy orchestration.

bation likewise act on different update mechanisms in these backbones. A shared representation must therefore specify the intended control effect while allowing each backbone to implement that effect differently. Separately configurable interventions also make it possible to compare individual channels with their combination under the same evaluation protocol.

Search-State Assisted Policy Orchestration (SAPO) addresses this question through a state-to-intervention interface and backbone adapters. As illustrated in Fig. 1, the optimizer's recorded search state supplies feedback to an assistant policy. The policy outputs three intervention values, and the adapter translates each value into an executable operation for the selected backbone. Separately configurable channels support ablation, while the adapter makes their implementation explicit for each optimizer. Candidate generation remains with the backbone, and the rollout oracle and objective remain unchanged. Expert-SAPO instantiates this interface with hand-crafted rules. Imitation-SAPO approximates discretized rule outputs through supervised learning on expert-labeled search states.

Delivery-mechanism experiments show both the benefits and the limits of this control interface. Expert-SAPO-DE improves on DE, jDE, and SHADE under matched conditions, and backbone-specific implementations improve on their respective baselines in the Standard environment. Search-step scale gives the lowest single-channel median, but joint channel use has no established advantage over individual channels. Expert-SAPO-DE also has no established advantage over the tested deterministic schedule. Numerical benchmarks further show that the fixed rules behave differently across backbones. The evidence thus links the observed gains to particular policies and settings, rather than to a general advantage of adding control channels.

Together, these experiments test whether the interface is reusable across backbones and whether its observed gains persist beyond the rule-based DE instance.

2 Related Work

DE, PSO, and ES provide established backbones for derivative-free optimization when evaluations are simulation-driven (Storn & Price, 1997; Kennedy & Eberhart, 1995; Beyer & Schwefel, 2002; Bäck & Schwefel, 1993; Amaran et al., 2016). Their performance depends on both candidate-generation rules and the regulation of search controls: an overly large step can waste budget, whereas an overly small step can promote premature convergence.

Parameter control and adaptive evolutionary algorithms have demonstrated the importance of dynamically controlling search behavior (Eiben et al., 1999; Karafotias et al., 2015; Bischl et al., 2023). jDE associates DE parameters with individuals, whereas SHADE maintains parameter memories using historical success information (Brest et al., 2006; Tanabe & Fukunaga, 2013). Deterministic

schedules adjust search-step scale according to budget progress. These methods are usually defined for a specific backbone and its internal parameters, such as F/CR in DE, the inertia weight of PSO, or the mutation scale of ES. SAPO organizes optimizer-side control into shared control categories and maps them to different backbones through adapters.

Hyper-heuristics, adaptive operator selection, and algorithm selection regulate which search procedures or operators are applied over time (Burke et al., 2013; Drake et al., 2020; Kerschke et al., 2019; Hutter et al., 2009). SAPO addresses a narrower control problem: it defines a low-dimensional, interpretable interface for DE, PSO, and ES and tests whether its rule outputs can be imitated from labeled search states.

3 Problem Definition and Evaluation Setup

We formulate budget-constrained rollout optimization over nine-dimensional delivery-mechanism parameters. For a candidate mechanism θ, a multi-day simulator driven by historical orders returns platform cost, lateness rate, and completion rate. The optimizer receives only these objective evaluations, without an analytical model, gradients, or access to the simulator's internal state.

The mechanism parameter vector is

$$\theta = (b_0, b_1, r_{\text{on}}, r_{\text{early}}, \tau_1, \tau_2, p_1, p_2, w_{\text{bias}})^\top \in \mathbb{R}^9. \tag{1}$$

Here, b_0, b_1 control the base and distance-related payout, r_{on} and r_{early} denote online and early-arrival rewards, τ_1, τ_2 and p_1, p_2 denote timeliness thresholds and penalty levels, and w_{bias} denotes the dispatch/payout bias weight. All candidate solutions are mapped into the feasible domain Θ after box clipping and structural repair. If $\tau_2 \le \tau_1$, we set $\tau_2 \leftarrow \tau_1 + 1$; if $p_2 < p_1$, we set $p_2 \leftarrow p_1$. Parameter bounds and simulation-protocol details are provided in the Supplementary Material.

The business objective uses a fixed-weight scalarization:

$$\begin{aligned} F_{\text{biz}}(\theta) = {} & \text{payout_per_order}(\theta) \\ & + w_{\text{late}}\, \text{late_rate_done}(\theta) \\ & - w_{\text{comp}}\, \text{completion_rate}(\theta), \end{aligned} \tag{2}$$

where $w_{\text{late}} = 10$ and $w_{\text{comp}} = 5$ are fixed in all experiments. The evaluation oracle can be written as $\mathcal{E} : \Theta \times \mathbb{Z} \to (F_{\text{biz}}, \mathbf{m}, t_{\text{wall}})$, where the second input is the random seed and $\mathbf{m}$ denotes auxiliary business metrics. All algorithms are compared using the same scalarized objective, defined in Eq. (2) and evaluated by the oracle.

The business objective is to minimize the rollout-oracle value $F_{\text{biz}}(\theta)$ over Θ. Under a fixed budget B, the optimization procedure seeks the lowest observed objective b_B while using the same feasible domain, oracle, and random-seed protocol. SAPO controls this search process while keeping the business objective, simulator, candidate-solution domain, and number of oracle calls fixed.

4 The SAPO Framework

SAPO separates the choice of a search intervention from its execution within a backbone optimizer. Search-step scale, relative acceptance slack, and conditional perturbation define the policy outputs; backbone-specific adapters translate them into update operations. The channels can be varied independently while retaining the same candidate representation and evaluation oracle.

The oracle evaluates each candidate, the state extractor summarizes the resulting population, and the policy selects the next intervention; the adapter applies that intervention during subsequent candidate generation and selection (Fig. 2).

4.1 Framework Abstraction

Search control under a fixed objective

The acceptance slack β denotes a **relaxation of the selection criterion during the search stage**; a larger β means looser selection. The business objective in Eq. (2) continues to be evaluated by the rollout oracle.

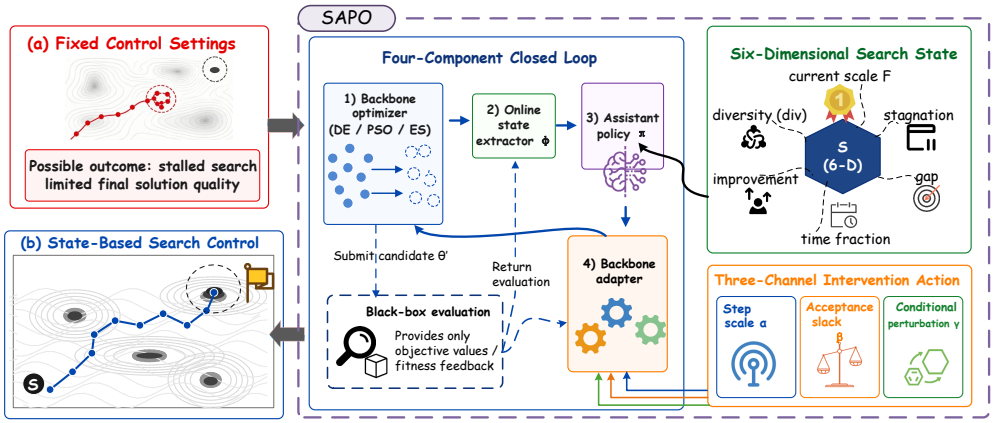


Figure 2: Operational view of the SAPO framework. The backbone optimizer maintains the population and generates candidates, which the oracle evaluates using the fixed rollout objective. The state extractor builds the six-dimensional search state, and the assistant policy maps it to a three-channel intervention. The backbone adapter implements the intervention in DE, PSO, or ES. Arrows indicate evaluation feedback, state information, and control signals between the components.

The adapter determines how each control output modifies the backbone update. Expert-SAPO supplies these outputs through hand-crafted rules, while Imitation-SAPO predicts discretized versions of the same outputs. This separation permits policy substitution within a specified adapter and channel ablation without changing the rollout objective.

State, policy, and adapter

The backbone optimizer $b \in \{\mathrm{DE}, \mathrm{PSO}, \mathrm{ES}\}$ maintains the population and generates candidate solutions. The online search state $s_t = \phi(P_t, f_t, t/T)$ is constructed from the population P_t, historical objective feedback f_t, and budget progress. The assistant policy π outputs $a_t = \pi(s_t)$ according to the state. The backbone adapter ψ_b maps the unified intervention a_t to the backbone's control variables $u_t = \psi_b(a_t)$. Expert-SAPO and Imitation-SAPO use this (s_t, a_t) interface; outcome-driven policy learning remains a future extension.

Online search state

SAPO uses a six-dimensional state vector with fixed ordering:

$$s_t = \big[\mathrm{div},\ \mathrm{stagnation},\ \mathrm{impr},\ \mathrm{gap},\ t_{\mathrm{frac}},\ F\big]. \tag{3}$$

For DE, the state contains diversity, stagnation, recent improvement, the population-to-best gap, budget progress, and the current scale. Here, div is the mean per-dimension standard deviation of the population in the original coordinates; $\mathrm{stagnation}$ is the number of evaluations since the best observed objective last improved; impr is the relative magnitude of the most recent improvement; $\mathrm{gap} = (\mathrm{median}(f(P_t)) - f_{\mathrm{best}})/(|\mathrm{median}(f(P_t))| + \epsilon)$ characterizes the relative gap between the median population objective and the best observed objective; $t_{\mathrm{frac}} = t/T$ is budget progress; and F is the current search-step scale (the scaling factor in DE and a backbone-specific scale feature in cross-backbone experiments). The assistant policy uses statistics recorded by the optimizer, without access to simulator internals or the true optimum.

Three-channel intervention space

The assistant policy outputs a three-channel intervention

$$a_t = \big(a_t^{\mathrm{exp}},\ a_t^{\mathrm{acc}},\ a_t^{\mathrm{rec}}\big), \tag{4}$$

corresponding to search-step scale, relative acceptance slack, and conditional perturbation. We call the three channels exploration, acceptance, and recovery; Table 1 gives their meanings and adapter mappings.

Table 1: SAPO Three-Channel Intervention Space and Backbone Mappings

Channel	Semantics	DE mapping	PSO mapping	ES mapping
Step scale	Regulates movement scale	Effective F	Inertia/velocity scale	Mutation scale σ
Acceptance slack	Allows limited short-term deterioration	Relaxed selection	Relaxed particle update	Relaxed center update
Perturbation	Injects perturbation under low diversity/stagnation	Mutant perturbation	Non-elite particle perturbation	Offspring perturbation

The search-step scale a_t^{exp} controls the scale of candidate movements (the effective F in DE, the inertia/velocity scale in PSO, and σ in ES). The relative acceptance slack $\beta = a_t^{\mathrm{acc}}$ relaxes greedy selection during the search stage. Conditional perturbation a_t^{rec} injects small-scale Gaussian perturbations into mutants, particles, or offspring when extremely low diversity coincides with stagnation. These perturbations aim to increase local variation without a global random restart.

In DE, let the parent objective be f_i and the trial objective be f'. The selection criterion is

$$\mathrm{Accept}(\theta') \;\Leftrightarrow\; f' \le f_i \;\vee\; \Big(\beta > 0 \;\wedge\; f' - f_i \le \beta \cdot \max(1, |f_i|)\Big). \tag{5}$$

When $\beta = 0$, the criterion reduces to standard greedy selection in DE.

4.2 Expert-SAPO-DE Instance and Algorithmic Procedure

Expert-SAPO is a hand-crafted policy π_{exp} that maps s_t to $(a_t^{\mathrm{exp}}, a_t^{\mathrm{acc}}, a_t^{\mathrm{rec}})$. Its rules provide a concrete state-feedback controller whose channels can be examined separately. Coupling this policy to the DE adapter yields Expert-SAPO-DE.

The main hyperparameters of Expert-SAPO are as follows. The stagnation condition is $\mathrm{stagnation} \ge W$, with $W = 6$ objective evaluations. Let $\lambda = \mathrm{clip}((D^\star - \mathrm{div})/D^\star, 0, 1)$ denote the diversity deficit ($D^\star = 0.10$). The search-step scale is updated by the following rules. If $\mathrm{stagnation} \ge W$ or $\mathrm{div} < 0.05$, then $F \leftarrow \min(0.9,\, 1.15F + 0.05)$. Otherwise, if $\mathrm{impr} > 0.01$ and $\mathrm{div} > D^\star$, then $F \leftarrow \max(0.2,\, 0.90F)$. Otherwise, $F \leftarrow \mathrm{clip}(F + 0.02(D^\star - \mathrm{div}),\, 0.2,\, 0.9)$. The relative acceptance slack is updated as $\beta \leftarrow \min(0.08,\, 0.02 \cdot \mathbf{1}[\mathrm{stagnation} \ge W] + 0.02\lambda)$. When $\mathrm{div} < 0.02$, $\mathrm{stagnation} \ge W$, and $\lambda > 0$, $a_t^{\mathrm{rec}} \leftarrow 0.05\lambda$; otherwise, $a_t^{\mathrm{rec}} \leftarrow 0$.

The complete threshold settings and hyperparameter values are given in the Supplementary Material; the rules above define the controller used in all reported experiments.

Expert-SAPO-DE embeds the SAPO orchestrator in standard DE/rand/1/bin. In each round, s_t is computed and π_{exp} is evaluated *before* trial generation. The resulting F, β, and a_t^{rec} control differential mutation, the selection rule in Eq. (5), and mutant perturbation. The base differential term, binomial crossover, and feasible-domain repair remain unchanged. Algorithm 1 specifies the resulting update sequence.

In Algorithm 1, the budget counts all evaluation requests, including initialization and cache hits. Record(x, f) increments n by one and updates $(\theta_{\mathrm{best}}, f_{\mathrm{best}})$, τ, and r when a new best is observed. The archive and cache bookkeeping are independent of population replacement and are documented in the Supplementary Material. The Gaussian components are independent, with coordinate-wise standard deviations $\gamma \odot (u - \ell)$.

4.3 Supervised Rule Imitation and Cross-Backbone Extension

Imitation-SAPO (supervised rule-imitation evaluation). We train Imitation-SAPO to assess how accurately a supervised model can reproduce discretized rule outputs at the (s_t, a_t) interface. The model is a random forest trained on synthetic search states labeled with discretized outputs from Expert-SAPO. The input is the six-dimensional state in Eq. (3); the output is one of $3 \times 3 \times 3 = 27$ action-grid combinations (search-step scale $\{0.3, 0.55, 0.8\}$, relative acceptance slack $\{0, 0.04, 0.08\}$, and conditional perturbation $\{0, 0.03, 0.06\}$). The dataset contains 8000 samples divided into training, validation, and test subsets in a 60%/20%/20% split. The random forest contains 200 trees with a maximum depth of 12. The test-set accuracy is 0.995625 and the macro-

Algorithm 1 Expert-SAPO-DE

Require: oracle $\mathcal{E}$, bounds $[\ell, u]$, repair, budget B, population size N with $B \geq N \geq 4$, F_0, CR

Initialize feasible population $P = \{x_i\}_{i=1}^{N}$ and $F \leftarrow F_0$
$n \leftarrow 0$, $\tau \leftarrow 0$, $r \leftarrow 0$, $f_{\text{best}} \leftarrow \infty$, $\theta_{\text{best}} \leftarrow \emptyset$
For $i = 1, \ldots, N$: $f_i \leftarrow \mathcal{E}(x_i)$; RECORD$(x_i, f_i)$
while $n < B$ **do**
  Compute s_t by Eq. (3), using P, $\{f_i\}$, f_{best}, stagnation $= n - \tau$, impr $= r$, $t_{\text{frac}} = n/B$, and F
  $(F, \beta, \gamma) \leftarrow \pi_{\text{exp}}(s_t)$, with $\gamma = a_t^{\text{rec}}$
  for $i = 1, \ldots, N$ **do**
    If $n = B$, **return** $(\theta_{\text{best}}, f_{\text{best}})$
    Sample distinct indices $a, b, c \in \{1, \ldots, N\} \setminus \{i\}$
    $v \leftarrow x_a + F(x_b - x_c) + \mathcal{N}(0, \gamma \odot (u - \ell))$
    $\theta' \leftarrow \text{repair}(\text{clip}(\text{BinCross}(x_i, v; CR)))$
    $f' \leftarrow \mathcal{E}(\theta')$; RECORD$(\theta', f')$
    if Eq. (5) holds for (f', f_i, β) **then**
      $x_i \leftarrow \theta'$, $f_i \leftarrow f'$
    end if
  end for
end while
return $(\theta_{\text{best}}, f_{\text{best}})$

F1 is 0.9969. The training target is the discretized expert action, so prediction fidelity and online optimization performance are evaluated separately.

Cross-backbone adapters. The three categories share a common intervention intent across DE, PSO, and ES, whereas the rules and adapter mappings remain backbone-specific (Table 1). In the DE implementation, F in Algorithm 1 sets the scale of rand/1 mutation, β controls relaxed selection, and γ sets the mutant-perturbation scale. In PSO, the search-step scale channel controls the inertia-weight and velocity-limit multipliers; the acceptance channel relaxes particle updates, and the perturbation channel modifies non-elite particle positions. In ES, the search-step scale channel controls σ, the acceptance channel relaxes center updates, and the perturbation channel modifies offspring.

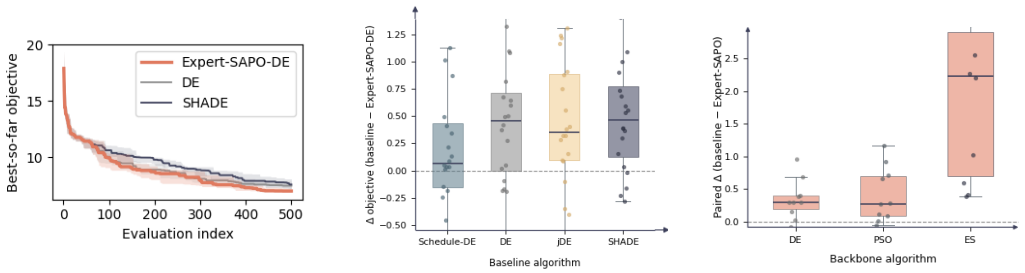
5 EXPERIMENTAL DESIGN

We evaluate overall performance and the contribution of individual design choices under matched protocols. Comparisons with DE, jDE, and SHADE quantify performance on the Canonical delivery task; Schedule-DE tests state feedback against a predefined schedule; and single-channel ablations test whether combining interventions is beneficial. Cross-backbone and scenario comparisons probe changes in optimizer dynamics and delivery conditions. Supervised imitation separates rule-prediction fidelity from online performance, while numerical benchmarks delimit the scope of the fixed rules beyond delivery optimization.

The numerical suites use different dimensions and budgets, so comparisons remain within each protocol; detailed run counts and coverage are provided in the Supplementary Material. Supervised imitation uses 8,000 synthetic labeled states without simulator calls.

Within the same experiment, all algorithms use the same F_{biz}, feasible domain, repair rules, evaluation budget, population size, and random-seed protocol. The acceptance slack of the assistant policy affects only the acceptance criterion for trial solutions; the oracle objective definition remains fixed. Candidate solutions are always generated by the backbone optimizer. The main performance experiment compares DE, jDE, SHADE, Schedule-DE, and Expert-SAPO-DE. Schedule-DE uses $F = 0.7$ for the first 30% of the budget and $F = 0.4$ thereafter, to compare scheduled search-step scale control with state-feedback control.

Paired experiments use the paired Wilcoxon signed-rank test (`alternative=less`). Holm correction is used for multiple comparisons. We report the median, interquartile range (IQR), win/loss counts, raw p, and, where needed, p_{holm}. Significance is claimed only when the corrected $p < 0.05$.



(a) Main-task convergence (b) Main-task paired gains (c) Cross-backbone paired gains

Figure 3: Rollout optimization results. (a) Convergence on the Canonical main task over 20 seeds × 500 evaluations. Curves show the median best observed objective, and shaded regions show the IQR; smaller values indicate better performance. (b) Seed-level paired improvements of Expert-SAPO-DE relative to each main-task baseline. (c) Paired improvements of the SAPO implementations relative to their respective backbones in the Standard environment over 10 seeds. In (b) and (c), values are the baseline final objective minus the corresponding SAPO final objective; positive values favor SAPO.

6 Results and Analysis

The best observed objective is the primary metric, with smaller values indicating better performance. The alternative hypothesis is that Expert-SAPO-DE, or a cross-backbone SAPO implementation, yields a lower objective than its corresponding baseline.

6.1 Gains over DE Baselines Do Not Establish an Advantage over Scheduling

Expert-SAPO-DE attained the lowest median final objective, 6.982, across 20 seeds on the Canonical nine-dimensional delivery task. The convergence curves showed lower objectives than DE and SHADE early in search, with the difference persisting to the budget limit (Fig. 3(a)).

Expert-SAPO-DE had lower final objectives than DE, jDE, and SHADE; paired Wilcoxon tests gave $p = 0.0032$, 0.0036, and 0.0002, respectively (Table 2(a)). Schedule-DE also outperformed fixed-parameter DE ($p = 0.022$, 14/6 wins/losses, median improvement 0.205). Expert-SAPO-DE had a lower median than Schedule-DE (6.982 versus 7.175), but this difference was not statistically significant ($p = 0.108$).

The controllers followed different effective-F trajectories, although their final-objective comparison was not significant. Plain DE keeps the effective search-step scale at $F = 0.5$ throughout the 500-evaluation budget, and Schedule-DE follows a profile defined by budget segments. Expert-SAPO-DE instead starts with an effective scale near 0.57 and reaches a median effective F of approximately 0.89–0.90 in the middle and final budget segments.

6.2 Scale Control Has the Lowest Single-Channel Median

The ablation compares each intervention alone with the full rule-based controller. Exploration-only had the lowest median objective (4.469), followed by full Expert-SAPO-DE (4.579), acceptance-only (4.650), and recovery-only (4.749) in the Standard environment (Table 2(b)). None of the paired comparisons with the full version met the significance criterion $p_{\text{holm}} < 0.05$ after Holm correction. Exploration-only was significantly better than recovery-only ($p_{\text{holm}} = 0.0073$); the complete pairwise table is provided in the Supplementary Material.

Because the full controller did not differ significantly from the single-channel variants after correction, the ablation does not establish a joint-channel gain or isolate a causal contribution from the effective-F trajectory.

Table 2: Final objectives for the Canonical main task and Standard-environment channel ablation. Brackets give [Q25, Q75]. In (a), W/L counts Expert wins/losses against the row baseline; p is the raw paired Wilcoxon value. The two panels use separate protocols, so their objective values are not compared across panels.

Algorithm or variant	Median	[Q25, Q75]	W/L	p
(a) Main task: Canonical, 20 seeds, budget 500				
Expert-SAPO-DE	6.982	[6.862, 7.242]	—	—
Schedule-DE	7.175	[7.035, 7.428]	14/6	0.108
DE	7.458	[7.227, 7.850]	15/5	0.0032
jDE	7.541	[7.248, 7.820]	16/4	0.0036
SHADE	7.566	[7.221, 8.064]	16/4	0.0002
(b) Channel ablation: Standard, 20 seeds, budget 500				
Expert-SAPO-DE [exploration]	4.469	[4.250, 4.689]		
Expert-SAPO-DE [full]	4.579	[4.378, 4.901]		
Expert-SAPO-DE [acceptance]	4.650	[4.562, 4.882]		
Expert-SAPO-DE [recovery]	4.749	[4.578, 4.868]		
Full vs. each single channel: no Holm-significant difference.				

Table 3: Cross-backbone and scenario-shift results. Values are median final objectives; p denotes raw paired Wilcoxon values. In (a), W/L counts SAPO wins/losses over the corresponding backbone. In (b), Expert denotes Expert-SAPO-DE and p compares it with DE.

(a) Cross-backbone
Standard, 10 seeds, budget 300

Backbone	Baseline	SAPO	W/L	p
DE	5.224	4.894	9/1	0.0029
PSO	4.857	4.275	9/1	0.0029
ES	8.406	6.101	10/0	0.0010

(b) Scenario shifts
Canonical, 20 seeds, budget 500

Scenario	Expert	DE	SHADE	p
InDomain	6.898	7.123	7.566	0.0086
LoadHigh	7.107	7.333	7.486	0.057
TimeShift	6.853	7.241	7.512	0.020

6.3 Backbone-Specific Adapters Improve Standard-Environment Performance

The Standard-environment comparisons examined whether the gains extended beyond DE. For DE, PSO, and ES, the SAPO variants had lower medians than their corresponding baselines (Table 3(a)); paired Wilcoxon tests gave $p = 0.0029$, 0.0029, and 0.0010, respectively.

Paired improvements for all three backbones were concentrated mainly in the positive region (Fig. 3(c)). These gains used backbone-specific rules and adapters rather than an unchanged policy transferred between optimizers. Because Standard and Canonical use different objective scales, the cross-backbone results are interpreted only within the same environment. Service-level diagnostics do not indicate that the gains were primarily driven by sacrificing lateness performance.

6.4 Scenario Shifts Preserve Lower Medians but Weaken Evidence under High Load

Expert-SAPO-DE had lower medians than DE in all three scenarios; the DE comparisons were significant for InDomain and TimeShift ($p = 0.0086$ and $p = 0.020$) but not for LoadHigh ($p = 0.057$). The full table also reports the SHADE comparisons.

TimeShift uses a shifted calendar window, with a 15-day rollout starting on 2021-08-16. The results show that the direction of SAPO's gain is consistent across the tested date windows and load configurations, although improvement magnitudes vary across scenarios and the statistical evidence is weaker under high load.

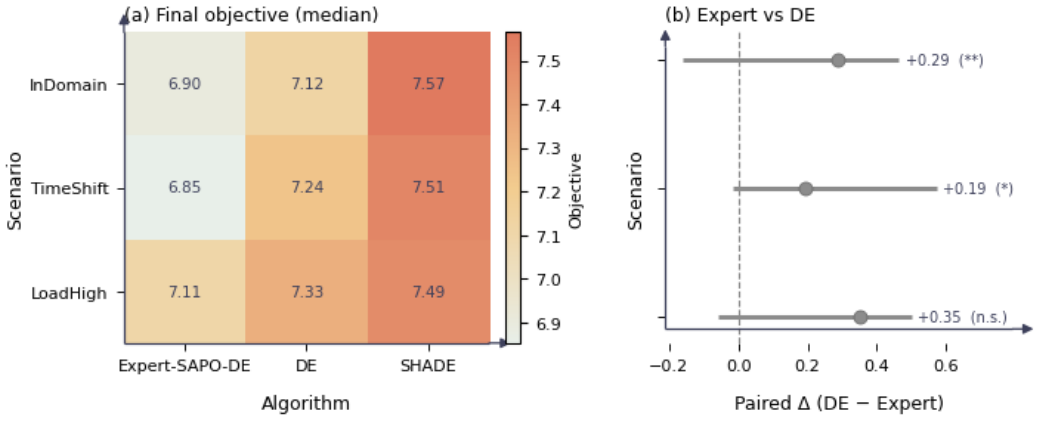


Figure 4: Scenario-shift results. The left panel shows the median final objective for three scenarios and three algorithms. The right panel shows the paired improvement of Expert-SAPO-DE over DE and the Wilcoxon test results. Positive values indicate that Expert-SAPO-DE has a lower final objective.

6.5 Imitation Fidelity and Benchmark Outcomes Depend on the Evaluation Target

On a held-out set of $n = 1600$, the random forest model achieves 0.996 test accuracy and 0.997 macro-F1. Feature-importance diagnostics for the discretized rule labels are reported in the Supplementary Material.

Imitation-SAPO-DE, Expert-SAPO-DE, and DE had medians of 7.731, 7.684, and 7.974, respectively; all pairwise comparisons were non-significant despite the high rule-prediction accuracy. The model therefore reproduced discretized expert decisions without establishing an online optimization gain; the complete online-imitation results are provided in the Supplementary Material.

The numerical benchmarks changed the relative performance of the backbone-specific implementations. Expert-SAPO-DE does not outperform DE on any function in CEC2020 and records 4/1/19 wins/ties/losses on BBOB. By contrast, Expert-SAPO-PSO has more favorable results on both suites, with 10/0/0 on CEC2020 and 22/2/0 on BBOB. Expert-SAPO-ES shows mixed function-level outcomes. The contrast between the PSO and DE counts indicates that the rollout results do not extend uniformly across numerical landscapes. Full function-level results are provided in the Supplementary Material.

7 Discussion and Conclusion

SAPO provides a state-to-intervention interface for search-step scale, relative acceptance slack, and conditional perturbation, with backbone-specific adapters for DE, PSO, and ES. On the Canonical delivery task, Expert-SAPO-DE significantly improves on DE, jDE, and SHADE; in the Standard environment, the SAPO implementations improve on their respective backbone baselines. Ablations identify search-step scale as the strongest single-channel variant by median objective, while supervised imitation demonstrates approximation of discretized rule outputs without establishing an online performance advantage.

The interface makes control decisions separately testable, with application evidence limited to delivery optimization under a fixed scalarized objective. Gains beyond the tested schedule or from joint channel use remain unestablished; outcome-driven learning and broader objective formulations are therefore priorities for further evaluation.

AI USE STATEMENT

The authors used generative AI tools solely for language editing, grammar checking, and formatting assistance. All scientific content, including the problem formulation, method design, experimental protocols, results, statistical analyses, and conclusions, was produced, verified, and approved by the authors. No generative AI system was used to generate experimental data, fabricate results, or replace authorship responsibilities.

DATA AND CODE AVAILABILITY

The historical delivery-order records and the production simulator used in this study are subject to platform access restrictions and cannot be released in raw form. To support reproducibility of the reported comparisons, we will release an anonymized reproducibility package that includes: (i) source code for the SAPO interface, Expert/Imitation policies, and DE/PSO/ES backbone adapters; (ii) experiment runners and configuration files that fully specify decision-variable bounds, repair rules, evaluation budgets, population sizes, random-seed protocols, and environment settings used in each phase; (iii) aggregated optimization logs sufficient to regenerate the reported tables, statistical tests, and figures; and (iv) scripts that rebuild all main-text and appendix results from these artifacts. Where platform data cannot be shared, the package documents the exact evaluation protocol and provides desensitized or synthetic substitutes that preserve the same input–output interface as the rollout oracle used in the paper. Access details and a persistent repository link will be provided in the camera-ready version (or upon acceptance), subject to completed desensitization review.

REFERENCES

S. Amaran, N. V. Sahinidis, B. Sharda, and S. J. Bury. Simulation optimization: A review of algorithms and applications. *Annals of Operations Research*, 240(1):351–380, 2016.

T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

H.-G. Beyer and H.-P. Schwefel. Evolution strategies–a comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, J. Thomas, T. Ullmann, M. Becker, A.-L. Boulesteix, D. Deng, and M. Lindauer. Hyperparameter optimization: Foundations, algorithms, best practices and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(2):e1484, 2023.

J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657, 2006.

E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013.

J. H. Drake, A. Kheiri, E. Özcan, and E. K. Burke. Recent advances in selection hyper-heuristics. *European Journal of Operational Research*, 285(2):405–428, 2020.

A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

R. Garnett. *Bayesian Optimization*. Cambridge University Press, Cambridge, U.K., 2023.

F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, 2009.

Y. Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011.

G. Karafotias, M. Hoogendoorn, and A. E. Eiben. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation*, 19(2):167–187, 2015.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proc. IEEE International Conference on Neural Networks*, pp. 1942–1948, 1995.

P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary Computation*, 27(1):3–45, 2019.

R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

R. Tanabe and A. Fukunaga. Success-history based parameter adaptation for differential evolution. In *Proc. IEEE Congress on Evolutionary Computation*, pp. 71–78, 2013.