RLCG: When Reinforcement Learning Meets Coarse Graining

Anonymous Author(s)
Affiliation
Address
email

Abstract

Coarse graining (CG) algorithms are widely used to speed up molecular dynamics (MD) simulations. Recent data-driven CG algorithms have demonstrated competitive performances to empirical CG methods. However, these data-driven algorithms often rely heavily on labeled information (e.g., force), which is sometimes unavailable, and may not scale to large and complex molecular systems. In this paper, we propose Reinforcement Learning for Coarse Graining (RLCG), a reinforcement-learning-based framework for learning CG mappings. Particularly, RLCG makes CG assignments based on local information of each atom and is trained using a novel reward function. This "atom-centric" approach may substantially improve computational scalability. We showcase the power of RLCG by demonstrating its competitive performance against the state-of-the-arts on small (Alanine Dipeptide and Paracetamol) and medium-sized (Chignolin) molecules. More broadly, RLCG has great potential in accelerating the scientific discovery cycle, especially on large-scale problems.

1 Introduction

Molecular dynamics (MD) simulations [1, 2] are commonly used to probe and predict the atomic interactions of molecules across a wide range of scientific fields, including drug discoveries [3, 4, 5], protein folding predictions [6, 7], and battery material simulations [8, 9]. One major challenge in MD applications is the high computational cost, often induced by the spatial and temporal scale of the problem [10, 11]. Among the multiple approaches proposed to trade high resolution of the simulation for a better speed, e.g., adopting a larger time step [12], the application of coarse graining (CG) algorithms has been the most popular one. Coarse graining algorithms group atoms into different "beads" [13, 14], based on a mapping that can either be empirically defined or learned in a data-driven fashion [15, 14]. By representing a group of atoms as a bead, coarse graining algorithms reduce the spatial resolution of the system and are thus easier to simulate. Moreover, coarse-grained models tend to have smoother energy landscapes because of the removal of small fluctuations. Consequently, their simulations can be conducted with much larger time steps, leading to even faster simulation performances [16, 15].

A critical component of a successful CG algorithm is the CG mapping that assigns atoms to beads. To this end, several approaches have been proposed to define a CG mapping. Empirical methods that predefine a grouping rule (e.g., a "four-to-one" mapping), are widely used, see, e.g., MARTINI [17, 18]). Other algorithms adopt more automated procedures to the mapping process, such as extensions to MARTINI , as well as algorithms based on the graph structure of molecules [19, 20, 21]. More

Submitted to NeurIPS 2022 AI for Science Workshop.

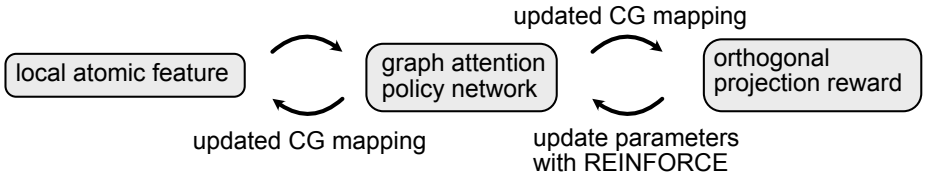

Figure 1: Major components of RLCG. The reinforcement learning framework facilitates efficient decision and learning with the policy network using only local information of each atom.

recently, the rise of artificial intelligence (AI) has brought about data-driven algorithms that learn a CG mapping by training neural networks with MD trajectory data. By learning the CG mapping purely from data, these algorithms eliminate the presence of human-imposed biases, and have seen success on small and medium-sized molecules. For example, Coarse-grained Autoencoder (CGAE) [16, 22] uses an autoencoder to learn the mapping by considering the CG space as its latent space. Another type of data-driven algorithm adopts a supervised learning approach that uses manually identified mappings as labels for training the mapping algorithms [23, 24].

There are two major caveats with the aforementioned algorithms. First, an appropriate loss function needs to be carefully designed in order to achieve a reasonable mapping. Such a loss function (e.g., in CGAE) may also require proper tuning and additional data beyond the coordinates of the atoms to compute, e.g., force labels, which can be hard to acquire. At a higher level, there lacks of an intuitive metric to assess the quality of CG mappings. As a consequence, measuring the quality of a CG mapping requires the presence of a downstream task. Secondly, graph-based methods suffer from limited scalability because of the computational cost of performing matrix manipulations. The same issue also arises in data-driven methods because these algorithms (e.g. CGAE) learn a mapping on the whole molecule, which typically has a computational complexity of at least $O(Nn)$ with N, n being the number of atoms and CG beads, respectively. In particular, these algorithms can take a substantial amount of time to converge for large molecules with thousands of atoms.

To address these issues, we propose, to the best of our knowledge, the *first* reinforcement learning framework to learn CG mappings, i.e., Reinforcement Learning for Coarse graining (RLCG). Our framework uses an atom-centric approach by making bead assignments per atom, requiring learning only a fixed number of parameters regardless of the size of the molecule. More specifically, the bead assignment is facilitated by a novel policy network architecture based on a graph attention (GAT) neural network [25], together with a novel but simple design of the reward function. These designs enable us to scale the algorithm to large molecules and provide us with an intuitive metric to measure the quality of the learned CG mapping.

The rest of this paper is organized as follows: Section 2 details the RLCG framework; Section 3 presents our experimental results on three molecules with different sizes, including Alanine Dipeptide, Paracetamol, and Chignolin; Section 4 compares our method to several related works and briefly concludes the paper and discusses some future directions to further improve RLCG.

2 RLCG: Architecture and design

RLCG learns a CG mapping by sequentially assigning individual atoms to different CG beads. At each iteration, it uses a graph attention policy network to featurize atom-specific information and make CG assignments on individual atoms. The updated CG mapping will receive an orthogonal projection reward which quantifies the quality of the mapping. RLCG learns the parameters of the policy network by routinely making assignments with different random initializations and updating the parameters under a reinforcement learning framework. The overall procedure is depicted in Fig. 1, and we will introduce the components in the remainder of this section.

2.1 Preliminaries

Let $\mathcal{T}$ be the trajectory of the molecule and $X_t \in \mathbb{R}^{n\times 3}$ be the coordinates of the molecule at time t, where n is the number of atoms. We want to learn a mapping $M(\cdot) : \mathbb{R}^{n\times 3} \mapsto \mathbb{R}^{N\times 3}$ that maps the atomic coordinates into the CG coordinates (geometric center of the corresponding atomic coordinates), where N is the number of CG beads and $N \ll n$. Let $M^{-1}(x_t; k) \in \mathbb{R}^{N_k\times 3}$ denote the atomic coordinates of the k-th CG bead with N_k atoms at the t-th time. For simplicity, let $M(i) \in [N]$ represent the CG assignment of the i-th atom.

2.2 The orthogonal projection reward

We need a metric to evaluate the quality of the current CG mapping. Intuitively, a good CG mapping should preserve intra-bead rigidity. In other words, the atoms mapped within one bead should have smaller relative movements (Fig. 2.a). Such within-bead rigidity may also benefit the training of the CG force field. Following this intuition, we introduce the orthogonal projection loss. More specifically, for two arbitrary frames of the atomic coordinates x and y, the orthogonal projection loss of a CG mapping, $M(\cdot)$, is defined as:

$$L_{\text{ortho}}(x, y, M) = \sum_{k=1}^{K} w_k \min_{\Omega^\top_{x,y,k,M}\Omega_{x,y,k,M}=I} \|\Omega_{x,y,k,M} M^{-1}(x; k) - M^{-1}(y; k)\|_F,$$

where w_k is the mass of the k-th bead divided by the mass of the molecule, $\|\cdot\|_F$ is the Frobenius norm, and $\Omega_{x,y,k,M}$ is obtained via solving the orthogonal Procrustes analysis [26] by first performing a singular value decomposition: $M^{-1}(x; k)M^{-1}(y; k)^\top = U\Sigma V^\top$ and letting $\Omega_{x,y,k,M} = VU^\top$. Intuitively, the orthogonal projection loss is the residue of the projected atomic coordinates of each CG bead between two frames, weighted by the mass of each bead. A smaller orthogonal projection loss corresponds to a better rigidity within a bead, as atoms are relatively static to each other when the loss is small.

The orthogonal projection reward of the CG mapping M is thus defined as the negative expected orthogonal projection loss averaged over the entire trajectory $\mathcal{T}$. :

$$\text{Reward}(M) = -\mathbb{E}_{x,y\sim\mathcal{T}}\big[L_{\text{ortho}}(x, y, M)\big].$$

In practice, we used 1000 randomly sampled pairs of frames of coordinates to approximate the expectation above. This reward (or loss) only depends on the coordinates of the atoms and accurately reflects the quality of the mapping as shown in Supplementary Fig. S1.

2.3 The graph attention policy network

The policy network receives feature vectors of each atom in the molecule and generates two outputs: the selection output that determines which atom's CG assignment to change, and the decision output that determines the selected atoms' new CG assignment.

This design aims at explicitly decoupling from atom assignment, because not every atom needs to be re-assigned throughout the learning process. Moreover, the assignment action is based on merging the CG id of one atom to the other, which lends itself to the use of a graph attention network that explicitly models the relative interaction between atoms.

Feature vector. We adopt a similar featurization strategy as in literature using graph neural networks for molecules and graphs[22, 27]. For the i-th atom, the node feature is defined as:

$$\boldsymbol{F}_i^{(0)} = \text{one_hot}(\text{atom_type}(i)) \bigoplus \text{one_hot}(M(i)) \bigoplus \text{load}(i),$$

where $\bigoplus$ denotes concatenation, $\text{one_hot}(M(i))$ is the one-hot encoding of the CG assignment, and

$$\text{load}(i) = \frac{|\{k : M(k) = M(i), k = 1, \cdots, n\}|}{n}$$

represents the proportion of CG i. The node feature includes chemical information (atom type) as well as the mapping related information.

The edge feature between a pair of atoms i, j connected with a bond is simply:

$$E_{ij} = \text{one_hot}(\text{bond_type}(i, j)).$$

Intuitively, the algorithm will make grouping assignments based on the type of the atom and how it is connected to other atoms (e.g. a carbon atom may be more likely to be assigned to the same CG bead with the oxygen atom through their C=O bond than with another carbon atom through C-C bond, because the former combination has a stronger bond.). Moreover, based on the loading of the atom, the algorithm can balance the loading of each CG bead by converting the assignment of an atom from the larger bead to that of a smaller bead.

Selection layer. The selection layer consists of a graph attention network (GAT) layer:

$$\boldsymbol{F}_i^{(1)} = \alpha_{i,i}^{(1)} \Theta \boldsymbol{F}_i^{(0)} + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}^{(1)} \Theta \boldsymbol{F}_j^{(0)},$$

where $\mathcal{N}(i)$ is the index set of the neighbors of atom i defined by bond connections, Θ is an MLP and $\alpha_{i,j}$ is the attention coefficient computed using $\Theta, \boldsymbol{F}_i^{(1)}, \boldsymbol{F}_j^{(1)}, E_{ij}$, and an attention network, $\boldsymbol{a}$ [28, 25].

The readout layer is an MLP combined with a masked softmax:

$$\text{Selection} = \text{Softmax}(\text{MLP}(\boldsymbol{F}_i^{(1)}), i \in \text{bd}),$$

where bd is the boundary set such that $i \in \text{bd}$ if there exists $j \in \mathcal{N}(i)$ such that $CG(i) \neq CG(j)$. In other words, the algorithm only seeks to update the assignment of atoms which has neighboring atoms with a different CG assignment. This way the algorithm will avoid "incongruous" mapping where non-connected atoms are assigned to the same CG group.

Decision layer. The decision layer outputs the CG assignment of each atom. This is achieved with another GAT layer:

$$\boldsymbol{F}_i^{(2)} = \alpha_{i,i}^{(2)} \Theta \boldsymbol{F}_i^{(1)} + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}^{(2)} \Theta \boldsymbol{F}_j^{(1)}.$$

The assignment of each atom is drawn from a multinomial distribution with class probabilities following the attention coefficients: $\{\alpha_{ij} : j \in \mathcal{N}(i) \cup \{i\}\}$.

Hence, for each atom, the probability of being given the same assignment of its neighboring atom j is given by α_{ij}. Note that the decision is only limited to the neighboring atoms, which further reduces the undesirable scenarios where an atom is assigned with a CG bead different from any of its neighbors. Furthermore, the assignment is "relative": an atom is not directly assigned a CG id, but indirectly through the id of its neighboring atom. This resolves the potential "label permutation invariance" issue as the number that represents CG assignment may change with different episodes in the RL cycle (i.e. CG bead 1 can be bead 2 in a different episode).

2.4 The Reinforcement learning framework

The framework follows the standard REINFORCE algorithm [29]. For each episode, we use a random backbone initialization of the mapping and consecutively make CG assignments using the policy network until a predefined number of steps has been reached. This is listed in Algorithm 1 and illustrated in Fig. 2.b. Therefore, as an atom-centric approach, each individual atom makes the mapping decision based on its surrounding environment.

Note that the actual reward we use to train the policy network is the difference between the reward of the current step and the previous one as we want to maximize "gain" in reward for each step.

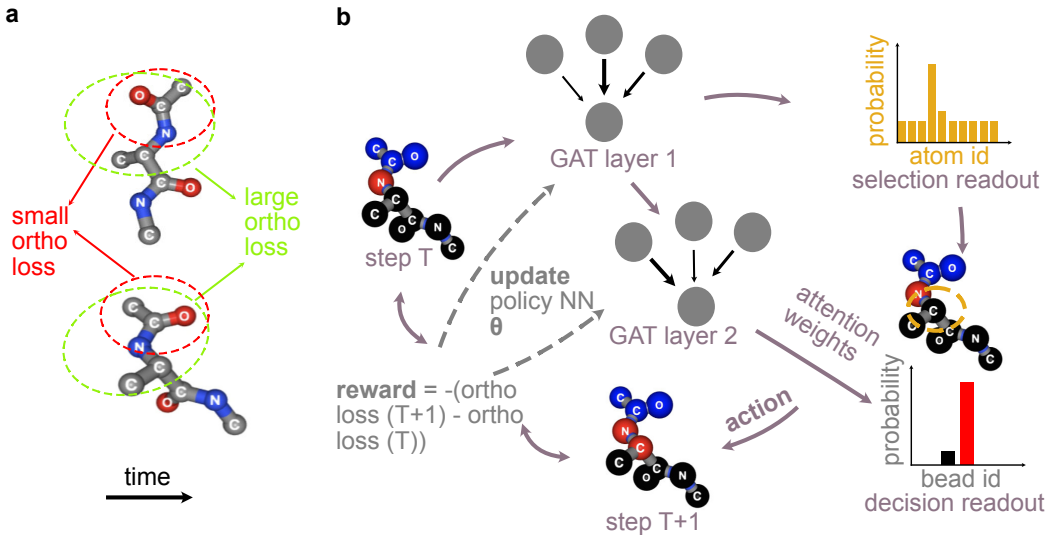

Figure 2: Diagram for RLCG. **(a).** Illustration of the orthogonal loss. **(b).** GNN architecture and RL training routine for one episode.

Algorithm 1: RLCG evaluation steps for one episode

Input: policy network $\pi_\theta(\cdots)$, *initialization* method, number of steps S.
$M \leftarrow$ *initialization*
$R =$ *empty set*
for $s \leftarrow 1$ **to** S **do**
  $r = \text{Reward}(M)$
  $\boldsymbol{F}_i^{(0)} \leftarrow \text{one_hot}(\text{atom_type}(i)) \bigoplus \text{one_hot}(M(i)) \bigoplus \text{load}(i), \forall i$
  $\boldsymbol{F}_i^{(1)} = \alpha_{i,i}^{(1)} \Theta \boldsymbol{F}_i^{(0)} + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}^{(1)} \Theta \boldsymbol{F}_j^{(0)}$
  $\eta \leftarrow \text{Softmax}(\text{MLP}(\boldsymbol{F}_i^{(1)}), i \in \text{bd})$
  $\boldsymbol{F}_i^{(2)} = \alpha_{i,i}^{(2)} \Theta \boldsymbol{F}_i^{(1)} + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}^{(2)} \Theta \boldsymbol{F}_j^{(1)}$
  $M(\eta) \leftarrow \{\alpha_{ij} : j \in \mathcal{N}(i) \cup \{i\}\}$
  $\boldsymbol{F}_\eta^{(0)} \leftarrow \text{one_hot}(\text{atom_type}(\eta)) \bigoplus \text{one_hot}(M(\eta)) \bigoplus \text{load}(\eta)$
  $R = R \cup \{\text{Reward}(M) - r\}$

3 Experiments

3.1 Experiment setup

We test the performance of RLCG on three molecules:

Alanine Dipeptide. This molecule contains 10 heavy atoms (22 atoms in total). The trajectory data was obtained from the mdshare dataset and downloaded from `https://markovmodel.github.io/mdshare/ALA2/`. We randomly selected $10,000$ frames for training and learned a CG mapping with 2 beads on this molecule.

Paracetamol. This molecule contains 11 heavy atoms (20 atoms in total). The trajectory data was obtained from the MD17 dataset [30] and downloaded from `http://www.sgdml.org/#datasets`. We randomly selected $10,000$ frames for training and learned a CG mapping with 3 beads on this molecule.

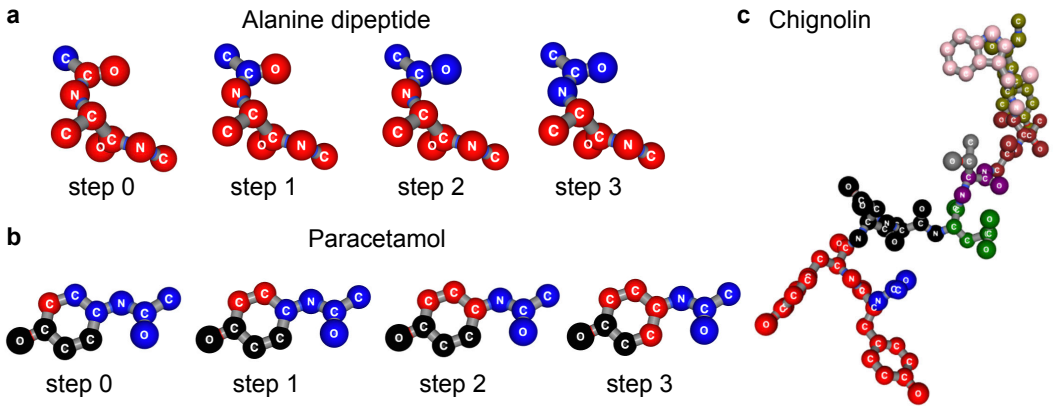

Figure 3: Training results for the three molecules. **(a)**. Example evaluation trajectory for Alanine Dipeptide with 2 beads. Step 0 represents the initial CG assignment. Different colors represent different CG beads. **(b)**. Example evaluation trajectory for Paracetamol with 3 beads. **(c)**. Learned CG mapping for Chignolin with 10 beads.

Chignolin. Chignolin is a 10-amino acid mini-protein with 99 heavy atoms (175 atoms in total). The trajectory data was simulated using HTMD (High-Throughput Molecular Dynamics) [31] and downloaded from `http://pub.htmd.org/Chignolin_trajectories.tar.gz`. We randomly selected $10,000$ frames for training and learned a CG mapping with 10 beads on this molecule.

The simulation details for Alanine Dipeptide and Chignolin can be found in [23]. These two datasets have also been studied extensively as benchmark datasets in [22]. For all datasets, we only consider the heavy atoms for simplicity. To test the performance of RLCG, we trained the policy network for 1500 episodes on Alanine Dipeptide and Paracetamol, and 2500 episodes on Chignolin. Within each episode, we ran the policy network for 5 steps on Alanine Dipeptide and Paracetamol, and 50 steps on Chignolin with random initialization.

We also compared RLCG with CGAE. The training loss for CGAE includes the reconstruction error and the force regularization. Note that the force label is only available in Paracetamol dataset. As a baseline, we compared RLCG with random partition where each CG bead has at least one atom assigned to it. We performed a 5-fold cross validation, where each algorithm was trained on 4 folds and tested on the held-out fold. See Fig. S2 for the evolution of the testing orthogonal loss for RLCG. Even though the algorithms compared are not supervised approaches, 5-fold cross validation may account for potential variance during the learning and testing process and enable a more fair comparison.

3.2 Simulation results

As an illustrative example, we plotted the evolution of the CG mapping using RLCG on Alanine dipeptide within one episode (Fig. 3.a). As can be seen (Fig. 3.a), RLCG automatically balances the CG loading by stopping further updates after step 3 (step 4 is not shown because it is the same as step 3). On Paracetamal, RLCG also yields congruous mappings within 3 steps (Fig. 3.b). Note that RLCG will group the C atom with the one on the benzene ring instead of the N atom (step 2 to 3, red bead), indicating the algorithm has learned the correct relative bond information between atoms. RLCG also learns congruent mapping with reasonably balanced CG loadings for Chignolin (Fig. 3.c).

RLCG achieves a competitive orthogonal loss across all the algorithms compared (Table 1). Note that for Alanine Dipeptide, the slight higher loss of RLCG than CGAE is due to the randomness in initialization for RLCG, which can be further improved.

Table 1: Orthogonal projection loss on three molecules, mean $\pm$ SD

Algorithm	Alanine Dipeptide	Paracetamol	Chignolin
random partition	0.428 ± 0.038	0.543 ± 0.045	68.59 ± 2.20
CGAE	$\mathbf{0.332 \pm 0.002}$	0.257 ± 0.123	43.326 ± 7.652
CGAE w/ force regularization	/	0.208 ± 0.081	/
RLCG	0.361 ± 0.03	$\mathbf{0.161 \pm 0.011}$	$\mathbf{20.145 \pm 0.312}$

4 Conclusions & Discussions

4.1 Conclusions

In this paper, we present RLCG, a reinforcement learning framework that is designed to efficiently learn the coarse-grained mapping of a molecule. RLCG adopts an atom-centric decision mechanism and uses GAT to generate CG assignments, which is capable of fast simulations even on large molecules. Our algorithm has the potential to expedite the scientific discovery cycle if further extended to large molecules.

The RLCG framework has connections to other RL-based graph partitioning algorithms, but possesses distinct advantages. For example, several vertex-centric methods are proposed for graph partitioning [32, 33]. However, these algorithms do not use a standard RL framework and partition a graph "on the go", which may not generalize to different datasets. GNN-based RL algorithms have also been proposed [27]. However, such methods only consider graphs without chemical information, and only apply to bi-partitions. There are also algorithms that fragment a molecule by breaking their SMILES strings [34] or chemical bonds [35]. However, these methods are mainly designed for decomposing a large molecule into motifs and hence may only result in larger fragments and may not be directly applicable to coarse graining.

4.2 Discussions and future work

Multiple improvements can be made to speed up the RLCG framework. First, the computation of the orthogonal projection reward is local: changing the assignment of one atom will only affect the CG partition of two CG beads. This property may enable a much faster computation of the orthogonal projection reward. Second, adding more layers to the policy network may enable the network to handle more complicated structures. For example, more GNN layers would allow the policy network to see more hops away, and consequently handle rings of different numbers of atoms. Third, the RL framework is flexible and can process chemical information, such as the rotation angle, as well as the designer's preference for the downstream task, e.g., using more CG beads for a benzene ring to preserve its planar property which will potentially improve the training of the CG force field. Lastly, future work also includes experimenting on larger molecules and also testing the generality of the trained policy network on new and unseen molecules.

References

[1] Dennis C Rapaport and Dennis C Rapaport Rapaport. *The art of molecular dynamics simulation*. Cambridge University Press, 2004.

[2] Martin Karplus and Gregory A Petsko. Molecular dynamics simulations in biology. *Nature*, 347(6294):631–639, 1990.

[3] David W Borhani and David E Shaw. The future of molecular dynamics simulations in drug discovery. *Journal of Computer-aided Molecular Design*, 26(1):15–26, 2012.

[4] Jacob D Durrant and J Andrew McCammon. Molecular dynamics simulations and drug discovery. *BMC Biology*, 9(1):1–9, 2011.

[5] Marco De Vivo, Matteo Masetti, Giovanni Bottegoni, and Andrea Cavalli. Role of molecular dynamics and related methods in drug discovery. *Journal of Medicinal Chemistry*, 59(9):4035–4061, 2016.

[6] William C Swope, Jed W Pitera, and Frank Suits. Describing protein folding kinetics by molecular dynamics simulations. 1. theory. *The Journal of Physical Chemistry B*, 108(21):6571–6581, 2004.

[7] Yuji Sugita and Yuko Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chemical Physics Letters*, 314(1-2):141–151, 1999.

[8] Kevin Leung and Joanne L Budzien. Ab initio molecular dynamics simulations of the initial stages of solid–electrolyte interphase formation on lithium ion battery graphitic anodes. *Physical Chemistry Chemical Physics*, 12(25):6583–6586, 2010.

[9] Ken Tasaki, Katsuya Kanda, Shinichiro Nakamura, and Makoto Ue. Decomposition of lipf6and stability of pf 5 in li-ion battery electrolytes: Density functional theory and molecular dynamics studies. *Journal of The Electrochemical Society*, 150(12):A1628, 2003.

[10] Dominik Marx and Jurg Hutter. Ab initio molecular dynamics: Theory and implementation. *Modern Methods and Algorithms of Quantum Chemistry*, 1(301-449):141, 2000.

[11] Christian Kandt, Walter L Ash, and D Peter Tieleman. Setting up and running molecular dynamics simulations of membrane proteins. *Methods*, 41(4):475–488, 2007.

[12] Tianze Zheng, Weihao Gao, and Chong Wang. Learning large-time-step molecular dynamics with graph neural networks. *arXiv preprint arXiv:2111.15176*, 2021.

[13] Robert E Rudd and Jeremy Q Broughton. Coarse-grained molecular dynamics and the atomic limit of finite elements. *Physical Review B*, 58(10):R5893, 1998.

[14] Soumil Y Joshi and Sanket A Deshmukh. A review of advancements in coarse-grained molecular dynamics simulations. *Molecular Simulation*, 47(10-11):786–803, 2021.

[15] Jiang Wang, Simon Olsson, Christoph Wehmeyer, Adrià Pérez, Nicholas E Charron, Gianni De Fabritiis, Frank Noé, and Cecilia Clementi. Machine learning of coarse-grained molecular dynamics force fields. *ACS Central Science*, 5(5):755–767, 2019.

[16] Wujie Wang and Rafael Gómez-Bombarelli. Coarse-graining auto-encoders for molecular dynamics. *npj Computational Materials*, 5(1):1–9, 2019.

[17] Siewert J Marrink, H Jelger Risselada, Serge Yefimov, D Peter Tieleman, and Alex H De Vries. The MARTINI force field: Coarse grained model for biomolecular simulations. *The Journal of Physical Chemistry B*, 111(27):7812–7824, 2007.

[18] Luca Monticelli, Senthil K Kandasamy, Xavier Periole, Ronald G Larson, D Peter Tieleman, and Siewert-Jan Marrink. The MARTINI coarse-grained force field: Extension to proteins. *Journal of Chemical Theory and Computation*, 4(5):819–834, 2008.

[19] Tristan Bereau and Kurt Kremer. Automated parametrization of the coarse-grained MARTINI force field for small organic molecules. *Journal of Chemical Theory and Computation*, 11(6):2783–2791, 2015.

[20] Michael A Webb, Jean-Yves Delannoy, and Juan J De Pablo. Graph-based approach to systematic molecular coarse-graining. *Journal of Chemical Theory and Computation*, 15(2):1199–1208, 2018.

[21] Thomas D Potter, Elin L Barrett, and Mark A Miller. Automated coarse-grained mapping algorithm for the MARTINI force field and benchmarks for membrane–water partitioning. *Journal of Chemical Theory and Computation*, 17(9):5777–5791, 2021.

[22] Wujie Wang, Minkai Xu, Chen Cai, Benjamin Kurt Miller, Tess Smidt, Yusu Wang, Jian Tang, and Rafael Gómez-Bombarelli. Generative coarse-graining of molecular conformations. *arXiv preprint arXiv:2201.12176*, 2022.

[23] Brooke E Husic, Nicholas E Charron, Dominik Lemm, Jiang Wang, Adrià Pérez, Maciej Majewski, Andreas Krämer, Yaoyi Chen, Simon Olsson, Gianni de Fabritiis, et al. Coarse graining molecular dynamics with graph neural networks. *The Journal of Chemical Physics*, 153(19):194101, 2020.

[24] Zhiheng Li, Geemi P Wellawatte, Maghesree Chakraborty, Heta A Gandhi, Chenliang Xu, and Andrew D White. Graph neural network based coarse-grained mapping prediction. *Chemical Science*, 11(35):9524–9531, 2020.

[25] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[26] John C Gower and Garmt B Dijksterhuis. *Procrustes problems*, volume 30. OUP Oxford, 2004.

[27] Alice Gatti, Zhixiong Hu, Tess Smidt, Esmond G Ng, and Pieter Ghysels. Graph partitioning and sparse matrix ordering using reinforcement learning and graph neural networks. *arXiv preprint arXiv:2104.03546*, 2021.

[28] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

[29] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

[30] Stefan Chmiela, Alexandre Tkatchenko, Huziel E Sauceda, Igor Poltavsky, Kristof T Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, 3(5):e1603015, 2017.

[31] S Doerr, MJ Harvey, Frank Noé, and G De Fabritiis. Htmd: High-throughput molecular dynamics for molecular discovery. *Journal of Chemical Theory and Computation*, 12(4):1845–1852, 2016.

[32] Mohammad Hasanzadeh Mofrad, Rami Melhem, and Mohammad Hammoud. Revolver: Vertex-centric graph partitioning using reinforcement learning. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 818–821. IEEE, 2018.

[33] Mohammad Hasanzadeh Mofrad, Rami Melhem, and Mohammad Hammoud. Partitioning graphs for the cloud using reinforcement learning. *arXiv preprint arXiv:1907.06768*, 2019.

[34] Marco Podda, Davide Bacciu, and Alessio Micheli. A deep generative model for fragment-based molecule generation. In *International conference on artificial intelligence and statistics*, pages 2240–2250. PMLR, 2020.

[35] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *International conference on machine learning*, pages 4839–4848. PMLR, 2020.

Appendix

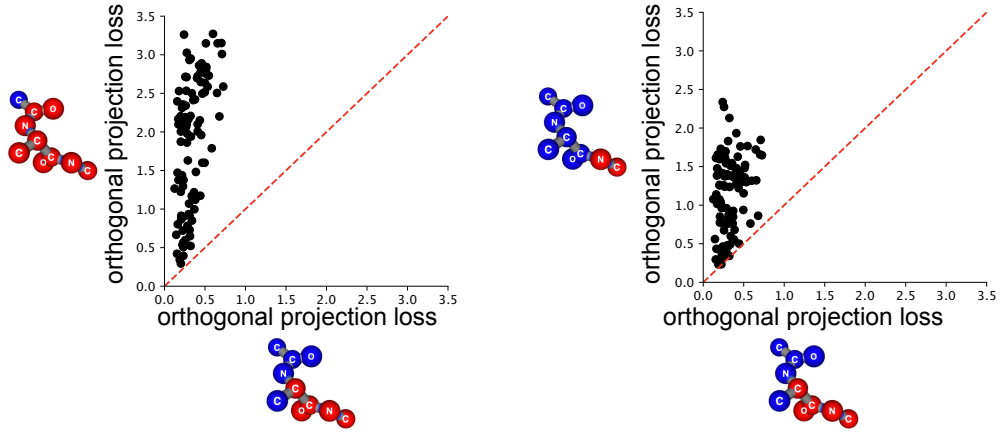

Figure S1: **Example orthogonal projection loss on different CG mappings of Alanine dipeptide.** We selected three CG mappings and we computed their orthogonal projection loss based on coordinates of Alanine dipeptide of 100 randomly sampled times. Each dot represents the orthogonal projection loss of two mappings (illustrated with the icons) on a given time point. It can be seen that the better mapping (represented by the x-axis) has a smaller orthogonal projection loss than the other two mappings. Note that there is variance in the loss under the same CG mapping (dots are scattered), hence we computed the loss based on 1000 randomly selected time points in practice, to reduce the effect of this variance.

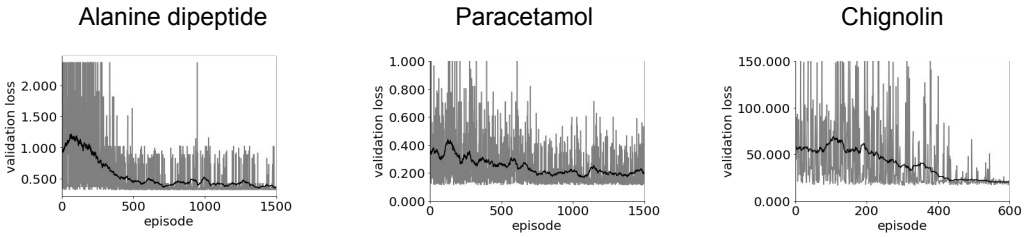

Figure S2: **RLCG training validation error.** Orthogonal loss of the CG mapping returned by RLCG during training process per episode (grey traces). Black traces are 50-episode moving averages. For Chignolin we only show the first 800 episodes since the training process already converges within 800 episodes.