Modeling Crystal Structure Prediction Using Reinforcement Learning: Final Report

Madison Davis
Harvard Computer Science Undergraduate
madisondavis@college.harvard.edu

Patrick Pariseau
Harvard GSAS Bioengineering PhD
patrickpariseau@g.harvard.edu

Pedro Manon
Harvard Computer Science Undergraduate
pedromanon@college.harvard.edu

Abstract

Currently, there are no existing general methods to analytically predict the stable crystal structure of ionic molecules. With a massive state space of potential atomic arrangements–spanning across different elements, their varying possible charges, and their relative position to other atoms in a crystal–computational simulation of new candidate chemicals/materials with a desired set of properties has become common practice. These computational simulators of atomic arrangements in a crystalline solid rely on pre-programmed, simple heuristics that apply to only a small subset of chemicals. For chemical arrangements without these loose sets of rules, iteration by trial and error must be undertaken. Zamaraeva et al. propose the use of reinforcement learning strategies, specifically the REINFORCE policy gradient algorithm, to assess whether policies for finding stable crystal structures can be found.[1] In this paper, we replicate the partial findings of Zamaraeva's work and attempt to push it forward further by implementing Natural Policy Gradient, Proximal Policy Gradient, and varied baseline estimation techniques to reduce variance.

1 Introduction

1.1 'The Crystallization Problem' and Motivation for Using RL

In chemistry, while there exist probabilistic physical rules underlying whether two atoms will bond with one another, the problem of understanding interactions of many atoms collectively forming a crystal is high-dimensional and difficult to solve in an exact analytical formulation. Indeed, two charged atoms, or ions, will tend to form bonds with one another if their interactions (motivated by accepting/donating of electrons) lead to a collective lower energy state. However, when more than two ions are present in a system, the distribution of electron orbitals from one atom amongst multiple other atoms can occur in many different ways. The distribution of orbitals, in turn, will influence the energy of an overall macroscopic atomic structure (i.e., a crystal) containing these atoms. When the number of atoms is high, as in the repeating structure of a crystal, the possible state space governing the sharing of electron orbital distributions explodes. It has been observed, fortunately, that of these potential atomic arrangements/positions of atoms relative to one another, that those arrangements with the lowest energies are most likely to be physically realizable in a laboratory setting. Existing computational crystalline structure simulators incorporate patterns derived from experiment to help guide generation of potential low energy crystals, but these patterns are not general and are limited by the extent of previous experimentation. Thus, these simulators often resort to trial and error based swapping of atoms in a structure to find potential low energy crystal structure solutions. Both from a scientific and engineering perspective, the use of reinforcement learning techniques to derive a policy governing efficient traversal through an energy state space/landscape is of great interest. Indeed, finding and subsequent use of a goal-directed policy, rather than previous guess-and-check methods, may help reveal underlying trends in the stable structures of ionic compounds bearing physical implications. Since these policies correspond to physical actions performed on a crystal's atomic arrangement, these policies may in turn help inform chemists' understanding of the natural laws governing macroscopic atomic interactions.

1.2 Terminology

Before we dive into the procedure the scientists used to tackle the problem, we will define the Reinforcement Learning terms and functions used in the paper:

- MDP: A Markov Decision Process is a framework used to model decision-making by using an action-state environment to identify the best policy, ie the best action to take at any given state. For the purposes of this project, the MDP is modeling crystal growth and it is being used to identify the best actions to take to achieve more desirable crystalline states.
- States: Each state is an ionic chemical structure with an associated energy as a feature. The energy is normalized by a Z-score. The ionic solids are formed in layers called modules. Each layer can be further divided up into small blocks called sub-modules, ranging in size from 1 to 4 atoms. For each state i, there are a total of N_i atoms.
- Actions: A set of $k = 9$ predefined actions, where each action instructs the model how to change a chemical structure. These actions range from swapping two atoms to swapping it with an entirely new structure. In the paper, they are labeled as Actions 1 through 7, 9, and 10. As quoted from the paper, actions 1, 9, and 10 swap the positions of two, three, or $N_i - 1$ atoms in structure i, respectively. Action 2 switches the positions of two sub-modules within the structure. Action 3 alters the instructions used to assemble the structure (such as unit cell angles or module packing-type). Action 4 switches the positions of two modules within the structure. Action 5 doubles the structure along a chosen unit cell axis. Action 6 creates a new structure where the unit cell is the same size or smaller than the current structure. Action 7 creates a new structure of any size, with limits.
- Policy Function: Our choice of action is determined where with probability (1-ϵ) we use a softmax policy function based on our state space, θ, and suggested action; and with probability ϵ, we choose a random action.
- θ: The current policy is parameterized by θ, a feature vector indicating the probability of choosing one of the 9 discrete actions based on the current energy.
- Reward Function: The reward function is based on the negative change in normalized energy (i.e., the more an action reduces energy, the greater the reward). The reward is also calculated three kinds of negative rewards that correspond to when a performed action results in (1) no new information/an unrealizable structure (a configuration that violates known limitations on atomic organization), (2) the same energy as before, (3) or a structure that was made in a previous run.

1.3 Original Procedure

At this stage, we will lay out the original methodology in a step-by-step process:

- Start with $n = 100$ crystalline structures of the same compound. This group of structures is called a generation. Each generation undergoes a change to make the next generation.
- How do we go from one generation to the next? For each generation, we run FUSE on all of its structures. FUSE is an application of the Basin Hopping technique on ionic compounds. Basin Hopping (BH) is an algorithm that tries to minimize the energy of a given structure. The task for BH to minimize energy on ionic compounds is nontrivial, so FUSE incorporates an algorithmic package called GULP to aid BH. GULP is given the chemical formula and knowledge of ionic chemical properties to infer the new material structure.

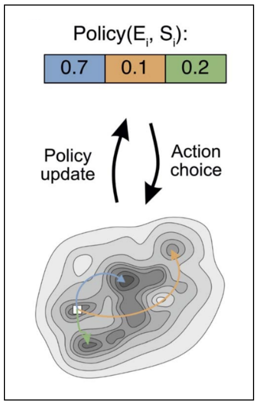


Figure 1: θ Array Representation with Policy (Energy$_i$, State$_i$) and State-Space Visualization.[1]

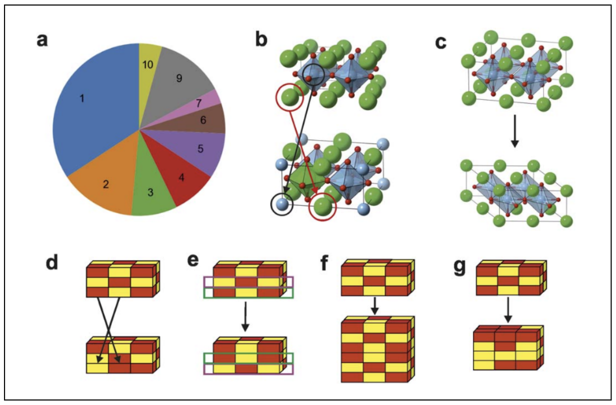


Figure 2: Action Types. (a) Probability distribution of Actions 1-7, 9-10. (b) Visual of Actions 1, 9, 10. (c-g) Visual of Actions 3, 2, 4, 5, and 6-7, respectively.[1]

- Once we're done with BH, we'll focus our attention on the structure with the minimized energy. We'll choose an action for this structure we're on to see how we can make it even more stable. With a small probability, we'll uniformly choose one of the $k = 9$ predefined actions. With a larger probability, we'll use a softmax to decide across possible actions based on a θ parameter. θ is simply the probability distribution over all the actions.
- After every H generations, we'll update our θ based on what we've seen so far, that is, the states and actions we explored and their respective changes in energies over the generations. We call a grouping of H generations and the data we've collected on it an "episode". Each update of θ takes into account entropy regularization, a normalized reward, and the policy function for that episode.
- This process will repeat for an extended period of time (in the researcher paper, effectively a While-True loop until the algorithm has quantified its findings and felt it has searched through all possible findings). Once done, it will end and report results.

1.4 Questions That Arise

In Policy Gradient algorithms for RL, our question focuses around the authors' rationale for updating the θ vector that parameterizes their policy. We propose there is value in using Policy Gradients, as the current approach does not take care of *expected* values over the entire parameter space nor possible trajectories of state-action pairs seen beyond the choices made in the episode (ie not looking at the entire episode-set of generations as a whole, just the choices we made).

2 Methodology

In the methodology section, we'll focus on the dataset description and the PG algorithms we explore.

2.1 Dataset

The data for our project is sourced from the GitHub provided in the referenced publication. The data is synthetically generated from the crystal structure prediction (CSP) simulator FUSE, which uses chemical properties of ionic solids to infer material structure given a proposed chemical formula. FUSE uses an algorithmic enhancement and numerical solver package called GULP. Specifically, GULP helps optimize for the lowest energy structure under a given action/change to the compound. The compositions in the original study focused on the possible chemical combinations yielded by the field $Y^{3+} - Sr^{2+} - Ti^{4+} - O^{2-}$. We have focused on the singular compound Y_2O_3 from this field for testing, selecting for repeated runs of this chemical over testing with other molecules. FUSE generates a simulated numerical dataset defining the material bonds, energy, and atomic arrangement in MySQL. We have evaluated several features in the dataset and in the code to evaluate our performance, namely: (1) the time taken to find an optimal low energy structure, (2) the average number of moves required to find said structure and (3) the difference between the lowest energy structure in the initial population and the lowest energy structure in the resulting population.

2.2 Coding

The two main folders of the github project, rlcsp and FUSE_RL, consist of just over 50,000 lines of code. We dealt with 3,000 of these lines of code that had to be thoroughly annotated to understand the premise of their procedure in code format. We eventually inserted approximately 350 lines of code for technical depth. The 3 files containing the relevant sections of code are: (1) rlcsp/input_FUSE/input_FUSE_RLCSP_Y203.py, (2) FUSE_RL/fuse_rl/run_fuse.py, and (3) rlcsp/input_FUSE/rlcsp/rlcsp.py. The first of these files contains simple parameters that can be adjusted to modify the starting and ending states of the crystal structure MDP studied, for Y_2O_3. Additionally, this file contains communication initialization fields for the mySQL server storing databases for our generated chemical data. Second, the file run_fuse.py contains sequential calls to rlcsp.py to coordinate execution of the generation, traversal, and saving of data related to finding an optimal policy during crystal structure stabilization. Finally, rlcsp.py contains the majority of our implemented code, including functions to implement NPG, PPO, and varied baseline functions. Please see the full public view of the repository here, or open the URL at https://github.com/Madison-Davis/CS1840FinalProject.

2.3 Algorithms

Of the 350 lines we added, the majority of it went into creating the three main RL concepts described below. As previously mentioned, all of these algorithms are being tested in the last stage where we decide to update θ:

- PPO: One of our motivations is to try to balance speed in calculations with a policy whose changes between steps will not throw us too-far off course. PPO can then be a valid alternative to the current approach for how we update θ and likewise the policy, as it incorporates a Lagrangian Relaxation of PG algorithms to ease computational complexity. PPO does work best in continuous spaces, and while we have a discrete number of actions, our states can be very large if we extrapolate to any kind of compound-input and not constraining to just our phase field; in other words, we want this model to be robust in the face of possibly infinite kinds of compound combinations. This is why PPO is a plausible starting point for potential improvement. Various papers have cited the superiority of PPO in such conditions.[2] Even if we wanted to only focus on a singular phase field (which may allow for a more discrete state space), PPO has been shown in research to still offer significant results.[3]

- NPG: Natural Policy Gradient may converge faster than REINFORCE because it does not risk making a large step that overshoots the optimal reward state (i.e., the global minimum energy state). Overshooting with the wrong action (such as replacing the atomic arrangement with an entirely new arrangement) when we were near the global minimum may place us in a state that requires many more steps to return to a near end-goal state. While REINFORCE, as implemented in the original published RL-crystal structure prediction paper, is also a policy gradient method, it does not incorporate a limit on how much our updated policy can deviate from the previous policy. NPG is an approximation for the Trust Region Policy Optimization approach, which limits the distance between the trajectory distributions for our old and new policies. It places a restriction on the deviation of our new policy from its previous iteration by updating its learning rate based on a desired distance between distributions, the gradient of the objective function, and the Fisher Information Matrix. One can think of this then as the algorithm taking into account its parameter space (the Fisher Matrix) and acting in accordance with it. This reduction of overshoot is another benefit of policy gradients over value-based policy iteration, as our actions will be more gradual and less likely to induce large changes in our state.

- Variations In Baseline Functions: In trying to calculate the policy gradient through PPO, the update to θ can occur with a high degree of variance meaning that the resulting MDP can be unreliable, especially with a small action-state environment like the one we are working with. To get a more confident MDP, we need to implement some variance reduction method. The PPO algorithm already uses a baseline function to reduce variance, however, we wanted to test and experiment with different baseline functions to see which would perform better in our environment. The first baseline function is one modeled on the general entropy of the system. The second baseline function is the standard Value function estimation such that the baseline function subtraction with the Q function results in an Advantage function-based policy gradient algorithm. Finally, we also thought to include a baseline function that takes ana average of the first two functions to experiment if that would perform better. Our approach was informed by the desire to reduce the variance of the PPO algorithm and calculate a more confident θ.

2.4 Experiments and Technical Depth

Here, we describe what experiments we are running, the parameters involved, and what data we are gathering. We focus on a single compound, $Y2O3$, to conduct five different experiments based on the algorithms described above: RLCSP, which serves as a control from what the original researchers used; PPO using a pre-defined B1 function as our baseline function; PPO using a pre-defined B2 function as our baseline function; PPO using a combination of B1 and B2 as our baseline function, specifically $0.7{\cdot}\text{B2}+0.3{\cdot}\text{B1}$ so as to put more emphasis on an Advantage-styled function; and NPG. RLCSP utilizes entropy regularization, the policy function π, and the reward function. B1 refers to an entropy difference function between our chosen action and a chosen baseline action. B2 refers to a Value Estimation Function. We ran each test twice and averaged the results of the data we wanted to collect. We collected the following data:

Data Type	Data Description and Motivation For Inclusion
Lowest Energy Achieved	The lowest energy achieved in a structure across all Basin Hops (BH), expressed in eV per atom. We wish to see if our algorithms perform as well as RLCSP.
Time Ran	We set the timer to one hour to see how long the BH searching would continue. The BH only stops if the algorithm is confident we are done searching through all the structures. If the time went beyond one hour, we cut it off arbitrarily.
Moves Sequence	An array representing the number moves it takes for a single basin hop to produce the next stable generation. It can take several attempts of a determined move for stability to hold across the structures, and we would not like to have to wait an extended period of time to find the next generation. If we've chosen good actions so far, then we could expect the mean number of moves to decrease over time, as the policy can lead to more stable structures.
Moves Average	The average value of the moves sequence.
Generation Sequence	An array where each entry is the length/number of generations where we see a "flatline" in energy or decrease in energy. For example, if our minimized energy across 5 generations was 5, 4, 3, 2, 5, then the generation sequence would append the value 4 (the number of moves prior to an increase in energy). The reason for collecting this data is we'd like to visualize the trend of these generation sequence lengths over time. For example, does the descent to lower energies get shorter/stepper over time? In a good policy-updating model, we may expect to do less exploration as time goes on, and so generation sequences may on average be longer over time.
Generation Average	The average value of the generations sequence. As mentioned above, we expect that for a better model, the average generation sequence length may increase, indicating either we are making more stable progress or hit a "pit" in which we stay more often at a minimized value.

Energy Sequence	An array where each entry is the localized minimum energy in a generation. Repeat energies between successive generations were not included in the array. If these arrays are graphed, the graph would have a generally oscillating nature, and this is because our action may at any point decide to explore entirely new structures. Exploration usually causes increases, or spikes, in the minimized energy for the next generation, while exploitation thereafter seeks to minimize it. Our goal then is to graph not the sequence but the minimum energies right before an exploration phase. We want to see if we generally find that our minimized energy stays or goes down over time.

For parameters, we set the initial generation to $n = 100$ crystalline structures. We also set an episode length of $H = 5$. Mathematically, PPO and NPG both rely on sampling trajectories for expectations. The probability distribution of sampling these trajectories is notated as p_{π_θ}. We set the number of trajectories to sample equal to 10 to allow for room to calculate expectations while also not being computationally expensive. A trajectory is simply a list of successive state-action pairs plausible via BH.

3 Results

3.1 Move, Generation, and Time Results

Figures 3 and 4 showcase the results across all five experiments for Move, Generation, and Time data. Figure 3 focuses on each run individually, while Figure 4 is an average of the two runs (per experiment). In Figure 3, we plotted the Move Sequence and Generation Sequence for each of our two runs on a distribution scale to showcase each run's mean, median, mode, and STD. In Figure 4, we showcase the Average Time Ran (cut off at 1 hour), Average Lowest Energy Structure Achieved, Average Number of Moves, and Average Number of Generations.

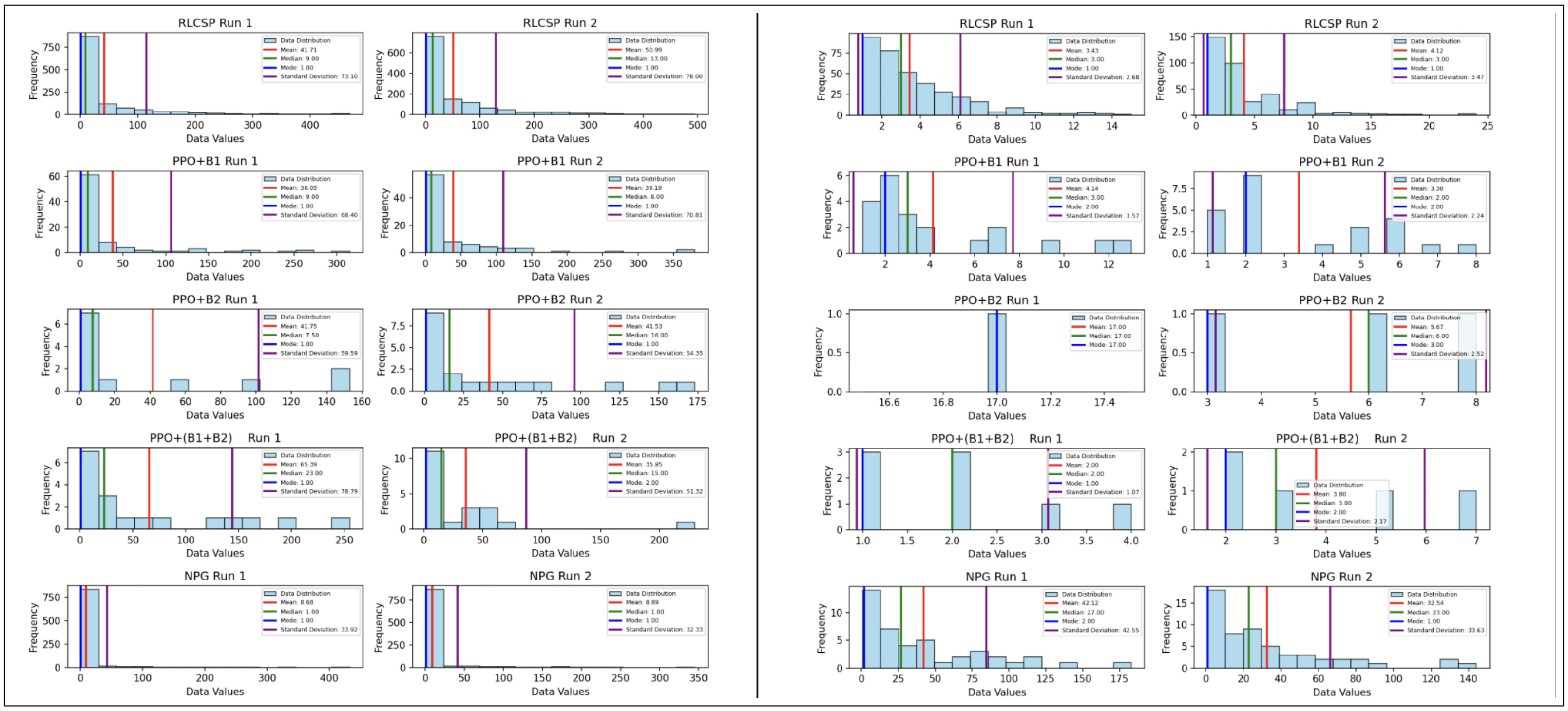
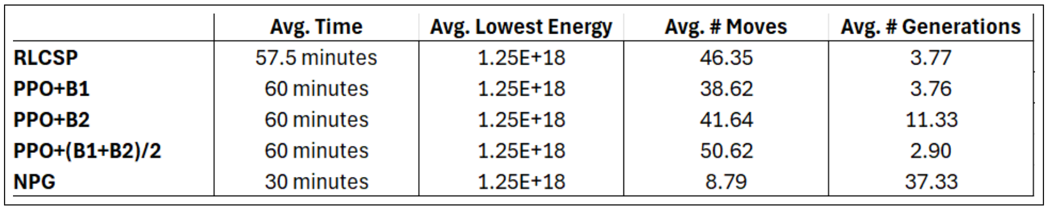


Figure 3: Move Number (Left) and Generation Sequence (Right) Results Per Experiment

From the results, we see that in terms of time ran, one of the RCSLP runs and both of the NPG runs finished before the 1-hour cutoff mark, meaning that they had searched through all possible



	Avg. Time	Avg. Lowest Energy	Avg. # Moves	Avg. # Generations
RLCSP	57.5 minutes	1.25E+18	46.35	3.77
PPO+B1	60 minutes	1.25E+18	38.62	3.76
PPO+B2	60 minutes	1.25E+18	41.64	11.33
PPO+(B1+B2)/2	60 minutes	1.25E+18	50.62	2.90
NPG	30 minutes	1.25E+18	8.79	37.33

Figure 4: Average Moves, Average Generations, and Other Numerical Results per Experiment

findings of the initial generation sequence. PPO runs did not finish and were cut off at the 1-hour mark. When it came to the average moves, from smallest to largest, we list out the experiments: NPG, PPO with B1, PPO with B2, RLCSP, and PPO with a combination of B1 and B2. When it came to the average generations, from smallest to largest, we list out the experiments: PPO with a combination of B1 and B2, PPO B1, RLCSP, PPO B2, and NPG.

3.2 Energy Sequence Results

Figure 5 showcases the results across all five experiments on an individual-run basis for the Energy Sequence Results. Recall we do not graph the entire sequence, as by nature it oscillates, but rather we graph the minimum energies right before an exploration phase.

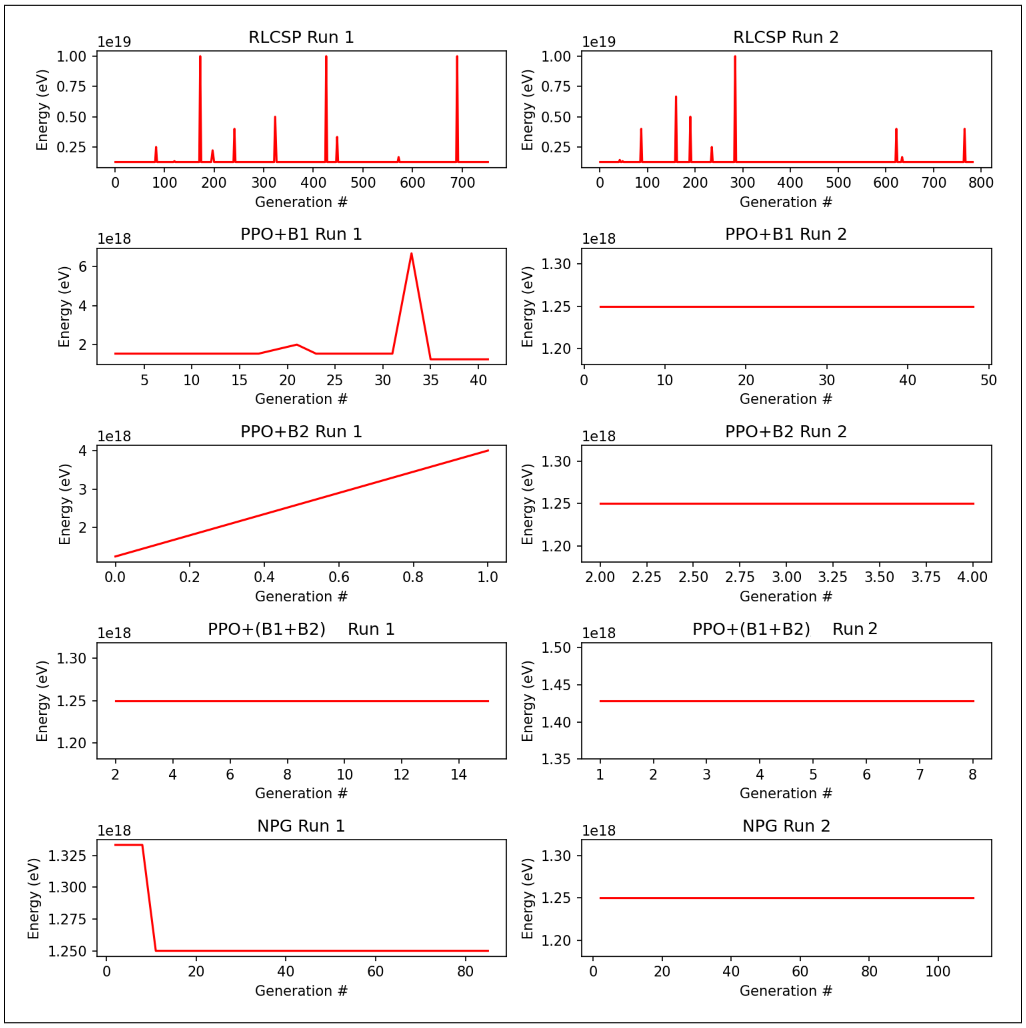


Figure 5: Average Moves, Average Generations, and Other Numerical Results per Experiment

From Figure 4, we can see that NPG and PPO matched performance of finding the lowest energy relative to RLCSP. From Figure 5, we see that RLCSP oscillates between small values and large values and also had a lot of generations. NPG had fewer generations than RLCSP, and PPO had fewer generations than NPG. PPO generally stayed flat along the generations with the exception of a spike in PPO + B1 Run 1 and PPO + B2 in Run 1. NPG stayed flat or consistently went down.

4 Interpretations

4.1 Move and Generation Interpretations:

For Moves, we look primarily at the average value for each experiment. We can expect that a model that is more adept at finding and sticking to minimized energy structures may need less moves to produce a stable result. We assume a collection of structures that are closely-related and also have near-minimized energy structures, due to their close structural similarity. If we have entered this collection using a given policy on a given structure, the probability of finding another low energy structure with the same or similar policy is high, given our assumption of nearby structures whose energies are also low. Because NPG on average had the least number of moves, we can interpret this then as the model finding a good θ parameter for us to then properly act upon a structure. It appears PPO with a combination of both baselines did the worst. This may be because of the fact that we were subtracting off two baselines B1 and B2 from a Q function compared to just one baseline. This would indicate to the model that there was ”less advantage” than there really was, and so it would discourage the model in cases where it may have actually produced good results. This provides insight on the necessity of choosing and evaluating a proper baseline.

For Generations, we look primarily at the average value for each experiment. For a desirable policy-updating model, we may expect to do less exploration (i.e., restart with a new set of crystal structures) as time goes on, and so single generation sequences may on average be longer over time, avoiding an ”oscillating” nature of small energies and large energies. We saw that PPO with a combination of B1 and B2 produced the smallest average generation sequence length, whereas NPG produced the largest generation sequence length. We can interpret this then as NPG focusing less on exploration and more on exploitation, indicating its policy update correlates well with a crystalline-structure problem. A question is raised as to why PPO using an individual baseline performs better compared to PPO with a combined baseline. The line of reasoning follows from that discussed above, wherein a double baseline has the potential to artificially inflate penalties for suboptimal, yet still potentially advantageous moves.

For Average Time, we observe that NPG completes the energy landscape traversal to locate the global minimum energy structure in half the average time of RLCSP. This result once again speaks to the benefit for NPG as a desired method for updating θ so as to choose actions that lead to quicker understanding of the state space.

4.2 Energy Interpretations:

For the Energy Sequence, we want to analyze the nature of the slope of the lines we plot. A slope that is flat indicates we have found a local or global minimized energy structure and make no progress beyond that, effectively putting us in a ”pit” of minimized energy. We see that RLCSP oscillates, indicating that the RLCSP algorithm does not take into nature far-off past history it has seen related to minimized energy structures. This maintains a dynamic level of exploration throughout a given run. PPO runs either stayed flat or in some runs oscillated. The single run where PPO increased (PPO + B2 Run 1) only had one generation, in which case the graph is misleading. In taking the x-axis into consideration, we may be inclined to interpret then is PPO + B2 Run 1 models more of what we see in PPO + B1 Run 1 and RLCSP, where we have oscillation. RLCSP had more volatile oscillating structures, with its max oscillation at times going up to $1e + 19$ compared to PPO B1 (at most $6e + 18$) and PPO B2 (at most $4e + 18$). We can infer then that PPO improves upon RLCSP by providing more stable convergence to a global minimum energy structure by limiting exploration as the energy landscape has been continually explored. Runs PPO B1+B2 appear to be completely flat, indicating we may have hit a local minimum ”pit” in our energy structures. This flat-like nature may suggest that the policy updates we are doing, while providing stability, are not extreme enough to allow us to choose an action to explore the entire collection of minimized, similar energy structures. This is also supported by the fact that PPO never finished within the hour setting. NPG, on the other hand, stays at a flat line or even decreases over time. Unlike the flat lines of PPO+B1 and PPO+B2, given the eventual convergence of NPG, we may infer a more progressive search-and-updating scheme that (1) takes into past considerations of minimized energy structures and (2) whose policy update is also extreme enough such that actions can permit explorations of nearby structures that can be minimized further.

The overall interpretation for why NPG may be better in this approach compared to PPO comes down the mathematical modeling of NPG and PPO. NPG and PPO both stem from TRPO. NPG is an approximated and pre-conditioned gradient method of TRPO, whereas PPO relies on an approximated Lagrangian Relaxation of TRPO. By a pre-conditioned gradient method, we are referring to that the gradient is calculated based off of the parameter space (through the Fisher Matrix) and not any expected objective. Natural phenomena behave in a similar way to NPG, where actions are made based on the current environment space. Moreover, PPO's approximation method with a Lagrangian Relaxation, while can allow it to be simpler, may not capture the nuances necessary for a continuous, complex environment. These mathematical interpretations can be reasons for why NPG provides a competitive edge.

5 Conclusion

Crystal structure prediction using informed policies may be better suited to the task of generating physically realizable materials than current heuristic or trial-and-error techniques. As there are rules imposed by nature on the formation of these crystals, albeit unknown, discerning a policy for generating such structures has been postulated to more reliably converge to low energy states given a particular set of molecules. In this work, we replicated the partial findings of Zamaraeva et al.'s implementation of the REINFORCE policy gradient algorithm to predict low energy crystal structures. Further, we compare this implementation to our own additions of Natural Policy Gradient and Proximal Policy Gradient, as well as various baseline techniques for variance reduction. Through repeated simulation of these algorithms, we find that NPG quickly obtains a policy for traversing the landscape of potential stable crystal structures and arrives at the global minimum structure. Likewise, PPO–paired individually with each of the two implemented baselines–and NPG both demonstrate a lower average required number of moves to reach the global minimum structure versus RLCSP (the implementation of Zamaraeva et al.'s REINFORCE). Finally, we observe NPG and PPO to perform a higher or similar level of exploration, demonstrated by average number of crystal generations, to RLCSP. Taken together, these findings suggest that the more constrained policy updates enforced by NPG and PPO can actually promote beneficial explorations which lead to convergence on lowest energy crystal structures. Thus, we conclude that careful selection of appropriate RL techniques can offer enhancements in finding stable energy structures, which may in turn aid in physical material realization efforts compared to traditional computational crystal structure prediction simulators.

6 References

1. Zamaraeva, Elena, Christopher M. Collins, Dmytro Antypov, Vladimir V. Gusev, Rahul Savani, Matthew S. Dyer, George R. Darling, Igor Potapov, Matthew J. Rosseinsky, and Paul G. Spirakis. "Reinforcement Learning in Crystal Structure Prediction." Digital Discovery, September 26, 2023. https://pubs.rsc.org/en/content/articlelanding/2023/dd/d3dd00063j.
2. Tang, Yunhao, and Shipra Agrawal. "Discretizing Continuous Action Space for On-Policy Optimization." arXiv.org, March 19, 2020. https://arxiv.org/abs/1901.10500.
3. Kalusivalingam, Aravind Kumar, Amit Sharma, Neha Patel, and Vikram Singh. "Optimizing Industrial Systems through Deep Q-Networks and Proximal Policy Optimization in Reinforcement Learning." International Journal of AI and ML. Accessed December 13, 2024. https://www.cognitivecomputingjournal.com/index.php/IJAIML-V1/article/view/47.