Goal directed molecule generation using Monte Carlo Tree Search

Anand A. Rajasekar[1], Karthik Raman[1,3,4], and Balaraman Ravindran[2,3,4]

[1]*Department of Biotechnology, IIT Madras*
[2]*Department of Computer Science and Engineering, IIT Madras*
[3]*Initiative for Biological Systems Engineering, IIT Madras*
[4]*Robert Bosch Centre for Data Science and Artificial Intelligence (RBC-DSAI), IIT Madras*

Abstract

One challenging and essential task in biochemistry is the generation of novel molecules with desired properties. Novel molecule generation remains a challenge since the molecule space is difficult to navigate through, and the generated molecules should obey the rules of chemical valency. Through this work, we propose a novel method, which we call ***unitMCTS***, to perform molecule generation by making a unit change to the molecule at every step using Monte Carlo Tree Search. We show that this method outperforms the recently published techniques on benchmark molecular optimization tasks such as QED and penalized logP. We also demonstrate the usefulness of this method in improving molecule properties while being similar to the starting molecule. Given that there is no learning involved, our method finds desired molecules within a shorter amount of time.

1 Introduction

One of the fundamental problems in chemistry is the design of novel molecules with specific properties. Molecule generation is essential for drug design and material science. Nevertheless, the task remains a difficult one due to the vastness of the chemical space. It is estimated that the number of drug-like molecules is around 10^{23} – 10^{60} [Polishchuk et al., 2013]. Also, the chemical space is discrete with high sensitivity of properties to molecular structure. Hence, there is an increasing need for efficient models focussed on specific applications that guide molecule generation.

In recent years, various ML approaches have been deployed in molecular design. Generally, molecules are represented as SMILES strings [Gómez-Bombarelli et al., 2016, Blaschke et al., 2018] (a linear string notation), fed as input to the model. One of the primary challenges with the SMILES notation is to ensure the validity of generated strings since the generated strings may or may not correspond to a molecule. A better way of representing molecules is with molecular graphs where a node represents an atom, and an edge represents a bond. This method's advantage is the stepwise generation of molecular graphs [Li et al., 2018a,b], which is valid at every step. One common strategy for molecular graph generation is using a generative model such as Generative Adversarial Network or Variational Auto Encoder. Although generative models seem a viable option for generation, these models cannot optimize for a particular property. Hence, an additional step is usually employed that involves optimization in the latent space of the model using a property-oriented decoder [Samanta et al., 2019], Particle Swarm Optimization [Winter et al., 2019], or Bayesian Optimization [Jin et al., 2018]. Non-convexity and high dimensionality of the latent space usually render the task of optimization highly difficult. A second strategy is based on reinforcement learning, which involves an agent that interacts with the environment to learn and make decisions that maximize the cumulative reward. Again, the featurization of molecules here can be done with SMILES representation. Using RL techniques with SMILES representation [Guimaraes et al., 2018, Olivecrona et al., 2017] struggle in

Preprint. Under review.

generating valid molecules. However, search-based techniques such as MCTS with SMILES can help maintain chemical validity [Yang et al., 2017] by removing invalid SMILES strings. Using molecular graphs representation with RL provides 100% validity [You et al., 2018]. All the methods mentioned above involves the use of a dataset. MolDQN [Zhou et al., 2019] pursued a different approach by constructing a Markov Decision Process with unit modification to a molecule as actions and used DQN to approximate the molecules' value. Molecules are represented using fingerprints, and this method learns to generate molecules without the help of a dataset. PGFS Gottipati et al. [2020] proposed multiple modifications to an existing molecule at every step. This method constitutes an MDP with valid reactions as possible actions, while reactants and products are the states. We follow the MDP structure defined by MolDQN [Zhou et al., 2019]; however, we do not allow bridge atoms so that the generated molecules look feasible. Algorithmically, our method combines Monte Carlo Tree Search, a well-established technique, and below mentioned formulation of MDP in a novel way, which is critical for its success.

2 Markov Decision Process

We define the generation process for our method using the MDP, M = $\big(S, A, R, P\big)$ which is defined as follows.

State space, S consists of all valid molecules, $s \in S$. At $t = 0$, s_0 begins with nothing or a specific molecule (for constrained optimization), and at every step, the molecule is modified. The number of steps allowed from the starting state is limited according to the task at hand.

Action space, A consists of all possible modifications that can be performed on a molecule. Given any molecule, all modifications that can be applied to it falls under one of four categories;

- Atom addition - Let K be the set of atoms that are allowed for addition. For any given molecule M, all atoms in set K are added to molecule M at every location that results in a chemically valid bond. These additions result in several possible new molecules.
- Bond addition - Any two atoms a and b that allow for a valid addition of bond in molecule M is included in this list of molecules
- Bond removal - Any two atoms a and b connected by a bond in Molecule M can be removed and included in this list of molecules.
- Bond replacement - Any two atoms a and b connected by a bond can be replaced with another bond subject to valency conditions. All valid replacements are included in this list of molecules.

Atom removal is not included here since when an atom becomes disconnected from a molecule, it is automatically removed. Together all these modifications correspond to actions that can be performed on the existing molecule M. We do not allow modifications that introduce bridge atoms in the molecule.

Reward, R associated with the generated molecule is used as feedback. We do not reward the system at every step. After each episode, properties associated with the final generated molecule are used to update the nodes' value.

Transition probability, P - Since the MDP is deterministic, each action can correspond to a single new state. Hence, the probability of reaching that state is one while all other states are zero.

3 Proposed method

In a regular tree search, one evaluates a node, its descendants, and so on until a final solution is obtained. However, this brute force search is not efficient for tasks that exponentially increase the number of nodes as we go further down the tree. One such example that has a vast number of possibilities is the game of Go. However, humans have achieved a state of the art performance in Go with the help of MCTS Silver et al. [2016]. MCTS makes the tree search faster by arriving at a policy that gives more importance to favorable descendants while lesser importance to

others. This way, only a minimal number of nodes are explored in order to obtain the optimal solution.

Monte Carlo Tree Search consists of four steps, which are repeated for a certain number of times. The four steps are:

- Selection - This step involves selecting a node that favors both exploration and exploitation among all descendants of a parent. The strategy used to select the node is called tree policy and is defined in Equation (1).
- Expansion - Once we reach the tree's leaf node, possible future states of the node are added to the tree, thus expanding it.
- Simulation - After a new node is added, a simulation is performed for a specified rollout depth using a simple rollout policy.
- Backpropagation - The resulting state's value is then used to update V and n of nodes in the selected path of the tree.

$$child = \underset{i}{argmax} \ \frac{V_i}{n_i} + c\sqrt{\frac{lnN_i}{n_i}} \tag{1}$$

where V_i is the value of the node, n_i and N_i are the numbers of times the node, and its parent have been visited, and c is a hyperparameter that decides the importance of exploration and exploitation.

Algorithm 1: unitMCTS

for *each mcts step* **do**
  while *not leaf* **do**
    Select - Pick a child node based on tree policy in Equation (1)
  end
  Expand - Add top k children to the tree and initialize the number of visits, n with one and value, V with reward obtained by ϵ-greedy policy for each new node. Backpropagate value and visits till the root of the tree.
  Simulate - Execute the rollout policy until rollout depth and pick the final molecule.
  while *not root* **do**
    Backpropagate - Scale the reward exponentially using a scaling factor α. Increment V and n of the node in the selected path by the scaled reward of molecule and one, respectively. Move to the parent of the node.
  end
end

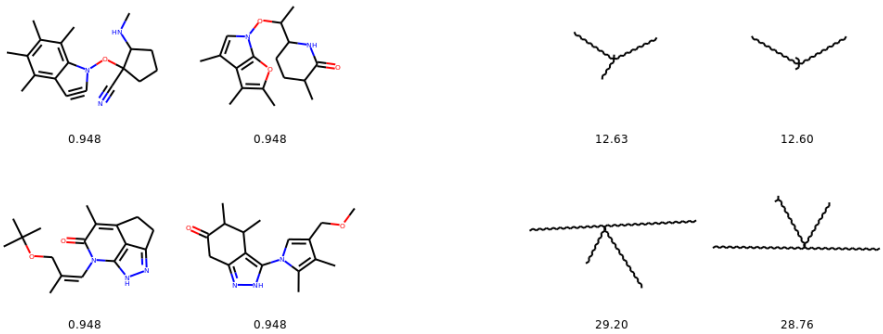
4 Results & Discussion

Setup - RDKit [Landrum, 2016] is used for the molecule environment. The maximum steps per episode are 38 and 84 for **unitMCTS-38** and **unitMCTS-84** respectively, for all molecular optimization tasks starting from scratch and 20 for constrained optimization tasks, being set similar to MolDQN[Zhou et al., 2019] for a fair comparison. ***unitMCTS*** uses three atoms, i.e., Carbon, Nitrogen, and Oxygen for molecule generation. We optimize for two metrics, namely Quantitative Estimate of Drug Likeness (QED) and Penalized logP. QED is a weighted sum of a molecule's fundamental properties, such as its solubility, molecular weight, etc. It has a bounded range of [0,1]. Penalized logP is the logarithm of the partition ratio of solute between octanol and water subtracted by synthetic accessibility score and long cycles. It has a range of $(-\infty, \infty)$.

Baselines - We compare our method with the following baselines. JTVAE [Jin et al., 2018] uses a graph representation of molecules combined with Variational Auto Encoder for generating molecular graphs and Bayesian Optimization on latent space for optimizing property scores. ORGAN [Guimaraes et al., 2018] follows a text representation of molecules coupled with RL based generation. ChemTS [Yang et al., 2017] uses the Monte Carlo Tree Search for SMILES generation with RNN based rollout policy. We also compare our work with GCPN [You et al., 2018], which uses a graph convolutional network for molecule generation, and MolDQN [Zhou et al., 2019], which uses

Table 1: Top 3 molecules obtained by each method on QED and Penalized logP tasks

	Penalized logP			QED		
	1st	2nd	3rd	1st	2nd	3rd
ORGAN	3.63	3.49	3.44	0.896	0.824	0.82
JTVAE	5.3	4.93	4.49	0.925	0.911	0.91
ChemTS	6.56	6.43	6.34	—	—	—
GCPN	7.98	7.85	7.8	**0.948**	0.947	0.946
MolDQN-bootstrap	11.84	11.84	11.82	**0.948**	0.944	0.943
Ours (unitMCTS-38)	**12.63**	**12.6**	**12.55**	**0.948**	**0.948**	**0.948**
MSO	26.10	—	—	**0.948**	—	—
PGFS	27.22	—	—	**0.948**	—	—
Ours (unitMCTS-84)	**29.20**	**28.76**	**28.73**	**0.948**	**0.948**	**0.948**



(a) QED optimization

(b) Penalized logP optimization

Figure 1: Top molecules generated by our method

Bootstrapped DQN to estimate the value of molecules. Above mentioned baselines limit the number of steps per episode to 38. We also compare our method with PGFS [Gottipati et al., 2020], which proposes a forward synthesis framework powered by reinforcement learning, and MSO [Winter et al., 2019], which generates molecules using Particle Swarm Optimization. To perform a fair comparison with the last two baselines, we set the number of steps per episode to 84, equal to the maximum bond number present in the dataset used by PGFS.

To measure the effectiveness of molecule generation using MCTS, we have evaluated our approach on two standard tasks, namely Property optimization and Constrained optimization.

4.1 Property optimization

This task aims to generate molecules with high QED and Penalized logP scores. Table 1 summarizes the property optimization results obtained by our method compared to other approaches. unitMCTS-38 outperforms all the baselines with an average improvement of 60% over GCPN and 6% over MolDQN-bootstrap on the Penalized logP task. It outperforms the baselines on the QED task as well. unitMCTS-84 outperforms both the baselines on the Penalized logP task. All of our generated molecules for both QED and Penalized logP tasks look realistic and are displayed in Figure 1. Given that only valid molecules are added to the tree, our method's validity is 100%.

4.2 Constrained optimization

In this task, we chose 800 molecules with the lowest Penalized logP scores in the ZINC [Irwin et al., 2012] dataset and performed a search to improve the Penalized logP while maintaining similarity with the starting molecule at various thresholds. Table 2 summarizes the performance of our method compared to the baselines. Our method outperforms the baselines at all thresholds. The tree expansion

Table 2: Mean and S.D. of Penalized logP improvement in constrained optimization tasks

δ	JT-VAE	GCPN	MolDQN-bootstrap	Ours (unitMCTS-20)
0.0	1.91 ± 2.04	4.20 ± 1.28	7.04 ± 1.42	9.47 ± 2.38
0.2	1.68 ± 1.85	4.12 ± 1.19	5.06 ± 1.79	8.21 ± 1.99
0.4	0.84 ± 1.45	2.49 ± 1.30	3.37 ± 1.62	6.21 ± 1.15
0.6	0.21 ± 0.71	0.79 ± 0.63	1.86 ± 1.21	3.44 ± 1.80

for this task is slightly different from the above tasks since only the top k molecules that pass the similarity check are added to the tree. Hence, the success rate of our method is 100%.

5 Future work

Our work primarily deals with unit modification to a molecule at every step. Molecule modification with non-unit changes is one promising direction for future work where molecules are added instead of atoms. Our future work in this area is to propose MolMCTS that adds molecules at every step, similar to MDP proposed by PGFS [Gottipati et al., 2020].

References

Pavel Polishchuk, Timur Madzhidov, and Alexandre Varnek. Estimation of the size of drug-like chemical space based on gdb-17 data. *Journal of computer-aided molecular design*, 27, 08 2013. doi: 10.1007/s10822-013-9672-4.

Rafael Gómez-Bombarelli, David Duvenaud, José Hernández-Lobato, Jorge Aguilera-Iparraguirre, Timothy Hirzel, Ryan Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4, 10 2016. doi: 10.1021/acscentsci.7b00572.

Thomas Blaschke, Marcus Olivecrona, Ola Engkvist, Jürgen Bajorath, and Hongming Chen. Application of generative autoencoder in de novo molecular design. *Molecular Informatics*, 37(1-2): 1700123, 2018. doi: 10.1002/minf.201700123. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/minf.201700123`.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs, 2018a.

Yibo Li, Liangren Zhang, and Zhenming Liu. Multi-objective de novo drug design with conditional graph generative model, 2018b.

Bidisha Samanta, Abir De, Gourhari Jana, Pratim Kumar Chattaraj, Niloy Ganguly, and Manuel Gomez-Rodriguez. Nevae: A deep generative model for molecular graphs, 2019.

Robin Winter, Floriane Montanari, Andreas Steffen, Hans Briem, Frank Noé, and Djork-Arné Clevert. Efficient multi-objective molecular optimization in a continuous latent space. *Chem. Sci.*, 10:8016–8024, 2019. doi: 10.1039/C9SC01928F. URL `http://dx.doi.org/10.1039/C9SC01928F`.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *Proceedings of Machine Learning Research*, 80:2323–2332, 10–15 Jul 2018. URL `http://proceedings.mlr.press/v80/jin18a.html`.

Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-reinforced generative adversarial networks (organ) for sequence generation models, 2018.

Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de novo design through deep reinforcement learning, 2017.

Xiufeng Yang, Jinzhe Zhang, Kazuki Yoshizoe, Kei Terayama, and Koji Tsuda. Chemts: an efficient python library for de novo molecular generation. *Science and Technology of Advanced Materials*, 18(1):972–976, 2017. doi: 10.1080/14686996.2017.1401424. URL https://doi.org/10.1080/14686996.2017.1401424. PMID: 29435094.

Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6410–6421. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7877-graph-convolutional-policy-network-for-goal-directed-molecular-graph-generation.pdf.

Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N. Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific Reports*, 9(1), Jul 2019. ISSN 2045-2322. doi: 10.1038/s41598-019-47148-x. URL http://dx.doi.org/10.1038/s41598-019-47148-x.

Sai Krishna Gottipati, Boris Sattarov, Sufeng Niu, Yashaswi Pathak, Haoran Wei, Shengchao Liu, Karam M. J. Thomas, Simon Blackburn, Connor W. Coley, Jian Tang, Sarath Chandar, and Yoshua Bengio. Learning to navigate the synthetically accessible chemical space using reinforcement learning, 2020.

David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016. URL http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

Greg Landrum. Rdkit: Open-source cheminformatics software. 2016. URL https://github.com/rdkit/rdkit/releases/tag/Release_2016_09_4.

John J. Irwin, Teague Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman. Zinc: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 52(7):1757–1768, 2012. doi: 10.1021/ci3001277. URL https://doi.org/10.1021/ci3001277. PMID: 22587354.