

## Review 4

Bilal Akliai

4 Jul 2024

**What I did** I read the paper about the NN-Descent algorithm. I implemented it in Python, but it was not really efficient (I didn't try to optimize it because parallelization seemed to be not really efficient in Python).

So, I switched to C++. I implemented all the algorithms that I had before + the NN-Descent algorithm. It uses Python to construct the fingerprints, and then with Pybind11 it works in C++.

I added parallelization in the NN-Descent algorithm. Thanks to it, we can see a real difference of insertion time.

For example, for 30k compounds, it takes only 2.8s for the insertion time. For the search time, it takes 4s without HNSW, and 6.6ms with HNSW (similar to what we had in Python 2 weeks ago). The problem is that we have only a score of 28%. I think that it can be an error of implementation (I noticed an error in my implementation of the NN-Descent algorithm and I didn't have time to change it for today). So, I will try to correct it, and I hope that it will improve the result.

```

Calculating nearest neighbors without HNSW...
Time taken without HNSW: 3.9953598976135254 seconds

WITH HNSW
Saving fingerprints to file...
Saving active molecules...
Saving inactive molecules...
Total fingerprints saved: 30000
Initializing HNSW structure from file: fps.txt
Starting to insert elements into HNSW...
Calculating nearest neighbors with HNSW...
Time taken with HNSW: 0.006613969802856445 seconds
Percentage of correct neighbors found using HNSW: 28.00%
[39, 159, 194, 433, 659, 746, 951, 952, 958, 959, 960, 1096, 1153, 1248, 1251, 1
267, 1292, 1315, 1320, 1321, 1322, 1323, 1324, 1371, 1372, 1380, 1407, 1441, 296
1, 4712, 5016, 5027, 12650, 16246, 16315, 20692, 20693, 20779, 20800, 20819, 209
37, 21066, 27176, 27224, 27530, 28027, 28741, 29601, 29647, 29866]
[32, 182, 221, 232, 233, 413, 440, 478, 513, 523, 542, 1225, 1247, 1251, 1252, 1
265, 1277, 1278, 1297, 1320, 1321, 1359, 1371, 1372, 1441, 2961, 3081, 3702, 485
1, 5289, 11501, 12984, 13077, 16315, 16454, 20800, 20841, 20937, 20983, 21066, 2
1241, 21243, 24959, 25811, 25812, 27987, 28027, 28741, 28759, 29647]
insert_list :2.819134921
k_nn_search :0.006560485
distance :0.00269557700000000184
nn_descent_full :2.7971787519999998
transform_heaps_to_sets_full :0.0
sample :0.0187455630000000277
reverse :0.010083429
sample_full :1693964574.0 (parallel)
sample_full2 :0.013700257999999984
update_nn_full :0.0
merge_heaps_full :0.010223112999999964
neighborhood :0.0012586840000000001
searchLayer :0.0
process_entry :2.52911527600000064 (parallel)
parallel_for :0.8800532059999999
process_entry2 :0.4012692649999999 (parallel)
parallel_for2 :0.800924787
process_entry3 :5.051753425999986 (parallel)
parallel_for3 :1.0966475490000003
bilal@bilal-Aspire-A315-58:~/2A/Internship/Python/cpp$

```

Figure 1: 50-KNN for 30k compounds, with K=50, efConstruction=100, rho=5 and delta = 0.001

For 110k compounds, it takes 11.3s for the insertion time (compared to about 6min in Python 2 weeks ago). So, we really improved that time! For the search time it's also correct (about 8.2ms). However, we still have the same problem: the result is not good !!

```

Calculating nearest neighbors without HNSW...
Time taken without HNSW: 16.056755542755127 seconds

WITH HNSW
Saving fingerprints to file...
Saving active molecules...
Saving inactive molecules...
Total fingerprints saved: 113741
Initializing HNSW structure from file: fps.txt
Starting to insert elements into HNSW...
Calculating nearest neighbors with HNSW...
Time taken with HNSW: 0.008219003677368164 seconds
Percentage of correct neighbors found using HNSW: 2.00%
[39, 159, 194, 659, 746, 951, 952, 958, 960, 1153, 1248, 1267, 1320, 1321, 1324,
 1371, 1372, 1380, 1407, 1441, 2961, 4712, 5016, 5027, 12650, 16246, 20693, 2077
 9, 20800, 20819, 20937, 27176, 27530, 28027, 31161, 32576, 33152, 33841, 34127,
 34130, 34167, 39889, 41461, 44050, 44226, 45522, 46658, 46680, 52758, 109295]
[27, 140, 224, 232, 349, 412, 429, 540, 734, 971, 1185, 1276, 1291, 1297, 1474,
 3703, 4102, 4992, 4993, 5209, 5482, 12652, 13561, 20695, 20937, 24785, 24893, 28
 792, 30454, 30965, 31890, 33890, 34292, 34959, 35237, 36724, 38191, 38336, 43518
 , 45543, 45711, 46780, 46781, 49957, 52676, 65804, 78928, 106870, 109319, 113486
 ]
insert_list :11.329613181
k_nn_search :0.008149617
distance :0.003414464999999998
nn_descent_full :11.242427960999999
transform_heaps_to_sets_full :0.0
sample :0.069998371000000227
reverse :0.04943765
sample_full :8135726852.0 (parallel)
sample_full2 :0.05146387199999923
update_nn_full :0.0
merge_heaps_full :0.039973679000000753
neighborhood :0.0015016340000000002
searchLayer :0.0
process_entry :6.644707859999996 (parallel)
parallel_for :3.324617351
process_entry2 :2.4115535059999775 (parallel)
parallel_for2 :3.178864923
process_entry3 :92.09463139399824 (parallel)
parallel_for3 :4.668491489000001

```

Figure 2: 50-KNN for 110k compounds, with K=50, efConstruction=100, rho=5 and delta = 0.001

By changing parameters, I can have good results. For example, if I put  $K=400$  (number of neighbors for each node) and  $efConstruction=800$ . I can have 90% which is very good. However, the problem is that changing these parameters impact a lot the memory (400 neighbor for each node is too much!!) and these 2 parameters increase also a lot the search time: so it is not what we want. Indeed, the search time is now about 200ms (compared to about maximum 10 with normal parameters). The insertion time is still very good (22s compared to 6min in Python).

```

Calculating nearest neighbors without HNSW...
Time taken without HNSW: 15.244003534317017 seconds

WITH HNSW
Saving fingerprints to file...
Saving active molecules...
Saving inactive molecules...
Total fingerprints saved: 113741
Initializing HNSW structure from file: fps.txt
Starting to insert elements into HNSW...
Calculating nearest neighbors with HNSW...
Time taken with HNSW: 0.21846222877502441 seconds
Percentage of correct neighbors found using HNSW: 90.00%
[39, 159, 194, 659, 746, 951, 952, 958, 960, 1153, 1248, 1267, 1320, 1321, 1324,
 1371, 1372, 1380, 1407, 1441, 2961, 4712, 5016, 5027, 12650, 16246, 20693, 2077
9, 20800, 20819, 20937, 27176, 27530, 28027, 31161, 32576, 33152, 33841, 34127,
34130, 34167, 39889, 41461, 44050, 44226, 45522, 46658, 46680, 52758, 109295]
[194, 746, 951, 952, 958, 960, 1153, 1248, 1267, 1321, 1324, 1371, 1372, 1380, 1
407, 1441, 2961, 4712, 5016, 5027, 12650, 16246, 20693, 20779, 20800, 20819, 209
37, 27176, 27530, 28027, 31161, 32576, 33152, 33841, 34127, 34130, 34167, 39889,
41461, 44024, 44050, 44132, 45522, 46658, 46680, 52758, 105739, 107890, 109295,
113306]
insert_list :22.15065835
k_nn_search :0.218283138
distance :0.053584862999999578
nn_descent_full :22.060983241
transform_heaps_to_sets_full :0.0
sample :0.072788504000000104
reverse :0.075929356999999999
sample_full :227794491565.0 (parallel)
sample_full2 :0.064760697000000031
update_nn_full :0.0
merge_heaps_full :0.0419644170000005805
neighborhood :0.07141143399999995
searchLayer :0.0
process_entry :22.407408058999845 (parallel)
parallel_for :4.5542629530000002
process_entry2 :3.1221730619999226 (parallel)
parallel_for2 :3.811102476
process_entry3 :67822.52359502218 (parallel)
parallel_for3 :13.558934009

```

Figure 3: 50-KNN for 110k compounds, with  $K=400$ ,  $efConstruction=800$ ,  $\rho=5$  and  $\delta = 0.001$

So, the objective is to have this kind of result but with small parameters (so, it will be good for the memory, and the search time). I will try to correct my implementation of the NN-Descent algorithm (maybe it will solve this problem).

For 200k compounds we have also good results: 78% with an insertion time of 45s. However, it's also with big parameters, so the search time is very long (about 0.3s).

```

Time taken without HNSW: 27.0845308303833 seconds

WITH HNSW
Saving fingerprints to file...
Saving active molecules...
Saving inactive molecules...
Total fingerprints saved: 200000
Initializing HNSW structure from file: fps.txt
Starting to insert elements into HNSW...
Calculating nearest neighbors with HNSW...
Time taken with HNSW: 0.29201436042785645 seconds
Percentage of correct neighbors found using HNSW: 78.00%
[35, 106, 156, 456, 1054, 1605, 1660, 1804, 2059, 9004, 9424, 12055, 14500, 14501, 15162, 15178, 15179, 15187, 15479, 19967, 22859, 23205, 26786, 27117, 27119, 31191, 31201, 31238, 33020, 36467, 36756, 36833, 36834, 36835, 38827, 84025, 96758, 103342, 103415, 105234, 121613, 127987, 164603, 164640, 164641, 165372, 165408, 166862, 166863, 196405]
[35, 106, 456, 1702, 1804, 1852, 2059, 9424, 14500, 14501, 15162, 15178, 15179, 15187, 15479, 17517, 17666, 19967, 22859, 23205, 27117, 27119, 31191, 33020, 36467, 36756, 36833, 36834, 36835, 38827, 38851, 84025, 96758, 103338, 103342, 103415, 104577, 105234, 105276, 121613, 127987, 144575, 164603, 164641, 165372, 165402, 165769, 166862, 166863, 196405]
insert_list :44.420716757
k_nn_search :0.291942931
distance :0.081154096000000625
nn_descent_full :44.25695861
transform_heaps_to_sets_full :0.0
sample :0.1635925659999965
reverse :0.130554523
sample_full :558565805464.0 (parallel)
sample_full2 :0.13590240300000156
update_nn_full :0.0
merge_heaps_full :0.0931404470000024
neighborhood :0.07803509800000008
searchLayer :0.0
process_entry :89.99042049199937 (parallel)
parallel_for :9.652527251
process_entry2 :16.105486943999896 (parallel)
parallel_for2 :7.945747228
process_entry3 :145395.42229636945 (parallel)
parallel_for3 :26.380804187

```

Figure 4: 50-KNN for 200k compounds, with  $K=400$ ,  $efConstruction=800$ ,  $\rho=5$  and  $\delta = 0.001$



I have also another problem. I have to optimize the memory, because it don't work for more than 200k or 300k compounds (the process is killed cause of not enough memory). I've tried not to store all the fingerprints but only the positions with "1"s (as there are a lot of "0"s) but this doesn't allow me to process many more molecules.

**What I still have to do** For the moment, I didn't try to optimize the search time (use techniques for Jaccard similarity for accelerating computations like we said for example...).

**Objectives** I think that firstly I have to resolve the problem of accuracy (without using big parameters to maintain a good search time). Then, I have to optimize the memory to be able to test with more compounds. Next, I will have to try to optimize the search time (it seemed to be not really adapted to parallelization: I tried it a bit but I didn't succeed). And, finally I will have to use the latest papers about HNSW to have the more efficient structure as possible.