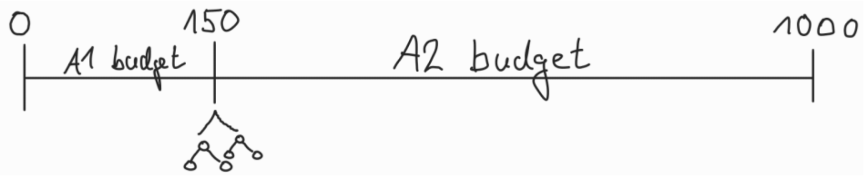
Towards Dynamic Switching in Per-Run Algorithm Selection

July 3, 2025

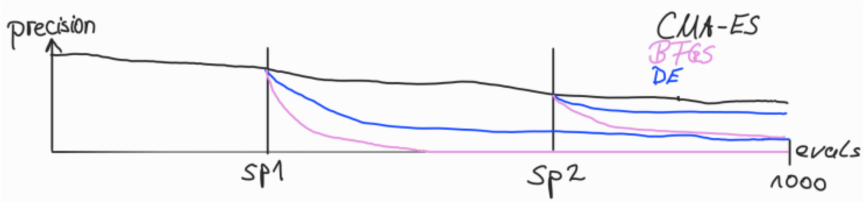
RWTH AACHEN UNIVERSITY

Why is Dynamic Switching Necessary?



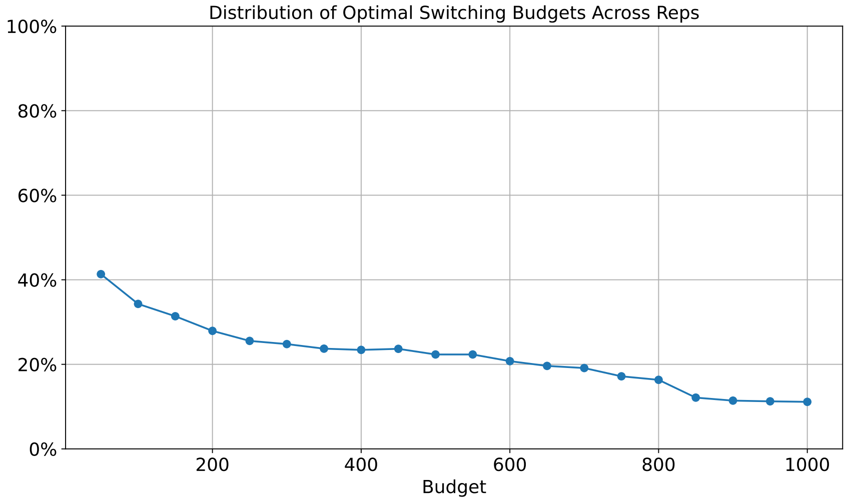
- Approach from Kostovska et al. 2022:
  1. After 150 evaluations, calculate ELA features from CMA-ES samples
  2. Random Forest regression models predict target precisions of the six $\mathcal{A}2$ algorithms
  3. Choose $\mathcal{A}2$ algorithm with lowest predicted precision
  4. Warm-start second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

Why is Dynamic Switching Necessary?



- Switching point 1 is optimal here

Why is Dynamic Switching Necessary?



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

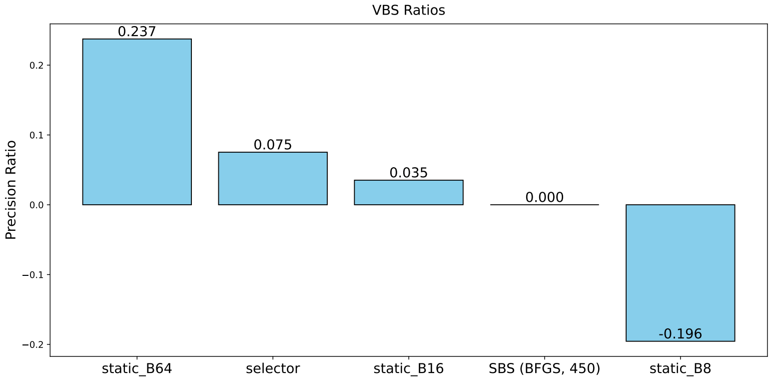
- Predict the precisions of all six algorithms, switch to the algorithm with the lowest precision

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
- Metric:

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

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

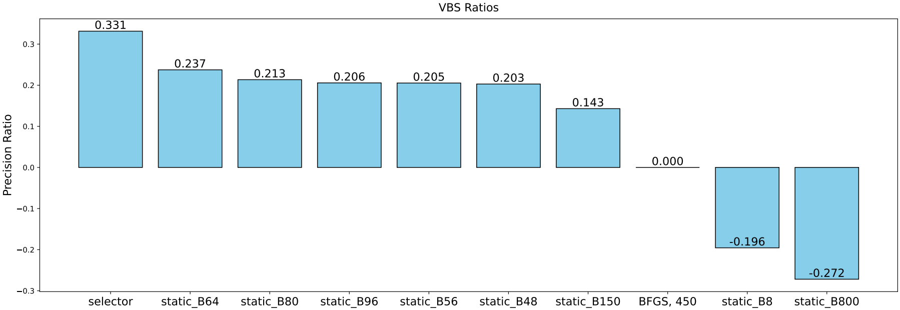
Second Approach

Let $s_1, \ldots, s_n$ be the switching points, $a_1, \ldots, a_n$ be the algorithm selectors

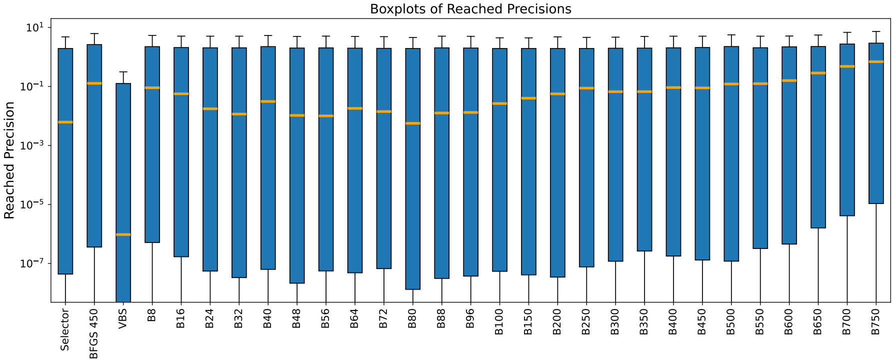
1. Record the performances of $a_1, \ldots, a_n$ on first five instances
2. If a_i performs best on function f, define all s_l as the optimal switching point for all runs on that function for all $l \geq i$
3. Define binary switching models for each switching point
4. Train them on first five instances
5. Evaluation on instances 6 and 7

$\Rightarrow$ Switching decision is now function-specific and takes algorithm selector performance into account

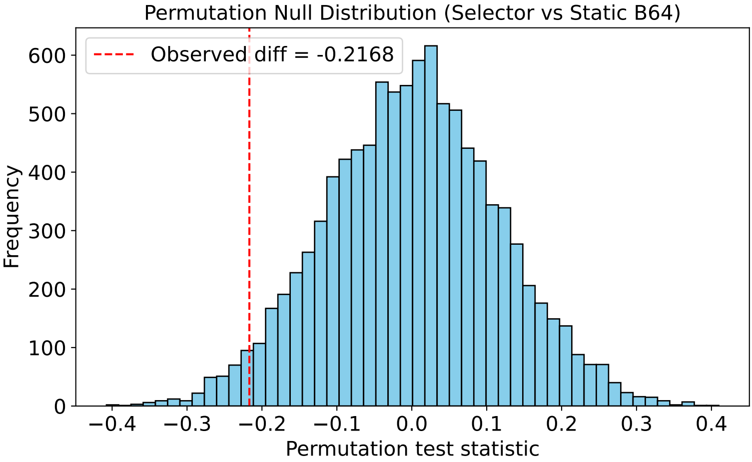
Second Approach: Results



Second Approach: Results



Second Approach: Statistical Significance



- p-value: 0.029

Conclusion

- 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

Detailed Boxplots