THE JOURNAL OF PHYSICAL CHEMISTRY B
A JOURNAL OF THE AMERICAN CHEMICAL SOCIETY

pubs.acs.org/JPCB Article

MolOpt: Autonomous Molecular Geometry Optimization Using Multiagent Reinforcement Learning

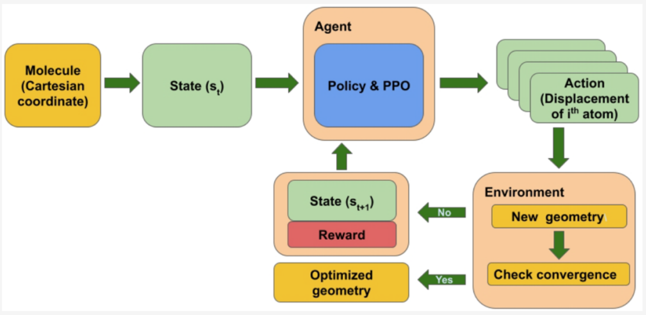
Published as part of The Journal of Physical Chemistry B virtual special issue "Machine Learning in Physical Chemistry Volume 2".

Rohit Modee, Sarvesh Mehta, Siddhartha Laghuvarapu, and U. Deva Priyakumar*

Cite This: https://doi.org/10.1021/acs.jpcb.3c04771 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: Most optimization problems require the user to select an algorithm and, to some extent, also tune it for better performance. Although intuition and knowledge about the problem can speed up these selection and fine-tuning processes, users often use trial-and-error methodologies, which can be time-consuming and inefficient. With all of that in mind and much more, the concept of "learned optimizers", "learning to learn", and "meta-learning" has been gathering attention in recent years. In this article, we propose MolOpt that uses multiagent reinforcement learning (MARL) for autonomous molecular geometry optimization (MGO). Typically MGO algorithms are hand-designed, but MolOpt uses MARL to learn a learned optimizer (policy) that can perform MGO without the need for other hand-designed optimizers. We cast MGO as a MARL problem, where each agent corresponds to a single atom in the molecule. MolOpt performs MGO by minimizing the forces on each atom of the molecule. Our experiments demonstrate the generalizing ability of MolOpt for the MGO of propane, pentane, heptane, hexane, and octane when trained on ethane, butane, and isobutane. In terms of performance, MolOpt outperforms the MDMin optimizer and demonstrates performance similar to that of the FIRE optimizer. However, it does not surpass the BFGS optimizer. The results demonstrate that MolOpt has the potential to introduce innovative advancements in MGO by providing a novel approach using reinforcement learning (RL), which may open up new research directions for MGO. Overall, this work serves as a proof-of-concept for the potential of MARL in MGO.



1. INTRODUCTION

Neural network potentials (NNPs) learn to approximate the potential energy surface (PES) as a high dimensional function (HDF) f by learning from existing reference data. Once trained NNPs can successfully circumvent the need to solve the electronic Schrödinger equation explicitly as it has learned the mapping $f(Z_i, r_i) \rightarrow E$, where Z_i are the nuclear charges and r_i are the atomic positions.[1−7] Machine learning (ML) methods in general have been successful in improving computational chemistry algorithms leading to accelerated property prediction and chemical space exploration.[8] Recently, much emphasis has been on developing efficient ML-based search algorithms to explore chemical space,[9−12] but the same is not the case for conformational space. There are very few attempts to develop an efficient ML-based search algorithm that can explore the conformational space, i.e., probe the potential energy surface (PES).[13−17] These ML-based search algorithms have applications in 3D structure generation[18,19] and molecular geometry optimization (MGO). MGO aims to find the nearest/local molecular conformation with minimum potential energy on PES, starting from a given initial 3D conformation. MGO is an essential part of computational chemistry because any studies related to equilibrium geometries demand searches for minima on PES. Over the past several decades, there have been a variety of well-established optimization methods, such as conjugate gradient (CG), steepest descent (SD), Newton−Raphson, and quasi-Newton methods, to solve this task of MGO.[20−24]

These geometry optimization methods involve using the PES's first-order or second-order derivatives. The examples of first-order optimization algorithms are the steepest descent (SD)[25,26] and conjugate gradient (CG).[27] These first-order optimization algorithms use gradient information to perform optimization. The steepest descent method takes the next step

Received: July 15, 2023
Revised: November 1, 2023
Accepted: November 1, 2023

by searching for the steepest direction to minimize the function's value given the current point. This is done by taking a step in the direction of the negative gradient, and step size is calculated using a line search. Conversely, the conjugate gradient method involves the use of gradient information to compute n-conjugate (A-orthogonal) directions and takes a gradient direction descent step in these n-conjugate directions at each iteration, thereby reaching the minima in *n* number of steps. One can see that using the n-conjugate directions, the CG method avoids moving in the zigzag fashion during optimization, i.e., CG takes a step along these n-conjugate directions only once. SD and CG require gradient calculations to decide the update direction and have very slow convergence (more number of steps) as compared to second-order optimization methods.[22]

Second-order optimization methods make use of Hessian of the PES for optimization. The benefit of second-order optimization methods is that the use of Hessian and gradient algorithms provides a much better update step than the step taken with only gradient information. An example of second-order optimization algorithms is the Newton−Raphson method. At each iteration of the Newton−Raphson method, the PES is approximated as a quadratic function locally, and the optimization step is computed as the step toward the minima of this local approximation. Even though these sophisticated, second-order optimization methods converge in fewer steps, they still need to compute the Hessian and its inverse at each iteration, which is computationally expensive. Hence each iteration of second-order optimization methods is computationally expensive compared to first-order optimization methods but takes fewer steps to converge. On the other hand, quasi-Newton methods like BFGS[28−31] achieve performance similar to the second-order optimization algorithms by circumventing the need to calculate the Hessian and its inverse at each iteration explicitly. These methods instead make a lower-rank approximation to the Hessian using the displacement vectors and gradients and then take a Newton-type update step based on this approximation.

Developing these algorithms is a laborious process, one that needs to be formulated and validated iteratively.[32] Lately, the focus has been on devising new methods to machine-learn molecular features, resulting in robust representations. Just as deep learning (DL) has been successful in automating feature engineering, automating algorithm design could open new avenues and change the way we design algorithms. Automating algorithm design and learning a "learned optimizer" may outperform current hand-designed optimizers.[32−34]

Lately, there has been some progress in developing customized optimizers using DL to handle various optimization problems. For instance, Egidio et al. introduced a "step-size policy" that predicts the step size for the L-BFGS algorithm using the local information at the current position.[33] Andrychowicz et al. developed learning optimization algorithms using a supervised learning technique using long- and short-term memory networks (LSTMs). They showed that their learning optimization algorithms could solve simple convex optimization problems and were able to optimize neural networks.[35] Metz et al. have discussed the difficulties in training the learned optimizers, and by analyzing the trained optimizer, they desire to acquire wisdom that may transfer back to hand-designed optimizers.[34] Using RL, Li and Malik developed learned optimizers for different classes of convex and nonconvex objective functions and showed that the autonomous optimizers converge in fewer steps and/or reach better optima than hand-designed optimizers.[32] Ahuja et al. have designed an RL-based optimizer that adds a corrective term to the BFGS algorithm.[36]

Motivated by the aforementioned methods, we introduce MolOpt, a novel approach for autonomous molecular geometry optimization (MGO) that utilizes multiagent reinforcement learning (MARL). By defining the input as an atomic environment vector (AEV) and forces on each atom, we are able to develop an RL-based model known as MolOpt, which outputs displacements of each atom in the molecule. These displacements are used to update the Cartesian coordinates of the atoms in the molecule. MolOpt is a MARL-based model in which each atom is treated as an agent. This formulation allows us to use AEVs as input; this mitigates several problems, viz. 1. Due to MARL formulation, the policy is defined for atoms; hence, MolOpt can handle the different sizes of molecules and is permutationally invariant. 2. As we used AEV as input, we had a fixed-size vector incorporating rotational and translational invariance into the model. All of these above properties enable us to output action as displacement, which can be used to update the molecular structure directly in the Cartesian coordinate. Our model MolOpt is novel because it is a denovo learned optimizer for MGO. MolOpt is independent of other optimizers as it does not require other optimizers for training or testing. In the methods section, we briefly introduced MARL and described the MGO problem as MARL formulation, followed by the data set used for training and testing and "training and implementation details". In the results section, we demonstrated the effect of various input "feature vectors" or state representation s_t and architectural differences on the performance of MolOpt. We show the ability of our learned optimizer, MolOpt, to perform MGO on different classes of alkanes, demonstrating the transferability of our model to different molecules. We also compare MolOpt with other optimizers such as MDMin, FIRE, and BFGS. MolOpt is the first of its kind to apply reinforcement learning to MGO without any dependence on other hand-designed optimizers. Our work serves as a proof-of-concept for the potential of MARL in this domain, opening up new research directions for MGO. The main contributions to the paper are as follows:

- Formulation of molecular geometry optimization as Multiagent RL (MARL) problem. Where each atom is an agent, thus allowing us to have the same policy across different molecular sizes and mitigate the problem of permutation transformation with the molecules.

- We have developed a *"learned optimizer"* which is in contrast to the hand-designed optimizers available for MGO.

- Our MolOpt model is an nonhistory based *"learned optimizer"* which needs only a single previous state to predict actions. Which is in contrast to other models which need multiple previous states as observation to predict next action.

- MolOpt's learned policy incorporates the principles of chemistry to optimize molecular geometry with a gradient-based local optimization approach.

- Transferability. Our optimizer trained on small molecules, i.e., ethane and butane (includes 2 isomers) can be used to optimize larger molecules such as heptane (includes 9 isomers) and octane (includes 18 isomers).

2. METHOD

2.1. Preliminaries. In reinforcement learning, the agent chooses actions $a_t \in \mathcal{A}$ at each time step t, thus changing the state $s_t \in \mathcal{S}$ of the environment in a random manner, and gets feedback based on the outcome of the action. The feedback is commonly provided as a reward or cost $r_t \in \mathcal{R}$. The agent's goal is to take appropriate actions based on the observation/state s_t that maximizes the cumulative reward or minimizes cumulative cost over all time steps.

To this effect, we formulate MGO as an RL problem by defining the potential energy surface (PES) as a gamelike environment for repeated exploration of conformation space. An essential aspect of solving an RL problem is by learning a policy using a neural network that can predict appropriate actions by observing different states and points along the surface of the objective function, PES, in this case. Finally, we train our model using a popular RL algorithm known as proximal policy optimization (PPO) to learn the optimal policy. In the following subsections, we formulate geometry optimization as an RL problem.

2.2. Multiagent Reinforcement Learning (MARL). A reinforcement learning problem is generally represented as a Markov decision process (MDP). We define finite horizon MDP with continuous state and action space as a tuple $(\mathcal{S}, \mathcal{A}, p_o, p, \mathcal{R}, \gamma)$, where a set of states $\mathcal{S}$ encodes the information or knowledge about the environment at various moments of time; a set of actions or decisions $\mathcal{A}$ that helps to move from one state to another; $p_o: \mathcal{S} \to \mathcal{R}^+$ is the probability density over initial states, a transition probability function $p: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R}^+$ that defines the probability of moving from one state to another on taking a particular action; and a reward function $\mathcal{R}: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that defines the goodness or badness of taking a particular action at some given state and $\gamma \in (0, 1]$ is the discount factor.

By solving the RL problem, we aim to learn a stochastic policy $\pi: \mathcal{S} \times \mathcal{A} \to \mathcal{R}^+$, which is a conditional probability density over actions given the current state, such that the expected cumulative reward is maximized. In other words, we aim to find a mapping from states to action, called a policy (denoted as π), with a maximum achievable total reward. The policy π is often parametrized using a neural network.

To formulate geometry optimization as an MDP, let us consider an N atom molecule having Cartesian coordinates $x \in \mathbb{R}^{3N}$. Here we define each atom as an agent making this formulation a Multiagent MDP[37] (MMDP). However, MMDP's presume all agents get the same reward. Shapley, in 1953 introduced stochastic Games (aka Markov Games), in which he allowed a unique reward function for each agent.[38]

Partially-Observable Stochastic Games ("POSG") (Lowe et al., 2017),[39] defined below, is an extension of Stochastic Games to the situation in MDP where we have only a partially observable state (similar to a partially observable MDP),[37] and is the model we use throughout this paper.

Definition: POSG is a tuple $\langle \mathcal{S}, \mathcal{N}, \{\mathcal{A}_i\}, \mathcal{P}, \{\mathcal{R}_i\}, \{\Omega_i\}, \{O_i\} \rangle$ where

- $\mathcal{S}$ is set of all possible states.
- $\mathcal{N}$ is number of agents. The set of agents in $[\mathcal{N}]$.
- $\mathcal{A}_i$ is the set of possible actions for agent i.
- $\mathcal{P}: \mathcal{S} \times \prod_{i \in [\mathcal{N}]} \mathcal{A}_i \times \mathcal{S} \to [0,1]$ is the (stochastic) transition function.
- $\mathcal{R}_i: \mathcal{S} \times \prod_{i \in [\mathcal{N}]} \mathcal{A}_i \to \mathbb{R}$ is the reward function for agent i.
- Ω_i is the set of possible observations for agent i.
- O_i: $\mathcal{A}_i$, $\times$ $\mathcal{S} \times \Omega_i$ is the observations function.

2.2.1. Parameter Sharing. "Nonstationarity" is a fundamental problem in cooperative MARL.[40] Each agent's policy evolves during learning, while it is also part of the environment from the perspective of other agents. This is known as the ringing effect, in which the information oscillates between agents during learning, significantly slowing the convergence. Increasing centralization during learning can mitigate the problem of slow convergence due to nonstationarity.[41] One of the centralized cases of learning is parameter sharing. The concept of parameter sharing is quite common throughout deep learning. In MARL, parameter sharing[42,43] refers to a learning algorithm that acts on behalf of every agent while using and making updates to a collectively shared policy.[42]

2.3. Atomic Environment Vector (AEV). The atomic environment vector (AEV) captures the atomic environment around each atom. AEVs are constructed from "symmetry functions", which encode each atom's radial and angular environment. As described by Smith et al. in ref 44, we have used a modified version of the original Behler and Parrinello symmetry function (BPSF).[1] AEV comprises a radial part and an angular part that encode the radial and angular environments around the atom, respectively. As summarized in Table 4, we have five variants of the MolOpt model, each employing different state representations and architectures. A detailed discussion of these five variants can be found in the results section. This section explicitly highlights the AEV component of the state representation used in these variants. For variants 1 and 2, we utilized 32 evenly spaced radial shifting parameters for the radial part and eight radial and eight angular shifting parameters for the angular part. Given that there are two atom types (C and H), this results in a 256-length AEV, where radial and angular parts are 64 and 192 lengths, respectively. In contrast, for variants 4 and 5, we employed 16 evenly spaced radial shifting parameters for the radial part and eight radial and four angular shifting parameters for the angular part, which results in a 128-length AEV, where radial and angular parts are 32 and 96 lengths, respectively.

2.4. Formulation. Coming back to geometry optimization as an MDP, let us consider an N atom molecule having Cartesian coordinates $x \in \mathbb{R}^{3N}$. We define each atom as an agent making this formulation an multiagent MDP[37] (MMDP); the number of agents $\mathcal{N}$ depends on the number of atoms in the molecule. As seen in Figure 1 we compute rotationally and translationally invariant state representation s_t at time step t. State representation s_t encodes the 3D structure of the molecule by the virtue of AEV, atom type and unit forces F_x, F_y, F_z. Each of these entities of state s_t are for a single atom in the molecule, i.e., dimensions of AEV = $N \times 128$, atom-type one-hot vector = $N \times$ number of atomic species (2 in our case) and unit forces = $N \times 3$, hence dimension of s_t = $N \times 133$. It should be noted that the state representation s_t is different for different variants, see Table 4.

- State, s_t = [AEV, atom-type (one-hot vector), F_x^t, F_y^t, F_z^t] where F_x^t, F_y^t, F_z^t are the unit forces in x,y,z direction at time t.
- Action, a_t = $[D_x, D_y, D_z]$ (displacement of atom in x,y,z direction).
- Reward, r_t = total reward (see eq 3), where F_r is the resultant force (eV/Å) on each atom.

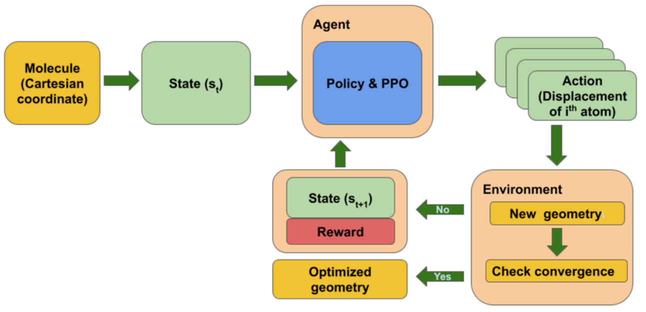


Figure 1. Workflow of the MolOpt model. The molecule's structure is in Cartesian coordinates, which are used to compute rotational and translational invariant state representation. State representation consists of atomic environment vector (AEV), one-hot encoding of atom type, and unit forces. The shared policy network receives atomic state representation as an observation and predicts actions based on those observations. These actions are modeled as displacements of each atom of the molecule in the Cartesian coordinate system. As intended, the actions produce displacement of each atom, resulting in a new conformation. We then calculate the new conformation's energy and forces to check if optimization has reached convergence and calculate reward to check the goodness and badness of the action. If convergence is not achieved, we compute the invariant state representation of the new conformation and repeat the process.

$$\text{atomic reward} = \begin{cases} -1 & \text{if } F_r > 10 \\ 0 & \text{if } F_r < 10 \text{ or } F_r > = 1 \\ 1 & \text{if } F_r < 1 \text{ or } F_r = 0.1 \\ 20 & \text{if } F_r < 0.1 \text{ or } F_r > = 0.01 \\ 500 & \text{if } F_r < 0.01 \\ 2000 & \text{if } F_r^{\max} < 0.01 (\text{converged}) \end{cases} \tag{1}$$

$$\text{team reward} = \text{mean}(-\log(F_r)) \tag{2}$$

$$\text{total reward} = \text{atomic reward} + \text{team reward} \tag{3}$$

The atomic agents take in the state representations as observations and use policy (π) to predict atomic actions (a_t). The next conformation in the optimization trajectory is obtained by using the actions (atomic displacements) and the previous conformation as follows $x_{t+1} = x_t + a_t$. As we can compute state representation s_t from x_t (Cartesian coordinates) we can write $s_{t+1} = s_t + a_t$. Now that we have a new molecular conformation in the optimization trajectory, we check for optimization convergence. If convergence is reached, we stop the optimization; if convergence is not reached, we again compute the state representation of new conformation from Cartesian coordinates. It is passed to the agent, and this loop continues as shown in Figure 1. We use the proximal policy optimization (PPO) algorithm to train our policy network. PPO can be categorized as the policy-based RL method that aims to learn the optimal policy (π^*). A policy is a mapping function that predicts the appropriate action given a state that results in the maximum possible cumulative reward.

We have designed a custom reward function (eq 3), which is a combination of the atomic reward and team reward (molecular reward). Now, why is this custom reward function important? One must realize that in our multiagent RL (MARL) formulation we predict atomwise actions, which leads to the "nonstationarity" problem. There are various ways to mitigate the "nonstationarity" problem; in our work, we use the reward function as an in-direct communication method to overcome the "nonstationarity" problem. Using the team reward function component, we provide the agent with information about what is happening with other agents (atoms). As shown in eq 3, the total reward is the summation of the atomic and team rewards. The atomic reward is given to each atom based on the resultant force (F_r) on that particular atom (see eq 1). Atomic reward aims to reduce the forces on an individual atom. In molecular geometry optimization, the displacement of a single atom leads to changes in the forces of other multiple other atoms. Hence, we introduce team reward to prevent actions that will cause the F_r on other atoms to increase dramatically. Team reward eq 2 aims to reduce forces on all atoms without drastically increasing forces on other atoms. In eq 1, $F_r^{\max}$ is the maximum resultant force on a particular atom in a molecule. If $F_r^{\max}$, which represents the largest resultant atomic force in the molecule, is below 0.01 eV/Å, we conclude that the optimization has converged and terminated the episode.

2.5. Data Set. We generated a data set containing initial conformers of alkanes known as ALINCO. ALINCO data set contains geometries of ethane, *n*-butane, and isobutane. Below are the steps followed to generate the ALINCO data set.

1. From SMILES generate "10*n atoms" structures for each isomer using RDKit.[45] Therefore, we get a total of 360 structures consisting of 80, 140, and 140 of ethane, *n*-butane, and isobutane, respectively.
2. We then add random noise with mean = 0 and std = 0.1 to the geometries generated in step 1. We repeat this step 5 times, generating 5x, giving 1800 structures.
3. Combine the initial 360 and perturbed 1800 structures to get a total of 2160 structures.
4. Optimize these 2160 structures using the BFGS algorithm provided in the Atomic Simulation Environment (ASE)[46] package. We use ANI1ccx[47] for energy and force calculation.
5. From optimization trajectory sample 0, 1, 2, 4, 6, 8 initial frames and also sample every *i*th frame from 10th till 10th last frame at the interval of 5.

In total, ALINCO has 42,262 structures of ethane, *n*-butane, and isobutane.

2.6. Training and Implementation Details. We implemented MolOpt using RLLIB and PPO implemented in RLLIB to train the policy network. The objective function in eq 4 is optimized using PPO. In eq 4 the parametrized policy is given as $\pi_\theta(a_t|s_t)$ which predicts action a_t given the state s_t, and $\pi_{\theta\text{old}}(a_t|s_t)$ represents the older iteration of the policy network. The clipping parameter ε guarantees that while updating the policy, we do not make excessively large updates, and we have set its value to 0.3. The advantage function $A(s_t, a_t)$ is defined as the advantage function that determines how good, or bad, the action a_t is on an average for a given state s_t. Other hyperparameters used during training are listed in Table 1.

$$L(\theta) = \mathbb{E}_t\left[\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}(a_t|s_t)}}A(s_t, a_t),\right.\right.$$
$$\left.\left.\text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon\right)A(s_t, a_t)\right)\right] \tag{4}$$

Table 1. Hyperparameter Used during Training and Evaluation of MolOpt

parameters	value
entropy coefficient	0.0001
KL coefficient	1.0
KL target	0.01
gamma	1.0
clip parameter	0.3
vf clip parameter	10.0
horizon	20
lr	5×10^{-5}
train batch size	2048

To train our model, we sample initial conformation from the ALINCO data set. We then compute rotationally and translationally invariant state representation (s_t). Agent takes (s_t) as an input and predicts actions (a_t). We get a new conformation at time $(t + 1)$ in the optimization trajectory. We then calculate the energy and forces of the new conformation using ANI1ccx.[47] For the agent to understand whether the action a_t taken was good or bad, we calculate the reward as given in equation 3.

Once trained, we evaluate the model using a test set that contains alkanes (C_nH_{2n+2}) where n = 3, 5, 6, 7, and 8. We generate 10 structures each of 36 isomers across different alkanes (C_nH_{2n+2}) where n = 3, 5, 6, 7, and 8 using RDKit.[45] In total, we have 360 structures in our test set.

3. RESULTS

In this section, we show the ability of our learned optimizer, MolOpt, to perform geometry optimization on different classes of alkanes. We compare five different variants of MolOpt based on their state representation and architectural differences. The performance evaluation metric for MolOpt is based on energy and all-atom RMSD. The all-atom RMSD is calculated by aligning the molecules to remove any translational and rotational transformations prior to computation. All models were trained on ALINCO data set and evaluated on a test set containing 360 structures of alkanes; details about the number of isomers and structures in the test set are summarized in Table 2.

Table 2. Test Set That Contains 360 Structures of Alkanes (C_nH_{2n+2}) where n = 3, 5, 6, 7, and 8

molecule name	no. isomers × no. structures	no. atoms
propane	1 × 10	11
pentane	3 × 10	17
hexane	5 × 10	20
heptane	9 × 10	23
octane	18 × 10	26
total	360	

The performance of these five variants has been summarized in Table 3. We can see that variant 5, which has state representation $s_t = [\text{AEV, atom-type}, F_x, F_y, F_z, \Delta F_x, \Delta F_y, \Delta F_z]$ achieves the best performance on the test set containing 360 structures with mean $\Delta E = -0.57$ kcal/mol and standard deviation $\Delta E = 0.67$ kcal/mol. In terms of RMSD, variant 5 achieves an overall mean RMSD of 0.107 ± 0.078 Å. We further discuss each of these variants in the following subsections.

3.1. Flavours of MolOpt/Ablation Study.

3.1.1. Variant 1. We consider variant 1 as the baseline. As seen in Table 4, the state representation (s_t) of variant 1 consists of atomic

Table 3. Geometry Optimization Performance of Different Flavors of MolOpt on a Test Set Containing 360 Structures[a]

variant	no. structures	mean ΔE (kcal/mol)	STD ΔE (kcal/mol)	mean RMSD (Å)	STD RMSD (Å)
variant 1	360	−3.51	2.12	0.18	0.08
variant 2	360	−1.64	1.25	0.15	0.09
variant 3	360	−1.34	1.14	0.14	0.07
variant 4	360	−0.99	0.94	0.13	0.08
variant 5	360	−0.57	0.67	0.11	0.08

[a]Mean $\Delta E = \frac{1}{N}\sum_{i=0}^{N} E^i_{\text{BFGS}} - E^i_{\text{MolOpt}}$ both E^i_{BFGS} and E^i_{MolOpt} are optimized energies and i runs over the structures in the test set, i.e., N = 360. Similarly, STD ΔE represents the standard deviations within the ΔE. We calculate the all-atom root mean squared deviation (RMSD) between the optimized BFGS structure and the optimized MolOpt structure. The mean RMSD $= \frac{1}{N}\sum_{i=0}^{N} \text{RMSD}^i$ where i runs over the structures in the test set. The STD RMSD represents the standard deviations with the RMSD.

environment vectors (AEV) of length 256, atom-type as the one-hot vector of length 2, unit forces of length 3, and resultant force of length 1, hence the length of s_t is 262.

The idea is to provide the model with the local atomic environment of each atom using AEV. Unit forces and the resultant force on each atom provide the model with the gradient information. The unit forces are defined as $F_x = \frac{dE}{dx}$, $F_y = \frac{dE}{dy}$, $F_z = \frac{dE}{dz}$ and resultant force is $F_r = \frac{dE}{dr}$, where E is the energy and r is the position. The policy network has four multilayered perceptrons (MLP) known as MLP_{aev}, MLP_{f}, MLP_{int}, and MLP_{v}. MLP_{aev} and MLP_{f} produce refined aev and force features, respectively. We add these features and pass them through the interaction MLP, MLP_{int}, which predicts actions. We use separate MLP_{v}, which acts as a value function. The architecture of these MLPs is shown in Table 4. We use LeakyReLU in MLP_{aev} and tanh for all other MLPs as the activation function.

Performance of variant 1 is summarized in Supporting Information (SI) Tables S1 and S2. We can see that the mean ΔE and mean RMSD values increase with the size of the alkanes. Variant 1 achieves the overall mean ΔE of −3.51 ± 2.12 kcal/mol and RMSD of 0.18 ± 0.08 Å. In SI Table S2, the overall mean RMSD before optimization is 0.25 ± 0.07 Å and after optimization mean RMSD is 0.18 ± 0.08 Å. Considering that it is a baseline model, these results are very encouraging.

3.1.2. Variant 2. Variant 2 is similar to variant 1, except the s_t does not have resultant force F_r hence the length of s_t is 261. We have excluded the resultant force, as its information is already present in unit forces, and the resultant force in itself does not add much to improve the accuracy of the MolOpt model. Performance of variant 2 is summarized in SI Table S3 and S4. Variant 2 achieves the overall mean ΔE of −1.63 ± 1.24 kcal/mol and RMSD of 0.15 ± 0.09 Å. In SI Table S4, the overall Mean RMSD before optimization is 0.25 ± 0.07 Å and after optimization mean RMSD is 0.15 ± 0.09 Å.

3.1.3. Variant 3. The idea is to evaluate the performance of the MolOpt model in the absence of local chemical environment information; hence variant 3 receives only information about the type of atoms and the forces on each atom; AEV is excluded from the state representation as shown in Table 4 therefore the length of s_t is 5. The architecture of the policy network is shown in Table 4. We use tanh as the activation function for all MLPs.

Table 4. Comparison of Five Different Variants of MolOpt Based on Their State Representation and Architectural Differences[a].

variants	state	architecture
variant 1	[Atom-type, AEV, F_x, F_y, F_z, F_r]	$\text{MLP}_{\text{aev}} = 262 \times [128]^3$ $\text{MLP}_{\text{f}} = 4 \times [128]^4$ $\text{MLP}_{\text{int}} = 128 \times 3$ $\text{MLP}_{\text{v}} = 262 \times 1$
variant 2	[Atom-type, AEV, F_x, F_y, F_z]	$\text{MLP}_{\text{aev}} = 261 \times [128]^3$ $\text{MLP}_{\text{f}} = 5 \times [128]^4$ $\text{MLP}_{\text{int}} = [128]^3 \times 3$ $\text{MLP}_{\text{v}} = 261 \times [128]^2 \times 1$
variant 3	[Atom-type, F_x, F_y, F_z]	$\text{MLP}_{\text{f}} = 5 \times [128]^5 \times 3$ $\text{MLP}_{\text{v}} = 5 \times [128]^3 \times 1$
variant 4	[Atom-type, AEV, F_x, F_y, F_z]	$\text{MLP}_{\text{aev}} = 133 \times [128]^3$ $\text{MLP}_{\text{f}} = 5 \times [128]^4$ $\text{MLP}_{\text{int}} = [128]^3 \times 3$ $\text{MLP}_{\text{v}} = 133 \times [128]^2 \times 1$
variant 5	[Atom-type AEV, F_x, F_y, F_z, ΔF_x, ΔF_y, ΔF_z]	$\text{MLP}_{\text{aev}} = 136 \times [128]^4$ $\text{MLP}_{\text{f}} = 8 \times [128]^3$ $\text{MLP}_{\text{int}} = [128]^3 \times 3$ $\text{MLP}_{\text{v}} = 136 \times [128]^2 \times 1$

[a]We represent $128 \times 128 \times 128$ as $[128]^3$ and so forth in the architecture column. MLP stands for multi-layered perceptron. Subscripts aev stand for the atomic environment vector, f for forces, int for interaction, and v for the value function. F_x, F_y, F_z, ΔF_x, ΔF_y, ΔF_z are unit and delta unit forces in the x,y,z direction respectively. F_r is the resultant force.

Performance of variant 3 is summarized in SI Tables S5 and S6. Variant 3 achieves the overall mean ΔE of -1.33 ± 1.14 kcal/mol and RMSD of 0.14 ± 0.07 Å. In SI Table S6, the overall mean RMSD before optimization is 0.25 ± 0.07 Å and after optimization mean RMSD is 0.14 ± 0.07 Å.

3.1.4. Variant 4. From variant 3, we see that there is a slight improvement in the performance after the removal of AEV from the state representation. This necessitates changes in the AEV; therefore, variant 4 has an AEV of length 128, contrary to the 256 used in the previous variants. The length of s_t is 133 due to reduction in length of AEV. Variant 4 is similar to variant 2, except AEV has been changed.

Performance of variant 4 is summarized in SI Tables S7 and S8. Variant 4 outperforms all previous variants to achieve the overall mean ΔE of -0.98 ± 0.95 kcal/mol and RMSD of 0.13 ± 0.08 Å. In SI Table S8, the overall mean RMSD after optimization mean RMSD is 0.13 ± 0.08 Å.

3.1.5. Variant 5. Variant 5 is our best-performing variant. In variant 5, the state representation (s_t) consists of atomic environment vectors (AEV), atom-type as one-hot vector, unit forces, and unit delta forces as shown in Table 4, the length of s_t is 136.

Here variant 5 receives not only information about the atomic environment, type of atoms, and unit forces but also information about changes in forces on each atom. Unit force $F = \frac{\text{d}E}{\text{d}r}$ and change in force $\Delta F = \frac{\text{d}F}{\text{d}r}$, where E is the energy and r is the position.

Performance of variant 5 is summarized in SI Tables S9 and S10. Extra added information about the change in forces in the state representation improves the performance of variant 5. Variant 5 achieves the overall mean ΔE of -0.57 ± 0.67 kcal/mol and RMSD of 0.10 ± 0.08 Å. In SI Table S10, the overall mean RMSD after optimization mean RMSD is 0.10 ± 0.08 Å. In Figure 2 we have shown how the RMSD values decrease after optimization using the MolOpt model and how the RMSD values increase as the size of the alkanes increases. The RMSD values are calculated by using BFGS optimized structures as reference; hence, zero in Figure 2 indicates BFGS optimized structures.

As can be seen, RMSD values are very close to BFGS optimized structures. In next section we benchmark variant 5 with few other optimizers, viz., FIRE and MDMin.

3.2. MolOpt (Variant 5) Benchmark. Over the past several decades, extensive work has delivered many popular optimization methods, such as steepest descent, conjugate gradient, Newton−Raphson, and BFGS, to name a few. These methods share one commonality: they are all hand-designed, i.e., human experts carefully ideate, design, and validate these algorithms, hence developing these algorithms is a laborious process. Taking inspiration from DL, which was able to automate feature design,

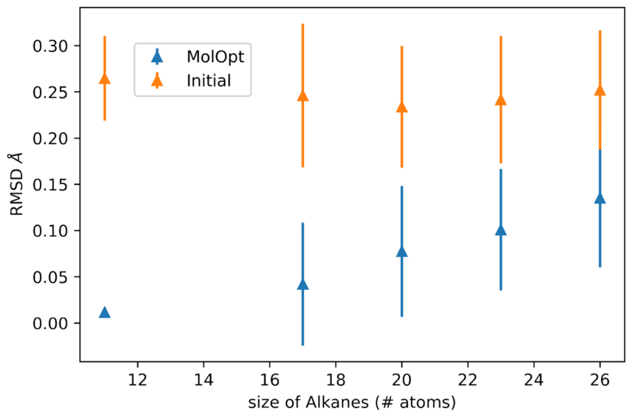


Figure 2. Geometry optimization of 360 structures using variant 5. The plot shows the difference in the RMSD values of the different classes of alkanes before and after optimization.

we have tried to automate algorithm design and learn a "learned optimizer" that may outperform current hand-designed optimizers. This section shows how our MolOpt model, a "learned optimizer", compares with other optimizers. We have compared our model MolOpt with three other optimizers, viz., MDMin, FIRE,[20] and BFGS.[28−31] All three optimizers are provided in the ASE package.[46]

The MDMin is a modified version of the typical velocity-Verlet MD method, which numerically solves Newton's second law. However, the dot product between the momenta and the forces is evaluated at each time step. If the dot product is zero, then it means that the system has just departed through a (local) minima on the PES; the kinetic energy is large and about to decrease again. At this juncture, the momentum is set to zero. Contrary to MD, all atomic masses are set to one. Fast inertial relaxation engine (FIRE)[20] is based on conventional MD with further velocity modifications and adaptive time steps. The Broyden, Fletcher, Goldfarb, and Shanno algorithm, or BFGS Algorithm, is a second-order optimization algorithm. BFGS is an example of Quasi-Newton methods in which an approximate Hessian is computed for optimization problems, where the second derivative is very expensive to calculate and cannot be computed for all practical purposes. The BFGS is one of the most widely used second-order algorithms for numerical optimization.

As seen from the SI Table S11, we perform better than the MDMin optimizer by achieving an overall mean ΔE of 0.97 ± 1.06 kcal/mol. The positive mean indicates that MolOpt, on average, reached energy levels that were lower than the MDMin optimized energy by 0.97 kcal/mol. We have summarized MolOpt comparison with FIRE and BFGS in SI Tables S12 and S13, respectively. Our model achieves an overall mean ΔE of

-0.18 ± 0.25 kcal/mol, an RMSD of 0.04 ± 0.04 Å and ΔE of -0.53 ± 0.64 kcal/mol, an RMSD of 0.10 ± 0.08 Å compared to FIRE and BFGS, respectively. In Figures 3, 4, 5, and 6, we have compared the geometry optimization trajectory of four different structures with empirical formulas C_5H_{12}, C_7H_{16} and C_8H_{18} with different optimizers. MolOpt performance is better than that of MDMin and similar to that of FIRE.

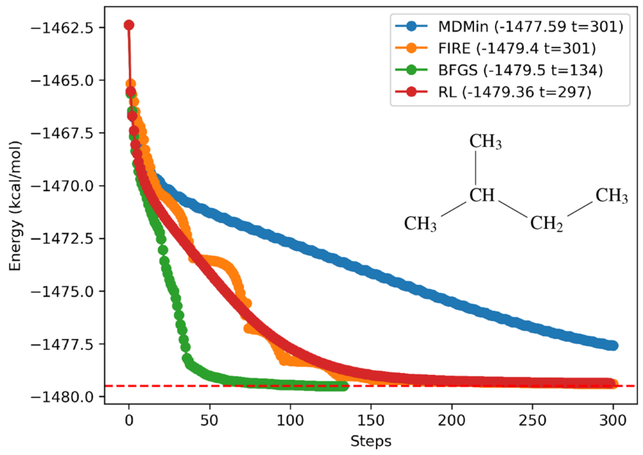


Figure 3. Plot shows geometry optimization trajectory of isopentane (empirical formula C_5H_{12}) using four different optimizers, viz., MDMin, FIRE, BFGS, and RL.

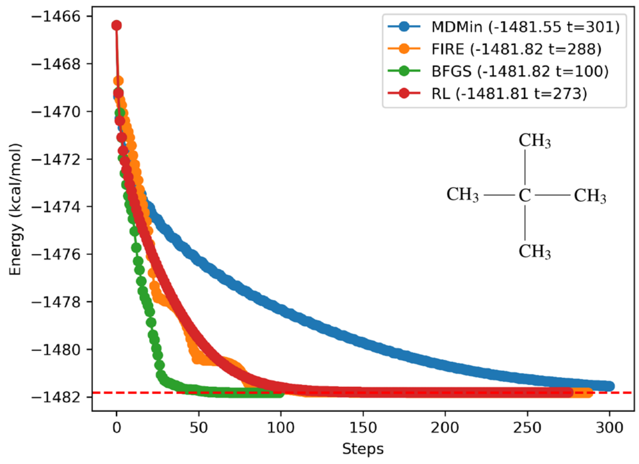


Figure 4. Plot shows geometry optimization trajectory of neopentane (empirical formula C_5H_{12}) using four different optimizers, viz., MDMin, FIRE, BFGS, and RL.

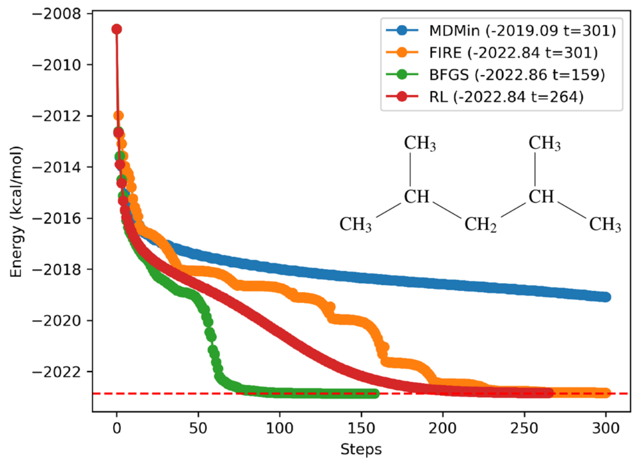


Figure 5. Plot shows geometry optimization trajectory of 2,4-dimethyl pentane (empirical formula C_7H_{16}) using four different optimizers, viz., MDMin, FIRE, BFGS, and RL.

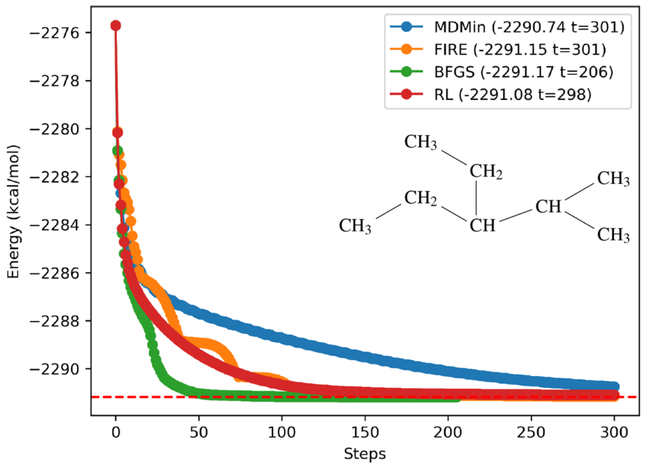


Figure 6. Plot shows geometry optimization trajectory of 3-ethyl-2-methyl pentane (empirical formula C_8H_{18}) using four different optimizers, viz., MDMin, FIRE, BFGS, and RL.

Unlike BFGS, which iteratively approximates the Hessian matrix using information from all previous time steps, MolOpt receives state information only at the current time step t, resulting in a less accurate inverse Hessian estimation. As a result, MolOpt may struggle to predict step sizes as precisely as BFGS, leading to more steps being required to reach the minimum.

For the sake of completeness, we also compare the performance of MolOpt in terms of speed or computational time required to do 72 optimizations. Out of 360 structures in the test set we pick every 5th structure for optimization, hence the number 72. This was done to reduce the computational time and cost. In SI Table S14 we show time taken by each method to complete these 72 optimization. It is important to acknowledge that other established methods, available as packages, have been in existence for a long time with optimized code for enhanced performance. In contrast, our code was primarily developed to introduce novel algorithms and has yet to undergo optimization for performance.

4. CONCLUSION AND DISCUSSION

We have introduced MolOpt, a robust MARL-based algorithm for MGO. This work serves as a proof-of-concept for the potential of MARL in MGO. The formulation of MGO as a MARL problem where each agent corresponds to a single atom in the molecule allows us to use the same model architecture across different molecular sizes. MolOpt is a "learned optimizer", which is in contrast to the hand-designed optimizer available for MGO. We were able to incorporate chemistry into the learned optimizer. We trained our optimizer on a few alkane molecules, i.e., ethane and butane (which include two isomers of butane). We tested MolOpt on different and comparatively larger alkane molecules not present in the training set, such as propane, pentane, hexane, heptane (which consists of 5 isomers), and octane (which consists of 9 isomers). The performance of MolOpt on the test set, which also contains larger alkane molecules, such as heptane and octane, demonstrates its transferability to other molecules.

Currently, MolOpt uses AEVs, atom type (one-hot encoding), and forces as the state representation of each atom. Moreover, each agent can see only its own state. In the future, we aim to extend MolOpt to include information sharing and communication between the agents during optimization. In addition, we plan to explore the use of more advanced optimization methods, such as iteratively approximating the Hessian similar to BFGS, to improve MolOpt's performance. It should be noted that our focus in this paper was on demonstrating the feasibility of using MARL for autonomous molecular geometry optimization without any dependency on other hand-designed optimizers. We believe there is still scope for improvement and exploration to develop better MGO algorithms using RL. We hope our work will inspire and motivate further research in MGO using RL. As an afterthought, MolOpt has shown promise in predicting atom displacement using gradients and can potentially be used for molecular dynamics (MD) simulations. Training MolOpt to generate an MD simulation trajectory can offer a new perspective toward molecular dynamics simulations.

Further enhancements in algorithm and data are necessary to extend MolOpt's optimization capabilities to molecules containing elements such as C, H, N, and O. Future work in this area is ongoing in our group.

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jpcb.3c04771.

Tables S1 to S10 contains MGO performance of MolOpt variant 1 to 5. Benchmark results, comparing MolOpt variant 5 with other optimizers (MDMin, FIRE and BFGS) w.r.t. different alkanes are summarized in SI Tables S11 to S13. Wall clock time taken by each optimizer is summarized in SI Table S14 (PDF)

AUTHOR INFORMATION

Corresponding Author

U. Deva Priyakumar − *Center for Computational Natural Sciences and Bioinformatics, International Institute of Information Technology, Hyderabad 500032, India;* orcid.org/0000-0001-7114-3955; Email: deva@iiit.ac.in

Authors

Rohit Modee − *Center for Computational Natural Sciences and Bioinformatics, International Institute of Information Technology, Hyderabad 500032, India;* orcid.org/0000-0002-2012-8031

Sarvesh Mehta − *Center for Computational Natural Sciences and Bioinformatics, International Institute of Information Technology, Hyderabad 500032, India*

Siddhartha Laghuvarapu − *Center for Computational Natural Sciences and Bioinformatics, International Institute of Information Technology, Hyderabad 500032, India*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jpcb.3c04771

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

R.M. thanks TCS for the research fellowship. U.D.P. thanks DST-SERB (CRG/2021/008036) and IHub-Data, IIIT Hyderabad, for their support.

REFERENCES

(1) Behler, J.; Parrinello, M. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.* **2007**, *98* (14), 146401.

(2) Modee, R.; Laghuvarapu, S.; Priyakumar, U. D. Benchmark study on deep neural network potentials for small organic molecules. *J. Comput. Chem.* **2022**, *43* (5), 308−318.

(3) Schütt, K. T.; Arbabzadah, F.; Chmiela, S.; Müller, K. R.; Tkatchenko, A. Quantum-chemical insights from deep tensor neural networks. *Nature Communications* **2017**, *8*, 6−13.

(4) Schütt, K. T.; Sauceda, H. E.; Kindermans, P. J.; Tkatchenko, A.; Müller, K. R. SchNet - A deep learning architecture for molecules and materials. *J. Chem. Phys.* **2018**, *148* (24), 241722.

(5) Unke, O. T.; Meuwly, M. PhysNet: A Neural Network for Predicting Energies, Forces, Dipole Moments, and Partial Charges. *Journal of Chemical Theory and Computation* **2019**, *15* (6), 3678−3693.

(6) Modee, R.; Agarwal, S.; Verma, A.; Joshi, K.; Priyakumar, U. D. DART. Deep learning enabled topological interaction model for energy prediction of metal clusters and its application in identifying unique low energy isomers. *Phys. Chem. Chem. Phys.* **2021**, *23* (38), 21995−22003.

(7) Laghuvarapu, S.; Pathak, Y.; Priyakumar, U. D. BAND NN: A Deep Learning Framework for Energy Prediction and Geometry Optimization of Organic Small Molecules. *J. Comput. Chem.* **2020**, *41* (8), 790−799.

(8) Noé, F.; Tkatchenko, A.; Müller, K. R.; Clementi, C. Machine learning for molecular simulation. *Annu. Rev. Phys. Chem.* **2020**, *71*, 361−390.

(9) Bjerrum, E. J.; Margreitter, C.; Blaschke, T.; Kolarova, S.; de Castro, R. L. R. Faster and more diverse de novo molecular optimization with double-loop reinforcement learning using augmented SMILES. *J. Comput. Aided. Mol. Des.* **2023**, *37* (8), 373−394 arXiv:2210.12458.

(10) Popova, M.; Isayev, O.; Tropsha, A. Deep reinforcement learning for de novo drug design. *Sci. Adv.* **2018**, *4* (7), 1−28 arXiv:1711.10907.

(11) Zhou, Z.; Kearnes, S.; Li, L.; Zare, R. N.; Riley, P. Optimization of Molecules via Deep Reinforcement Learning. *Sci. Rep.* **2019**, *9* (1), 10752 arXiv:1810.08678.

(12) Modee, R.; Verma, A.; Joshi, K.; Deva Priyakumar, U. MeGen - generation of gallium metal clusters using reinforcement learning. *Mach. Learn. Sci. Technol.* **2023**, *4* (2), 025032.

(13) Gogineni, T.; Xu, Z.; Punzalan, E.; Jiang, R.; Kammeraad, J.; Tewari, A.; Zimmerman, P. TorsionNet: A reinforcement learning approach to sequential conformer search. *Adv. Neural Inf. Process. Syst.* **2020**, 1−12. arXiv:2006.07078

(14) Shin, K.; Tran, D. P.; Takemura, K.; Kitao, A.; Terayama, K.; Tsuda, K. Enhancing Biomolecular Sampling with Reinforcement Learning: A Tree Search Molecular Dynamics Simulation Method. *ACS Omega* **2019**, *4* (9), 13853−13862.

(15) Shamsi, Z.; Cheng, K. J.; Shukla, D. Reinforcement Learning Based Adaptive Sampling: REAPing Rewards by Exploring Protein Conformational Landscapes. *J. Phys. Chem. B* **2018**, *122* (35), 8386−8395 arXiv:1710.00495.

(16) Chan, L.; Hutchison, G. R.; Morris, G. M. Bayesian optimization for conformer generation. *J. Cheminform.* **2019**, *11* (1), 1−11.

(17) Fang, L.; Guo, X.; Todorović, M.; Rinke, P.; Chen, X. Exploring the Conformers of an Organic Molecule on a Metal Cluster with Bayesian Optimization. *J. Chem. Inf. Model.* **2023**, *63* (3), 745−752.

(18) Simm, G. N.; Pinsler, R.; Hernández-Lobato, J. M. Reinforcement learning for molecular design guided by quantum mechanics. *arXiv* **2020**, No. 2002.07717v2, DOI: 10.48550/arXiv.2002.07717.

(19) Simm, G. N. C.; Pinsler, R.; Csányi, G.; Hernández-Lobato, J. M. Symmetry-Aware Actor-Critic for 3D Molecular Design. *arXiv* **2011**, No. 2011.12747v1, DOI: 10.48550/arXiv.2011.12747.

(20) Bitzek, E.; Koskinen, P.; Gähler, F.; Moseler, M.; Gumbsch, P. Structural relaxation made simple. *Phys. Rev. Lett.* **2006**, *97* (17), 1−4.
(21) Leach, A. R. Empirical Force Field Models: Molecular Mechanics. In *Molecular Modelling: Principles and Applications*; Prentice Hall, 2001; Chapter 4, pp 165−252.
(22) Nocedal, J.; Wright, S. J. *Numerical optimization*; Springer Series in Operations Research and Financial Engineering; Chapman and Hall/CRC, 2006; pp 1−664.
(23) Schlegel, H. B. Exploring potential energy surfaces for chemical reactions: An overview of some practical methods. *J. Comput. Chem.* **2003**, *24* (12), 1514−1527.
(24) Schlegel, H. B. Geometry optimization. *Wiley Interdiscip. Rev. Comput. Mol. Sci* **2011**, *1* (5), 790−809.
(25) Hadamard, J. *Mémoire sur le problème d'analyse relatif à l'equilibre des plaques élastiques encastrées*; Imprimerie nationale, 1908; Vol. 33.
(26) Cauchy, A. Methode gènèrale pour la rèsolution des systemes d'èqations simultanees. *Comptes Rendus* **1847**, *25* (1847), 536.
(27) Hestenes, M.; Stiefel, E. Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Bur. Stand.* **1934**, *49* (6), 409.
(28) Broyden, C. G. The convergence of a class of double-rank minimization algorithms 1. General considerations. *IMA J. Appl. Math. (Institute Math. Its Appl.* **1970**, *6* (1), 76−90.
(29) FLETCHER, R. New approach to variable metric algorithms. *Comput. J.* **1970**, *13* (3), 317−322.
(30) Goldfarb, D. A Family of Variable-Metric Methods Derived by Variational Means. *Math. Comput.* **1970**, *24* (109), 23.
(31) Shanno, D. F. Conditioning of Quasi-Newton Methods for Function Minimization. *Math. Comput.* **1970**, *24* (111), 647.
(32) Li, K.; Malik, J. Learning to Optimize. *arXiv* **2016**, No. 1606.01885v1, DOI: 10.48550/arXiv.1606.01885.
(33) Egidio, L. N.; Hansson, A.; Wahlberg, B. Learning the Step-size Policy for the Limited-Memory Broyden-Fletcher-Goldfarb-Shanno Algorithm. *arXiv* **2021**, No. 2, No. 2010.01311v2, DOI: 10.48550/arXiv.2010.01311.
(34) arXiv:1810.10180 Metz, L.; Maheswaranathan, N.; Nixon, J.; Daniel Freeman, C.; Sohl-Dickstein, J. Understanding and correcting pathologies in the training of learned optimizers. *arXiv* **2019**, No. 1810.10180v5, DOI: 10.48550/arXiv.1810.10180.
(35) Andrychowicz, M.; Denil, M.; Gomez, S.; Hoffman, M. W.; Pfau, D.; Schaul, T.; Shillingford, B.; de Freitas, N. Learning to learn by gradient descent by gradient descent. *Adv. Neural Inf. Process. Syst.* **2016**, *29* (Nips), 3988−3996. arXiv:1606.04474
(36) Ahuja, K.; Green, W. H.; Li, Y. P. Learning to Optimize Molecular Geometries Using Reinforcement Learning. *J. Chem. Theory Comput.* **2021**, *17* (2), 818−825.
(37) Boutilier, C. Planning, learning and coordination in multiagent decision processes. *Proc 6th Conf. Theor. Aspects Rationality Knowledge* **1996**, 195.
(38) Shapley, L. S. Stochastic Games. *Proc. Natl. Acad. Sci.* **1953**, *39* (10), 1095−1100.
(39) Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *Adv. Neural Inf. Process. Syst.* **2017**, 6380−6391. arXiv:1706.02275
(40) Choi, S. P.; Yeung, D. Y.; Zhang, N. L. An environment model for nonstationary reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2000**, 307−313.
(41) Papoudakis, G.; Christianos, F.; Rahman, A.; Albrecht, S. V. Dealing with Non-Stationarity in Multi-Agent Deep Reinforcement Learning. *arXiv Prepr. arXiv1906.04737arXiv* **1906**, 04737.
(42) Gupta, J. K.; Egorov, M.; Kochenderfer, M. Cooperative Multi-agent Control Using Deep Reinforcement Learning. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 10642 LNAI* **2017**, *10642*, 66−83. arXiv:2005.13625
(43) Chu, X.; Ye, H. Parameter Sharing Deep Deterministic Policy Gradient for Cooperative Multi-agent Reinforcement Learning. *arXiv Prepr. arXiv1710.00336arXiv* **1710**, 00336.
(44) Smith, J. S.; Isayev, O.; Roitberg, A. E. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chem. Sci.* **2017**, *8* (4), 3192−3203.
(45) Landrum, G. *RDKit: A software suite for cheminformatics, computational chemistry, and predictive modeling*; Components 8, 2010
(46) Hjorth Larsen, A.; JØrgen Mortensen, J.; Blomqvist, J.; Castelli, I. E.; Christensen, R.; Dułak, M.; Friis, J.; Groves, M. N.; Hammer, B.; Hargus, C.; Hermes, E. D.; Jennings, P. C.; Bjerre Jensen, P.; Kermode, J.; Kitchin, J. R.; Leonhard Kolsbjerg, E.; Kubal, J.; Kaasbjerg, K.; Lysgaard, S.; Bergmann Maronsson, J.; Maxson, T.; Olsen, T.; Pastewka, L.; Peterson, A.; Rostgaard, C.; SchiØtz, J.; Schütt, O.; Strange, M.; Thygesen, K. S.; Vegge, T.; Vilhelmsen, L.; Walter, M.; Zeng, Z.; Jacobsen, K. W. The atomic simulation environment - A Python library for working with atoms. *J. Phys.: Condens. Matter* **2017**, *29* (27), 273002.
(47) Smith, J. S.; Nebgen, B. T.; Zubatyuk, R.; Lubbers, N.; Devereux, C.; Barros, K.; Tretiak, S.; Isayev, O.; Roitberg, A. E. Approaching coupled cluster accuracy with a general-purpose neural network potential through transfer learning. *Nat. Commun.* **2019**, *10* (1), 1−8.