Selector Setup

At each checkpoint:

- Use ELA-features + CMA-ES internal state to predict the precisions of the optimal algorithms at this checkpoint and all following checkpoints
- If the predicted precision of the current checkpoint is among the lowest of all predictions: Switch.
- If we switch, we then use a second selector that chooses the best algorithm at that checkpoint (also via regression models for the precision of the six algorithms)

Selector Implementation and Optimization

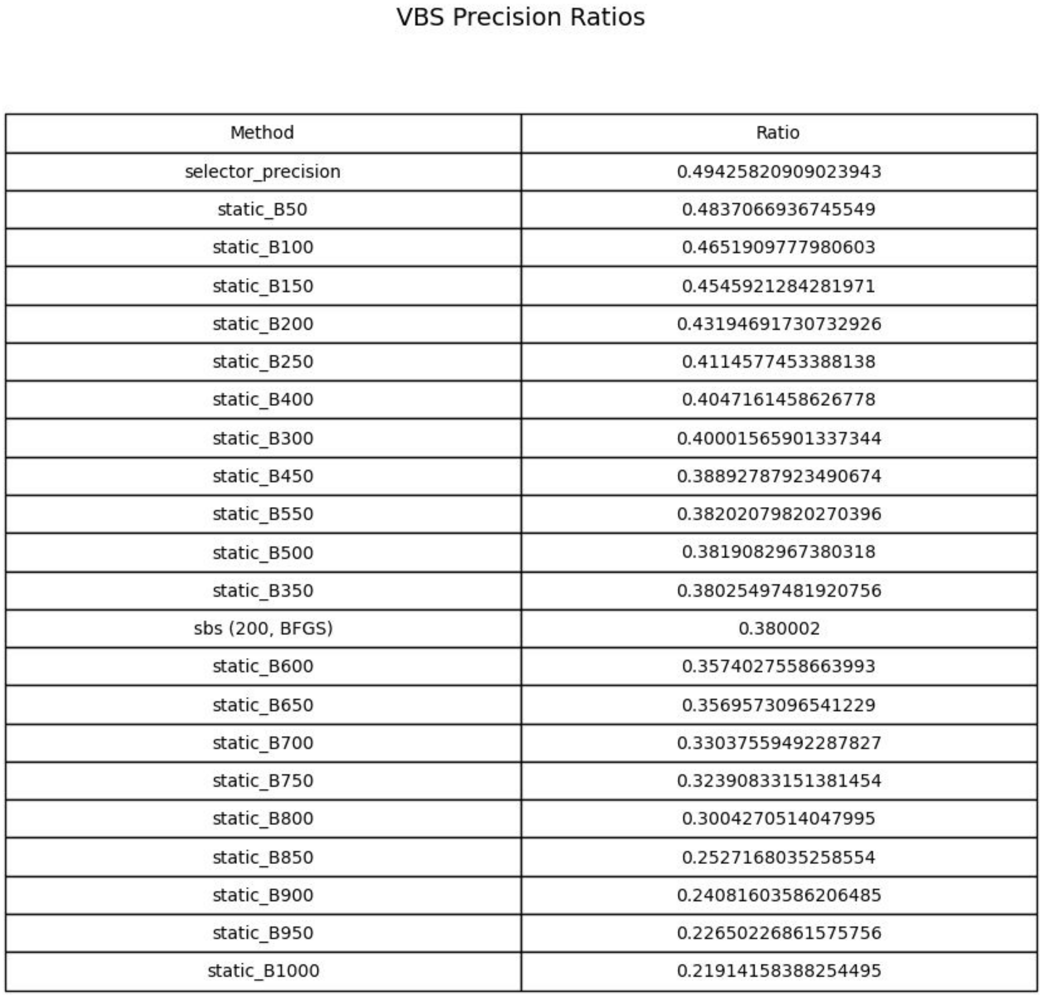
- For the models I used asf’s “Performance Model”
- Performance Model is a selector that trains regression models for all possible algorithms that predict the log-precision. It then chooses the algorithm with the lowest predicted precision (so exactly what we need)
- I optimized each Performance Model (so each collection of regressors) individually using asf’s tune_selector (SMAC with 75 trial points)
- Drawback: Each Regressor inside one Performance Model is assigned the same set of hyperparameters
- For the optimization metric, I used the sum of precisions of the predicted algorithms (so I look for the performance model for which, on average, the precision of the algorithms that it chooses is lowest)

Train and Test Setup

- I used five-fold cross-validation over the first five instances of each function to optimize the Performance Models
- After that optimization, I trained the Performance Models on all five instances
- I then recorded runs on instances 6 and 7 of each function (20 each) to use as test set

Results

static_B… means: Selector that always switches at that budget (so it only considers the performance of the algorithm selector at that budget, so static_B150 is the same setup as in the per-run paper). For each run, the precision of the VBS is defined as the lowest precision achieved by one algorithm across all of the budgets of that run

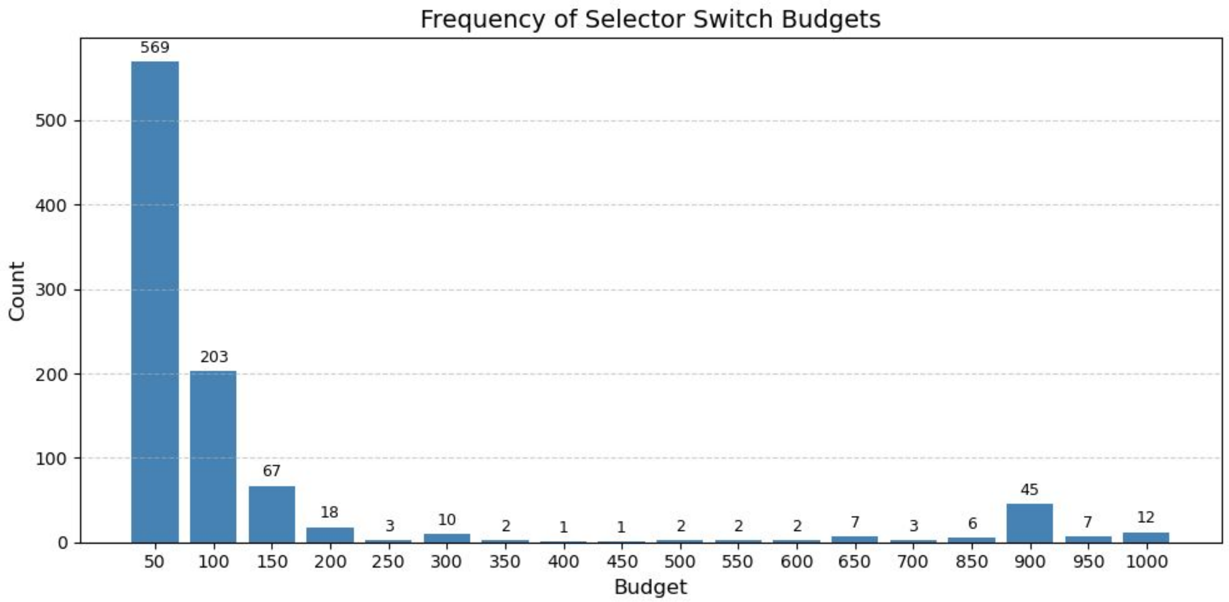


VBS Precision Ratios

Method	Ratio
selector_precision	0.49425820909023943
static_B50	0.4837066936745549
static_B100	0.4651909777980603
static_B150	0.4545921284281971
static_B200	0.43194691730732926
static_B250	0.4114577453388138
static_B400	0.4047161458626778
static_B300	0.40001565901337344
static_B450	0.38892787923490674
static_B550	0.38202079820270396
static_B500	0.3819082967380318
static_B350	0.38025497481920756
sbs (200, BFGS)	0.380002
static_B600	0.3574027558663993
static_B650	0.3569573096541229
static_B700	0.33037559492287827
static_B750	0.32390833151381454
static_B800	0.3004270514047995
static_B850	0.2527168035258554
static_B900	0.24081603586206485
static_B950	0.22650226861575756
static_B1000	0.21914158388254495

Results

The switches at later budgets (more than 600) are mostly all on the two easy functions sphere and linear slope. Whenever the switching decision is important, we heavily favor early switching.



Questions

- Is the optimization okay? Or should I do it differently?
- Is the train/test split okay?
  - Should we use more data, both for training and testing, e.g. 50-100 runs per instance?
  - Currently, I am not sure if the improved performance is just “lucky”
- Performance metric:
  - Currently, for each run, I divided the vbs precision by the selector’s precision
  - I added eps to both to avoid division by zero
  - And then averaged over all runs
- Cluster thesis application