Group 7

Project Results
Topic: Optimization of Molecules using RL.

Ghanendra Singh
Rana Kumar Jana

Link to Repository for codes running condition

- https://github.com/Ghanendra19213/Ghanendra_MT19213_RL_M2020/blob/master/Project/Moldqn_implementation.ipynb

Source code/implementations used

Mol-DQN using Pytorch

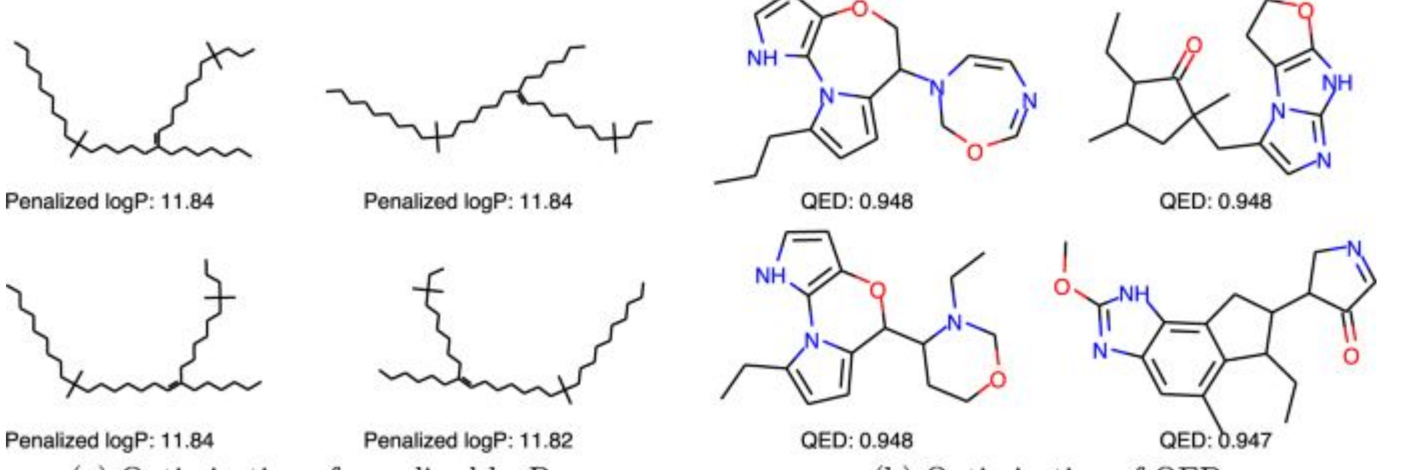
- https://github.com/google-research/google-research/tree/master/mol_dqn
- https://github.com/rdkit/rdkit/blob/master/Contrib/SA_Score/sascorer.py
- https://github.com/EXJUSTICE/MolDQN-pytorch
- https://github.com/aksub99/MolDQN-pytorch

Tensorboard in Jupyter Notebook

- lspvic/jupyter_tensorboard: Start Tensorboard in Jupyter Notebook

RDKit

- https://www.rdkit.org/docs/Cookbook.html#drawing-molecules-jupyter

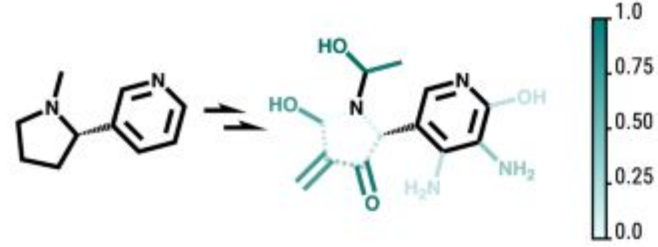


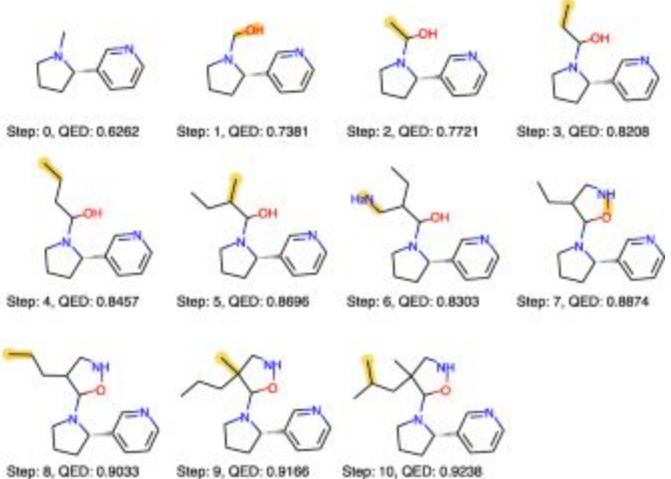
(a) Optimization of penalized logP

(b) Optimization of QED

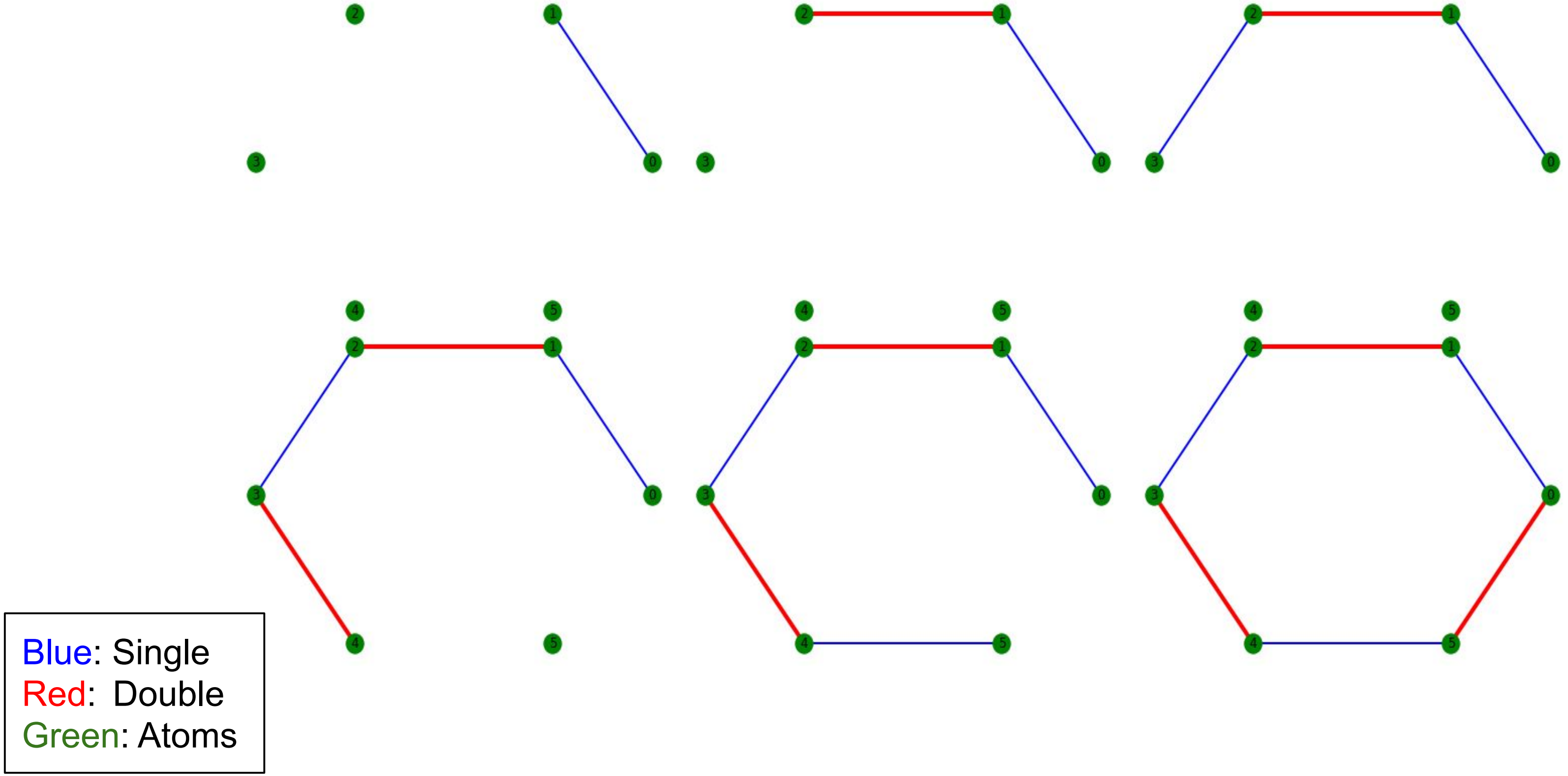
(a)

(b)



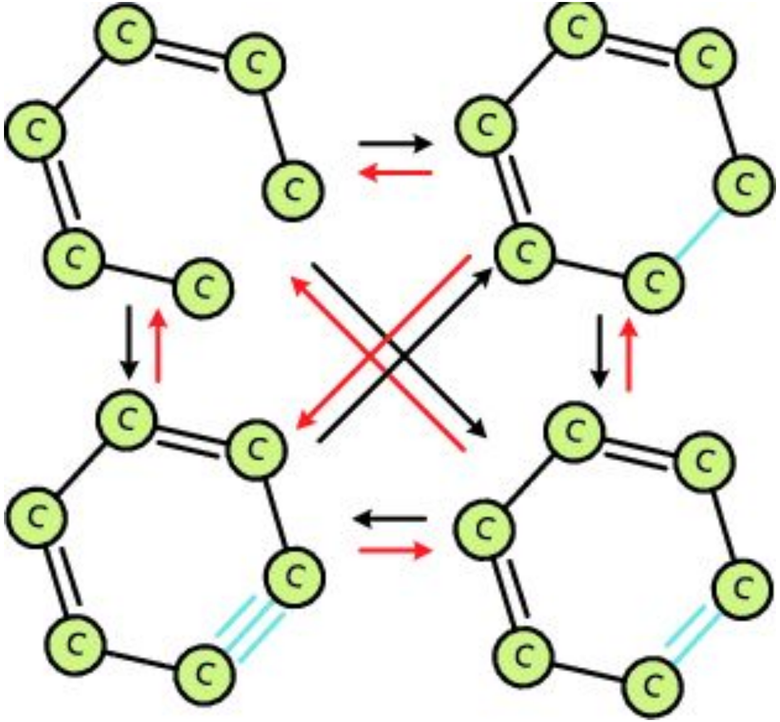


Bond formation Single and Double bond.

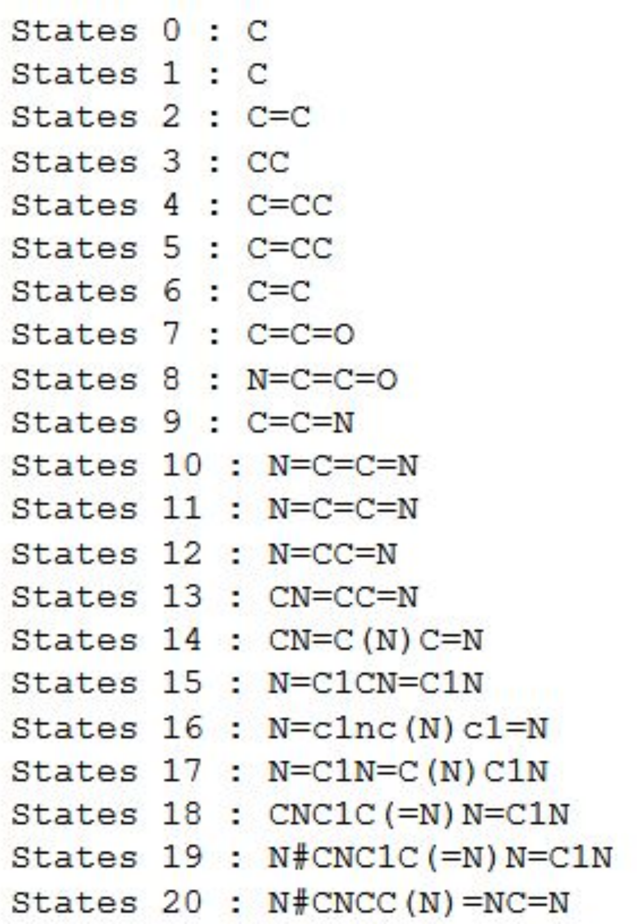


Blue: Single
Red: Double
Green: Atoms

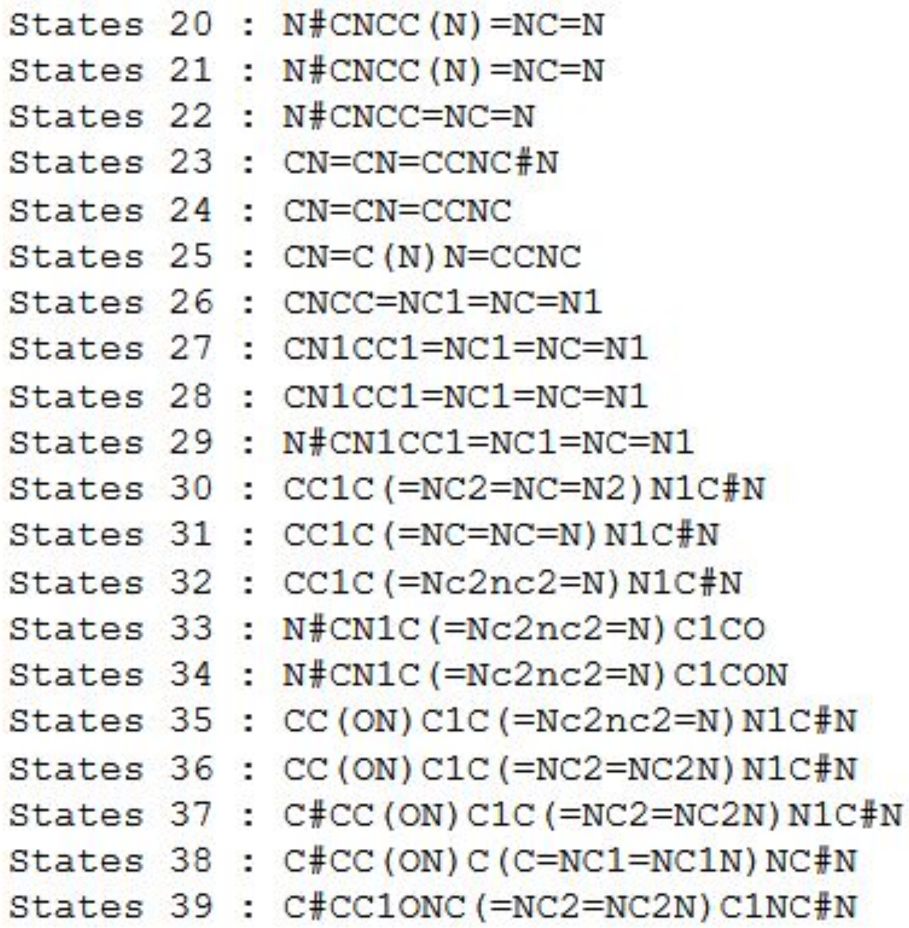
Aromatic ring formation



States of Molecules as SMILES

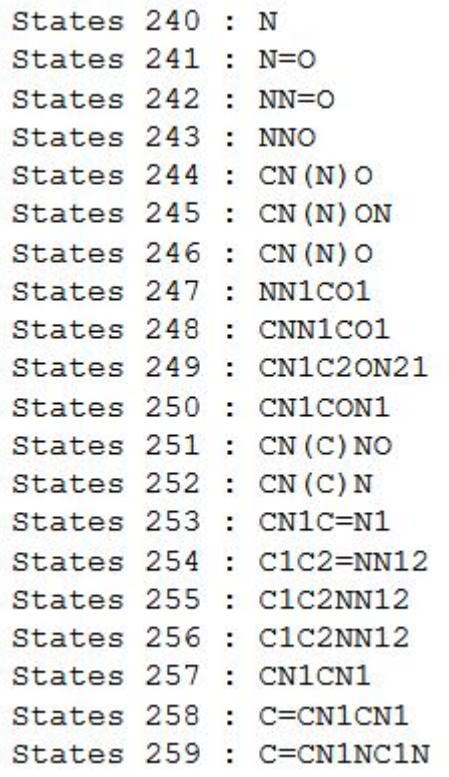


```
States 0 : C
States 1 : C
States 2 : C=C
States 3 : CC
States 4 : C=CC
States 5 : C=CC
States 6 : C=C
States 7 : C=C=O
States 8 : N=C=C=O
States 9 : C=C=N
States 10 : N=C=C=N
States 11 : N=C=C=N
States 12 : N=CC=N
States 13 : CN=CC=N
States 14 : CN=C(N)C=N
States 15 : N=C1CN=C1N
States 16 : N=c1nc(N)c1=N
States 17 : N=C1N=C(N)C1N
States 18 : CNC1C(=N)N=C1N
States 19 : N#CNC1C(=N)N=C1N
States 20 : N#CNCC(N)=NC=N
```

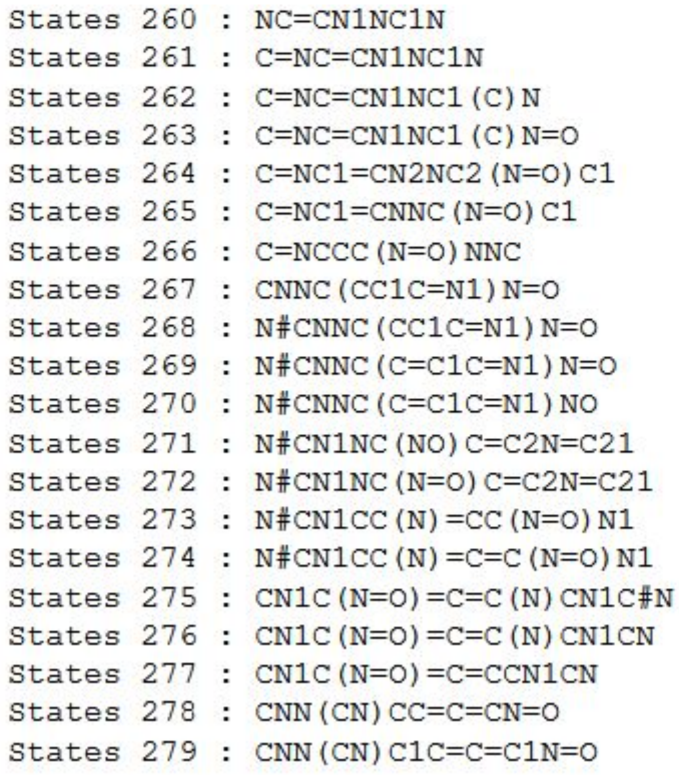


```
States 20 : N#CNCC(N)=NC=N
States 21 : N#CNCC(N)=NC=N
States 22 : N#CNCC=NC=N
States 23 : CN=CN=CCNC#N
States 24 : CN=CN=CCNC
States 25 : CN=C(N)N=CCNC
States 26 : CNCC=NC1=NC=N1
States 27 : CN1CC1=NC1=NC=N1
States 28 : CN1CC1=NC1=NC=N1
States 29 : N#CN1CC1=NC1=NC=N1
States 30 : CC1C(=NC2=NC=N2)N1C#N
States 31 : CC1C(=NC=NC=N)N1C#N
States 32 : CC1C(=Nc2nc2=N)N1C#N
States 33 : N#CN1C(=Nc2nc2=N)C1CO
States 34 : N#CN1C(=Nc2nc2=N)C1CON
States 35 : CC(ON)C1C(=Nc2nc2=N)N1C#N
States 36 : CC(ON)C1C(=NC2=NC2N)N1C#N
States 37 : C#CC(ON)C1C(=NC2=NC2N)N1C#N
States 38 : C#CC(ON)C(C=NC1=NC1N)NC#N
States 39 : C#CC1ONC(=NC2=NC2N)C1NC#N
```

Starting state S0 = N atom at iteration 240

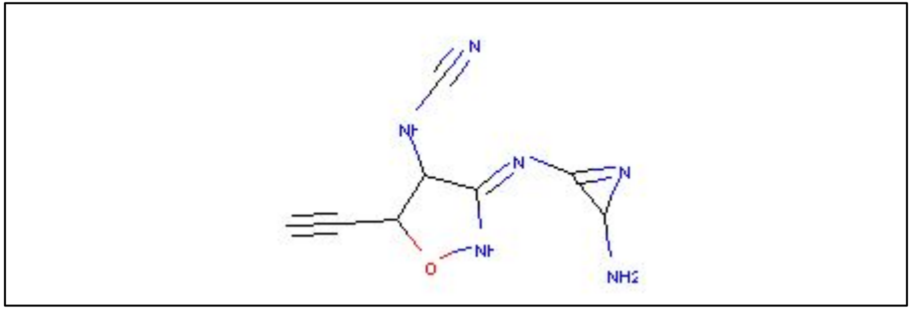


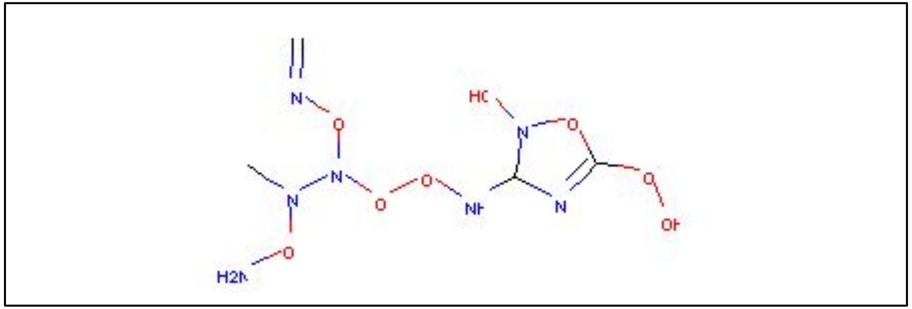
```
States 240 : N
States 241 : N=O
States 242 : NN=O
States 243 : NNO
States 244 : CN(N)O
States 245 : CN(N)ON
States 246 : CN(N)O
States 247 : NN1CO1
States 248 : CNN1CO1
States 249 : CN1C2ON21
States 250 : CN1CON1
States 251 : CN(C)NO
States 252 : CN(C)N
States 253 : CN1C=N1
States 254 : C1C2=NN12
States 255 : C1C2NN12
States 256 : C1C2NN12
States 257 : CN1CN1
States 258 : C=CN1CN1
States 259 : C=CN1NC1N
```

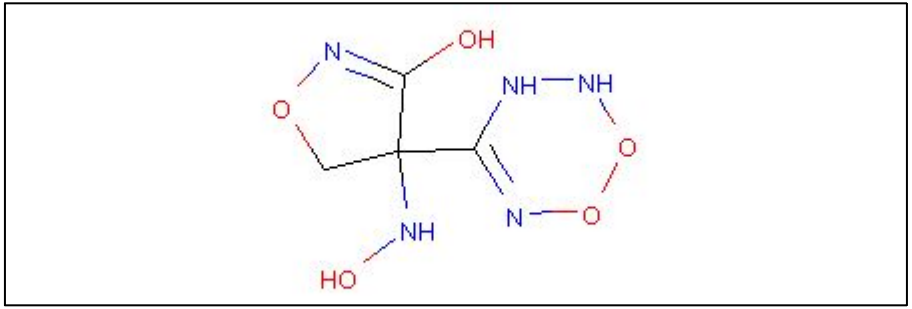


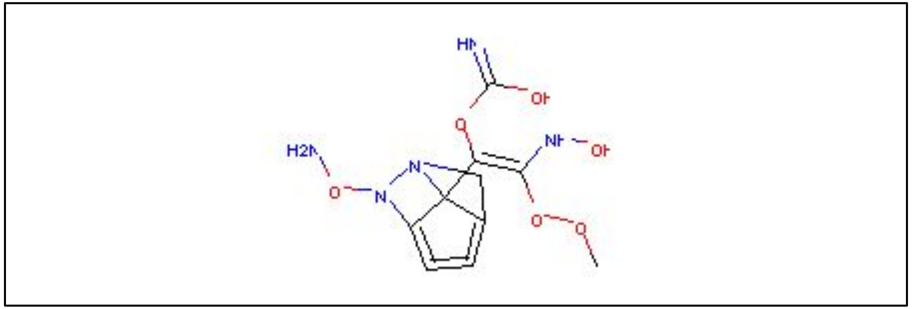
```
States 260 : NC=CN1NC1N
States 261 : C=NC=CN1NC1N
States 262 : C=NC=CN1NC1(C)N
States 263 : C=NC=CN1NC1(C)N=O
States 264 : C=NC1=CN2NC2(N=O)C1
States 265 : C=NC1=CNNC(N=O)C1
States 266 : C=NCCC(N=O)NNC
States 267 : CNNC(CC1C=N1)N=O
States 268 : N#CNNC(CC1C=N1)N=O
States 269 : N#CNNC(C=C1C=N1)N=O
States 270 : N#CNNC(C=C1C=N1)NO
States 271 : N#CN1NC(NO)C=C2N=C21
States 272 : N#CN1NC(N=O)C=C2N=C21
States 273 : N#CN1CC(N)=CC(N=O)N1
States 274 : N#CN1CC(N)=C=C(N=O)N1
States 275 : CN1C(N=O)=C=C(N)CN1C#N
States 276 : CN1C(N=O)=C=C(N)CN1CN
States 277 : CN1C(N=O)=C=CCN1CN
States 278 : CNN(CN)CC=C=CN=O
States 279 : CNN(CN)C1C=C=C1N=O
```

Sample Molecules generated





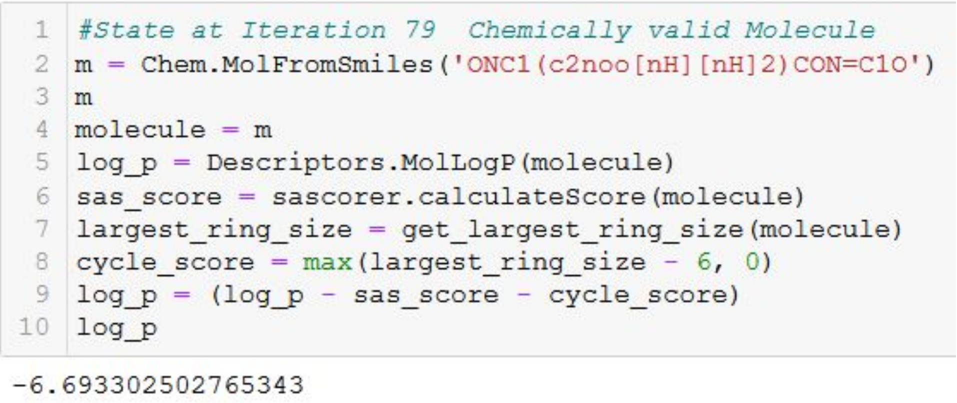




Penalized logP Score

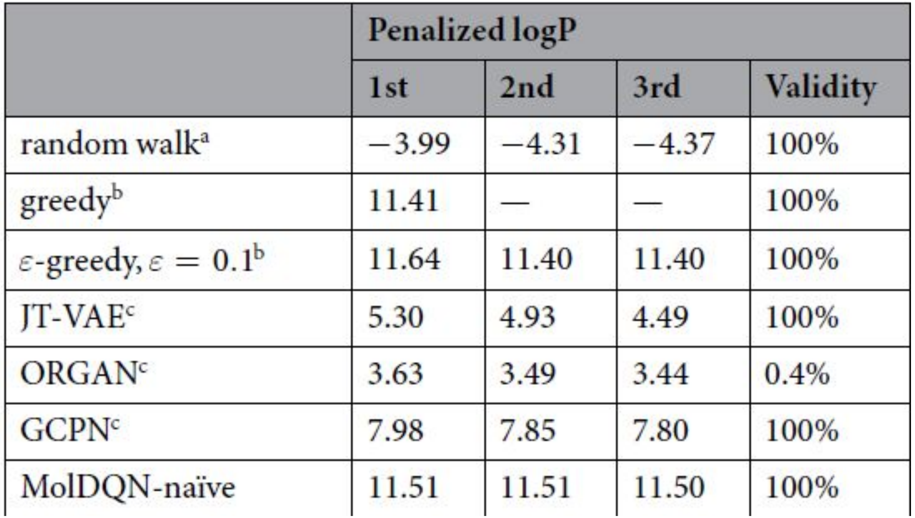
~ 6.7, Generated molecules are not drug like, importance of reward and need for using multiple objective using RL, But the SMILES string is **Chemically valid**.

Penalized logP = logP - SA score (Synthetic Accessibility)



```
#State at Iteration 79  Chemically valid Molecule
m = Chem.MolFromSmiles('ONC1(c2noo[nH][nH]2)CON=C1O')
m
molecule = m
log_p = Descriptors.MolLogP(molecule)
sas_score = sascorer.calculateScore(molecule)
largest_ring_size = get_largest_ring_size(molecule)
cycle_score = max(largest_ring_size - 6, 0)
log_p = (log_p - sas_score - cycle_score)
log_p
```

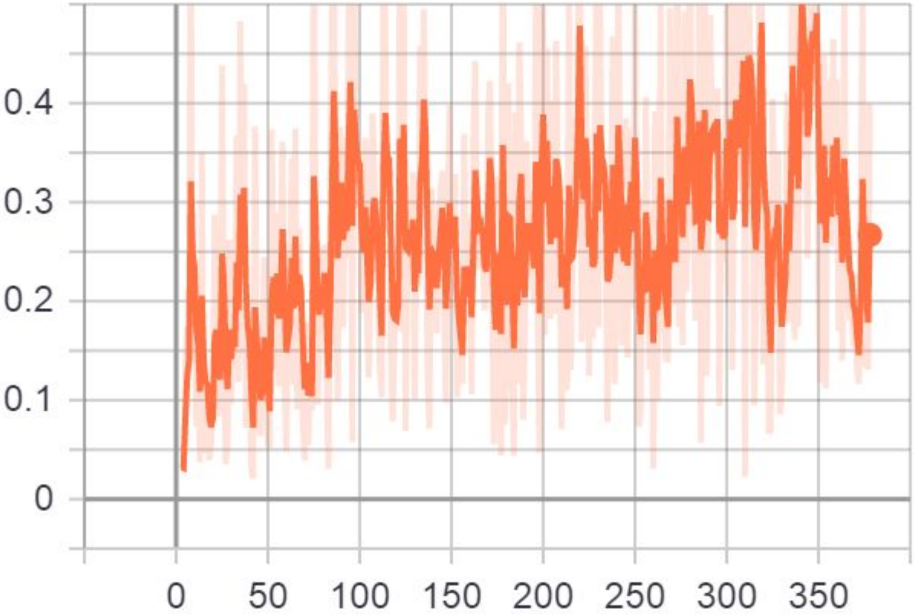
```
-6.693302502765343
```

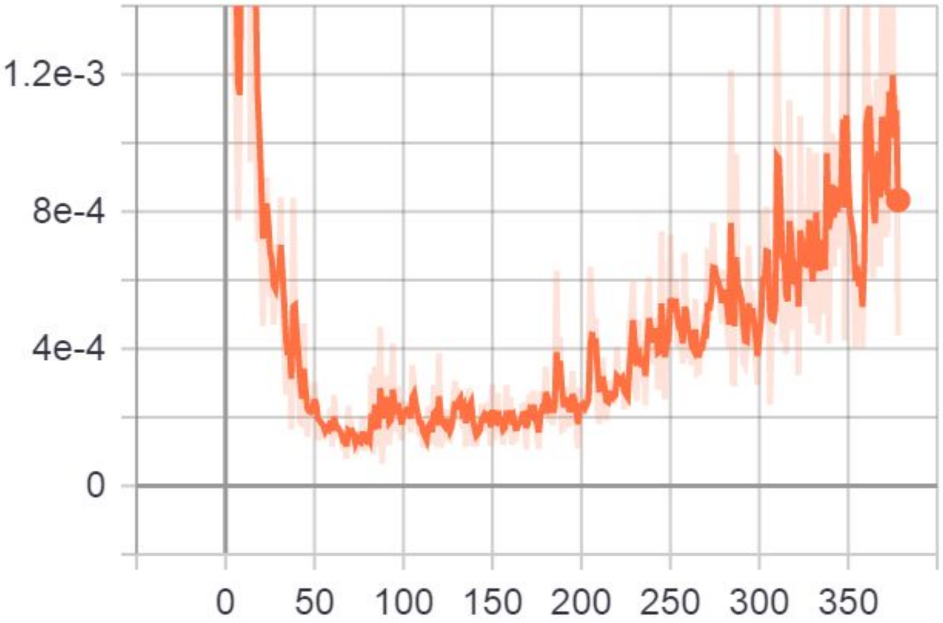


	Penalized logP			
	1st	2nd	3rd	Validity
random walk[a]	−3.99	−4.31	−4.37	100%
greedy[b]	11.41	—	—	100%
ε-greedy, $\varepsilon = 0.1$[b]	11.64	11.40	11.40	100%
JT-VAE[c]	5.30	4.93	4.49	100%
ORGAN[c]	3.63	3.49	3.44	0.4%
GCPN[c]	7.98	7.85	7.80	100%
MolDQN-naïve	11.51	11.51	11.50	100%

Results

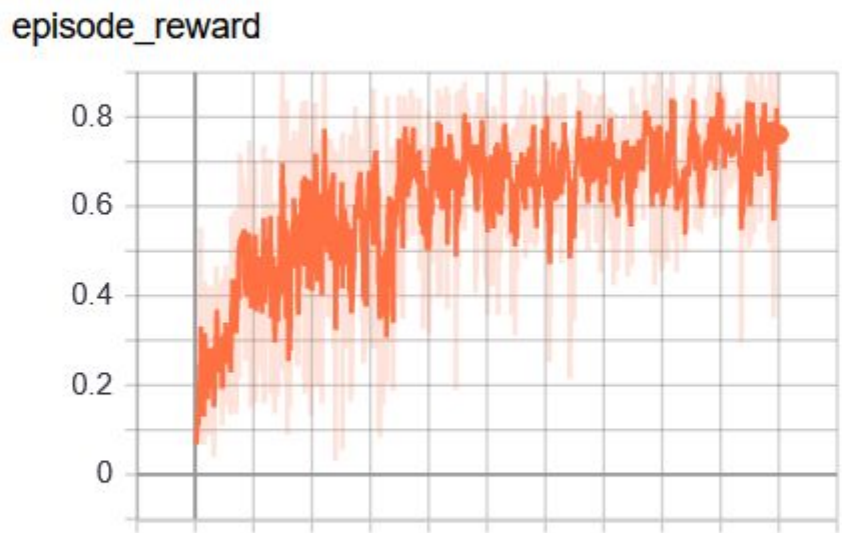
Single Property Optimization

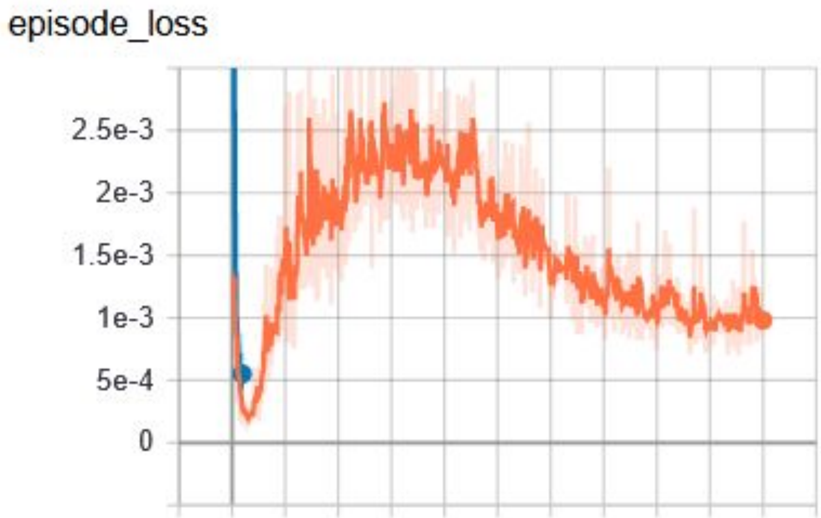




Results

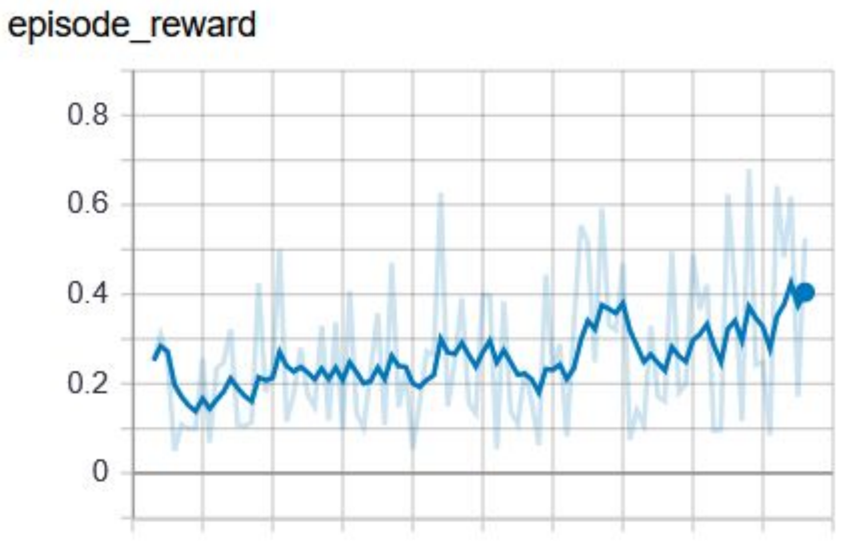
Multi Property Optimization

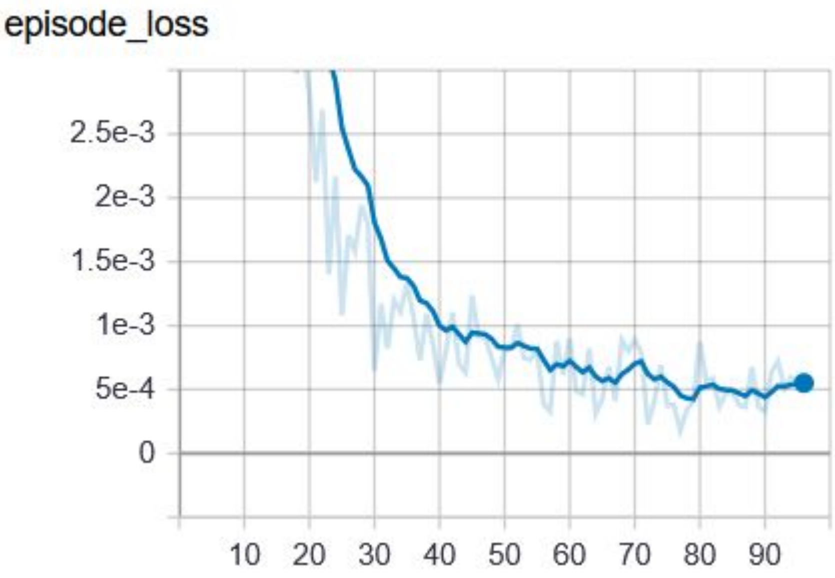

episode_reward


episode_loss

Single Property Optimization - 100 Episodes.

Demo run From Tensorboard


episode_reward


episode_loss
2.5e-3
2e-3
1.5e-3
1e-3
5e-4