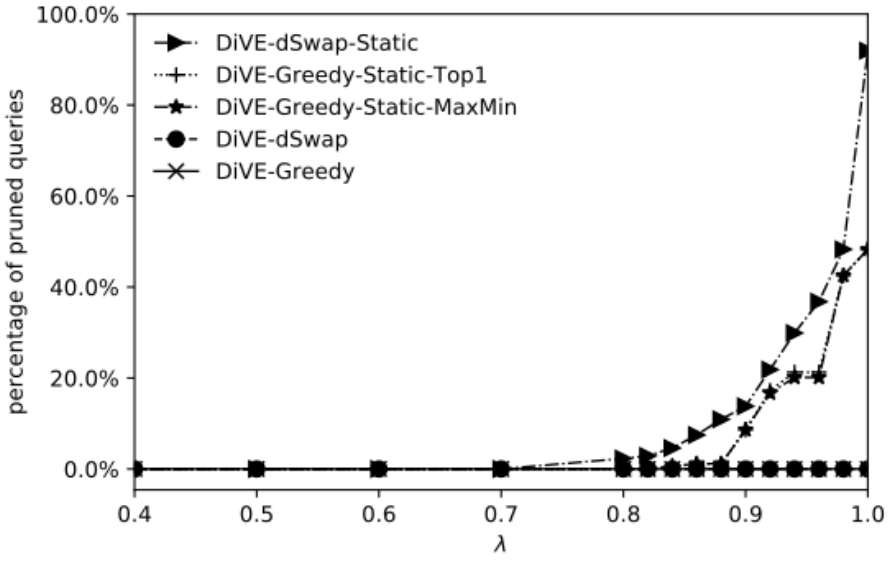
Greedy Static Max-Min vs Greedy Static Top-1

Greedy static Max-min

1. Get two most distant views as the initialization set S.
2. Execute query views of the initialization set and get the maximum importance score.
3. Use √2 as the maximum static bound, this maximum will not be changed until the end.
4. Use maximum importance score from the initialization set as the minimum bound.
5. Compute utility U of all views in X using √2 as the max bound and use actual diversity score.
6. Sort views in X based on the highest utility score, store in the list L.
7. Calculate Umax and Umin
8. Prune all views which have Umax < max(Umin)
9. Execute the first query view in L and calculate the real objective function F(S) as the *current_F(S)*.
10. If the importance score of executed query > current minimum bound, then update the minimum bound.
11. Always repeat step 7 and 8 after the minimum bound is updated.
12. Execute the next query view in L, calculate the real objective function F(S), if this F(S) higher than the current, then replace the current.
13. Repeat step 10 and 12 until there is no F(S) that higher than *current_F(S)*.
14. Go to the next iteration.

Greedy Static Top-1

1. Get two most distant views as the initialization set S.
2. Use √2 as the maximum static bound, this maximum will not be changed until the end.
3. Compute utility U of all views in X using √2 as the max bound and use actual diversity score.
4. Sort views in X based on the highest utility score, store in the list L.
5. Execute the first query view in L and calculate the real objective function F(S) as the *current_F(S)*.
6. If U of views in L < *current_F(S)*, those views will be pruned.
7. Execute the next query view, calculate the real objective function F(S), if this F(S) higher than the current, then replace the current.
8. Repeat step 6 and 7 till there is no F(S) that higher than *current_F(S)*.
9. Go to the next iteration.



The pruning performance of Greedy Max-min and Top-1 is very close. However, still swap has better performance compared to Greedy. The reason why we need swap is because Greedy starts with small number of views in the initialization. Only two views as the set S. Hence, while calculating the utility score of each views in X using static maximum bound and actual diversity score, there are a lot of views have same utility score. Due to of this, the chance of pruning is low.

However, Swap has bigger number of views in the initialization (e.g. five views). While calculating utility score of views in X, using static maximum bound and actual diversity score, the chance of views have same utility score is lower than Greedy.