Week 9: Function specific switching

- Idea: Measure the performance of the static algorithm selectors on instances 1 to 5
- I used a 4-fold cross validation where I trained the selectors on ¾ of runs, and then let them predict the best algorithm for the remaining quarter.
- For each function, I set the switching point as the budget for which the static selector performed at that budget best on that function (lowest sum of precisions)
- I then trained random forests, one for each possible switching budget, that predict whether or not to switch for a run (so a binary prediction here)

Week 9: Function specific switching

- In a first attempt, I set the switch decision to true for a function and a budget, only if that budget is the right budget for the function
- This resulted in poor performance, because each run for which the right budget was not detected, then continued without switching, which is basically always a bad decision
- So, I set the switch decision to true for a function and a budget, if that budget is greater or equal than the right budget for that function. (So that if a run is not detected to switch at the right budget, it will switch somewhere behind that)
- This resulted in good performance

Results if we consider multiples of 50 as switching points

Method	Ratio
selector_precision	0.6914739331516234
static_B50	0.7342292114590188
static_B200	0.7834688934562055
static_B100	0.8096604126452549
static_B150	0.8279748194510756
static_B350	0.8295645201063693
static_B450	0.8315831727495465
static_B300	0.833793375248887
static_B400	0.8454845500999881
static_B250	0.8521377125549351
static_B500	0.8646264151484486
static_B550	0.8651682121985461
static_B600	0.8680715726930405
static_B650	0.8727674677814952
static_B700	0.9691203768199774
static_B750	1.170709233204784
static_B800	1.2944068941955356
static_B850	1.612240193937837
static_B900	1.9356077011386117
static_B950	2.5804227966401223
static_B1000	3.9009662685339572

Results for multiples of 50 and multiples of 8 lower than 100

Method	Ratio
selector_precision	0.7348407859659998
static_B80	0.8343698876143243
static_B56	0.8431062571644096
static_B64	0.865747418437369
static_B24	0.8695766953339663
static_B40	0.874061958394088
static_B96	0.8749348244632318
static_B200	0.8864525814801185
static_B100	0.9095093121298923
static_B48	0.9111641348804176
static_B150	0.9256317196040164
static_B350	0.9270311535031466
static_B450	0.9288081993450942
static_B32	0.9297976509890551
static_B72	0.9301458725592493
static_B300	0.9307538689849849
static_B400	0.9410457603249096
static_B250	0.9469026246627765
static_B88	0.9556409676800052
static_B500	0.9578965899148633
static_B550	0.9583735408149151
static_B600	0.9609294063383146
static_B650	0.9650632630815067
static_B16	0.9933172712163364
static_B700	1.049883966110878
static_B750	1.227345224853125
static_B800	1.3362378641416175
static_B8	1.3913457921804038
static_B850	1.6160305989880546
static_B900	1.9006951677212154
static_B950	2.4683341759274477
static_B1000	3.63082555044471

Evaluation on new instances

- I also wanted to see if this works if we use runs on instances 6 and 7 (20 each) as a test set. Once again, I considered both sets of switching points
- The performance is worse here. Maybe this is because the distributions of training and test set are too different?

Method	Ratio
selector_precision	0.7397463435645291
static_B64	0.783110528315716
static_B80	0.7991784183080536
static_B96	0.8011234209716904
static_B56	0.8052528171186433
static_B48	0.8091478061854296
static_B100	0.8340053853588267
static_B72	0.8350493625064949
static_B550	0.8507966151371419
static_B300	0.8522574095492805
static_B400	0.8527850943443569
static_B350	0.857926650631531
static_B250	0.8623110386200014
static_B500	0.8793981739119423
static_B150	0.881782488942987
static_B600	0.8849761676564104
static_B450	0.8905819890916795
static_B650	0.8915562283902
static_B200	0.900379391354849
static_B88	0.9070488066040253
static_B24	0.9505310751659515
static_B700	0.9615915740558716
static_B32	0.9655715603363839
static_B16	1.011701664513621
static_B40	1.0258049077785734
static_B750	1.1243501419530186
static_B8	1.1955308516283178
static_B800	1.2689075171966206
static_B850	1.6472226745221685
static_B900	1.9641426220843177
static_B950	3.2024631000855757
static_B1000	8.075054376573444

Method	Ratio
static_B50	0.8052528171186433
selector_precision	0.8082308838341555
static_B100	0.8340053853588267
static_B550	0.8507966151371419
static_B300	0.8522574095492805
static_B400	0.8527850943443569
static_B350	0.857926650631531
static_B250	0.8623110386200014
static_B500	0.8793981739119423
static_B150	0.881782488942987
static_B600	0.8849761676564104
static_B450	0.8905819890916795
static_B650	0.8915562283902
static_B200	0.900379391354849
static_B700	0.9615915740558716
static_B750	1.1243501419530186
static_B800	1.2689075171966206
static_B850	1.6472226745221685
static_B900	1.9641426220843177
static_B950	3.2024631000855757
static_B1000	8.075054376573444

Questions

- Is the switching selection done right? I think we could probably improve here. Currently, around 80% of runs switch at the right switching point. We need a way to handle the remaining runs.
- I also tried to tune the switching models (according to F1 score), but this did not improve the performance. Should I tune them according to a different metric?
- The performance of the static selectors on the training and test data often does not really match. I think we need more data (so more runs per instance) to really understand this. Currently, the performance of a static depends on only a few (un)lucky algorithm choices