It does appear that Clint's 1210 2018 directory is more recent than the 11 10 2018 which is from github.

I had previously assessed that Clinton had a fork for storeandpop that we did not use.After looking further, I found this October 13 commit which has the \*new\* one that has the minimize argument. https://github.com/AdityaSavara/MSRESOLVESG/commit/e9c87973b67186e42375f38d03e57550a4dae52e

note that this is more recent store and pop is not inside the October 12 (12 10) directory. so I think that Clint worked on best mass frag chooser on his computer and never uploaded it even to his own github.

So I need to use the most recent MSRESOLVE (in my github) and to merge into it the ExtentOfSLSUniqueSolvable.py AND bestMassFragChooser.py from Clint’s directory.

1. Stuck a relatively new MSRESOLVE.py with 12-10-2018
   1. 1st I will check if the test files in that directory work still, or not.
      1. --at present they do not work, but I think that is just because they are configured to work with an older MSRESOLVE userinput file (and maybe older MSRESOLVE).
   2. running ExtentOfSLSUniqueSolvable.py does work, and that is probably what Clinton was working on.
2. Making a new directory using MSRESOLVE from github, and sticking in the ClintBestMassFragmentChooserDirectory inside to see if it can pass tests with the newest MSRESOLVE, especially if I change the embedded testing flag to false.
   1. Unsurprisingly, Clint’s bestMassFragChooser.py failed when the flag was true.
   2. Surprisingly, Clint’s bestMassFragChooser.py failed when the flag was false.
      1. The error was that rowSumsList was referenced before assignment.
      2. Comparing the before-and-after file, there are changes which I believe are just superficial prior to line 206 in Clint’s file, which is before line 170 in the old file.
      3. Clint’s file then makes something called “intensityMatrix” and “significanceMatrix”
      4. both Clint’s file and the old file have a loop. So I’m going to just copy and paste the old loop and clint’s non-testing loop to see what their differences are, in a new file.
         1. It looks like they are the same, other than some superficial changes and one missing line above MSRESOLVE.passesRowsSumChecks, needs to have rowSumsList=numpy.sum(currentRefDataas1sand0s, axis=0)
         2. I am going to put that line and its comments into clint’s code above the passesRowsSumCHecks of the non-testing loop.
         3. Now the testing passes, if the testing flag is set to false. That means clint’s code can now be uploaded without causing any problems.
   3. Now that we are using the most recent MSRESOLVE, the ExtentOfSLSUniqueSolvable.py file is giving an error because storeAndPop has arguments that have changed. Corrected that by removing the reverse order argument from this new Python file.
      1. Somewhat surprisingly, what came out was different compared to inside #1 above (that is, inside \1\12-10-2018\ClintBestMassFragmentChooser)
         1. 
         2. Now:
            1. 
         3. Changed to have argument of optimumType="Maximum", and now it did save the item while throwing out the last index.
            1. 

This is probably reasonable since I think we can only have ascending order lists for bisect.

* + - 1. Commented out the upper objective function definition in the file (and parallel list) and uncommented the lower versions.
         1. Now get the expected output which is equivalent to before, only reversed since we are required to do ascending for bisect.



* + - * 1. This means we are ready to go on to the next step, version 3.

1. Copy of version 2 as is. We now have a version of Clint’s code that is compatible with the newest store and pop, at least inside ExtentOfSLSUniqueSolvable.py
   1. now just need to see if it works and also integrate it into the BestMassFragChooser.
   2. Reading to the bottom of 0-ClintObjectivesWriteup.txt, we see that he recommends using the output from ExtentOfSLSSolvableUnique to determine the list of e.g. 10 best mass fragment combinations. Apparently he has not done that yet.
      1. He does have a line saying:

(extentSolvable, summedSignificance,

solvedSpecies, massFragsUsed) = ESUS.ExtentOfSLSUniqueSolvable(

massFragCombination, moleculesToMonitor,

relevantIntensities1sand0s, relevantSignificanceMatrix)

* + 1. and even a line saying:
       1. [topSolvabilitiesAndSignificance,
       2. topBestMassFragments, listUpdated] = MSRESOLVE.storeAndPop(
       3. topSolvabilitiesAndSignificance,
       4. (extentSolvable, summedSignificance),
       5. topBestMassFragments,
       6. massFragCombination,
       7. finalNumberOfCombinationsToKeep,
       8. reverseOrder=True
       9. )
    2. so it seems like it should at least run for testing purposes.
       1. With testing flag off, tests pasts in ~50 seconds by typing pytest.py
       2. Setting the flag to “True” again and trying to run pytest…
          1. Fails, because there is a reverse order argument inside. Changing it to optimumType="Maximum"
          2. Now the loop finished in seconds, though it did not pass.
          3. For test1, the items chosen are quite different, I don’t know why. I need to have some kind of well chosen unit test to know what to expect
          4. Found evidence that there is a problem with summed significances, see file 0-EvidenceOf…. Inside directory 3.
          5. Copying that file into directory 4 and continuing.

1. I did some diagnosing inside of 0-EvidenceOfProblemsWithSummedSignificance.docx which is in the directory ClintSLSUniqueSolvableIntegration\4\MSRESOLVESG-master\UnitTests\ClintBestMassFragmentChooser
   1. I found that was a problem with values of 0 having no significance. This was intentional, but is problematic for our current usage of the significance calculator.
2. I have edited MS resolve in this version
   1. I have fixed things inside IndElemSignificanceCalculator so that there is now
      1. indSummationTermCap = maxIntensityPossible/minThreshold – 1
      2. where maxIntensityPossible has a default of 1.
   2. Now it seems to work. I get good results.
      1. Test 2 and test 3 are slower than I expected, when using this. So I probably still need to keep working on things.
         1. I also want to export all 10 combinations, not just return them.
         2. I added some lines about that.
   3. Looking at the timing as is,
      1. with Clint’s version (useExtentOfSLSUniqueSolvable):
         1. It looks like test 2 takes 23 seconds
         2. It looks like test 3 takes 23 seconds also
         3. that is because it is running through all combinations for the 4 fragments. I need to do some additional testing to look at what happens with the non-Clint version
      2. turning off useExtentOfSLSUniqueSolvable:
         1. test 1: n/a
         2. test 2: 26 seconds.
         3. test 3: 18 seconds.
            1. This one should be shorter, because it has the pre-check.
      3. Overall, we find that \*in this version of the code\* useExtentOfSLSUniqueSolvable is slightly faster (and that’s with print statements on, it should get even faster with them off).
         1. We also know that it is more accurate in general at picking masses. We will look at that a little bit closer in the next version.
         2. More importantly: we need to turn separate the onTheFlySLS code from the if statement that’s accessed when using useExtentOfSLSUniqueSolvable because it is recalculating a bunch of things. So we will see how fast things are with that turned off. In version 6.
3. Made copy of 5
   1. Moved the on the fly SLS if statement out of the useExtentOfSLSUniqueSolvable
      1. test 2: 24 seconds, then 18 seconds.
      2. test 3: 23 seconds. 🡨 just statistics? Should be same. Ran again and got 21 s, then 24 seconds.
         1. It seems the “on the fly part” was not significant b/c only 10 values are being kept right now.
   2. Renamed things to test 11, 12,13 for when using useExtentOfSLSUniqueSolvable = True
   3. Trying to make a test 4 and 14 which has more molecules and fragments, to see how things look in that situation. Copied test\_3 and then edited it.
      1. Test 4 took: 148 seconds.
         1. So the times do start to show up with larger arrays to consider.\
      2. Test 5 like test 4, with no pre-check.
         1. 1092.4558 seconds.
      3. Test 14 took: 23 seconds
4. Stuck the new useExtentOfSLSUniqueSolvable into the same directory as bestMassFragChooser unit test.
   1. Making unit test use DefaultUserInput instead of UserInput.
   2. Need to turn off the ExportedSLSUniqueMasFragments since that may be slowing things down.
      1. Making G.SLSUniqueExport == 'no' in each of the test files.
      2. Oddly, now getting a different error.
      3. For reasons I haven’t figured out, I only got things to work if I used UserInput rather than DefaultUserInput.
      4. Tests 13 etc. are much faster now.
         1. Test 14 is still at ~24 sec
         2. Test 4 at ~87 seconds
         3. test 13 is now at ~1s
         4. Test 3 now at ~3 s
   3. made test 17 and started going through possibilities.
      1. Having: moleculesToMonitor=['Ethylene (Ethene)', 'Ethanol', 'Crotyl Alcohol' , 'Acetaldehyde'] and 8 mass frags gave 1562275 mass frag combinations, which took 200 seconds when the printing was turned off and 333 when the printing was turned on.
      2. Test 18 the same but add in CO (still picking 8 mass frags). Gave 2220075 mass frag combinations, which took 303 seconds with printing.
      3. Made test 8, which is doing the limiting checks to compare the speed, with onTheFlySLS as false.
         1. Time Taken: 884.58 seconds for 2220075 mass frag combinations.
         2. However, this did both the rough uniqueness check and the significance factor check.
         3. What if we do just the rough uniqueness check?
      4. Made test 9 to do rough uniqueness check only
         1. 337.8304946422577 with printing, so it is not actually really any faster than the SLS, which is surprising.
            1. After looking at the code, we apparently do have passesRowsSumChecks, so it’s not like there is “no” check.
      5. Test\_19 from 18, adding in CO2.
         1. 3108105 combinations, 401.3262655735016 seconds.
      6. Test\_20 add in H2
         1. 4292145 comibnations, 307 seconds. [though I made a slight algorithmic improvement]
      7. Test\_21 add in H2O
         1. 7888725, Combinations, 738 seconds
      8. Test\_22 add in crotonaldehyde
         1. 38567100 combinations, 2470 seconds.
5. now trying to make an improvement in how passesRowSumsCheck works.
   1. Will use test fourteen to see if there’s any timing improvement possible.
      1. 1st do several runs to get statistics: 22.168, 22.408, 22.286, 22.343, 21.029, 21.029, 21.12
      2. After first change: 20.65426754951477, 22.25106930732727, 20.99746537208557, 21.933459520339966
      3. After 2nd change:
         1. 20.460, 22.23, 22.22, 21.76
            1. Basically, no change in speed.
   2. Running test 17, 308 s with printing turned on. Probably not statistically different.
   3. Uploading to github.