Towards Dynamic Switching in Per-Run Algorithm Selection

July 9, 2025

RWTH AACHEN UNIVERSITY

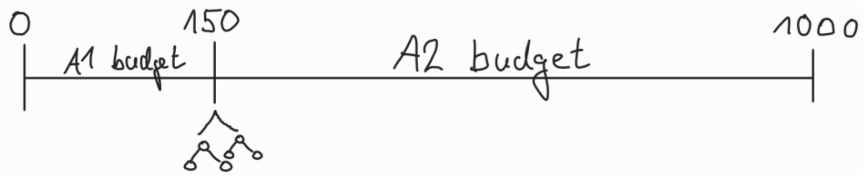
Continuous Black-Box Optimization

- Goal: Optimize a function $f : \mathbb{R}^D \to \mathbb{R}$
- Black-Box: Optimization is limited to sampling

Per-Instance Algorithm Selection

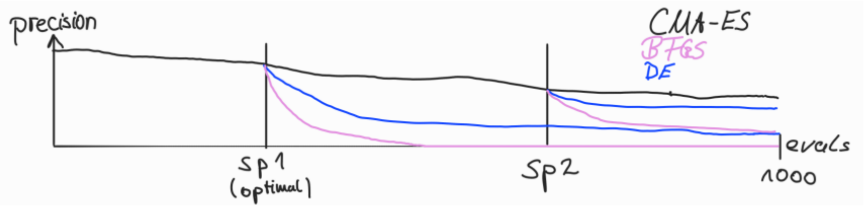
1. Sample f randomly
2. Compute ELA features from samples
3. Use features as input for a machine learning model that chooses the algorithm
4. Run the algorithm

Per-Run Algorithm Selection



- Approach from Kostovska et al. 2022:
  1. After 150 evaluations, calculate ELA features from CMA-ES samples
  2. Random Forest regression models predict target precisions of the six $\mathcal{A}2$ algorithms
  3. Choose $\mathcal{A}2$ algorithm with lowest predicted precision
  4. Warm-start second algorithm
- $\mathcal{A}1$ budget is static!

Why is Dynamic Switching Necessary?

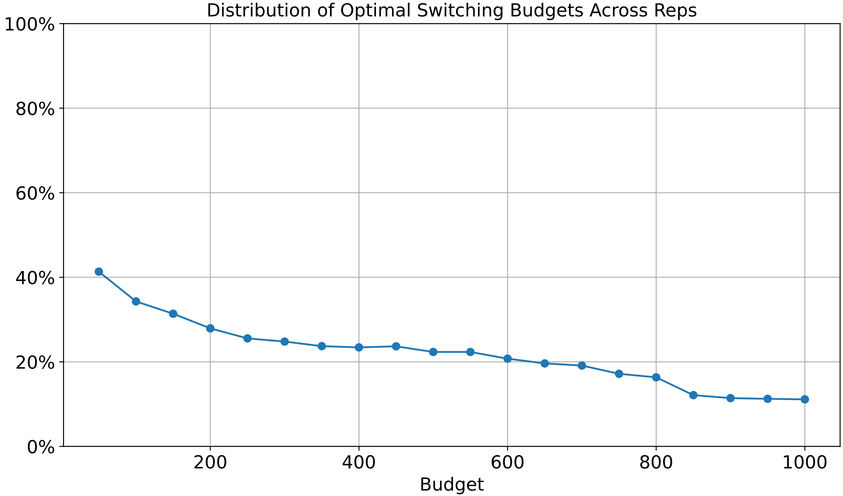


- Recording of 2400 runs conducted on BBOB functions

BBOB Test Suite (Hansen et al. 2016)

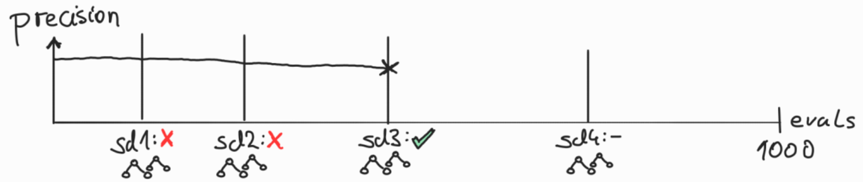
- Set of 24 synthetic noiseless black-box functions
- Each function has several instances
- Standard benchmarking set

Why is Dynamic Switching Necessary?

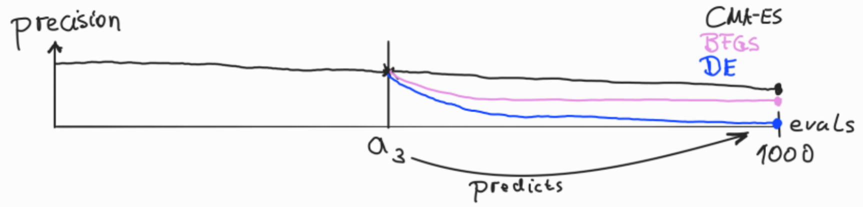


General Setup

1. Switching Decision



2. Algorithm Selection



First Approach

- For each run, we try to find the switching points defined as above
- Selection process consists of two parts:

Switching Decision

- Let $s_1, \ldots, s_n$ be the considered switching points
- At switching point s_i, we predict the precision of the best algorithm at $s_i, s_{i+1}, \ldots, s_n$
- We switch if the predicted precision of the current switching point is the lowest

Algorithm Selection

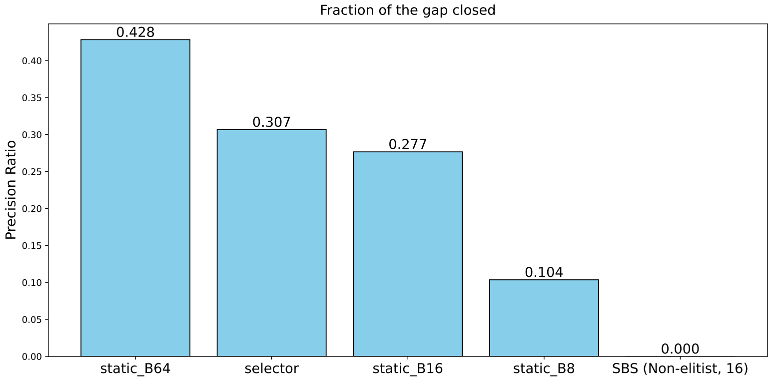
- Predict the precisions of all six algorithms, switch to the algorithm with the lowest precision

First Approach: Our setup

- Switching points: $8, 16, \ldots, 96, 100, 150, 200, \ldots, 1000$
- Machine learning input:
  - ELA features from CMA-ES samples
  - CMA-ES internal state during the last iteration
- Model training on first 5 instances of all BBOB functions, 20 runs each
- Evaluation on instances 6 and 7, 20 runs each
- Parameter tuning using ASF
- Baselines:
  - SBS: (Algorithm, Budget) combination that is best on the training set
  - VBS: Chooses the best switching point and algorithm at that switching point for each run
- Metric:

$$cg_{selector} = \frac{m_{SBS} - m_{selector}}{m_{SBS} - m_{VBS}}$$

First Approach: Results



Poor Performance

- Switching decision based on performance of best algorithm
  $\Rightarrow$ Switch too early
- ELA features do not capture enough run-specific information

Second Approach

Let $s_1, \ldots, s_n$ be the switching points, $a_1, \ldots, a_n$ be the algorithm selectors and $sd_1, \ldots, sd_n$ be the switching decision models

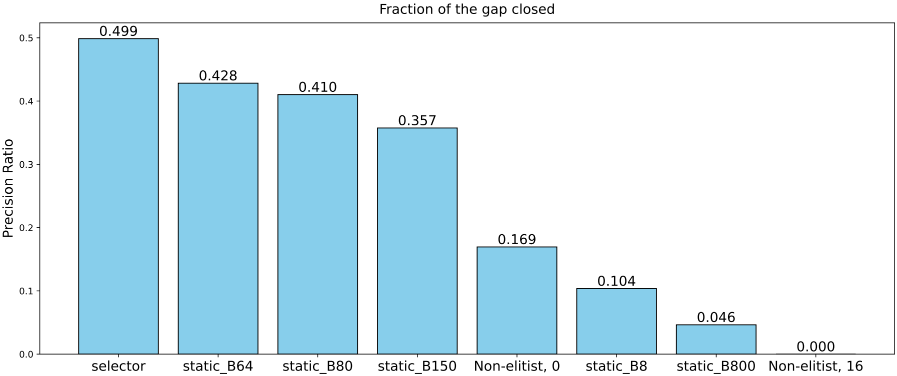
1. Evaluate Performances of $a_1, \ldots, a_n$

1. Record the performances of $a_1, \ldots, a_n$ on first five instances
2. For each function, determine which a_i performed best
   $\Rightarrow s_i$ is optimal for that function

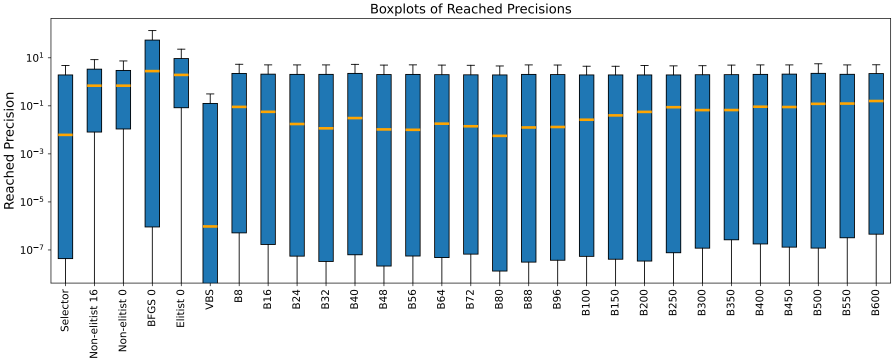
2. Training data for $sd_1, \ldots, sd_n$

1. For ELA-features and CMA-ES data belonging to a run at s_i, we label them as true iff s_i is greater or equal than the optimal switching point of the run's function
2. For each s_i, train a binary classifier, predicting whether or not to switch

Second Approach: Results



Second Approach: Results



Permutation Test: Selector vs Static Baseline

Setup

- Data: Precision values per run for selector and static baseline selector
- Test: Permutation test using SciPy with 10,000 resamples
- Statistic: Mean difference in reached precision

Null Hypothesis

- Null Hypothesis (H_0):
  The selector and static baseline have the same distribution of precision values. Any observed difference is due to random chance.

$\Rightarrow$ Found statistical significance

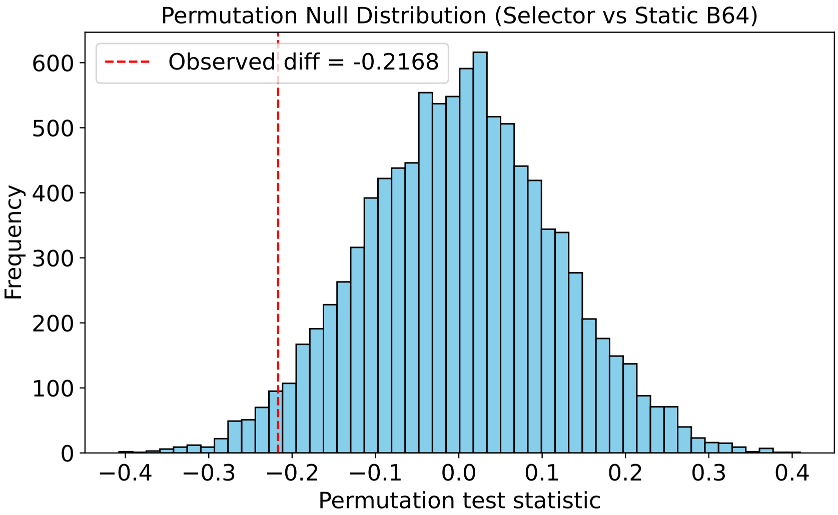
Next Step: In-Depth Empirical Evaluation

- Repeat experiments for dimension 10
- BBOB functions are purely synthetic
- Evaluation of the selector on LassoBench (Šehić et al. 2022)
  - Real-world HPO functions for weighted lasso regression
- Function from LassoBench do not have instances
  $\Rightarrow$ Train-test splits using different runs on each function
- And other real-world benchmarks

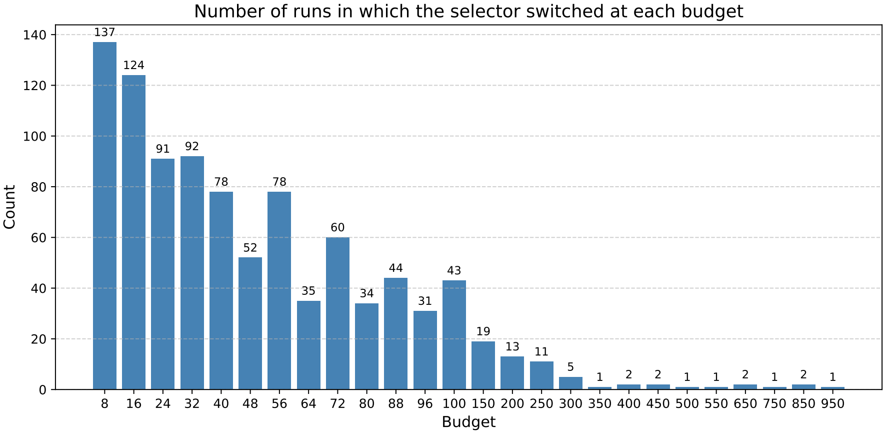
Take-Home Message

- Dynamic Switching leads to improvements over static switching
- First approach did not work
  - ELA features do not seem to capture run-specific information
  - Suboptimal performance of algorithm selectors across budgets
- Second approach leads to significant improvements over static switching
- Next step: Evaluate robustness of the selector

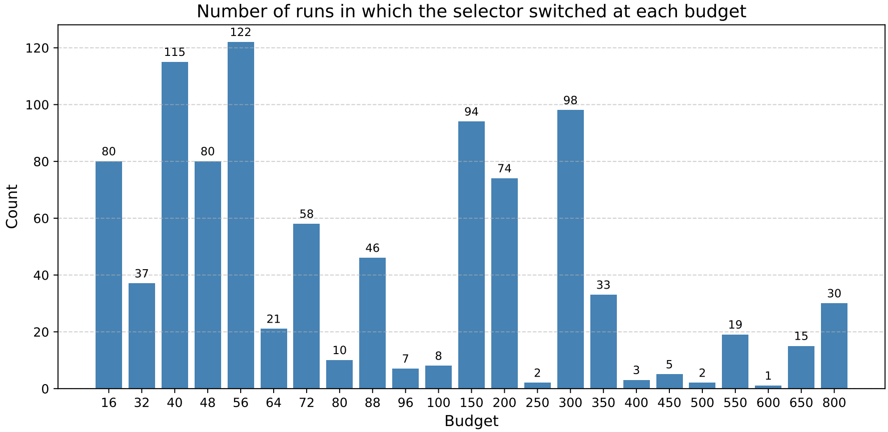
Permutation Test Results



Switching Budgets: First Approach



Switching Budgets: Second Approach



Algorithm Portfolio (Schröder et al. 2022)

- Broyden-Fletcher-Goldfarb-Shanno (BFGS): Deterministic line-search, approximates the hessian, local search
- Differential Evolution (DE): Random mutation of target solutions using binomial crossover
- Multi-Level Single Linkage (MLSL): Local searches based on clustering heuristics
- Particle Swarm Optimization (PSO): Swarm-based, velocity of particles gets adjusted based on their own and the global optimum
- CMA-ES elitism/non-elitism: In each iteration, does the offspring become the new population or does the new population consist of the best samples from both the old population and the offspring?