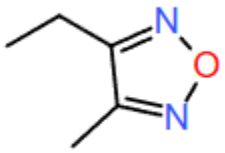
1. create a system of equation and approximate the error of each bead
2. calculate the average the appearance of each bead
3. Round 1 of bead reassignment
4. improvement from 0.626 to 0.662
5. Change from solving for absolute error to solving actual error

Error fixing:

1. Double bond connection in non-benzene 6-ring does not have enough constrains
   Here is what needed to be fix: for case D1 and D2, we need to check if the inner atoms are C and C else throw an error
   Case D3 the else should throw an error saying (double bond not mappable)
   Now we should have enough constraints for the code to work properly
   This now means that We potentially will have more cases to work with



Example: Here, we never checked if the atoms are C so we assign the beads as if they were all carbon -> lead to great error

After this error is fixed, We now have 64 cases left not working.

Easy Error fixing:

1. Add this line

```
                elem_a = atom[1].upper()
                elem_b = neighbor[1].upper()

                # if both are C → TC5
                if elem_a == 'C' and elem_b == 'C':
                    bead = "TC5" + generate_random_string()

                # if one is C and the other is N → TN6a
                elif (elem_a == 'C' and elem_b == 'N') or (elem_a ==
    'N' and elem_b == 'C'):
                    bead = "TN6a" + generate_random_string()
```

This is to fix the cases of inner double bonds atom with no outer connections of length 1.
After this error is fixed, We now have 57 cases left not working.

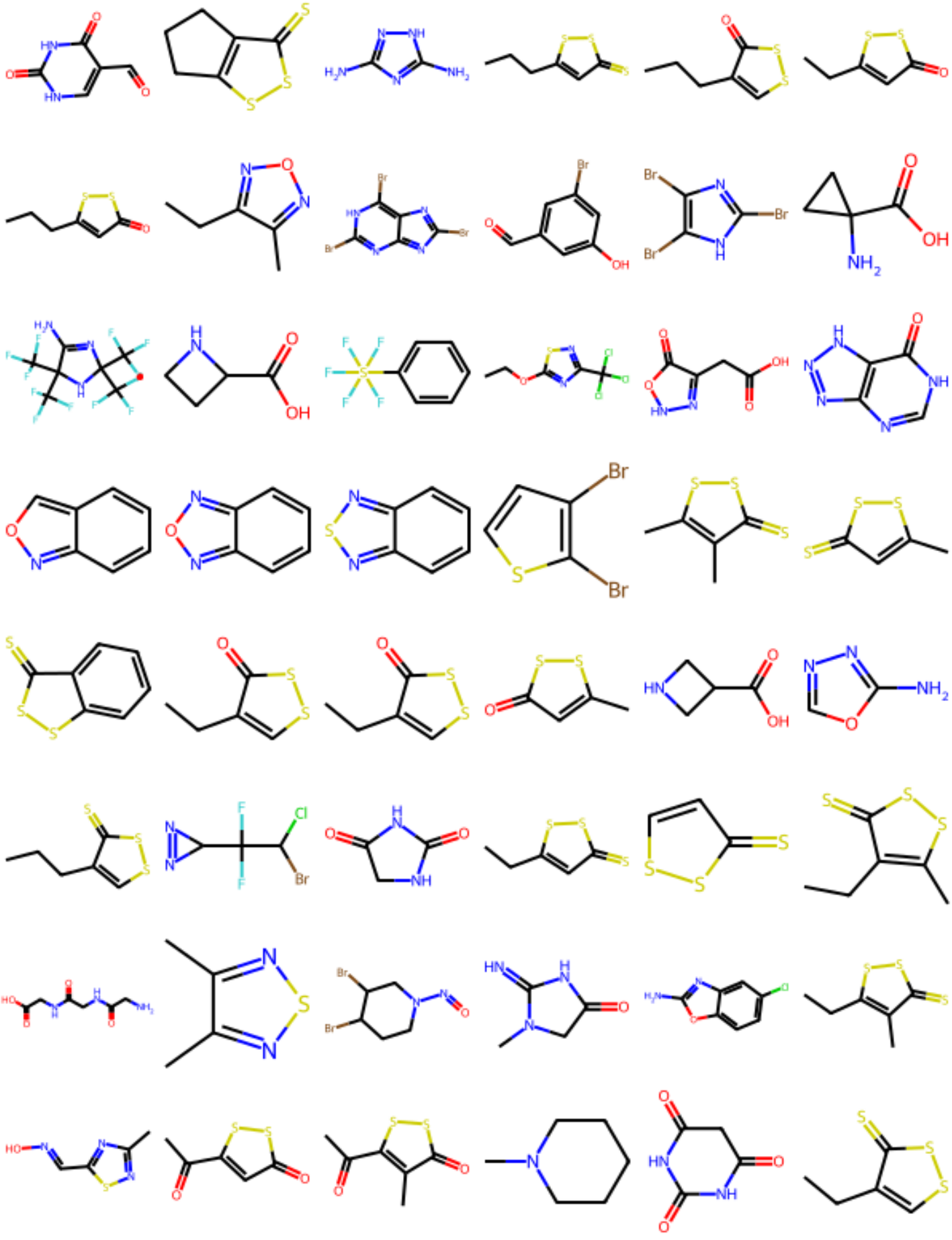
I will paste them all as a whole and then get all of the pictures to see the general images of what needs to be fixed.

1. 1195-08-0
2. 14085-33-7
3. 1455-77-2
4. 146252-77-9
5. 164584-60-5
6. 164584-61-6
7. 164584-62-7
8. 17647-69-7
9. 18874-52-7
10. 199177-26-9
11. 2034-22-2
12. 22059-21-8
13. 23757-42-8
14. 2517-04-6
15. 2557-81-5
16. 2593-15-9
17. 26537-53-1
18. 2683-90-1
19. 271-58-9
20. 273-09-6
21. 273-13-2
22. 3140-93-0
23. 3354-39-0
24. 3354-40-3
25. 3354-42-5
26. 35659-69-9
27. 35659-69-9
28. 3620-08-4
29. 36476-78-5
30. 3775-60-8
31. 38696-40-1
32. 427897-90-3
33. 461-72-3
34. 52514-89-3
35. 534-25-8

36. 55486-68-5
37. 556-33-2
38. 5728-21-2
39. 57541-73-8
40. 60-27-5
41. 61-80-3
42. 6125-90-2
43. 61444-96-0
44. 620957-93-9
45. 620957-94-0
46. 626-67-5
47. 67-52-7
48. 7113-30-6
49. 7411-23-6
50. 7491-74-9
51. 7682-38-4
52. 77-47-4
53. 77580-78-0
54. 865-49-6
55. 89-98-5
56. 94-97-3
57. 95-14-7

These are labels. I will now get the images in order of the label using chatgpt prompts:

write me a code that first read from the file non-working.txt, then for every line in that file, check the column 2 from the Table_S1_from_653.csv, then once you find a match, check the column 1 of the same matching row and retrieve the data there (should now be SMILES), then use RDKit to save a drawing of that molecules, there should be 57 total images like that, format them as 6x10, each image size 1x1, and DPI of the each image be 50x50



now from this let's group all the issues:

1. NH in 6-ring:
2. benzene-like 5-ring with C=N connections, N=N connections, with more atoms like O,S,N present in the ring
3. S-S connection in a ring
4. Connection between C=N and non-ring neighbors of size 1
5. S connecting to 6 other atoms
6. 4 ring has a NH
7. 3 ring has N=N

The biggest issues lies on issue 2 and 3.

Let's fix part 3 first:

This issue only appears in 5-ring section so we will only need to fix the 5 ring sections: (18 occurrences)

We will do this for both the 5-ring and 6-ring because they both handle 5-ring sections (6 will take over if 5 fail)

For both of those function, we need to add at the beginning handling of S cases. Loop through each atom and find one with 'S'

look to see if there is another 'S' as a neighbor (use the first one found)

if there is, assign them both as TC5, if there isn't skip

After this step issue 3 should be solved

We need to take a close look at the TC5 that we just assigned after testing, as there is a chance it does not work.

We also must take a step back because the bead assignment issue when merging is pending right now because the bead assignment is not final.

After this steps, 44 cases are left not working

Issue 2 is much harder to fix before it has lots of complexity:

Must do changes:

1. Big note is that 5-ring can also be all 1.5 bonds like benzene, there for all the bonds are essentially double bonds, for the fix, we need to consider a priority system on which pairs should be accounted for first.
2. For example: N=N and C=N should both be prioritized before C=C

First, we must investigate how it is currently functioning:

We know exactly where the function is handling the double bond (Step A)

Here is how Step A works:

Part 1

- **Loop over every atom** in this 5-ring.
- **Skip** any atom already mapped (`final[idx] != ""`).
- **Only consider** carbons (`'C'`) and nitrogens (`'N'`).
- **Inspect each inner bond** (`atom[4]`) and **select** any atom that participates in a double bond (`bond == 2`) or aromatic bond (`1.5`).
- **Record** a tuple (`pos, idx, type, has_foreign`):
    - `pos`: local index in `section`
    - `idx`: global atom index
    - `type`: `'c'` or `'n'`
    - `has_foreign`: for nitrogens, a boolean saying whether that atom has *any* outer connections (`atom[3]` non-empty)

After this loop, **`candidate_indices`** lists every C or N in the ring that is part of a double/aromatic bond and still unmapped.

This part is fine, however, sometimes we will need to consider unmapped atoms, we will do this in the checks at the end. We can overlook this part for now

Part 2

- **If exactly two** of our candidates are nitrogens, we want to narrow down to one (so we can pair the rest evenly).
- We remove the nitrogen that is **less "connected"** to foreign or to the other nitrogen.
- **If only one** nitrogen appears but the total candidate count is odd, we remove that nitrogen altogether so we can get an even number of carbons to pair.

This part is troublesome. It does not have enough rules to take care of all the cases that we have. This part group them as pairs so we need to change the rules on how the pairs should be grouped.

Part 3:

- We extract the **global indices** from `candidate_indices`, in the order they appear.

- We recursively build **all ways** to pair those indices up into disjoint 2-tuples (`(a,b)`), so that every candidate is in exactly one pair.

Part 4:

- For each candidate-pairing (`matching`), ensure:
  1. Neither atom is already mapped (`final[...] == ""`).
  2. The two atoms are actually **inner-bonded**: one appears in the other's `atom[4]` list.
- Collect those that pass into `valid_matchings`.
- Pick the **first** valid matching to use. If none survive, `pairs` is empty.

Part 5:

- For each pair `(a,b)`, we restore their atom records (`atomA`, `atomB`) to inspect element symbols.
- **C–C pairs**:
  1. If **both** have a single-atom foreign, the code signals an error—this is a TODO in the original.
  2. If **exactly one** has a size-1 foreign neighbor, we group that trio (`a`, `b`, and that foreign) under a bead whose key we select by matching the foreign element inside the `"CC(...)"` parenthesis in `martini_dict`.
  3. Otherwise (pure C–C without any qualifying foreign), we assign `"TC5"+random`.
- **Pairs involving N**: always `"TN6a"+random`.

Brainstorm what needs to be fixed:

I should work backwards here; there are a lot of steps that needs to be taken:

Part 5 fix: This is when we have all the pairings ready:

1. There should be a total of 3 cases: N=N, N=C and C=C
2. Check the two atoms in the pair:

If both of them are C and C:

We now check for the foreign sections of each atoms:

If both of them have a size 1 foreign neighbor:

We break the 4 atoms into 2 parts, each part have its inner atom and the corresponding foreign atom.

For each part we will use pick bead key where element is the element of the foreign atoms, the bond_order is the bond order between the foreign and its corresponding inner atom, and kind is T,

after finding the beads for each pairs, we assign the beads for each pairs

If only 1 of them have a size 1 foreign neighbor:

We keep this part the same as what is already in the code

If both of them dont have a size 1 foreign neighbor:

We assign them just like what is already in the code (TC5)

If one is C and one is N:

There is the issue of the half working beads where they must contain the exact bead type.

We fix this by assigning them as ? anyways and when it is time to make the ITP files,

I can remove the ? and therefore I can still tell which bead is made up from the ? from the text file

Shubha bead fixing is done; our result came out worse

Realization: to decrease the delta_G, we need to increase the water number, because the formula is octanol-water, We have been doing it in reverse the whole time!

Round 2 of bead fixing:

C6 -> N6 (CNCC and CCCN)

SC6 -> SN6 (CNC)

SN3 -> SN6 (CCN)

SN3 -> SP1 (C=CN)

X3 -> X4 (C(F)(F)(F)F)

TC4 -> TC6 (C(C) benzene)

N6 -> N4 (C(O)(C)(C), C(O)(C)(CC))

TN6 -> TP4 (C(O) benzene)

N5 -> N2 (C(C)(C)(CO), CCCO)

SP4 -> SP3 (C(N)(=O)(C))

SN5 -> SN4 (CCO)

TN3 -> TN5 (CN)

X1h -> X2 (ClCCCl)

P2a -> P4a (N(C=O)(C)(C))

SN3a -> SC5 (CC=O)

SN3a -> SP1a (OC=O)

SC2 -> SC3 (C(C)(C)(C))

SN3a -> SN2a (NC=O)

SN6 -> SN4 (CC(O))

P5 -> P6 (S(=O)(C)(C))

SN3r -> SN2r (COC)

TN5a -> TN4a (C=O)