Assignment2-Exercise3

Jingwen Wei a1671836
Zhuoying Li (a1675725)
Puzhi Yao (a1205593)
Xueyang Wang (a1690260)

Algorithm:

New algorithm for Exercise3 (ComplexHeuristic_3) is based on the Best Algorithm designed in the Exercise 2(ComplexHeuristic_2). The main process is almost the same, but the EA part is modified to dynamic. First do a greedy algorithm to generate an acceptable packing plan. Then improve this packing plan with dynamic EA algorithm until the given time used up. After this, doing a 1 bit flip local search algorithm until the given time used up.
The whole algorithm can be describe like this:

```
Input: tour, item, limit-time
Output: packing plan

Item-value←evaluate(item)
while(capability enough){
      pack up item by order(item-value)
} //greedy
While(given time not finished){
      packingplan = dynamicEA(packingplan);
}//EA
While(given time not finished){
      packingplan = bitFlip(packingplan);
}//LS
Return packingplan
```

The whole idea of this algorithm is using greedy algorithm to get a start packing plan, then use dynamic EA to quickly get closer to the optimal solution, then use bit flip algorithm to make the solution more accurate.

In this algorithm, three different algorithm is combined:

Greedy algorithm:

We evaluate an item by its unit per value minus cost on the way. This is calculate like:

value/weight –cost per time * remain distance/(vmax - weight*(vmax – vmin)/capability).

DynamicEA:

This algorithm is based on EA. The original mutation possibility is like 1/n. But in dynamic EA the possibility is connected to the current not used packing capability. The possibility in dynamic EA is 1/n + remain-capability/capability. Changing the possibility like this is based on an idea that when the remain packing capability is large that means it is farer to the optimal solution so it need more change in the packing plan.

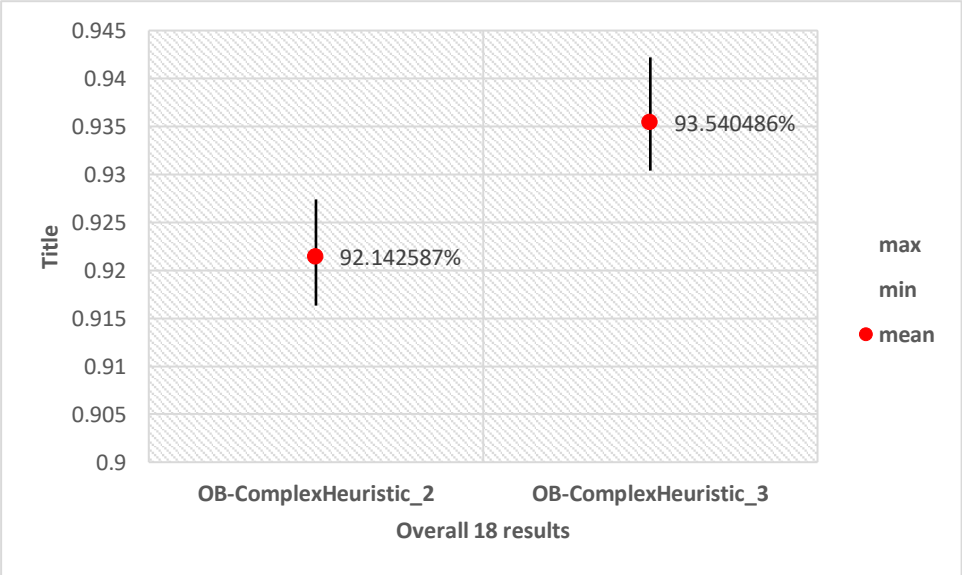
Bit Flip:

This is just the original given bit flip algorithm with a loop to run over and over again until reach the time.

Result:

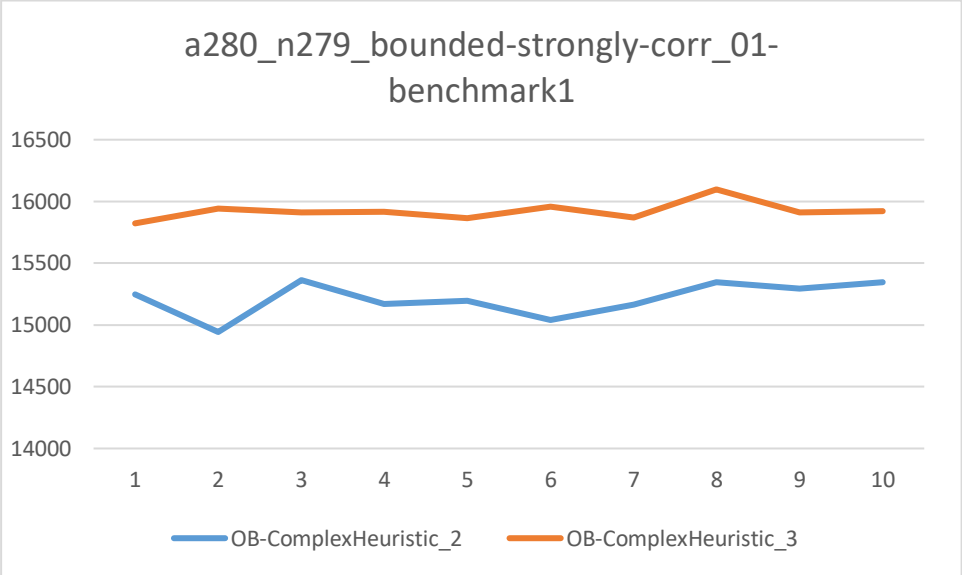
For the nine instance and two benchmark set we have test, we got 18 results in total. Compared the new algorithm (ComplexHeuristic_3) to the best algorithm (ComplexHeuristic_2) in Exercise 2, in 18 results, 10 results are better, 4 results are almost equal and 4 are worse. So in total the new algorithm is better.
And we found that for the instance that is bounded strongly, the new algorithm works better.

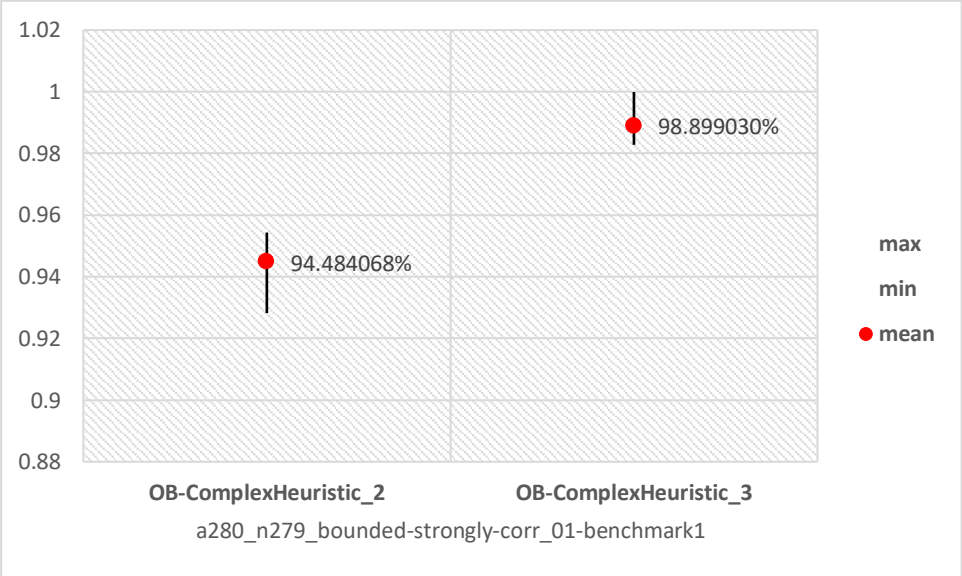
Here is the overall performance for all 18 results: (see independent performance in appendix)



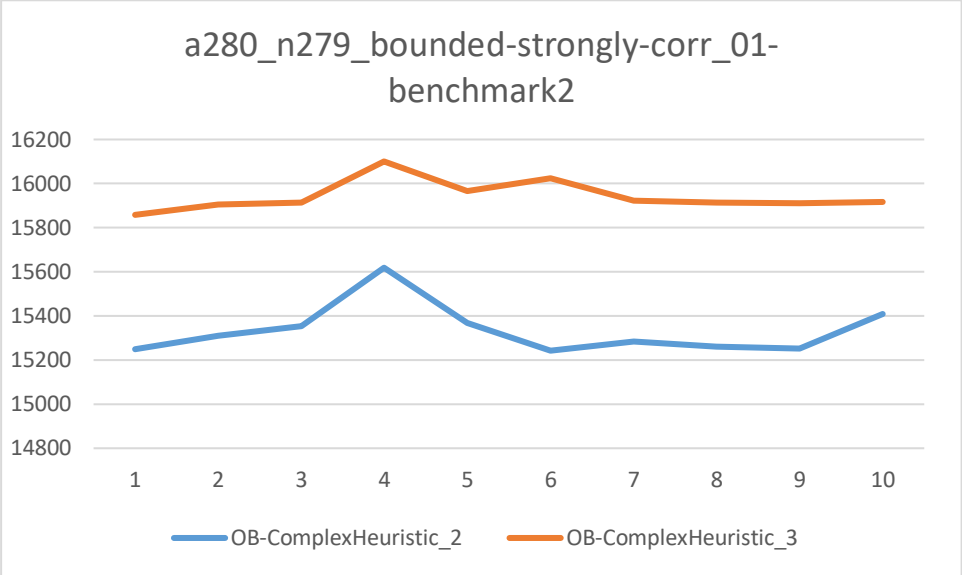
From this graph, we can see that the new algorithm (ComplexHeuristic_3) perform better overall.

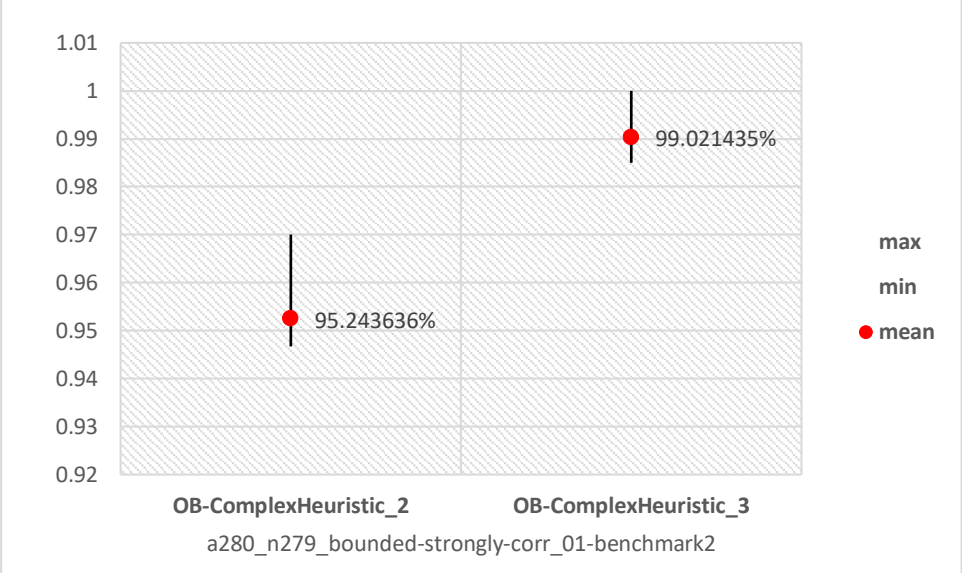
Appendix: The 18 results (ten times run each), scaled performance and the result table.



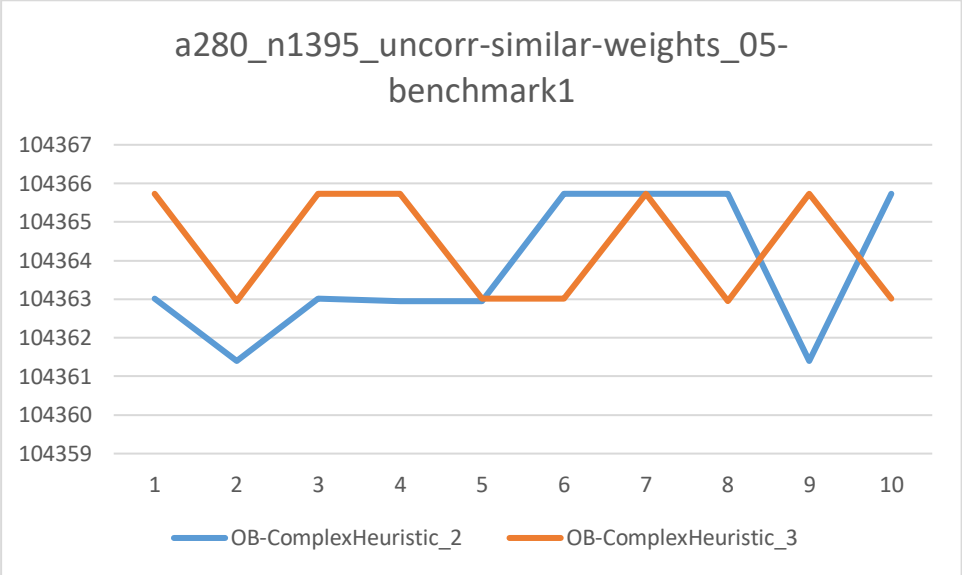


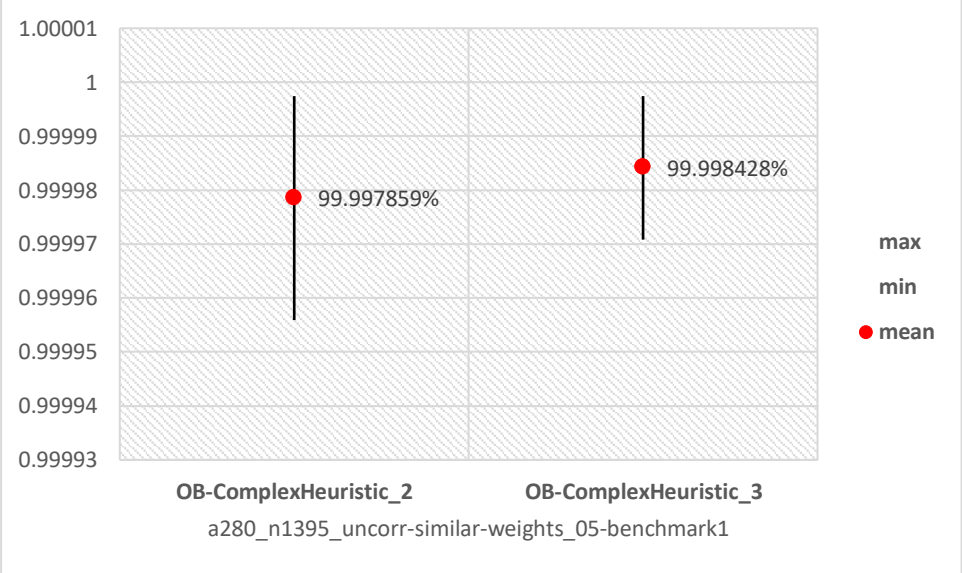
Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	15247.25	15822.71
2	14943.11	15943.56
3	15362.91	15913.41
4	15169.48	15914.71
5	15197.38	15866.76
6	15041.28	15959.22
7	15166.06	15869.03
8	15346.21	16097.52
9	15292.63	15910.12
10	15343.59	15920.51



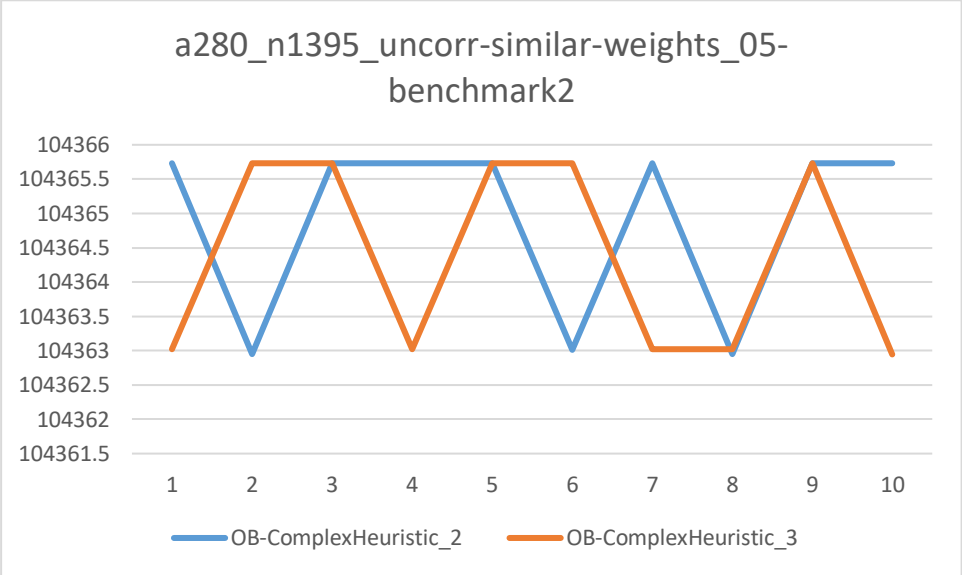


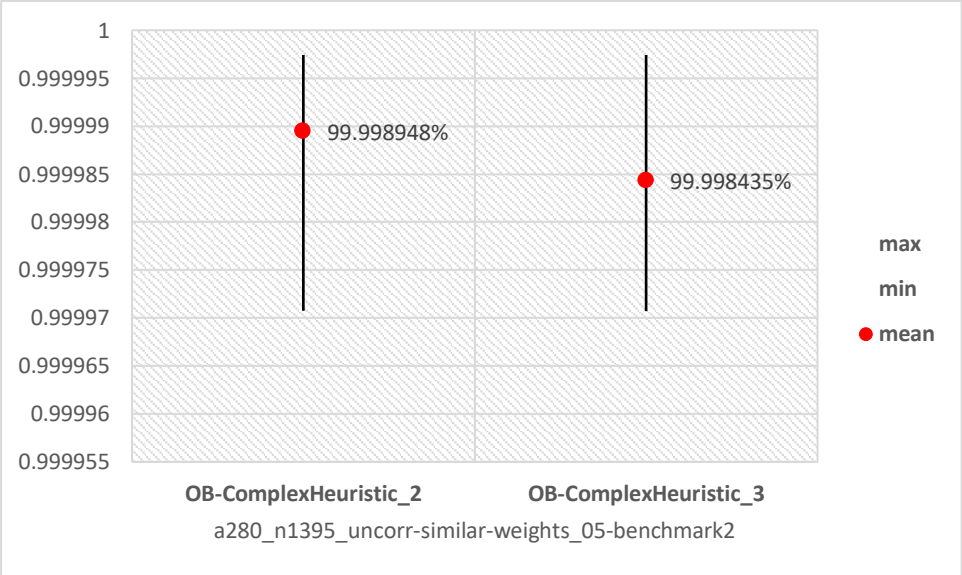
Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	15247.98	15858.19
2	15308.62	15905.02
3	15354.36	15914.8
4	15618.62	16100.37
5	15366.66	15965.66
6	15242.44	16023.94
7	15283.4	15922.42
8	15261.01	15912.56
9	15252.91	15909.6
10	15409.78	15915.61



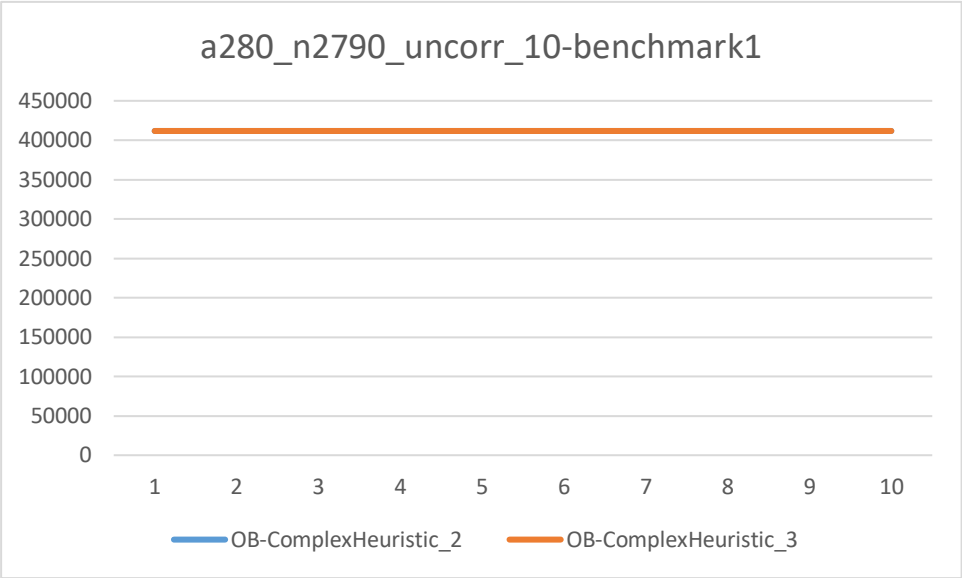


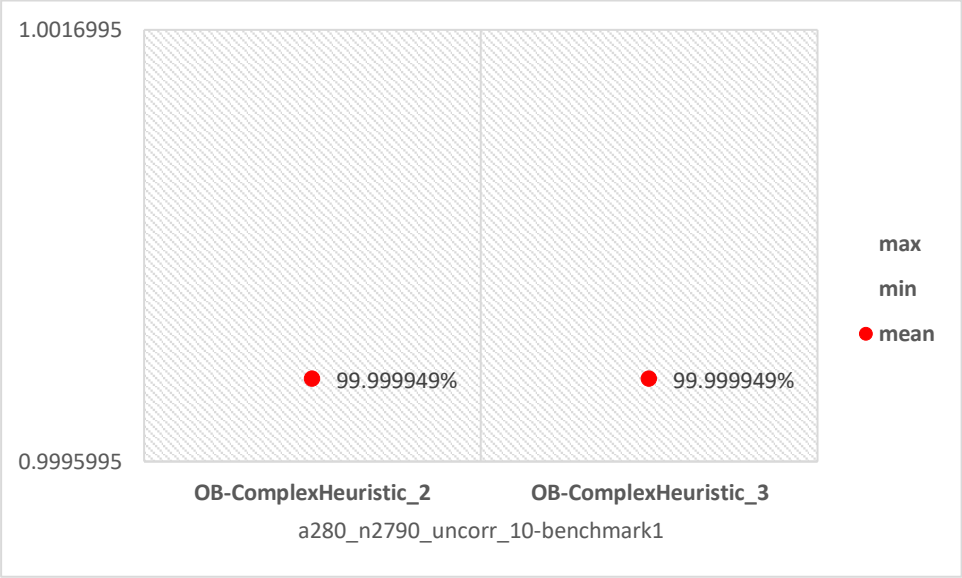
Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	104363	104365.7
2	104361.4	104363
3	104363	104365.7
4	104362.9	104365.7
5	104363	104363
6	104365.7	104363
7	104365.7	104365.7
8	104365.7	104363
9	104361.4	104365.7
10	104365.7	104363



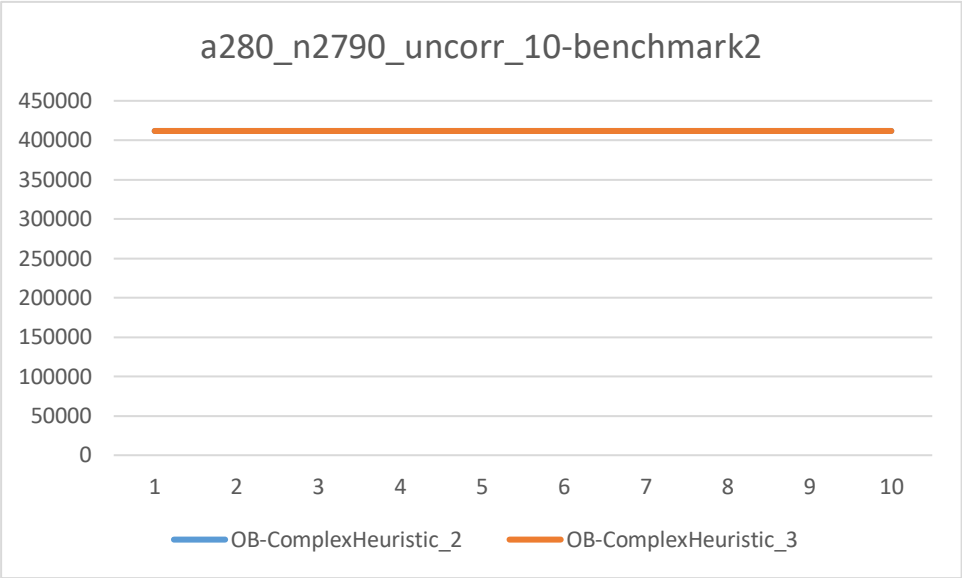


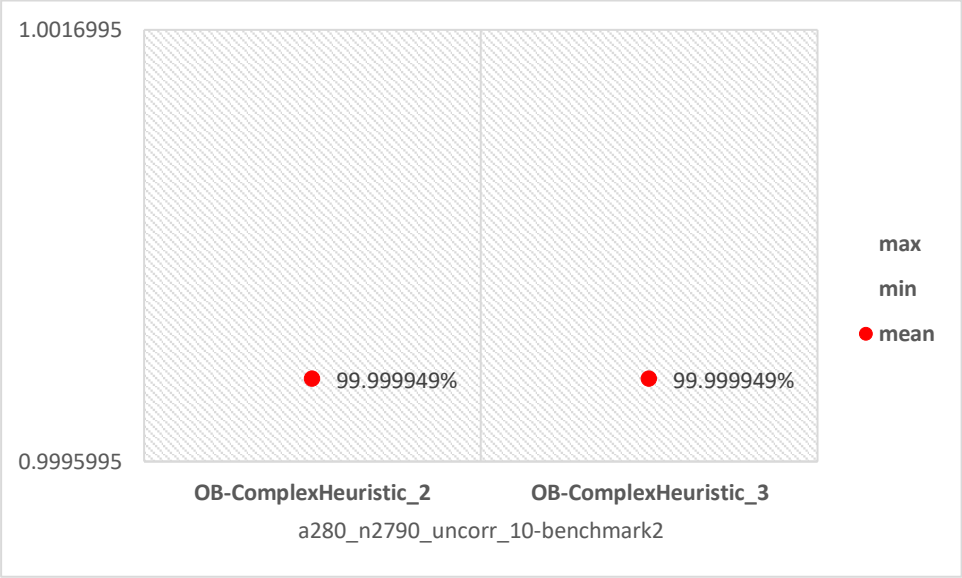
Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	104365.7	104363
2	104363	104365.7
3	104365.7	104365.7
4	104365.7	104363
5	104365.7	104365.7
6	104363	104365.7
7	104365.7	104363
8	104363	104363
9	104365.7	104365.7
10	104365.7	104362.9



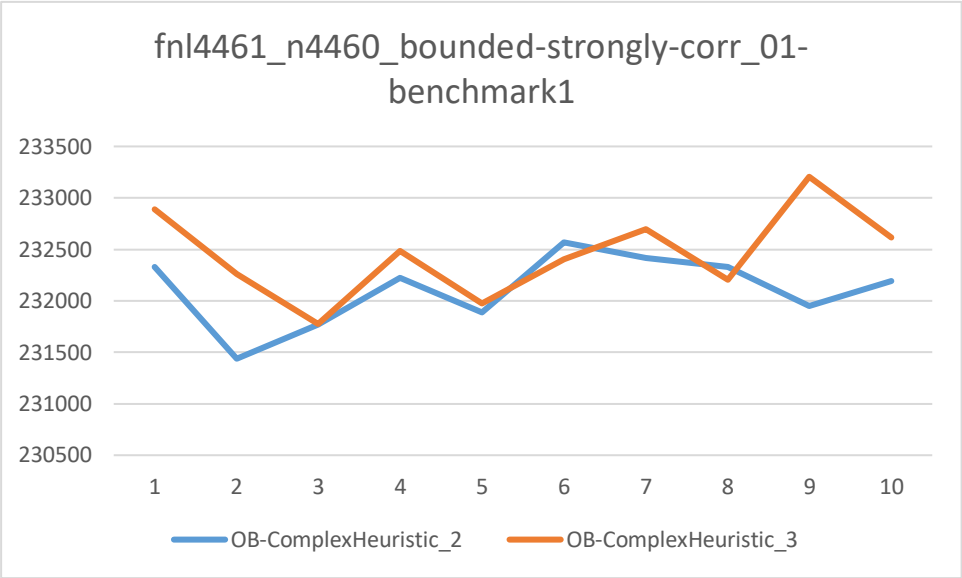


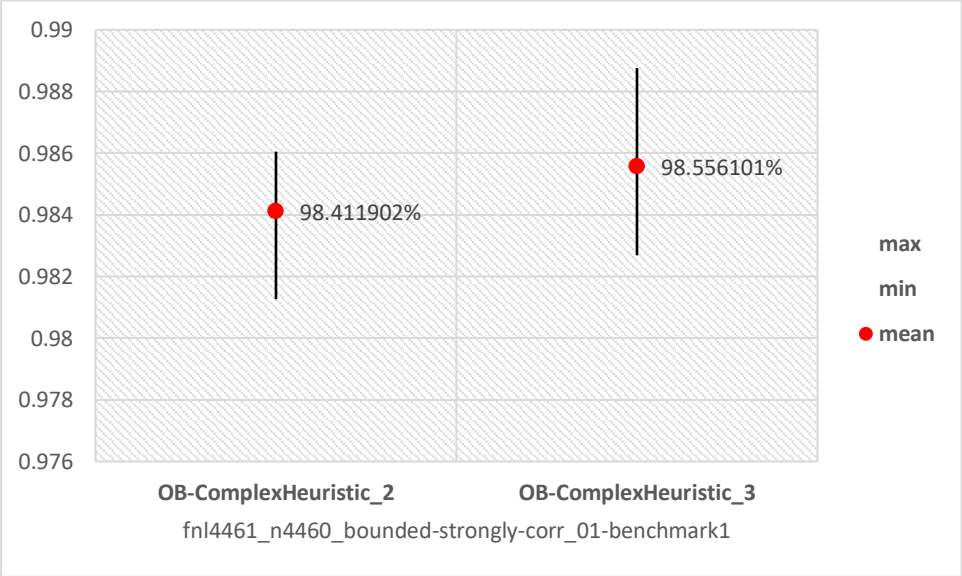
Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	411714.8	411714.8
2	411714.8	411714.8
3	411714.8	411714.8
4	411714.8	411714.8
5	411714.8	411714.8
6	411714.8	411714.8
7	411714.8	411714.8
8	411714.8	411714.8
9	411714.8	411714.8
10	411714.8	411714.8



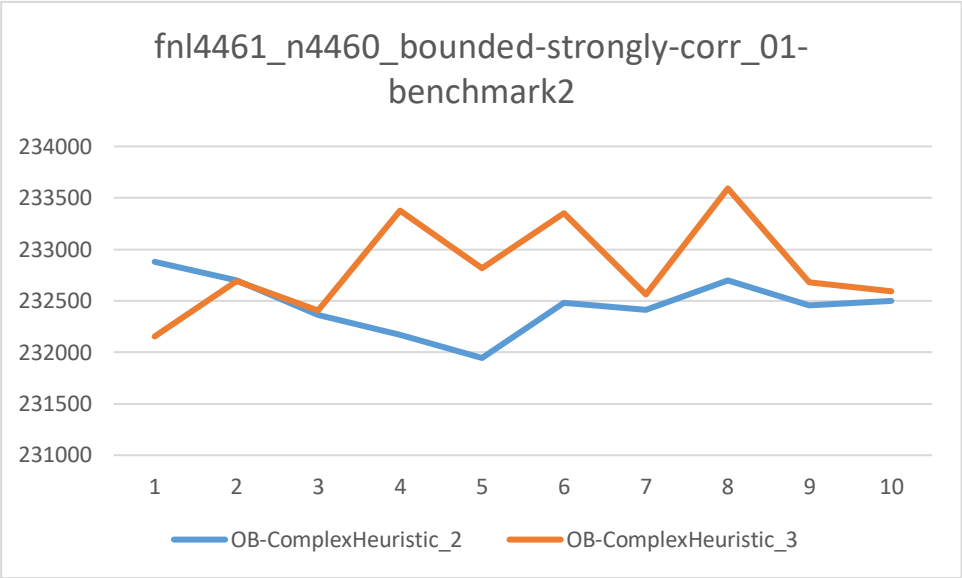


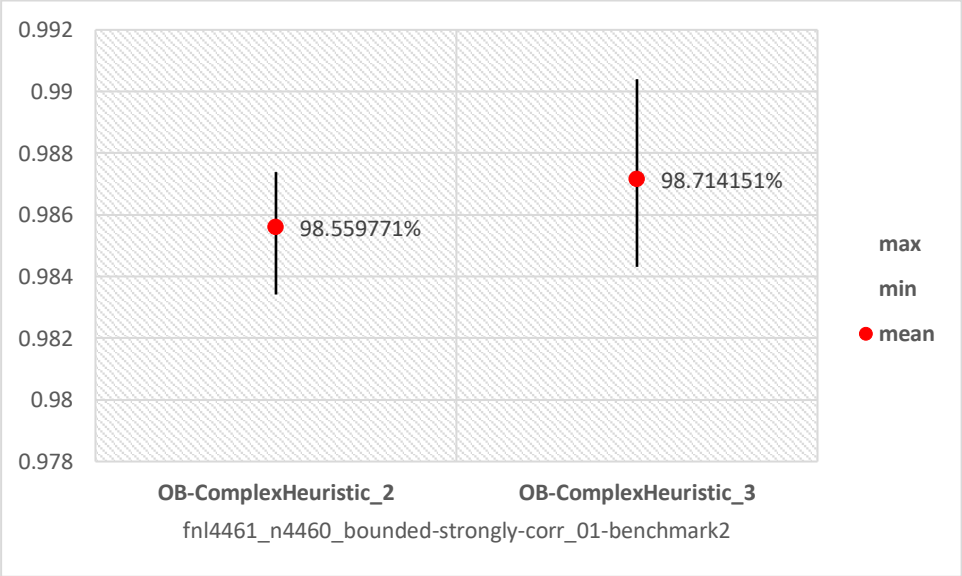
Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	411714.8	411714.8
2	411714.8	411714.8
3	411714.8	411714.8
4	411714.8	411714.8
5	411714.8	411714.8
6	411714.8	411714.8
7	411714.8	411714.8
8	411714.8	411714.8
9	411714.8	411714.8
10	411714.8	411714.8



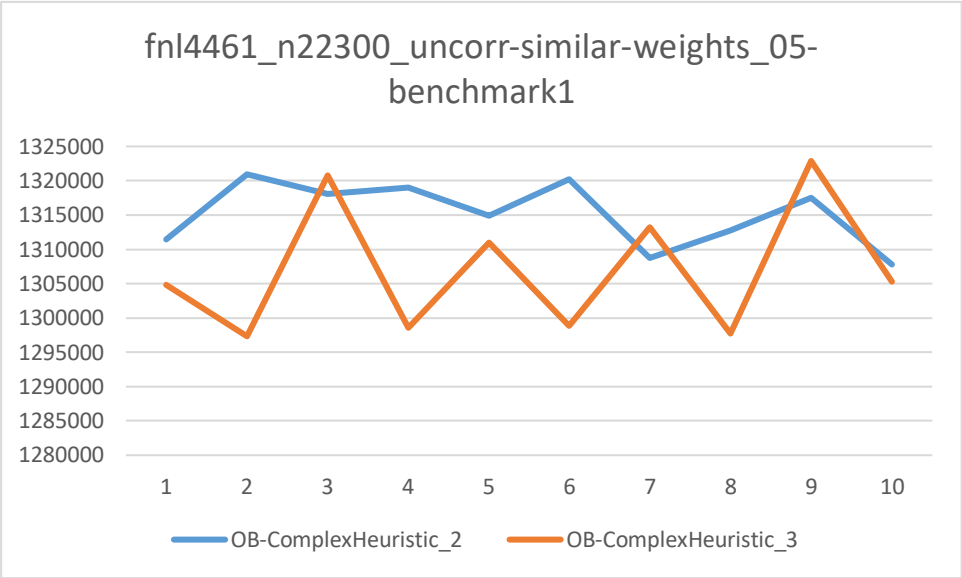


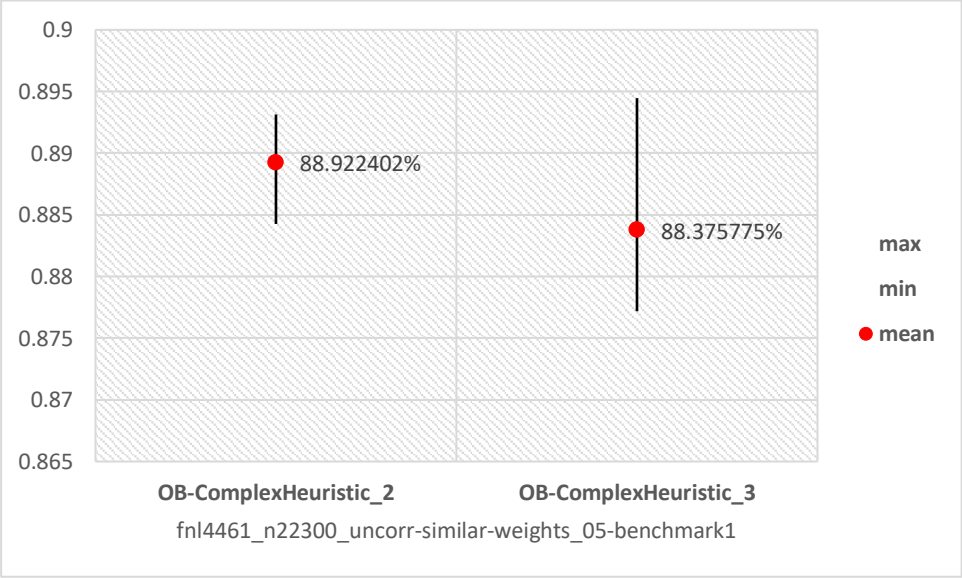
Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	232329.6	232890.7
2	231437.2	232258
3	231767.8	231775
4	232222.9	232481.6
5	231886.6	231975.9
6	232567.6	232404.6
7	232419	232696.4
8	232327.6	232201.8
9	231951.3	233205.1
10	232194.1	232615.7



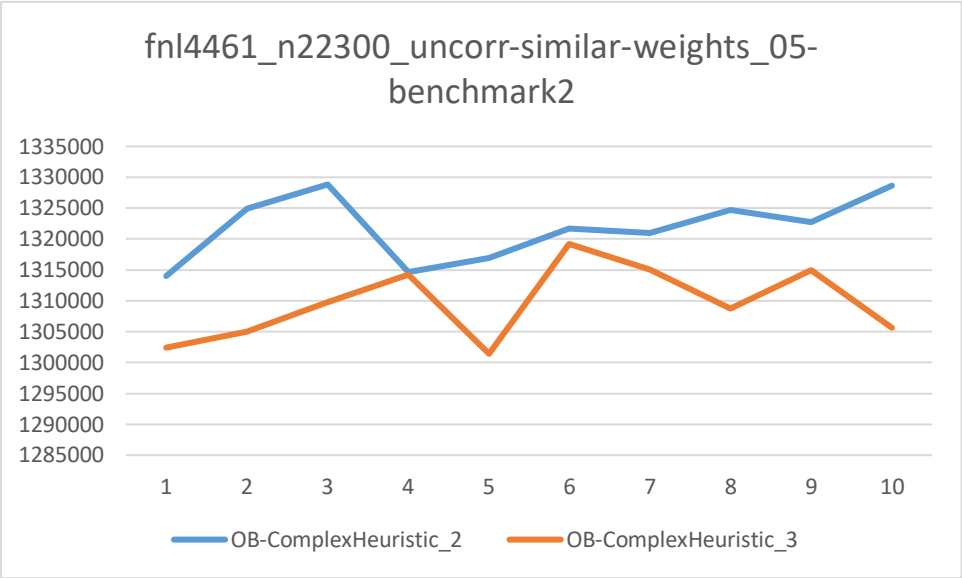


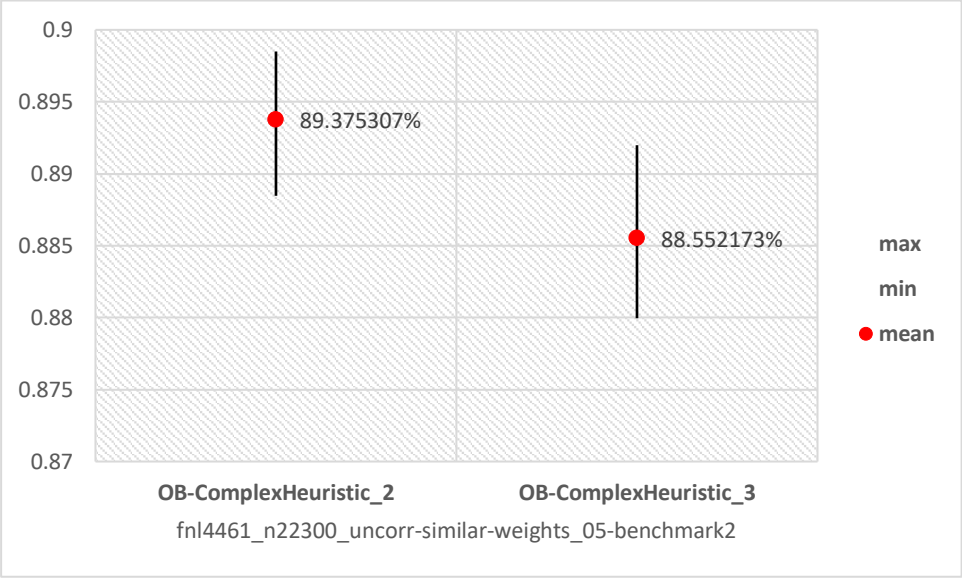
Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	232878.9	232153.8
2	232700.6	232694.9
3	232359.7	232403.6
4	232168.1	233379.2
5	231944.6	232814.4
6	232479.8	233354.2
7	232409.3	232562.9
8	232698.6	233591.9
9	232454.2	232682
10	232497.6	232595.6



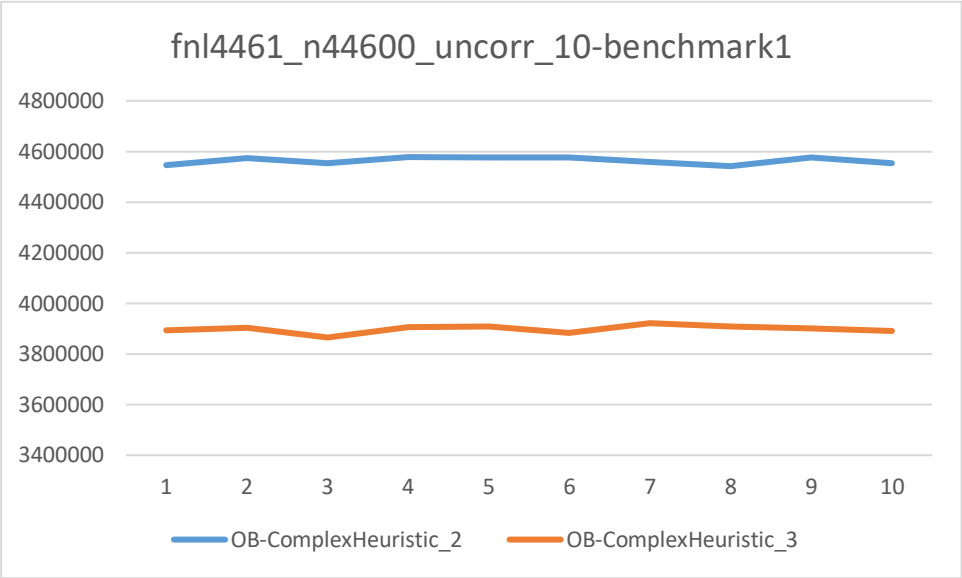


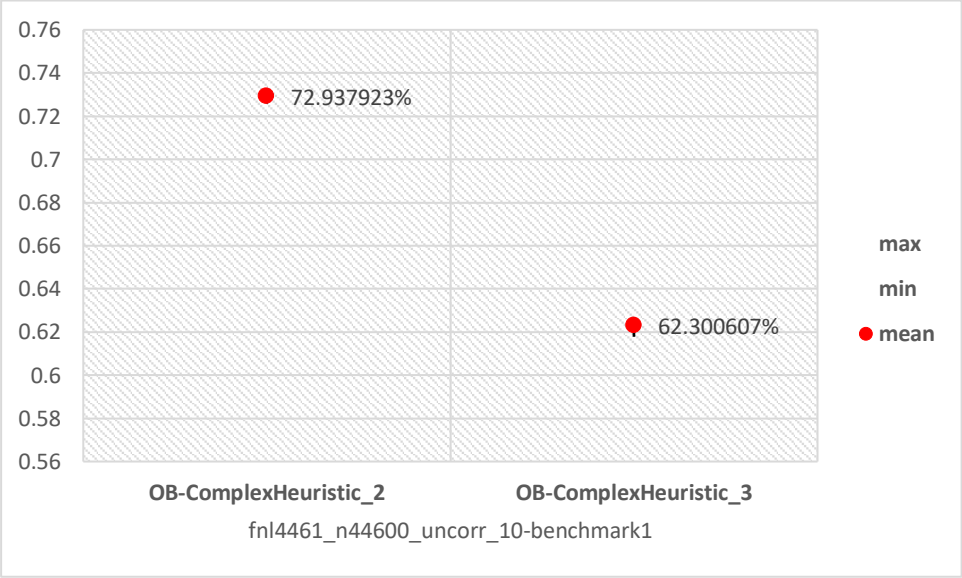
Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	1311443	1304841
2	1320939	1297327
3	1318050	1320747
4	1318965	1298529
5	1314888	1310983
6	1320246	1298894
7	1308732	1313252
8	1312733	1297728
9	1317497	1322876
10	1307792	1305264



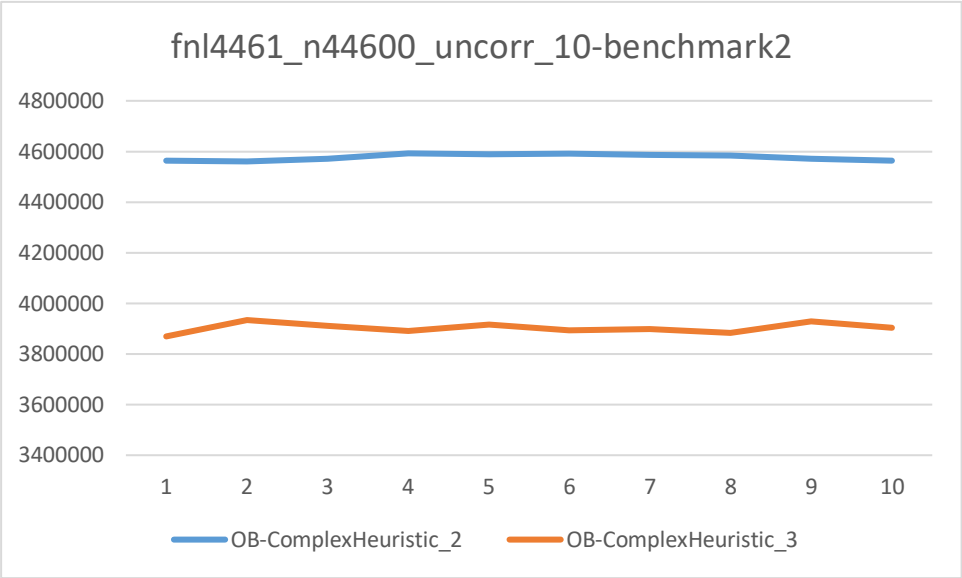


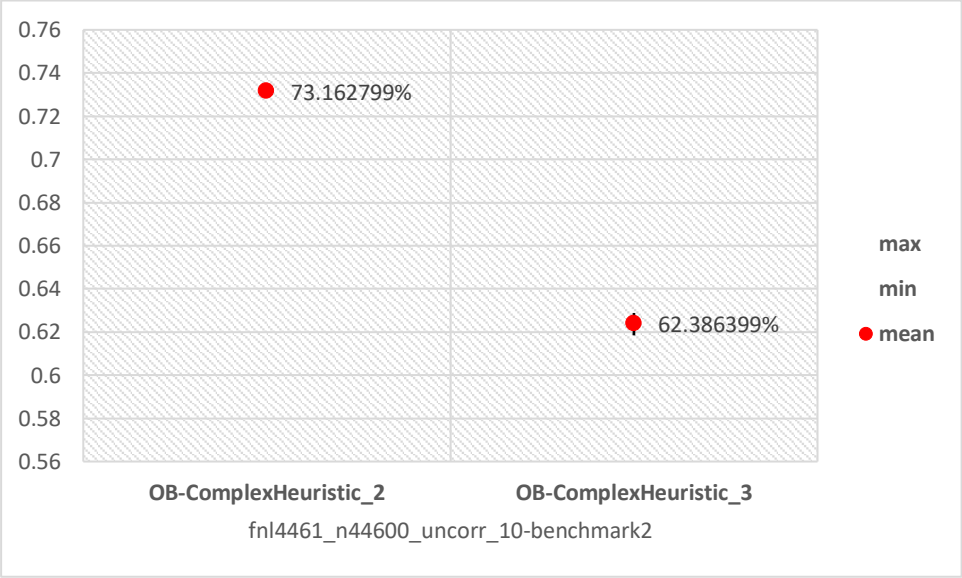
Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	1314018	1302451
2	1324928	1305053
3	1328841	1309751
4	1314694	1314262
5	1316962	1301425
6	1321728	1319206
7	1321003	1315066
8	1324727	1308751
9	1322746	1314966
10	1328622	1305598



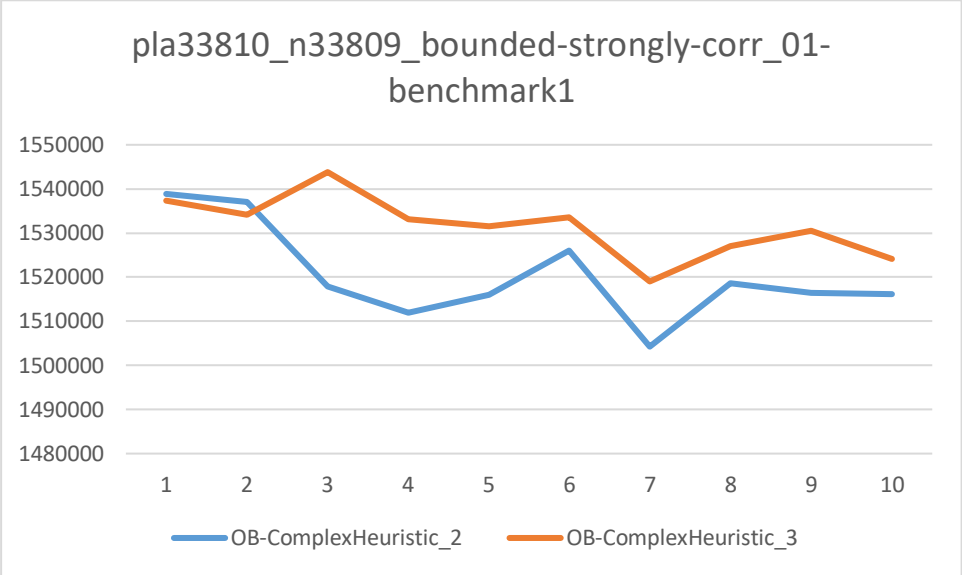


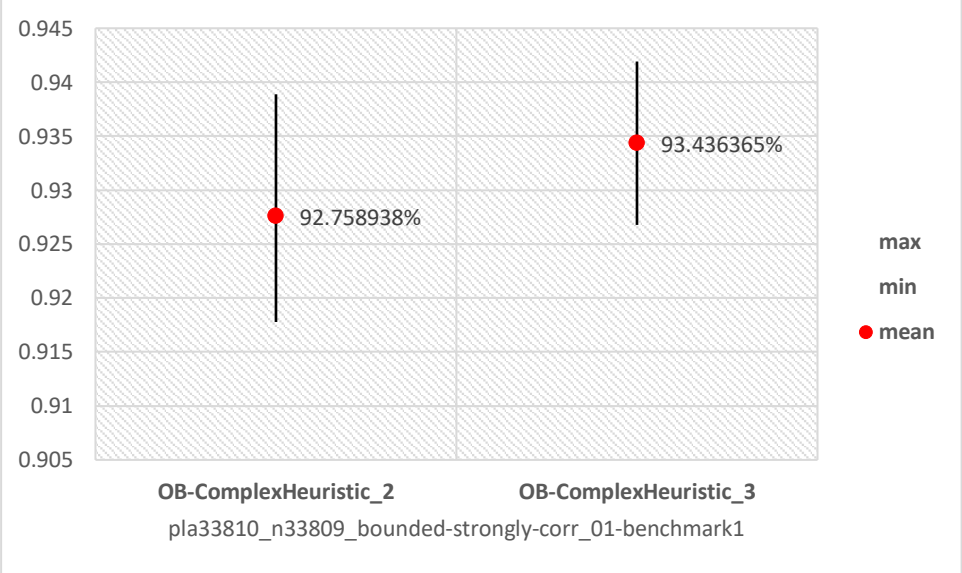
Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	4547005	3892400
2	4573001	3903657
3	4552921	3865066
4	4577778	3904687
5	4575772	3907422
6	4576698	3881939
7	4557977	3921503
8	4541861	3908202
9	4575507	3900201
10	4552902	3891429



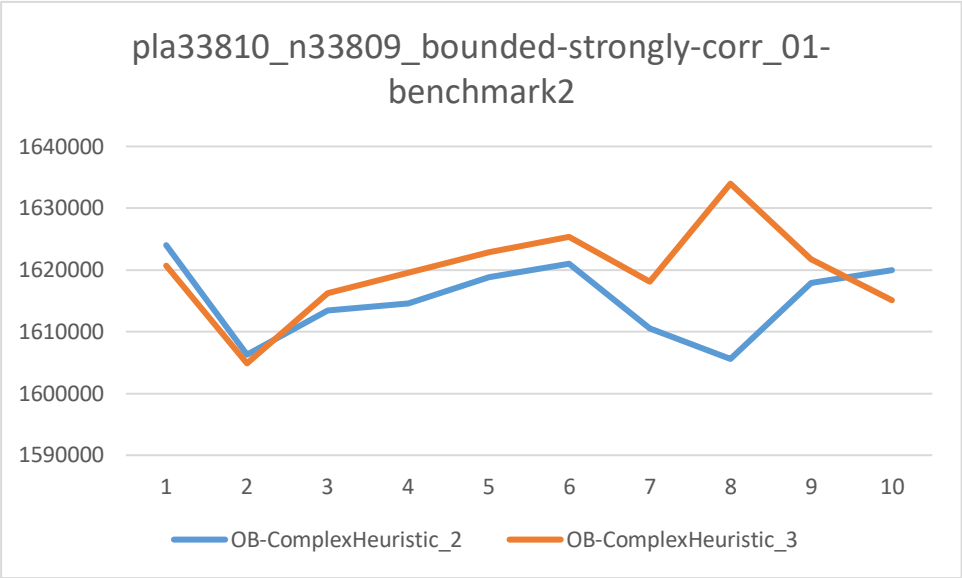


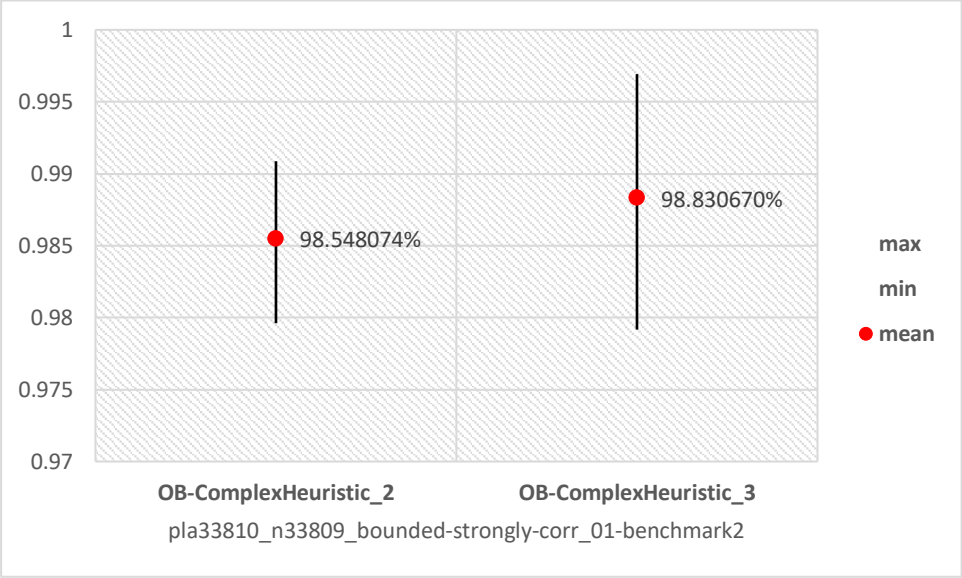
Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	4564535	3869267
2	4560588	3933829
3	4570277	3910569
4	4592485	3891886
5	4589963	3916488
6	4590697	3894269
7	4585348	3898760
8	4582988	3882250
9	4572315	3928688
10	4562913	3904174



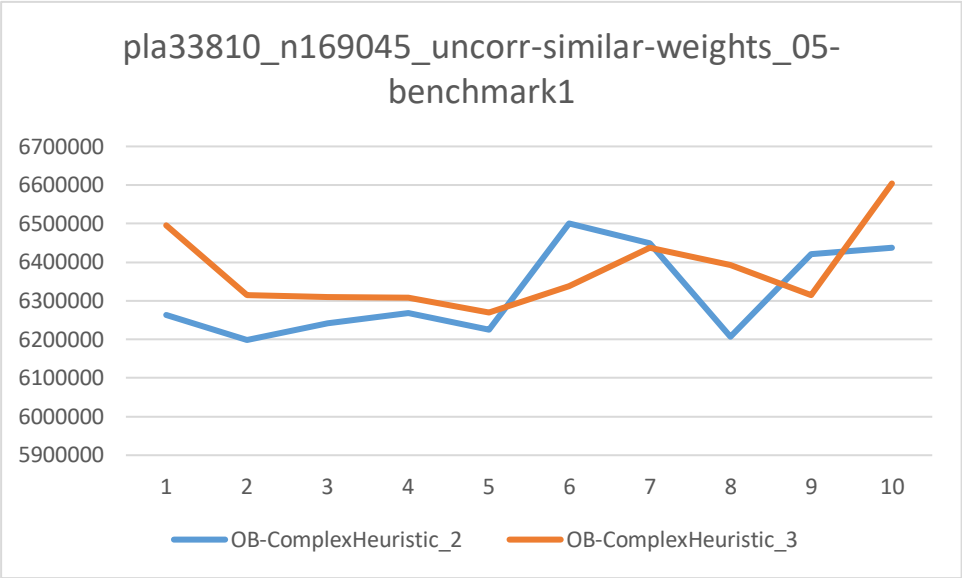


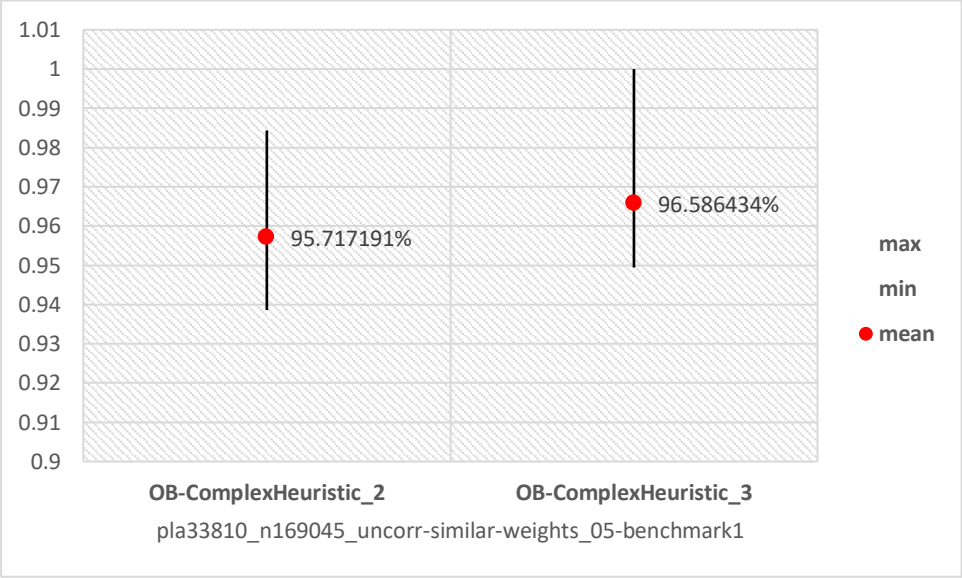
Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	1538867	1537308
2	1537006	1534078
3	1517869	1543809
4	1511999	1533128
5	1515937	1531567
6	1526077	1533598
7	1504249	1519025
8	1518616	1527091
9	1516447	1530518
10	1516198	1524173



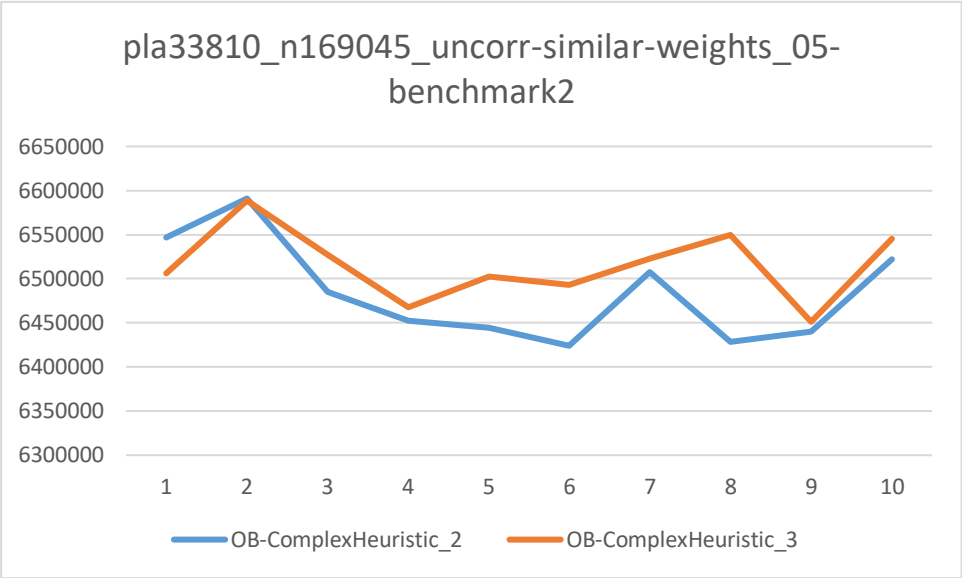


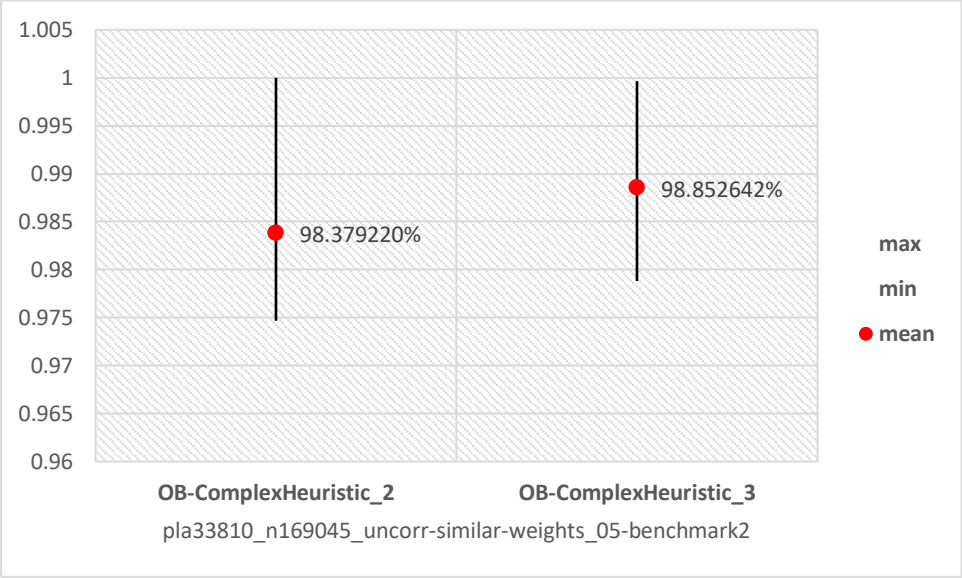
Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	1624010	1620641
2	1606271	1604867
3	1613396	1616211
4	1614572	1619572
5	1618867	1622853
6	1620950	1625361
7	1610573	1618059
8	1605587	1633965
9	1617921	1621775
10	1619961	1615124



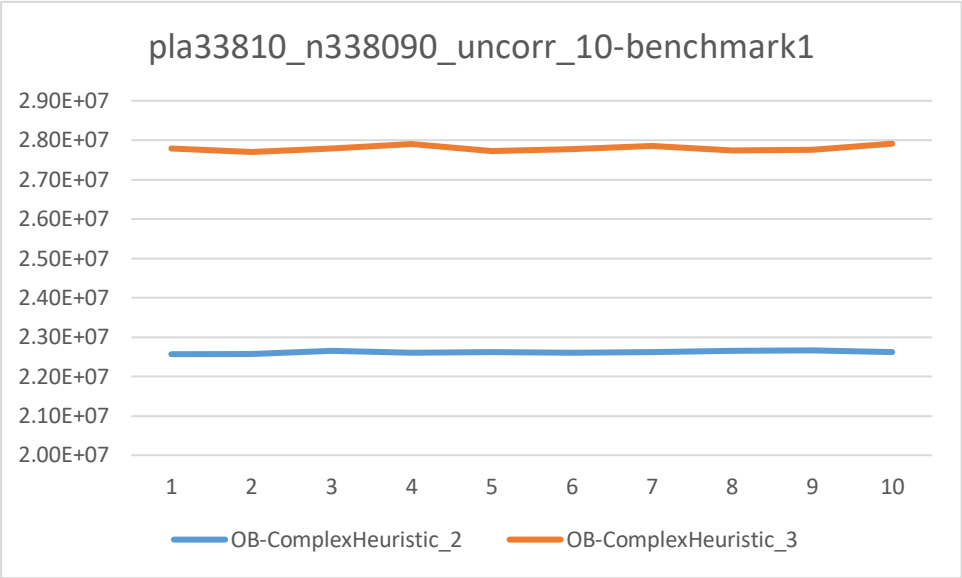


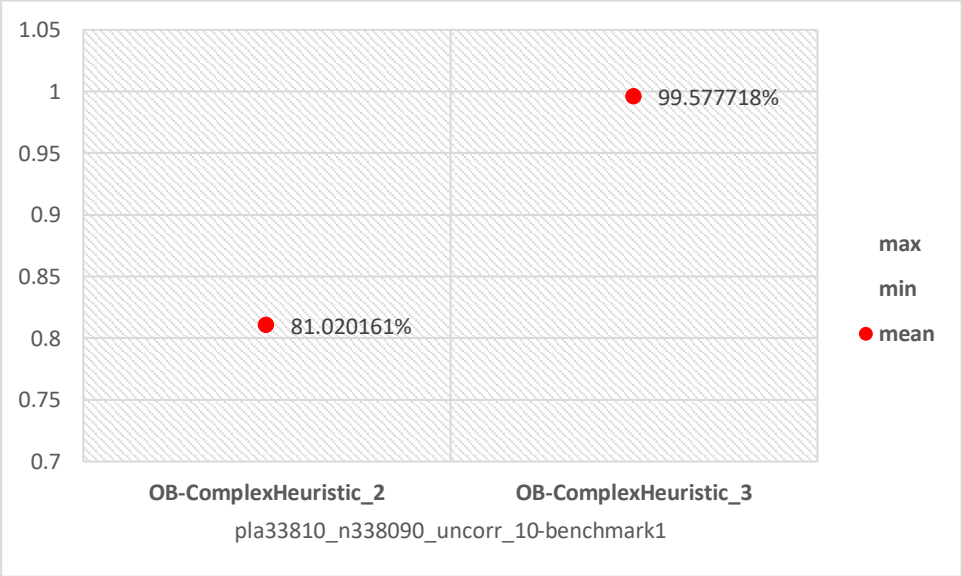
Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	6262770	6495402
2	6198483	6314439
3	6241744	6308979
4	6268844	6308759
5	6225329	6269686
6	6500398	6338730
7	6448448	6437346
8	6207246	6392919
9	6421054	6315353
10	6437242	6603992



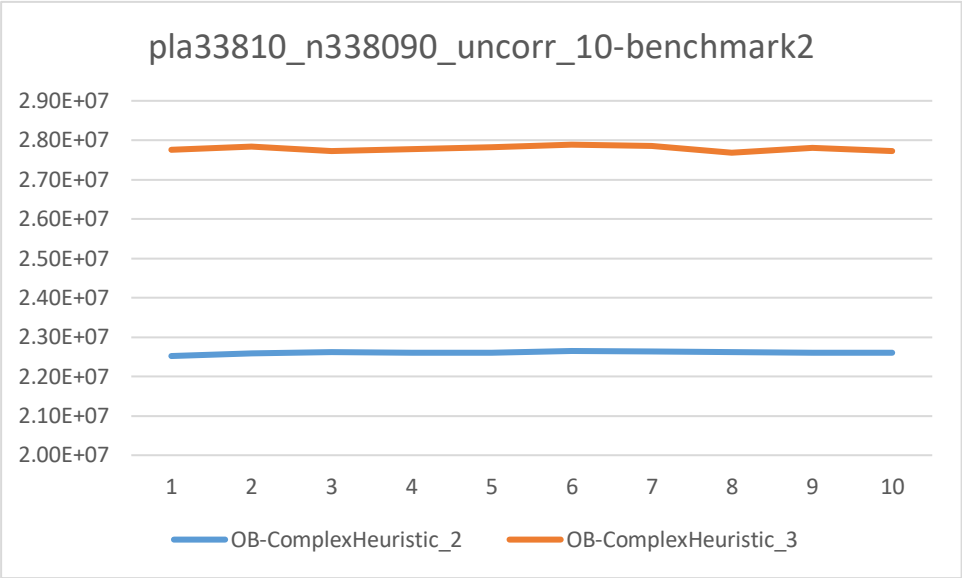


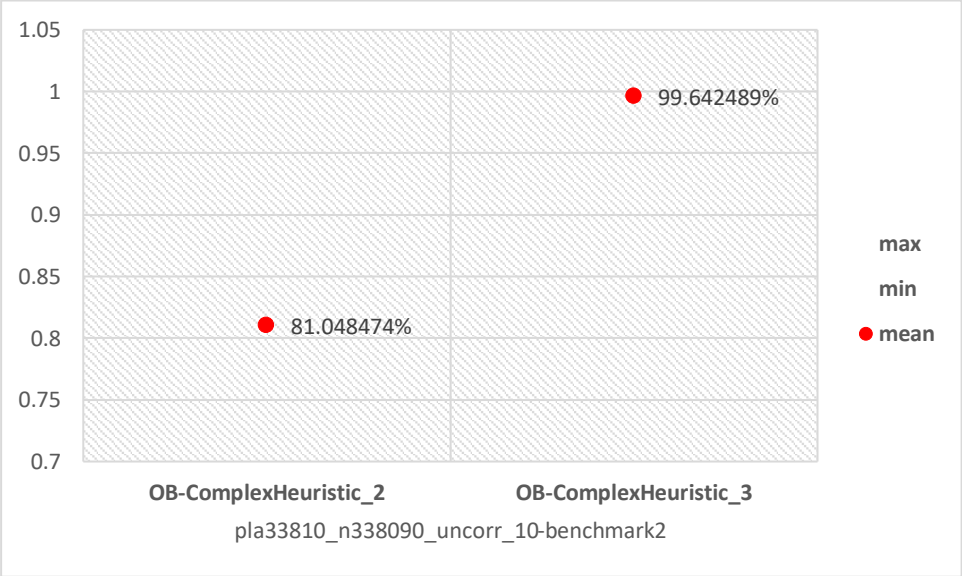
Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	6546961	6505899
2	6590976	6588574
3	6484963	6526997
4	6452476	6468003
5	6444671	6502685
6	6424041	6492749
7	6507367	6522626
8	6428149	6549756
9	6439959	6451214
10	6521941	6545032





Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	2.26E+07	2.78E+07
2	2.26E+07	2.77E+07
3	2.27E+07	2.78E+07
4	2.26E+07	2.79E+07
5	2.26E+07	2.77E+07
6	2.26E+07	2.78E+07
7	2.26E+07	2.79E+07
8	2.26E+07	2.77E+07
9	2.27E+07	2.78E+07
10	2.26E+07	2.79E+07





Run	OB-ComplexHeuristic_2	OB-ComplexHeuristic_3
1	2.25E+07	2.78E+07
2	2.26E+07	2.78E+07
3	2.26E+07	2.77E+07
4	2.26E+07	2.78E+07
5	2.26E+07	2.78E+07
6	2.26E+07	2.79E+07
7	2.26E+07	2.78E+07
8	2.26E+07	2.77E+07
9	2.26E+07	2.78E+07
10	2.26E+07	2.77E+07