Estimation of parameter importance and interaction with Random Forests

Leslie Pérez Cáceres

In this document we are testing how to use Random Forest to assess importance and interactions of parameters based on the data gathered by irace.

For the analysis below we use as example ACOTSP in two scenarios (short and long):

- 7 secs cut off time (short) / 20 secs cut off time (long)
- 11 parameters
- 400 instances sizes (1000,1500,2000,2500,3000,3500,4000,4500)
- 3000 experiments configuration budget / 5000 experiments configuration budget

We use random forest for predicting

1. configuration ranking
2. configuration ranking quartile
3. configuration performance (just to compare)

Models are trained using default settings of the package Random Forest, excepting the number of trees that was set to 300. We have access to the following measures that can be considered indicators of importance:

- mean_min_depth: mean depth of the subtree closest to the tree root, where a variable is root of the sub tree.
- no_of_nodes: number of nodes in which the variable was used to split
- mse_increase: increment of prediction mean squared error
- no_of_trees: number of trees in which the variable was used
- times_a_root: number of times a variable was selected as root variable
- accuracy_decrease: measure of the classification accuracy (only for classification models)

For this analysis we use data generated by irace, it is possible to select a subset of the data and perform the analysis. The data is imputed and there is a procedure to analyze importance based on a reference variable and a retraining scheme to assess real importance in conditional parameters. The instance is also used as a predictor in this data. (details in another document)

There are some things that should be investigated regarding the best way to use the models to asses interaction and importance:

1. Discretizing numerical variables for prediction: based on a comment I got that RF are biased to select numerical variables as split. This will be particularly interesting and in line with the fact that irace defines a sampling range around the current value.

2. Adjusting the number of trees and depth of them. My intuition is that smaller more smaller trees would be more useful in the task of detecting importance. This is because lower level splits are not as interesting and higher level splits. Also, to be used as a post-execution analysis tool the execution time required to build the model depends on these parameters.

3. How to understand how this importance or interaction is materialized i.e., which are the best parameter values ans how these interact. Can we have an heuristic idea of this interpreting the forest splits?

4. How to interpret instance importance and interaction when using models not predicting directly the performance. Can this help detecting heterogeneous sets?

1. Configuration ranking as dependent variable

In this example we use the *ranking as the dependent variable* when training, thus the trained random forest predicts que mean ranking of a configuration.

For training the model we must impute the dependent variable (ranking) since some configurations are not executed in some instances. We asume this configurations to be worst than the configurations executed in the instance, after imputation all configurations have a ranking assigned for each instance.

In these experiments the variable instance should not be a good as predictor compared to a model trained to predict performance directly. Despite this, interactions of the instance variable with other variables are possible and might indicate heterogeneity of the benchmark.

When having the short bugdet is expected that best performance its highly dependent on parameters, especially parameter which have a big effect on intensification of the search.

1.1 All data model

In this model all irace data execution was used for training. Note that due to the focused nature of irace data (given model convergence) there might be contradicting or not consistent interactions in the data.

We show importance measures ordered by the mean min depth given that the interaction analysis is based on this measure.

The initial parameter importance measures of the long benchmark:

```
load("model_data/model-acotsp1000-4500-ranking.Rdata")
importance_frame = model$importance_frame
important_parameters = model$important_parameters
full_interactions_frame = model$full_interactions_frame
interactions_frame = model$interactions_frame
kable(importance_frame[order(importance_frame[,"mean_min_depth"]),])  %>%
      kable_styling(latex_options="scale_down")
```

	variable	mean_min_depth	no_of_nodes	mse_increase	node_purity_increase	no_of_trees	times_a_root	p_value
9	localsearch	1.513333	11673	11095.40660	36736611	300	57	1
8	instance	2.003333	230875	7407.05141	90587926	300	0	0
5	dlb	2.080000	9578	9589.18283	27486731	300	37	1
11	q0	2.100000	52185	14476.20668	61102886	300	78	1
10	nnls	2.146667	69460	17400.80970	59853112	300	21	0
1	algorithm	2.183333	18344	7438.70516	31702412	300	81	1
2	alpha	2.673333	90098	11426.58140	46260673	300	9	0
3	ants	2.760000	85568	12016.18996	43561894	300	11	0
4	beta	3.150000	88332	8373.31471	37072211	300	2	0
13	rho	3.546667	84971	9182.43050	35770350	300	0	0
6	dummy	4.033333	66114	6590.37820	22056549	300	0	0
7	elitistants	4.166667	12703	332.70177	1999062	300	4	1
12	rasrank	5.066667	13110	77.17294	1832337	300	0	1

The initial parameter importance measures of the short benchmark:

```
load("model_data/model-acotsp1000-4500-low-ranking.Rdata")
importance_frame_low = model$importance_frame
important_parameters_low = model$important_parameters
full_interactions_frame_low = model$full_interactions_frame
```

```
interactions_frame_low = model$interactions_frame
kable(importance_frame_low[order(importance_frame_low[,"mean_min_depth"]),])  %>%
      kable_styling(latex_options="scale_down")
```

	variable	mean_min_depth	no_of_nodes	mse_increase	node_purity_increase	no_of_trees	times_a_root	p_value
9	localsearch	1.186667	5088	4912.2354	10950373.2	300	93	1
3	ants	1.810000	43232	6202.6054	14040421.0	300	58	0
10	nnls	2.160000	37818	4409.3503	10900283.7	300	45	0
8	instance	2.313333	123674	3730.7508	22197871.3	300	0	0
1	algorithm	2.383333	15852	5252.2123	7754571.1	300	11	1
13	rho	2.483333	43609	4362.2906	9478395.7	300	32	0
2	alpha	2.520000	46651	3975.3231	9571527.4	300	16	0
5	dlb	2.756667	6347	1543.2924	2721679.7	300	39	1
4	beta	3.133333	46062	3180.5041	7335376.5	300	5	0
12	rasrank	3.563333	12549	1402.9615	3435302.5	300	0	1
11	q0	3.926667	9831	555.1538	1638444.5	300	0	1
7	elitistants	3.963333	6226	323.8121	801844.2	300	1	1
6	dummy	3.996667	35310	1390.5434	3897781.7	300	0	0

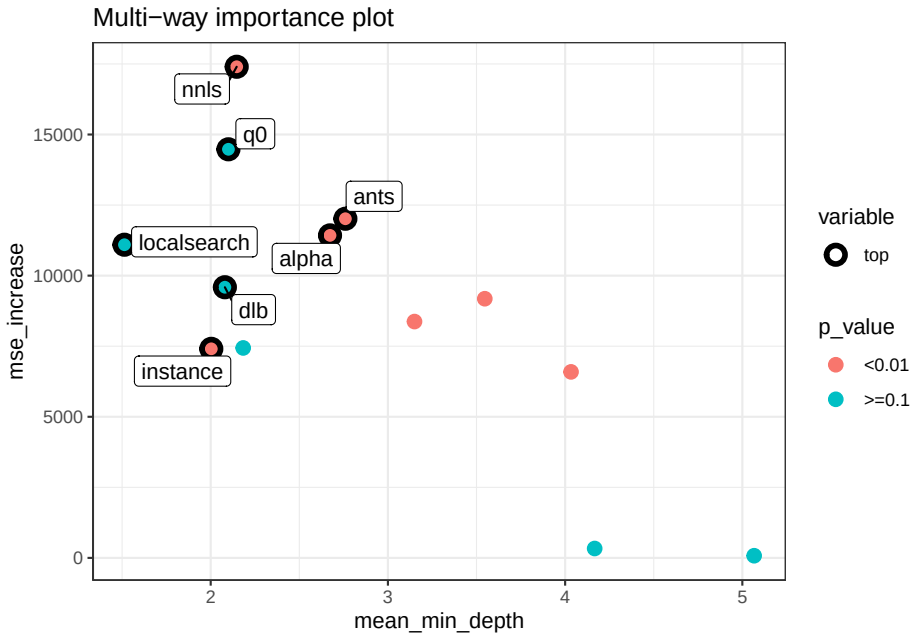
This list of important parameters is preliminary given that still has not been analized taking into account conditionality and the reference parameter. In this list, the most important parameter is localsearch in both benchmarks, this goes in line with what is known of ACOTSP. The appearance of the instance as a important parameter could signal some heterogeneity, that should be confirmed with an interaction. Also, there are some conditional parameters listed that will analyzed in a second step to determine if the detected importance can be associated directly to them or to their activation parameter. For example, nnls is conditional to localsearch and thus its importance might be derived from the use of localsearch.

We can plot two importance measures using the randomForestExplainer package, we choose the to show the mean min depth in the x axis and the mse increase in the y axis. Top 7 variables are highlighted. In this case, since we are not predicting performance the effect of instance variable in model performance must be interpreted carefully. These plots are interesting given that contrast importance for ranking prediction in terms of accuracy (mse_increase) and in terms of early importance the tree more in line with classification goals (mean min depth).

Importance of the long benchmark:

```
plot_multi_way_importance(importance_frame, size_measure = "p_value",
                          y_measure="mse_increase", x_measure="mean_min_depth",
                          no_of_labels=7)
```

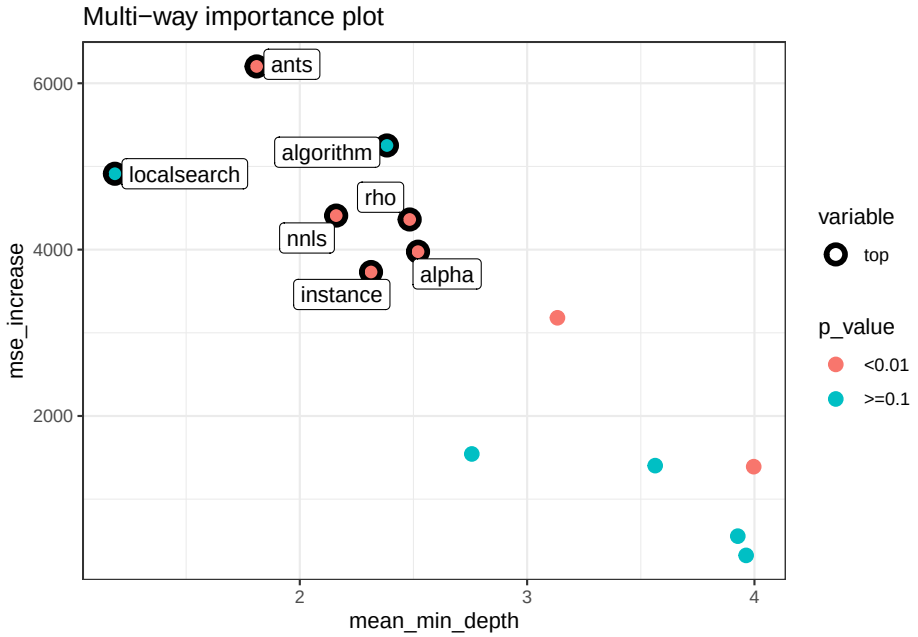
```
## Warning: Using alpha for a discrete variable is not advised.
```



Importance of the short benchmark:

```
plot_multi_way_importance(importance_frame_low, size_measure = "p_value",
                          y_measure="mse_increase", x_measure="mean_min_depth",
                          no_of_labels=7)
```

```
## Warning: Using alpha for a discrete variable is not advised.
```



In both benchmarks localsearch is the most important variable in terms of mean min depth, meaning that in general localsearch is early chosen to perform a split in the trees. On the other hand, for mse increase the mos important parameters are different in both benchmarks. In the long benchmark, both nnls and q0 are the most important. These parameters must be still analized to separate them from the effect of its activator parameter (localsearch and algorithm). For the short benchmark, the most important parameter are ants and

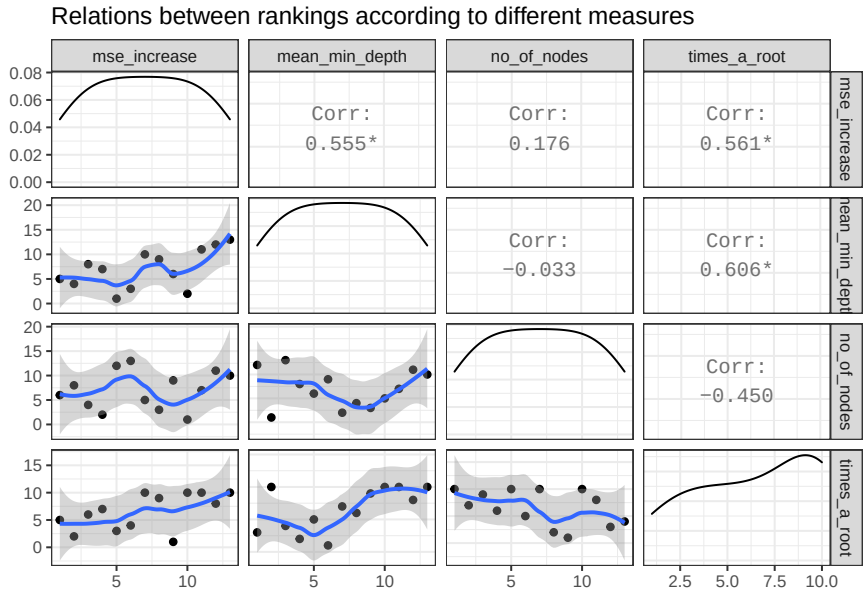
algorithm. Note that ants is located 3rd in the long benchmark and if we consider conditional parameters as their activators (nnls -> localsearch and q0 -> algorithm) we get the same set of parameters as the most important in both benchmarks (localsearch, ants, algorithm).

We can also visualize the relationship between importance measures, this could help to understand which indicator is more suited or can be used as a complement of other:

The long scenario:

```
plot_importance_rankings(importance_frame, measures=c("mse_increase", "mean_min_depth",
                                                      "no_of_nodes", "times_a_root"))
```

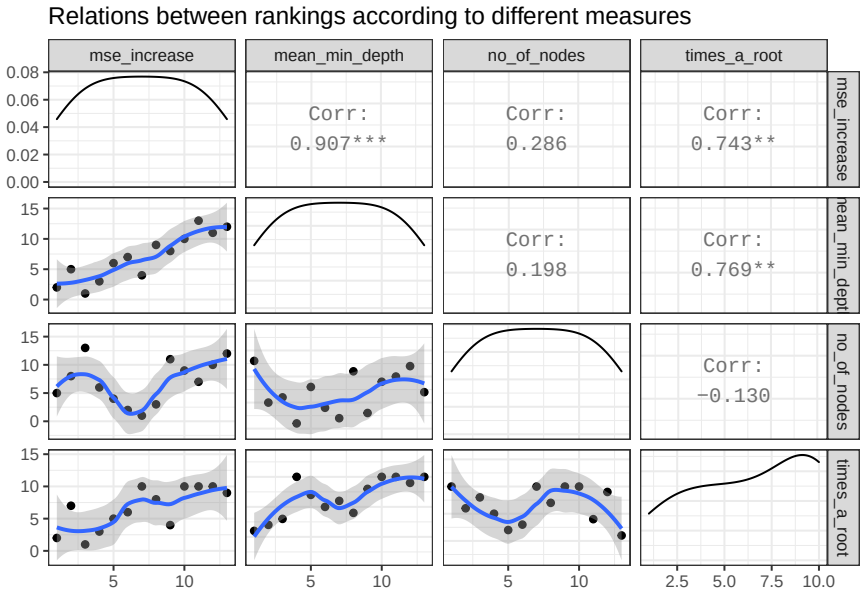
```
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```



The short scenario

```
plot_importance_rankings(importance_frame_low, measures=c("mse_increase", "mean_min_depth",
                                                      "no_of_nodes", "times_a_root"))
```

```
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```



The highest correlations are between the mse_increase, mean_min_depth and times_a_root. The times_a_root counts how many times a parameters was chosen as the first variable to perform a split of the data. This higher correlation could be showing that the mean_min_depth can be also used to predict importance to final ranking prediction. This relatively good correlation is probably a good indicator of the quality of a model to make importance assumptions. And also, that the min mean increase is valid a an importance predictor, this is important given that later this indicator is used to calculate interaction.

We perform an analysis of conditional parameters importance using a filtering and re training strategy and apply an irrelevant parameter filter by including a reference parameter. After this process the most important 5 parameters are detected.

Important parameters for long scenario:

```
print(important_parameters)
```

```
## [1] "localsearch" "instance"    "nnls"        "dummy"
```

Important parameters for short scenario:

```
print(important_parameters_low)
```

```
## [1] "localsearch" "ants"        "instance"    "algorithm"   "dummy"
```

Note that q0, which was detected initially as an important parameter has been removed. Its removal is related to its conditional dependency to the algorithm parameter. The mentioned dependency analysis discarded it as important by separating its effect of these parameters. The method uses to analyze conditional importance does not evaluate adding the activator parameter to the list of parameters (for now). But note that if that would be the case, the most important parameters would be the same.

In both cases the instance is listed as an important variable, this is expected as the scenario involves different instance sizes, and thus, different configurations may be more suited to solve certain instances. The importance of the instance for ranking prediction might be an indication of the level of heterogeneity of the scenario. This could signal that certain parameter values are better to solver certain instances, this should be more evident if initial iterations data is used as the model of irace has not yet converged.

Next, we run the interaction analysis only over important parameters. This is assuming the interactions one cares to detect are related to them, which might be not entirely correct ans should be evaluated as heuristic. The importance of interactions is calculated based on the mean min depth indicator in its conditional version.

Parameter interaction importance for the long scenario:

```
kable(full_interactions_frame) %>% kable_styling(latex_options="scale_down")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth
instance	localsearch	0.8873720	293	localsearch:instance	1.236667
dummy	dummy	1.0400114	292	dummy:dummy	1.343333
instance	dummy	0.8103982	291	dummy:instance	1.236667
nnls	localsearch	1.0100114	291	localsearch:nnls	1.470000
instance	instance	0.8786689	290	instance:instance	1.236667
nnls	dummy	0.9655063	289	dummy:nnls	1.470000
nnls	nnls	1.0540956	288	nnls:nnls	1.470000
instance	nnls	1.0470648	286	nnls:instance	1.236667
nnls	instance	1.1173493	286	instance:nnls	1.470000
localsearch	localsearch	1.7704209	285	localsearch:localsearch	1.323333
dummy	localsearch	1.9183163	265	localsearch:dummy	1.343333
dummy	instance	2.2761661	258	instance:dummy	1.343333
dummy	nnls	2.3509556	254	nnls:dummy	1.343333
localsearch	instance	2.9630262	243	instance:localsearch	1.323333
localsearch	dummy	3.3259613	237	dummy:localsearch	1.323333
localsearch	nnls	3.4337201	229	nnls:localsearch	1.323333

Parameter interaction importance for the short scenario:

```
kable(full_interactions_frame_low) %>% kable_styling(latex_options="scale_down")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth
ants	localsearch	1.0376712	292	localsearch:ants	1.613333
instance	instance	0.9120091	291	instance:instance	1.506667
ants	algorithm	1.0980594	290	algorithm:ants	1.613333
ants	instance	1.0887671	289	instance:ants	1.613333
instance	algorithm	1.2566781	289	algorithm:instance	1.506667
algorithm	algorithm	1.5773402	287	algorithm:algorithm	1.623333
instance	localsearch	1.2071918	286	localsearch:instance	1.506667
ants	dummy	1.3096119	285	dummy:ants	1.613333
dummy	dummy	1.2822146	285	dummy:dummy	1.660000
dummy	instance	1.3347945	283	instance:dummy	1.660000
dummy	algorithm	1.4149543	282	algorithm:dummy	1.660000
dummy	ants	1.4950685	280	ants:dummy	1.660000
ants	ants	1.5802740	279	ants:ants	1.613333
localsearch	localsearch	2.1241438	279	localsearch:localsearch	1.576667
dummy	localsearch	1.6763699	278	localsearch:dummy	1.660000
instance	dummy	1.5592466	277	dummy:instance	1.506667
instance	ants	1.7768493	274	ants:instance	1.506667
algorithm	localsearch	1.9726027	268	localsearch:algorithm	1.623333
algorithm	instance	2.1787671	262	instance:algorithm	1.623333
algorithm	ants	2.3752055	255	ants:algorithm	1.623333
algorithm	dummy	2.6898630	250	dummy:algorithm	1.623333
localsearch	instance	2.7838356	250	instance:localsearch	1.576667
localsearch	algorithm	2.9343607	242	algorithm:localsearch	1.576667
localsearch	dummy	3.2936301	235	dummy:localsearch	1.576667
localsearch	ants	3.4852055	231	ants:localsearch	1.576667

For this example we aggregate bidirectional (`param1:param2` and `param2:param2`) interactions, this is done given that we assume that the hierarchy of the interaction is not well represented in the forest and this should be analyzed separately. We also filter interactions using the reference parameter to remove all not relevant interactions. Once this process is done, the importance of most relevant interactions is summarized.

Relevant parameter interactions for the long benchmark:

```
kable(interactions_frame) %>% kable_styling(latex_options="scale_down")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth

Relevant parameter interactions for the short benchmark:

```
kable(interactions_frame_low) %>% kable_styling(latex_options="scale_down")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth
ants	localsearch	1.037671	523	localsearch:ants	1.613333

The long scenario filters out all interactions, while the short one detects one relavant interaction. These seem to be reasonable interactions based on the scenario. The fact that only the short benchmark has relevant interactions might indicate that the scenario is a bit more challenging to configure due to the tighter cut off time. Parameters might require values that increment intensification in order to solve well the instances.

2. Configuration ranking quartile as dependent variable

In this example we use the *ranking quartile as the dependent variable* when training, thus the trained random forest predicts the mean ranking quartile of a configuration. As a personal note, I think this makes the modeling task more a classification than regression task, thus the model was trained for **classification** instead of regression.

2.1 All data model

In this model all irace data execution was used for training. Note that due to the focused nature of irace data (given model convergence) there might be contradicting or not consistent interactions in the data.

The parameter important measures of the long benchmark:

```
load("model_data/model-acotsp1000-4500-qranking.Rdata")
importance_frame = model$importance_frame
important_parameters = model$important_parameters
full_interactions_frame = model$full_interactions_frame
interactions_frame = model$interactions_frame
kable(importance_frame[order(importance_frame[,"mean_min_depth"]),])  %>%
      kable_styling(latex_options="scale_down")
```

	variable	mean_min_depth	no_of_nodes	accuracy_decrease	gini_decrease	no_of_trees	times_a_root	p_value
3	ants	2.066667	31678	0.2141229	992.5850	300	61	0
4	beta	2.090000	34600	0.1647756	1117.6224	300	34	0
11	q0	2.135167	22324	0.1995098	760.5582	299	56	1
13	rho	2.280000	33248	0.1820312	1064.9947	300	9	0
2	alpha	2.360000	34114	0.1942330	1084.8124	300	12	0
10	nnls	2.453333	27741	0.1996888	868.7375	300	29	0
6	dummy	3.433333	25221	0.1567722	780.6749	300	0	0
1	algorithm	3.606667	7320	0.1007625	215.2854	300	44	1
12	rasrank	3.726667	5243	0.0186836	152.7772	300	11	1
9	localsearch	3.820000	7465	0.0731146	211.6164	300	24	1
8	instance	4.096667	73213	0.0059385	1210.2186	300	0	0
5	dlb	4.490000	5848	0.0670083	164.0032	300	19	1
7	elitistants	6.123333	4991	0.0126435	131.8775	300	1	1

The parameter important measures of the short benchmark:

```
load("model_data/model-acotsp1000-4500-low-qranking.Rdata")
importance_frame_low = model$importance_frame
important_parameters_low = model$important_parameters
full_interactions_frame_low = model$full_interactions_frame
interactions_frame_low = model$interactions_frame
kable(importance_frame_low[order(importance_frame_low[,"mean_min_depth"]),])  %>%
      kable_styling(latex_options="scale_down")
```

	variable	mean_min_depth	no_of_nodes	accuracy_decrease	gini_decrease	no_of_trees	times_a_root	p_value
4	beta	1.933333	21584	0.2438364	687.58408	300	61	0
13	rho	2.270000	19862	0.2556695	617.59206	300	45	0
2	alpha	2.526667	21442	0.1900077	678.25099	300	17	0
3	ants	2.623333	19924	0.2151947	624.15030	300	28	0
10	nnls	3.016667	17387	0.2291349	538.46669	300	14	0
9	localsearch	3.200000	4066	0.1547968	133.23288	300	47	1
1	algorithm	3.710000	6350	0.1879718	201.39472	300	22	1
8	instance	3.823333	39968	-0.0083003	542.53580	300	0	0
12	rasrank	3.866667	6298	0.0719390	202.48866	300	9	1
6	dummy	3.956667	15331	0.1842497	462.62918	300	1	0
7	elitistants	4.206667	3205	0.0297548	106.30968	300	20	1
5	dlb	4.340000	3122	0.1206997	97.42264	300	26	1
11	q0	4.396667	4642	0.0439562	142.60750	300	10	1

When compared to the model trained to predict ranking, the best parameters are different in a way that numerical parameters seem to be favored. This is interesting, and it might be worth to investigate why this effect is produced. The reason could be either that the use of the quartiles as dependent variable changes the importance of parameters for prediction or it might be linked to the classification training applied in this experiment.

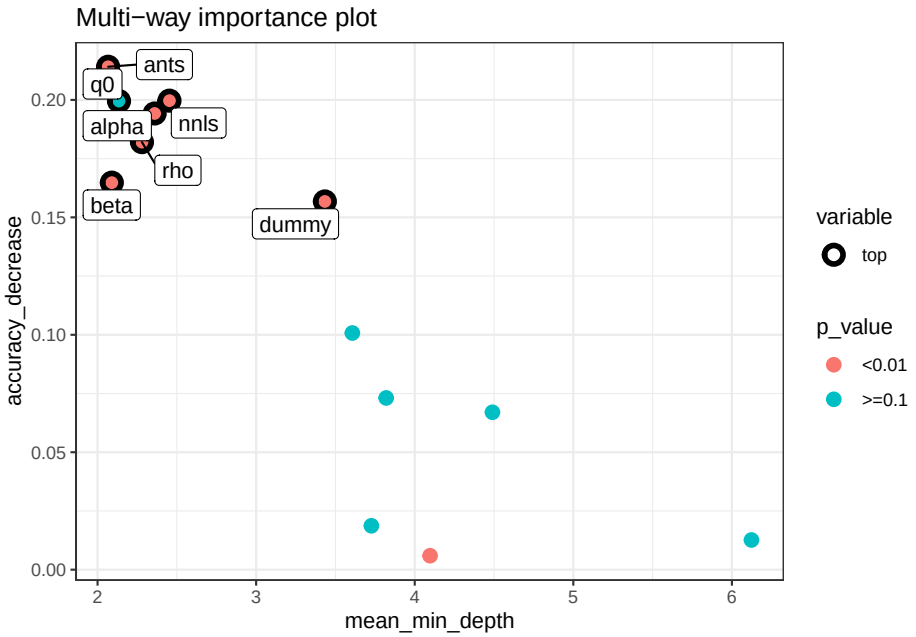
The fact that localsearch or algorithm was not ranked higher in importance indicates that the classification of configurations in quartiles maybe be masking the effects of these parameters, and thus, it might be an adequate method to assess parameter importance post-execution. This could be also due to the incomplete nature of the data.

We plot importance as above to visualize how the mse increase and mean min depth are related:

Importance of the long benchmark:

```
plot_multi_way_importance(importance_frame, size_measure = "p_value",
                          y_measure="accuracy_decrease", x_measure="mean_min_depth",
                          no_of_labels=7)
```

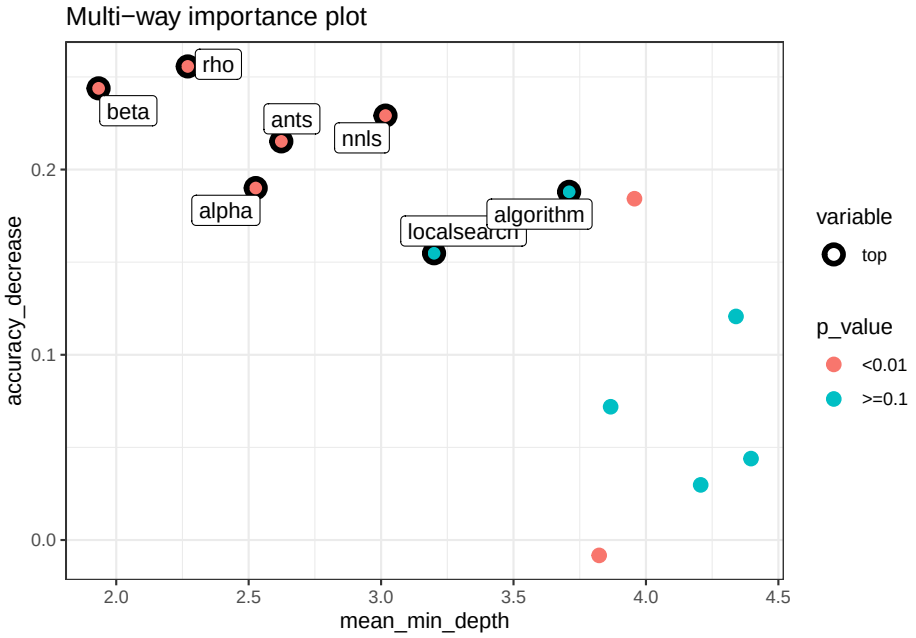
```
## Warning: Using alpha for a discrete variable is not advised.
```



Importance of the short benchmark:

```
plot_multi_way_importance(importance_frame_low, size_measure = "p_value",
                          y_measure="accuracy_decrease", x_measure="mean_min_depth",
                          no_of_labels=7)
```

```
## Warning: Using alpha for a discrete variable is not advised.
```

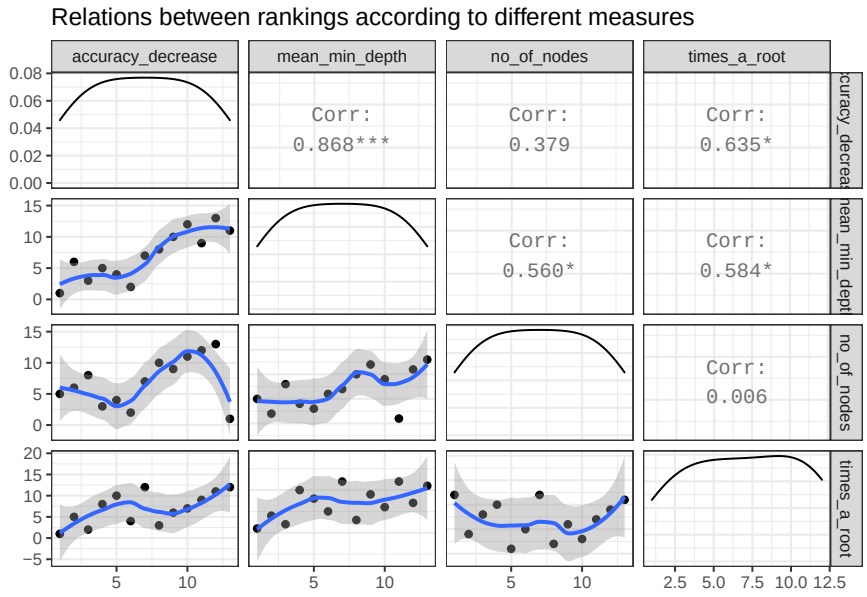


We visualize the relationship between importance measures, this could help to understand which indicator is more suited or can be used as a complement of other.

The long scenario:

```
plot_importance_rankings(importance_frame, measures=c("accuracy_decrease", "mean_min_depth",
                                                       "no_of_nodes", "times_a_root"))
```

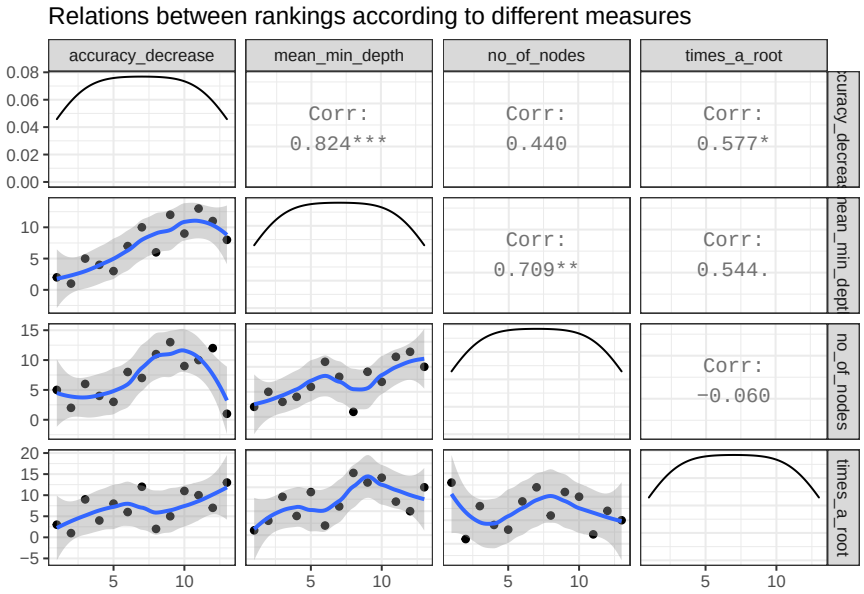
```
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```



The short scenario

```
plot_importance_rankings(importance_frame_low, measures=c("accuracy_decrease", "mean_min_depth",
                                                       "no_of_nodes", "times_a_root"))
```

```
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```



These plots showing that the mean min depth is more related the accuracy decrease than in the previous case to the mse_increase. This is in line with the idea that the mean min depth has a classification persperctive of importance.

We perform an analysis of conditional parameters importance. After this process the most important 5 parameters are detected.

Important parameters for long scenario:

```
print(important_parameters)
```

```
## [1] "ants"  "beta"  "rho"   "alpha" "dummy"
```

Important parameters for short scenario:

```
print(important_parameters_low)
```

```
## [1] "beta"  "rho"   "alpha" "ants"  "nnls"  "dummy"
```

In this case the most important parameters are detected as the numerical parameters alpha, beta, rho and ants in both benchmarks.

We continue to analyze the interaction. Given tha analysis above, if interactions with q0 or nnls are detected as important one could consider them as interactions with algorithm and localsearch respectively.

Parameter interaction importance for the long scenario:

```
kable(full_interactions_frame)  %>%  kable_styling(latex_options="scale_down")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth
beta	beta	1.058621	290	beta:beta	1.366667
alpha	beta	1.161690	289	beta:alpha	1.420000
dummy	beta	2.298483	286	beta:dummy	3.073333
ants	beta	1.584310	285	beta:ants	1.020000
rho	beta	1.573586	284	beta:rho	1.750000
ants	rho	2.114736	277	rho:ants	1.020000
rho	rho	1.997494	277	rho:rho	1.750000
alpha	rho	1.955126	276	rho:alpha	1.420000
beta	rho	2.099954	276	rho:beta	1.366667
beta	ants	2.091207	275	ants:beta	1.366667
dummy	rho	2.688621	275	rho:dummy	3.073333
alpha	ants	2.100276	274	ants:alpha	1.420000
dummy	ants	3.038207	274	ants:dummy	3.073333
alpha	alpha	2.192218	273	alpha:alpha	1.420000
ants	ants	2.474862	273	ants:ants	1.020000
beta	alpha	2.233598	273	alpha:beta	1.366667
ants	alpha	2.457023	271	alpha:ants	1.020000
rho	ants	2.460690	270	ants:rho	1.750000
dummy	alpha	3.304586	269	alpha:dummy	3.073333
rho	alpha	2.791448	266	alpha:rho	1.750000
beta	dummy	3.731621	211	dummy:beta	1.366667
dummy	dummy	4.651759	203	dummy:dummy	3.073333
rho	dummy	3.993138	203	dummy:rho	1.750000
alpha	dummy	4.083586	202	dummy:alpha	1.420000
ants	dummy	4.147241	200	dummy:ants	1.020000

Parameter interaction importance for the short scenario:

```
kable(full_interactions_frame_low) %>% kable_styling(latex_options="scale_down")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth
alpha	beta	1.285714	280	beta:alpha	1.850000
beta	beta	1.533071	278	beta:beta	1.180000
rho	beta	1.504500	278	beta:rho	1.760000
nnls	beta	2.098286	276	beta:nnls	2.200000
beta	rho	1.753191	273	rho:beta	1.180000
dummy	beta	3.187179	273	beta:dummy	3.423333
nnls	rho	2.078190	273	rho:nnls	2.200000
ants	beta	2.243000	272	beta:ants	1.870000
ants	rho	1.888286	271	rho:ants	1.870000
alpha	rho	1.968333	270	rho:alpha	1.850000
dummy	rho	3.178429	268	rho:dummy	3.423333
rho	rho	2.471286	268	rho:rho	1.760000
rho	alpha	2.610667	264	alpha:rho	1.760000
beta	ants	2.574000	262	ants:beta	1.180000
rho	ants	2.699000	262	ants:rho	1.760000
alpha	alpha	3.106417	261	alpha:alpha	1.850000
alpha	ants	2.716405	261	ants:alpha	1.850000
nnls	alpha	3.106417	261	alpha:nnls	2.200000
beta	alpha	2.891905	260	alpha:beta	1.180000
dummy	ants	3.630238	260	ants:dummy	3.423333
ants	ants	3.351214	259	ants:ants	1.870000
dummy	alpha	3.870024	258	alpha:dummy	3.423333
ants	alpha	3.301488	255	alpha:ants	1.870000
nnls	ants	3.542262	255	ants:nnls	2.200000
nnls	nnls	3.678679	253	nnls:nnls	2.200000
alpha	nnls	3.337821	249	nnls:alpha	1.850000
beta	nnls	3.198143	248	nnls:beta	1.180000
rho	nnls	3.355286	248	nnls:rho	1.760000
ants	nnls	4.035464	243	nnls:ants	1.870000
dummy	nnls	4.442214	242	nnls:dummy	3.423333
beta	dummy	3.774929	199	dummy:beta	1.180000
alpha	dummy	4.442333	186	dummy:alpha	1.850000
rho	dummy	4.591714	184	dummy:rho	1.760000
nnls	dummy	4.658952	182	dummy:nnls	2.200000
dummy	dummy	5.354762	180	dummy:dummy	3.423333
ants	dummy	4.927595	173	dummy:ants	1.870000

For this example we aggregate bidirectional (`param1:param2` and `param2:param2`) interactions, this is done given that we assume that the hierarchy of the interaction is not well represented in the forest and this should be analyzed separately. We also filter interactions using the reference parameter to remove all not relevant interactions. Once this process is done, the importance of most relevant interactions is summarized.

Relevant parameter interactions for the long benchmark:

```
kable(interactions_frame) %>% kable_styling(latex_options="scale_down")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth
alpha	beta	1.16169	562	beta:alpha	1.42

Relevant parameter interactions for the short benchmark:

```
kable(interactions_frame_low) %>% kable_styling(latex_options="scale_down")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth
alpha	beta	1.285714	540	beta:alpha	1.85
rho	beta	1.504500	551	beta:rho	1.76
nnls	beta	2.098286	524	beta:nnls	2.20

3.Configuration performance as dependent variable

In this example the ACOTSP data is used to predict solution quality similarly of how it is used in SMAC and GGA++. The intuition is that in these models instance is probably the best predictor of the quality due to the different sizes that compose the instance set and the nature of the configuration objective (tour length).

Parameter importance for the long benchmark:

```
load("model_data/model-acotsp1000-4500-performance.Rdata")
importance_frame = model$importance_frame
full_interactions_frame = model$full_interactions_frame
interactions_frame = model$interactions_frame
important_parameters = model$important_parameters
kable(importance_frame[order(importance_frame[,"mean_min_depth"]),]) %>%
      kable_styling(latex_options="scale_down")
```

	variable	mean_min_depth	no_of_nodes	mse_increase	node_purity_increase	no_of_trees	times_a_root	p_value
8	instance	1.120000	51860	8.291685e+13	2.129612e+17	300	89	0
10	nnls	2.170000	47474	3.041422e+12	5.721884e+15	300	61	0
3	ants	2.406667	55066	3.596844e+11	1.618010e+15	300	18	0
11	q0	2.686667	42418	4.538077e+11	1.260428e+15	300	13	0
9	localsearch	2.833333	13504	2.311053e+12	4.549491e+15	300	44	1
4	beta	2.873333	58975	3.811337e+10	1.241437e+15	300	7	0
2	alpha	2.933333	61327	6.514147e+10	1.204001e+15	300	8	0
1	algorithm	3.190000	14302	2.202469e+11	6.342228e+14	300	1	1
13	rho	3.210000	55117	3.000252e+10	1.084485e+15	300	5	0
5	dlb	3.396667	8549	2.258994e+12	4.312739e+15	300	49	1
6	dummy	3.926667	42148	-1.385570e+11	5.371808e+14	300	0	0
7	elitistants	3.976667	6690	4.073106e+09	2.558662e+14	300	2	1
12	rasrank	4.923333	5990	-3.624454e+10	1.613447e+14	300	3	1

Parameter importance for the short benchmark:

```
load("model_data/model-acotsp1000-4500-low-performance.Rdata")
importance_frame_low = model$importance_frame
full_interactions_frame_low = model$full_interactions_frame
interactions_frame_low = model$interactions_frame
important_parameters_low = model$important_parameters
kable(importance_frame[order(importance_frame_low[,"mean_min_depth"]),]) %>%
      kable_styling(latex_options="scale_down")
```

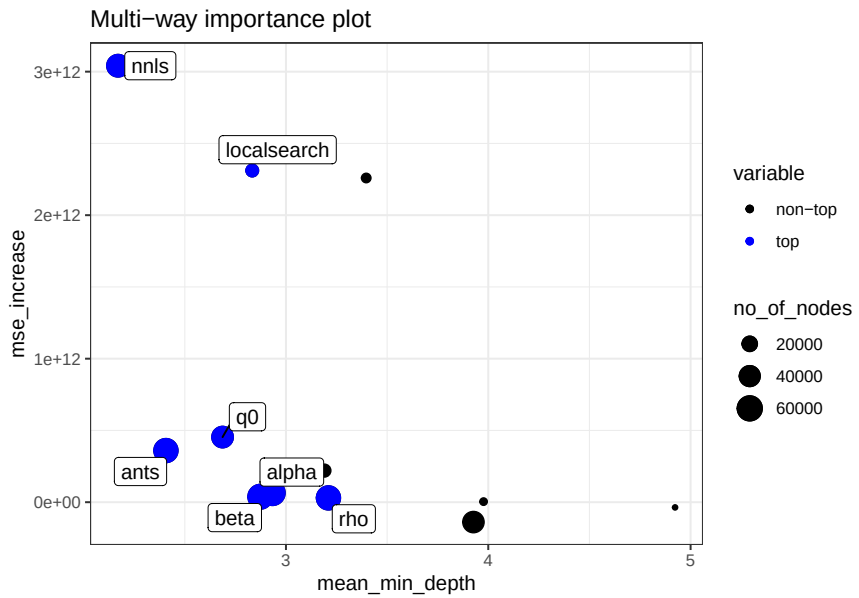
	variable	mean_min_depth	no_of_nodes	mse_increase	node_purity_increase	no_of_trees	times_a_root	p_value
8	instance	1.120000	51860	8.291685e+13	2.129612e+17	300	89	0
2	alpha	2.933333	61327	6.514147e+10	1.204001e+15	300	8	0
9	localsearch	2.833333	13504	2.311053e+12	4.549491e+15	300	44	1
10	nnls	2.170000	47474	3.041422e+12	5.721884e+15	300	61	0
4	beta	2.873333	58975	3.811337e+10	1.241437e+15	300	7	0
3	ants	2.406667	55066	3.596844e+11	1.618010e+15	300	18	0
13	rho	3.210000	55117	3.000252e+10	1.084485e+15	300	5	0
1	algorithm	3.190000	14302	2.202469e+11	6.342228e+14	300	1	1
5	dlb	3.396667	8549	2.258994e+12	4.312739e+15	300	49	1
6	dummy	3.926667	42148	-1.385570e+11	5.371808e+14	300	0	0
7	elitistants	3.976667	6690	4.073106e+09	2.558662e+14	300	2	1
12	rasrank	4.923333	5990	-3.624454e+10	1.613447e+14	300	3	1
11	q0	2.686667	42418	4.538077e+11	1.260428e+15	300	13	0

As expected the instance is the most important variable, the other parameters do not have the same ranking in the different benchmarks.

We plot the importance measures min mean depth and the mse increase to observe better this difference. The instance is removed in the plot so its possible to see more clearly other parameters.

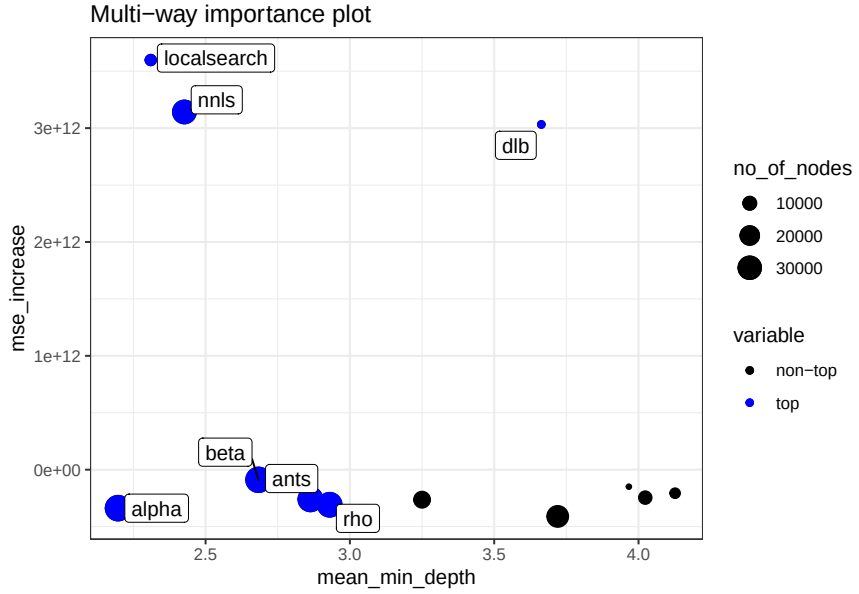
The long benchmark importance:

```
plot_multi_way_importance(importance_frame[importance_frame[,"variable"]!="instance",], size_measure =
                          y_measure="mse_increase", x_measure="mean_min_depth", no_of_labels=7)
```



THe short benchmark importance:

```
plot_multi_way_importance(importance_frame_low[importance_frame_low[,"variable"]!="instance",], size_mea
                          y_measure="mse_increase", x_measure="mean_min_depth", no_of_labels=7)
```



It seems that the values of mean min depth and the mse increase are not correlated, as most important

parameters in one measure, are not the most important in the other.

The localsearch parameter is known to have a strong effect in the performance of ACOTSP and thus, it is not surprising to find it between the most important followd by nnls, a parameter that is conditional to localsearch. It is possible to re-train the model to assess real importance of such conditional parameters, in this way it is possible to understand if the prediction benefits are derived from the root parameter. We perform this analysis filtering importance with the reference parameter and discarding irrelevant conditionals, selecting the 5 most important variables regarding the mean_min_depth.

Filtered important parameters for the long benchmark:

```
print(important_parameters)
```

```
## [1] "instance"    "nnls"        "ants"        "q0"          "localsearch"
## [6] "dummy"
```

Filtered important parameters for the short benchmark:

```
print(important_parameters_low)
```

```
## [1] "instance"    "alpha"       "localsearch" "beta"        "dummy"
```

We use the mean_min_depth measure as reference given that this measure can be extended to assess interactions and run the interaction analysis to find the importance of interactions between the selected important parameters. This is done by extending the definition of mean_min_depth to be calculated in max subtrees in which one variable is root.

For the long benchmark:

```
kable(full_interactions_frame)  %>%  kable_styling(latex_options="scale_down")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth
instance	localsearch	0.5604027	298	localsearch:instance	0.9766667
nnls	localsearch	1.5402685	298	localsearch:nnls	1.6200000
instance	instance	0.6863758	296	instance:instance	0.9766667
nnls	nnls	1.7695302	296	nnls:nnls	1.6200000
dummy	localsearch	1.5945302	295	localsearch:dummy	2.9000000
dummy	nnls	1.8338255	295	nnls:dummy	2.9000000
ants	localsearch	1.3676510	294	localsearch:ants	1.9733333
instance	nnls	0.9484564	294	nnls:instance	0.9766667
instance	q0	0.8595526	294	q0:instance	0.9766667
q0	instance	1.4130201	294	instance:q0	1.9233333
ants	nnls	1.3953020	293	nnls:ants	1.9733333
q0	localsearch	1.4562081	293	localsearch:q0	1.9233333
dummy	instance	2.2369799	292	instance:dummy	2.9000000
q0	nnls	1.5401342	292	nnls:q0	1.9233333
ants	instance	1.7563423	291	instance:ants	1.9733333
q0	q0	1.7022036	291	q0:q0	1.9233333
dummy	q0	1.9875615	290	q0:dummy	2.9000000
ants	q0	1.7024497	289	q0:ants	1.9733333
dummy	ants	2.3228188	286	ants:dummy	2.9000000
localsearch	nnls	2.7379866	286	nnls:localsearch	1.8866667
instance	ants	1.2963087	285	ants:instance	0.9766667
ants	ants	2.0751678	284	ants:ants	1.9733333
localsearch	localsearch	3.3430872	282	localsearch:localsearch	1.8866667
q0	ants	2.1734899	281	ants:q0	1.9233333
nnls	instance	3.9500336	255	instance:nnls	1.6200000
ants	dummy	2.3618345	252	dummy:ants	1.9733333
instance	dummy	2.0933781	252	dummy:instance	0.9766667
dummy	dummy	3.0237584	250	dummy:dummy	2.9000000
localsearch	instance	4.4595973	250	instance:localsearch	1.8866667
q0	dummy	2.7155034	241	dummy:q0	1.9233333
nnls	ants	5.3812081	222	ants:nnls	1.6200000
localsearch	ants	6.0107383	216	ants:localsearch	1.8866667
nnls	q0	6.2528076	207	q0:nnls	1.6200000
localsearch	q0	6.8712416	199	q0:localsearch	1.8866667
nnls	dummy	6.8114541	110	dummy:nnls	1.6200000
localsearch	dummy	7.4600447	92	dummy:localsearch	1.8866667

For the short benchmark:

```
kable(full_interactions_frame_low)  %>%  kable_styling(latex_options="scale_down")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth
beta	localsearch	1.129825	285	localsearch:beta	1.573333
alpha	localsearch	1.142339	284	localsearch:alpha	1.740000
instance	localsearch	1.189941	283	localsearch:instance	1.653333
beta	alpha	1.075825	282	alpha:beta	1.573333
instance	beta	1.180678	281	beta:instance	1.653333
alpha	alpha	1.266725	280	alpha:alpha	1.740000
alpha	beta	1.466175	276	beta:alpha	1.740000
beta	beta	1.480211	276	beta:beta	1.573333
instance	alpha	1.589591	275	alpha:instance	1.653333
instance	instance	1.366550	275	instance:instance	1.653333
beta	instance	1.363556	274	instance:beta	1.573333
dummy	localsearch	1.583275	274	localsearch:dummy	1.790000
alpha	instance	1.476351	273	instance:alpha	1.740000
localsearch	localsearch	2.115556	271	localsearch:localsearch	1.473333
beta	dummy	1.727251	269	dummy:beta	1.573333
dummy	alpha	1.867556	269	alpha:dummy	1.790000
instance	dummy	1.661520	268	dummy:instance	1.653333
dummy	beta	2.002526	267	beta:dummy	1.790000
dummy	instance	1.892538	262	instance:dummy	1.790000
dummy	dummy	2.086550	260	dummy:dummy	1.790000
alpha	dummy	2.006784	259	dummy:alpha	1.740000
localsearch	alpha	3.107871	242	alpha:localsearch	1.473333
localsearch	instance	3.180632	234	instance:localsearch	1.473333
localsearch	beta	3.517240	233	beta:localsearch	1.473333
localsearch	dummy	3.825263	213	dummy:localsearch	1.473333

We aggregate interactions with their inverse given that for now we are not interested in the direction of the interaction. Once we aggregate to have bidirectional interactions and filter dummy interactions, we get a matrix where the relevant importance of interactions are summarized:

For the long benchmark:

```
kable(interactions_frame) %>% kable_styling(latex_options="scale_down")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth
instance	localsearch	0.5604027	548	localsearch:instance	0.9766667
nnls	localsearch	1.5402685	584	localsearch:nnls	1.6200000

For the short benchmark:

```
kable(interactions_frame_low) %>% kable_styling(latex_options="scale_down")
```

variable	root_variable	mean_min_depth	occurrences	interaction	uncond_mean_min_depth
beta	localsearch	1.129825	518	localsearch:beta	1.573333
alpha	localsearch	1.142339	526	localsearch:alpha	1.740000
instance	localsearch	1.189941	517	localsearch:instance	1.653333