Adaptive Operator Selection for NSGA-II in a Vectorized Framework

First Author[1] [0000-0000-0000-0000] and Second Author[1]

Institution, Country
author@institute.edu

Abstract. Operator choice can strongly affect the performance of multi-objective evolutionary algorithms, yet the best operator may vary across problems and across stages of search. We present an adaptive operator selection (AOS) layer for NSGA-II that treats each (crossover, mutation) pipeline as a bandit arm and selects one arm per generation, updating an online policy using survival and diversity-based rewards (optionally including a hypervolume-improvement proxy). On 21benchmark problems from ZDT/DTLZ/WFG with 30seeds at a fixed evaluation budget, AOS improves the median normalized hypervolume from 0.934 to 0.942 with a median runtime overhead of 41%.

Keywords: Adaptive operator selection · Multi-armed bandits · NSGA-II · Multiobjective optimization

1 Introduction

Multiobjective evolutionary algorithms (MOEAs) are a default choice for black-box optimization problems with conflicting objectives, including engineering design and automated decision support. Most MOEAs rely on a small number of variation operators (crossover and mutation) to balance exploration and exploitation. However, operator choice is a major source of performance variability: operators that work well on one problem can stall on another, and the most effective operator can change as the population approaches the Pareto front. This motivates *adaptive operator selection* (AOS): selecting operators online based on feedback from the ongoing search, rather than relying on a single hand-tuned pipeline.

Most AOS mechanisms can be seen as two coupled components: (i) credit assignment (how to score an operator based on recent offspring) and (ii) an adaptation rule (how to turn scores into future operator-selection decisions). Following the bandit view of AOS [3], we model each candidate variation pipeline as an arm in a multi-armed bandit, and update an online policy from per-generation rewards. Compared to offline tuning, this approach aims to improve robustness across problems and reduce the need for manual operator configuration.

This paper focuses on NSGA-II [4], a widely used MOEA whose performance depends strongly on variation settings. We implement a generation-level AOS layer for NSGA-II in a vectorized optimization framework and evaluate it on standard benchmark suites. Our evaluation highlights both the potential benefits of online adaptation (substantial gains on some problems) and the practical costs (runtime overhead from bookkeeping and additional computations).

Contributions. This paper makes four contributions:

– A simple, reproducible AOS interface for NSGA-II that selects exactly one variation pipeline per generation.
– A reward design aligned with NSGA-II survivor selection, combining offspring survival and non-dominated insertion rates, with an optional bounded hypervolume-improvement proxy.
– An analysis workflow that logs per-generation operator choices and rewards, enabling inspection of *when* and *how* the policy switches operators during search.
– An empirical evaluation on classic MOEA benchmark suites [8,5,6] (ZDT/DTLZ/WFG) showing that AOS can improve median normalized hypervolume on difficult instances, while incurring measurable runtime overhead.

Organization. Section 2 reviews NSGA-II and the bandit perspective on AOS. Section 3 details the portfolio, reward signals, and policies. Section 4 summarizes implementation and reproducibility details. Sections 5–6 describe the experimental protocol and results, followed by discussion and threats to validity.

2 Background and related work

2.1 NSGA-II

NSGA-II [4] maintains a population and iteratively generates offspring via variation operators. Survivor selection is based on non-dominated sorting and crowding distance, promoting both convergence and diversity. Although the selection mechanism is parameter-light, practical performance depends on the chosen crossover/mutation pipeline and its hyperparameters, which are often selected by convention or tuned per problem.

2.2 Adaptive operator selection as bandits

In the multi-armed bandit (MAB) setting, a learner repeatedly selects an arm and observes a reward, aiming to maximize cumulative reward by balancing exploration and exploitation. Common policies include ε-greedy, upper-confidence bounds (UCB) [1], adversarial methods such as EXP3 [2], and Bayesian approaches such as Thompson sampling [7]. Bandit-based AOS [3] maps each operator (or operator pipeline) to an arm and uses online rewards to guide selection.

2.3 Credit assignment and non-stationarity

In evolutionary search, operator utility is inherently non-stationary: exploration-heavy operators can be useful early, while exploitation and local refinement become important as the population converges. This raises two practical design questions. First, how should rewards be defined so they are informative across problems and across stages? Second, which policies handle non-stationarity without over-reacting to noise? Sliding-window variants of classical bandit policies (e.g., UCB) and Bayesian policies with limited-memory reward histories are common approaches.

3 Method

3.1 Bandit formulation

Let $\mathcal{A} = \{1, \ldots, K\}$ be a portfolio of K candidate variation pipelines (arms). At each generation t, the algorithm selects one arm $a_t \in \mathcal{A}$, generates all offspring for that generation using that arm, observes a scalar reward $r_t \in [0, 1]$, and updates its policy. The objective is to maximize cumulative reward, which serves as a proxy for improved search progress.

3.2 Operator portfolios

We define a small *portfolio* of candidate variation pipelines, where each arm specifies one crossover operator and one mutation operator with fixed parameters. At generation t, NSGA-II selects exactly one arm and uses it to generate all offspring for that generation. This generation-level decision reduces reward noise (all offspring share the same operator) and keeps the online learning interface simple. It also mirrors how practitioners often tune operators: choosing a pipeline and running it for a while, rather than switching per mating event.

3.3 Reward signals

After survivor selection, we compute a reward that summarizes how useful the selected operator was for that generation. Let n_{off} be the offspring count, n_{surv} the number of offspring that survive into the next population, and n_{nd} the number of surviving offspring that are non-dominated in the next population. We define:

$$r_{\text{surv}} = \frac{n_{\text{surv}}}{n_{\text{off}}}, \qquad r_{\text{nd}} = \frac{n_{\text{nd}}}{n_{\text{off}}}. \tag{1}$$

Both rates are naturally bounded in $[0, 1]$ and are available for any problem without requiring additional reference information.

Algorithm 1 NSGA-II with generation-level AOS (sketch).

```
1: for $t = 1, 2, \ldots$ do
2:     Select arm $a_t$ using bandit policy (with optional warmup/floor).
3:     Generate offspring using the variation pipeline of $a_t$.
4:     Apply NSGA-II survivor selection to form next population.
5:     Compute reward $r_t$ from survival/diversity signals (and optional HV proxy).
6:     Update bandit policy with $(a_t, r_t)$.
7: end for
```

Optional hypervolume proxy. When reference information is available, we include a bounded proxy for hypervolume improvement, $r_{\mathrm{hv}} \in [0, 1]$. Let $\mathrm{HV}(F)$ be the normalized hypervolume of the current population front F, computed with a fixed reference point. We compute a relative improvement ratio

$$\rho_t = \frac{\mathrm{HV}(F_t) - \mathrm{HV}(F_{t-1})}{\max(|\mathrm{HV}(F_{t-1})|, \epsilon)}, \tag{2}$$

and squash it to $[0, 1]$ via $r_{\mathrm{hv}} = 0.5 + 0.5 \tanh(\rho_t)$. This keeps the reward scale consistent across problems and avoids unbounded updates on early generations where absolute HV values can be small.

The aggregate reward can be configured as a single component or as a convex combination:

$$r = w_{\mathrm{surv}} r_{\mathrm{surv}} + w_{\mathrm{nd}} r_{\mathrm{nd}} + w_{\mathrm{hv}} r_{\mathrm{hv}}. \tag{3}$$

In our implementation the weights are normalized to sum to one; if all are zero, we default to a balanced survival/ND reward.

3.4 Policies and practical details

We support several bandit policies (e.g., ε-greedy, UCB, EXP3, Thompson sampling) and two stabilization mechanisms: (i) a *warmup* minimum-usage rule that forces each arm to be tried at least a given number of times, and (ii) an *exploration floor* that mixes in a uniform arm draw with fixed probability. All random choices can be driven by an explicit policy RNG seed to support reproducibility.

Credit timing. Rewards are computed once per generation, after survivor selection has produced the next population. The policy is updated once per generation using the scalar reward, which matches the generation-level selection granularity.

4 Implementation and reproducibility

4.1 Integration into NSGA-II

The AOS layer is implemented as a controller that coordinates (i) a portfolio of named arms and (ii) a bandit policy. At the beginning of each generation, the controller selects an arm index and the algorithm uses the corresponding variation pipeline for all offspring of that generation. During the generation, the controller records the number of offspring created by the selected arm. After survivor selection, it computes reward components (r_{surv}, r_{nd}, and optionally r_{hv}), updates the policy, and emits a trace row for logging.

4.2 Determinism

To support reproducibility, we separate (a) the global run seed that drives variation and selection randomness from (b) an optional policy RNG seed that drives the stochastic policy components (e.g., ε-greedy exploration). With fixed seeds, the operator-selection sequence and rewards are deterministic for a given configuration.

4.3 Logging

We log two types of artifacts for analysis: (i) a per-generation trace (selected operator, reward breakdown, and batch size), and (ii) a summary table (number of pulls and mean reward per arm). These logs support post-hoc analysis of operator switching and help diagnose when a policy locks into a single arm early.

5 Experimental setup

5.1 Benchmarks and budget

We use standard multiobjective benchmark families: ZDT [8], DTLZ [5], and WFG [6]. The suite contains 21 problems (ZDT1–4,6; DTLZ1–7; WFG1–9). Each run is terminated after a fixed evaluation budget of 50,000 function evaluations, using 30independent seeds per problem.

5.2 Baseline and AOS configuration

The baseline is NSGA-II with a fixed SBX + polynomial mutation pipeline (commonly used default settings): SBX crossover probability 1.0 with distribution index $\eta_c = 20$, and polynomial mutation probability $1/n_{\text{var}}$ with $\eta_m = 20$. The AOS variant enables a portfolio of five crossover+mutation arms (Appendix, Table 4) and selects one arm per generation using an ε-greedy policy with $\varepsilon = 0.05$. Rewards use a weighted combination of survival rate, non-dominated insertion rate, and a bounded hypervolume proxy with weights $(0.4, 0.4, 0.2)$.

5.3 Metrics

We report normalized hypervolume (higher is better) and wall-clock runtime (lower is better). Hypervolume is a standard indicator for multiobjective performance [9]. To make results comparable across problems, we normalize hypervolume using a fixed reference front per benchmark problem and a reference point set to the front-wise maximum plus a small ϵ margin.

Table 1. Key experimental settings (baseline vs. AOS).

Setting	Value
Algorithm	NSGA-II
Population / offspring	100 / 100
Budget	50,000 evaluations
Seeds	30per problem
Problems	21(ZDT/DTLZ/WFG)
Baseline operators	SBX($p = 1.0$, $\eta_c = 20$) + PM($p = 1/n_{\text{var}}$, $\eta_m = 20$)
AOS policy	ε-greedy ($\varepsilon = 0.05$), 1 arm per generation
AOS reward	$0.4\, r_{\text{surv}} + 0.4\, r_{\text{nd}} + 0.2\, r_{\text{hv}}$
Portfolio	5 arms (Table 4)

Table 2. Median normalized hypervolume by problem family (baseline NSGA-II vs. baseline + AOS). Values computed from `experiments/ablation_aos_racing_tuner.csv`.

Variant	**ZDT**	**DTLZ**	**WFG**	**Average**
Baseline	0.990	0.799	**0.934**	0.908
Baseline + AOS	**0.990**	**0.880**	0.923	**0.931**

6 Results

We first summarize solution quality and runtime at the family level (ZDT/DTLZ/WFG), then inspect per-problem deltas, convergence behavior, and operator usage. Unless stated otherwise, we report medians over seeds.

6.1 Solution quality

Table 2 shows that AOS provides the largest benefits on the DTLZ family, while changes on ZDT are small and WFG shows a slight degradation on average. Across all problems, AOS improves per-problem median hypervolume on 9 out of 21 problems, with the largest gain on `dtlz6` (Δ=0.545) and the largest loss on `wfg3` (Δ=-0.028).

6.2 Runtime

Table 3 reports median runtime by family. The median runtime overhead (overall) is 41%, which reflects extra policy updates, reward computation, and logging-related bookkeeping. In settings where objective evaluations are expensive, this overhead is expected to be less pronounced in relative terms.

6.3 Convergence and operator usage

Figure 2 shows convergence in normalized hypervolume for representative problems. Figure 3 summarizes operator selection fractions by stage for the trace-

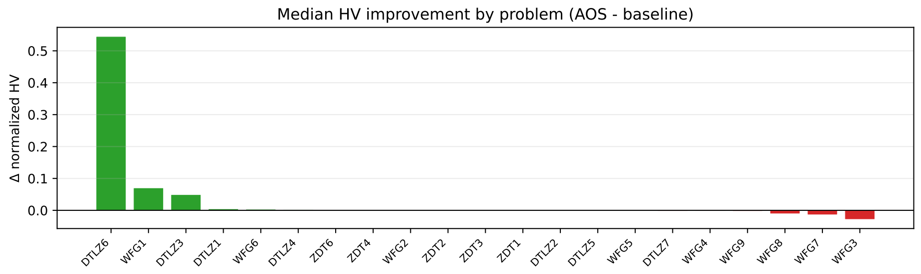

Fig. 1. Per-problem median hypervolume change (AOS minus baseline), sorted by Δ.

Table 3. Median runtime (seconds) by problem family (baseline NSGA-II vs. baseline + AOS). Values computed from `experiments/ablation_aos_racing_tuner.csv`.

Variant	**ZDT**	**DTLZ**	**WFG**	**Average**
Baseline	**0.89**	**1.00**	**1.02**	**0.97**
Baseline + AOS	1.36	1.39	1.84	1.53

exported problems. We observe that on some problems the policy quickly locks into a single arm (e.g., SBX+PM), while on others it maintains diversity in operator usage (e.g., mixing SBX+PM with UNDX+Gaussian).

7 Discussion

The results indicate that a simple generation-level AOS layer can yield meaningful gains on some problems (notably on DTLZ6), but also that “AOS by default” is not guaranteed to dominate a well-chosen fixed operator pipeline. This is consistent with the non-stationary and problem-dependent nature of operator utility.

Why can AOS underperform? First, bandit policies can lock into an initially lucky arm when exploration is low, preventing later-stage switches. Second, a portfolio that contains aggressive or highly disruptive operators can incur large opportunity costs when selected at the wrong stage. Third, reward definitions that are only weakly correlated with final hypervolume (e.g., survival credit without diversity) can reinforce short-term improvements that do not translate to better final fronts.

Practical improvements. Several modifications can improve robustness without changing the overall framework: (i) enforce a small warmup (`min_usage`> 0) to prevent early lock-in, (ii) adopt non-stationary policies (e.g., sliding-window UCB) to react to stage changes, (iii) use a “safer” portfolio composed of parameter variations of a strong baseline (e.g., SBX+PM with multiple η values)

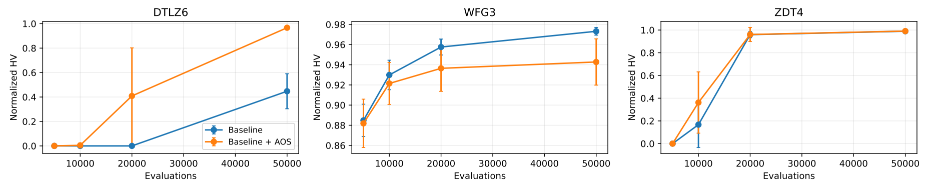

Fig. 2. Mean±std normalized hypervolume vs. evaluations for selected problems (baseline vs. AOS).

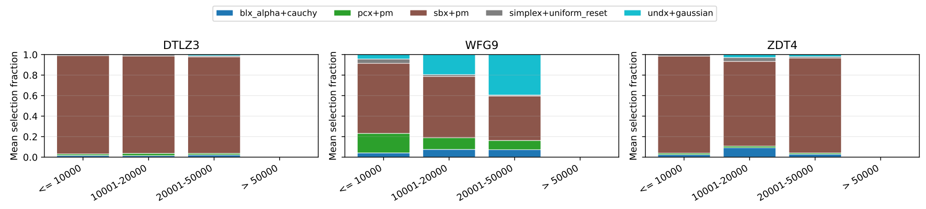

Fig. 3. Mean operator selection fractions by search stage for trace-exported problems (AOS variant).

rather than mixing in highly disruptive mutations, and (iv) tune reward weights to emphasize non-dominated insertion when diversity is critical.

8 Threats to validity

Portfolio dependence. Our conclusions depend on the chosen operator portfolio. A different set of arms (or different parameterizations) could shift both the magnitude and direction of the observed effects.

Reward design. We use a simple reward based on survival and non-dominated insertion, with an optional hypervolume proxy. More expressive credit assignment schemes (e.g., multi-step returns, per-offspring credit, or diversity-aware shaping) may produce different learning dynamics.

Benchmark scope. We evaluate continuous benchmark suites with standard dimensionalities. Results may not transfer directly to constrained problems, discrete encodings, or real-world objectives with expensive evaluations where overhead amortization differs.

Runtime measurement. Runtime overhead is measured in a vectorized engine, making controller overhead more visible. In other settings, the relative overhead of AOS may be smaller.

Table 4. Operator portfolio used by AOS. Mutation probability is set to $1/n_{\text{var}}$.

Arm	Crossover	Mutation
1	SBX (p=1.0, η=20)	Polynomial ($p=1/n_{\text{var}}$, η=20)
2	PCX (p=1.0, σ_η=0.1, σ_ζ=0.1)	Polynomial ($p=1/n_{\text{var}}$, η=20)
3	UNDX (p=1.0, ζ=0.5, η=0.35)	Gaussian ($p=1/n_{\text{var}}$, σ=0.1)
4	Simplex (p=1.0, ϵ=0.5)	Uniform reset ($p=1/n_{\text{var}}$)
5	BLX-α (p=0.9, α=0.5)	Cauchy ($p=1/n_{\text{var}}$, γ=0.1)

9 Conclusion

We presented a bandit-based adaptive operator selection layer for NSGA-II that selects one variation pipeline per generation and updates an online policy from survival and diversity-oriented rewards. Across standard benchmark suites, AOS improves median normalized hypervolume on a subset of problems and substantially improves performance on DTLZ6, while incurring a median runtime overhead of 41%.

Future work includes portfolio construction rules, more informative reward signals, non-stationary policies with explicit change detection, and cost-aware implementations that reduce controller overhead (e.g., by reusing selection statistics already computed by NSGA-II).

A Additional details

A.1 Operator portfolio

A.2 Per-problem tables

References

1. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning **47**(2–3), 235–256 (2002). https://doi.org/10.1023/A:1013689704352
2. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The nonstochastic multiarmed bandit problem. SIAM Journal on Computing **32**(1), 48–77 (2002). https://doi.org/10.1137/S0097539701398375
3. Da Costa, L., Fialho, Á., Schoenauer, M., Sebag, M.: Adaptive operator selection with dynamic multi-armed bandits. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO). pp. 913–920 (2008). https://doi.org/10.1145/1389095.1389272
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002). https://doi.org/10.1109/4235.996017
5. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In: Proceedings of the Congress on Evolutionary Computation (CEC). pp. 825–830 (2002). https://doi.org/10.1109/CEC.2002.1007032
6. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Transactions on Evolutionary Computation **10**(5), 477–506 (2006). https://doi.org/10.1109/TEVC.2005.861417

Table 5. Per-problem median normalized hypervolume at the final budget. Δ denotes AOS minus baseline (medians over seeds).

Problem	Baseline	AOS	Δ
zdt1	**0.991**	0.990	-0.000
zdt2	**0.982**	0.982	-0.000
zdt3	**0.996**	0.996	-0.000
zdt4	0.990	**0.991**	0.001
zdt6	0.980	**0.981**	0.001
dtlz1	0.915	**0.919**	0.005
dtlz2	**0.786**	0.785	-0.001
dtlz3	0.708	**0.757**	0.049
dtlz4	0.799	**0.801**	0.002
dtlz5	**0.964**	0.963	-0.001
dtlz6	0.423	**0.968**	0.545
dtlz7	**0.883**	0.880	-0.002
wfg1	0.439	**0.509**	0.070
wfg2	0.982	**0.983**	0.001
wfg3	**0.974**	0.946	-0.028
wfg4	**0.960**	0.957	-0.002
wfg5	**0.826**	0.825	-0.001
wfg6	0.833	**0.837**	0.003
wfg7	**0.966**	0.952	-0.014
wfg8	**0.744**	0.733	-0.011
wfg9	**0.934**	0.931	-0.003

7. Thompson, W.R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika **25**(3–4), 285–294 (1933). https://doi.org/10.1093/biomet/25.3-4.285
8. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation **8**(2), 173–195 (2000). https://doi.org/10.1162/106365600568202
9. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation **7**(2), 117–132 (2003). https://doi.org/10.1109/TEVC.2003.810758

Table 6. Per-problem median runtime (seconds) at the final budget. Δ denotes AOS minus baseline (medians over seeds).

Problem	**Baseline**	**AOS**	Δ
zdt1	**1.18**	1.40	0.22
zdt2	**0.93**	1.37	0.45
zdt3	**0.89**	1.37	0.48
zdt4	**0.80**	1.29	0.49
zdt6	**0.80**	1.28	0.48
dtlz1	**1.02**	1.36	0.34
dtlz2	**0.89**	1.41	0.52
dtlz3	**1.01**	1.33	0.32
dtlz4	**1.07**	1.41	0.33
dtlz5	**1.00**	1.38	0.38
dtlz6	**0.90**	1.36	0.47
dtlz7	**0.91**	1.43	0.52
wfg1	**0.92**	1.43	0.51
wfg2	**1.02**	1.44	0.42
wfg3	**1.03**	4.20	3.18
wfg4	**1.00**	1.41	0.42
wfg5	**0.97**	1.40	0.43
wfg6	**1.19**	1.99	0.80
wfg7	**1.01**	4.25	3.24
wfg8	**1.14**	1.93	0.79
wfg9	**1.41**	2.66	1.25