UNIVERSITY OF CALIFORNIA SAN DIEGO

Monte Carlo Tree Methods for Nonlinear Optimization

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science

by

Yaoguang Zhai

2024

DEDICATION

This dissertation is dedicated to my family, whose support and love have been the bedrock of my journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I extend my greatest acknowledgment to Professor **Sicun Gao** for his persistent support as the chair of my committee. His guidance has been absolutely invaluable. Professor Gao's insightful feedback, rigorous academic standards, and encouragement have been pivotal in navigating the complex journey of my research.

I am also profoundly grateful to the co-chair of my committee, Professor **Francesco Paesani**, whose expertise and mentorship have significantly shaped my academic and research trajectory. Professor Paesani's ability to challenge and inspire has not only honed my skills as a researcher but also deepened my appreciation and understanding of our field.

I am immensely grateful to Professor **Andrew Kahng** for his role as a committee member. Professor Kahng's profound expertise has greatly enhanced both my comprehension and methodology regarding my research topic.

My sincere thanks go to Professor **Lily Weng** for her invaluable contribution as a committee member. Her perceptive feedback have been fundamental to my growth and the attainment of this work.

I wish to express my deepest and most sincere gratitude to my mentors, who have been instrumental in my research and career journey.

My sincere thanks to Dr. **Andreas Goetz**, whose expertise and insights into computational knowledge have greatly enriched my understanding and appreciation of the field. His enthusiasm for science has inspired me profoundly.

To **Karl Schwamb** and **Peter Nguyen**, for their support and mentorship, guiding me through the complexities of my work with patience and wisdom.

I am deeply grateful to Dr. **Benjamin Yu** for his critical feedback and perspective on my work. His dedication to excellence and innovation has pushed me to reach higher, fostering a deeper understanding of my research domain.

To Dr. **Thomas Desautels**, for his encouragement and invaluable advice on my research path. His mentorship has been a pillar of strength, encouraging me to persevere through challenges with resilience and determination.

I extend my sincere thanks to Dr. **Soonho Kong**, for his analytical rigor and thoughtful guidance. His collective wisdom, mentorship, and friendship have left an indelible mark on my academic journey and personal development.

To Dr. **Bruno Dutertre**, for his meticulous attention and mentorship that has been critical in shaping my approach to scientific inquiry, instilling in me the values of integrity and diligence.

I am also thankful for the opportunity to my friends and colleagues in **Gao's group** and the **Paesani Research Group**. I would like to thank my collaborators, **Zhizhen Qin**, Dr. **Alessandro Caruso**, and Dr. **Sigbjørn Løland Bore**, who have contributed unique perspectives and expertise that have enriched this research.

Finally, my deepest gratitude goes to my family - my beloved wife, **Xu**, my inspiring child, **Erik**, my parents and my parents-in-law. Your enduring support has been my greatest strength throughout this extensive journey. I also hold a special place in my heart for my grandparents, whose memory and legacy have fortified me during this profound journey.

Chapter 2, in full, is a reprint of the material as it appears in “Monte Carlo Tree Descent for Black-Box Optimization,” Y. Zhai, S. Gao. in *Advances in Neural Information Processing Systems*, 2022, 35: 12581-12593. The dissertation author is the primary investigator and author of this paper.

Chapter 3, in full, is a reprint of the material as it appears in “Sample-and-Bound for Non-Convex Optimization,” Y. Zhai, Z. Qin, S. Gao. in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. The dissertation author is the primary investigator and author of this paper.

Chapter 4, in full, is a reprint of the material as it appears in “Active learning of many-body configuration space: Application to the Cs+–water MB-nrg potential energy function as a case study,” Y. Zhai, A. Caruso, S. Gao, F. Paesani, in *The Journal of Chemical Physics*, 2020, 152(14). The dissertation author is the primary investigator and author of this paper.

Chapter 5, in full, is a reprint of the material as it appears in “A “short blanket” dilemma for a state-of-the-art neural network potential for water: Reproducing properties or learning the underlying physics?" Y. Zhai, A. Caruso, S.L. Bore, Z. Luo, F. Paesani, in *The Journal of Chemical Physics*, 2023, 158(8). The dissertation author is the primary investigator and author of this paper.

VITA

2008	Master of Science, Royal Institute of Technology
2018	Master of Science, University of California San Diego
2018–2024	Research Assistant, University of California San Diego
2024	Doctor of Philosophy, University of California San Diego

PUBLICATIONS

Zhai, Y., Caruso, A., Gao, S., & Paesani, F. (2020). Active learning of many-body configuration space: Application to the Cs+–water MB-nrg potential energy function as a case study. The Journal of Chemical Physics, 152(14).

Zhai, Y., & Gao, S. (2022). Monte Carlo Tree Descent for Black-Box Optimization. Advances in Neural Information Processing Systems, 35, 12581-12593.

Zhai, Y., Caruso, A., Bore, S. L., Luo, Z., & Paesani, F. (2023). A "short blanket" dilemma for a state-of-the-art neural network potential for water: Reproducing experimental properties or the physics of the underlying many-body interactions?. The Journal of Chemical Physics, 158(8).

Zhai, Y., Qin, Z., & Gao, S. (2024). Sample-and-Bound for Non-Convex Optimization. In Proceedings of the AAAI Conference on Artificial Intelligence.

ABSTRACT OF THE DISSERTATION

Monte Carlo Tree Methods for Nonlinear Optimization

by

Yaoguang Zhai

Doctor of Philosophy in Computer Science

University of California San Diego, 2024

Professor Sicun Gao, Chair
Professor Francesco Paesani, Co-Chair

As highly nonlinear continuous functions become the prevalent model of computation, NP-hard optimization problems over the continuous domain pose significant challenges to AI/ML algorithms and systems, especially in terms of their robustness and safety. The key to nonlinear optimization is to efficiently search through input regions with potentially widely varying numerical properties to achieve low-regret descent and fast progress toward the optima. Monte Carlo Tree Search (MCTS) methods have recently been introduced to improve global optimization by computing better partitioning of the search space that balances exploration and exploitation.

This dissertation investigates the application of Monte Carlo tree methods for nonlinear optimization (encompassing black-box and non-convex optimization) to identify the global optimum, and the crafting of training datasets designed to boost transferability and reduce dataset size in computational molecular dynamics. To tackle global optimization challenges, the study integrates sampling strategies with MCTS frameworks, employing diverse local optimization techniques to highlight promising samples. These techniques span stochastic search, Gaussian Processes regression, numerical overapproximation of the objective function, and analysis of first- and second-order information. In the realm of training dataset development, computational simulations of water serve as a practical case study. An active learning framework is introduced to efficiently condense the size of the training dataset while preserving its quality and comprehensiveness. The research further explores model transferability by assessing various subsets for training set inclusion to the simulation of water molecules, thereby uncovering the model's adaptability challenges across different scenarios.

The findings affirm that Monte Carlo tree methods provide cost-effective strategies for managing the complexities inherent in state space exploration. By applying these methods to a range of application areas, the dissertation underscores the robustness and utility of sampling techniques in advancing machine learning research.

Chapter 1

Introduction

Global optimization is a classical problem that is notoriously difficult in both science [12, 89, 101, 119, 214, 235] and engineering [14, 39, 131, 162, 193]. As highly nonlinear continuous functions become the prevalent model of computation, these NP-hard optimization problems over the continuous domain pose significant challenges to AI/ML algorithms and systems, especially in terms of their robustness and safety. The key to nonlinear optimization is to efficiently search through input regions with potentially widely varying numerical properties to achieve low-regret descent and fast progress toward the optima, and to distinguish the global optimum from exponentially many potential local optima [89, 234]. Monte Carlo Tree Search (MCTS) methods have recently been introduced to improve global optimization by computing better partitioning of the search space that balances exploration and exploitation. To tackle global optimization challenges, this study integrates sampling strategies with MCTS frameworks, employing diverse local optimization techniques to highlight promising samples.

In this dissertation, we analyze the non-convex and black-box optimizations. The non-convex optimization, as one type of white-box optimizations, is characterized by scenarios where the objective function and constraints are fully accessible. This transparency enables the direct application of analytical techniques, such as gradient-based methods, to efficiently find local optimal solutions. In contrast, black-box optimization (BBO) addresses problems where the internal details of the objective function are unknown or cannot be directly accessed, often encountered in computational simulations and laboratory experiments. Due to the absence of an analytical form, BBO requires the adoption of heuristic or evolutionary algorithms, which rely on the function's output to iteratively identify optimal solutions. There are numerous approaches to tackling optimization problems, including model-based algorithms, search-based algorithms, and partition-based algorithms. Model-based approaches, such as Bayesian Optimization [93, 164], involve learning a surrogate function from samples of the unknown function and optimizing the surrogate rather than the original function. For highly nonlinear functions with high-dimensional input spaces, such methods are known to be costly because of the need for global modeling of the objective functions. On the other hand, search-based algorithms like simulated annealing [79] and Nelder-Mead [60] propose new samples iteratively by updating the sampling distributions over state space, aiming to converge towards the optima. However, these methods may struggle when the optimization landscape is highly non-convex [114] because they lack the mechanisms for understanding the function landscape. Partition-based algorithms, in contrast, divide the state space into partitions, choosing to sample within a selected partition in subsequent iteration [138, 223] (in the context of BBO) or employing pruning techniques to systematically narrow the search towards the global optimum (in non-convex optimization) [71, 142]. Many state-of-the-art partition-based algorithms [138, 190, 223] integrate MCTS [30] due to its efficiency in balancing exploitation and exploration within sampling processes.

We propose a new design of MCTS methods for BBO, with more emphasis on sample-efficient local descent, which can benefit the most from balanced exploitation and exploration. We use Bayesian optimization and local modeling as auxiliary metrics for guiding the search tree construction. At each node in our search tree, we iteratively collect samples in the neighborhood of some anchor point and also maintain a local Gaussian Process (GP) model for the neighborhood. The samples are chosen using sampling-based descent such as Stochastic Three Points methods (STP) [22], and they are also used to train the local GP models. These local models provide surrogate objectives to propose future samples without querying the ground truth function, and they also provide uncertainty metrics for exploration steps. We name our overall approach Monte Carlo Tree Descent (MCTD), because of the focus on faster descent led by samples that are managed by tree search, rather than using MCTS for explicit space partitioning.

For non-convex optimization, we propose an approach that combines the benefits of sampling-based and tree-based approaches as well as interval bounding and local optimization techniques, by taking advantage of the recent progress in Monte Carlo Tree Search (MCTS) methods. We assume that the analytic form of the objective function is known over a compact domain, so that we can use interval bounding [6] on the function value and its local first-order and second-order information in different parts of the MCTS design. A key feature of the Monte Carlo trees is that the growth of the tree is driven by samples rather than partitions, and hence the name *Sample-and-Bound*. By associating the analytic and estimated properties of adjustable neighborhoods around each sample, we design the MCTS algorithm to best balance exploration and exploitation based on the important numerical properties of the objective function. We evaluate the proposed algorithms against a wide range of existing algorithms and analyze the importance of various hyper parameters.

Another part of this dissertation talks about the application of machine learning models used in molecular dynamic simulations. Recent advancements in computer science have made computational simulations a crucial tool for gaining insights into the properties and behaviors of molecular systems [95, 113, 227]. These simulations are carried out by employing potential energy functions (PEFs), mathematical formulations designed to understand and predict the energetic interactions within molecular systems. PEFs establish the basis for simulating molecular behavior at the atomic level [91]. For instance, cutting-edge methodologies apply deep neural networks (DNNs) to learn the potential in a many-body expansion (MBE) form. MBE allows for expressing the total potential energy of a system composed of N (atomic or molecular) monomers ("bodies") as a sum of contributions from each n-body $(1 \leq n \leq N)$, exhibiting unprecedented accuracy and enabling predictive simulations from the gas to the condensed phases [148, 167]. These DNN-based potentials, leveraging the MBE, have become increasingly popular for computer simulations across a wide range of molecular systems, ranging from water [63, 198, 241, 244] and ionic liquids [115] to metals [120, 143, 229]. Such potentials are developed to learn the functional landscape in the configuration space efficiently and effectively. In developing DNN potentials, the quality and representativeness of the training data are crucial to determine the model's performance [94, 105, 247]. Effective sampling strategies are essential for capturing the diversity and complexity of the underlying population, ensuring the training dataset accurately reflects the variations and scenarios the model is likely to encounter. By carefully designing the sampling process, researchers and practitioners can build training datasets that are both manageable in size and highly effective in training robust and accurate models. In our approach, we developed an active learning framework to identify and select representative samples for the training set while excluding unnecessary ones, thereby reducing the cost of determining the ground truth value of the samples. Ideally, a model that reflects the functional landscape beyond its initial training scope is crucial for understanding the fundamental physics of molecular interactions. For instance, a DNN model trained on liquid water should accurately representing the system's behavior across different water phases, such as vapor, liquid, and various ice polymorphs, where the properties of the system significantly diverge. However, our work has shown the limitation of current DNN potentials in computing water molecule properties across various phases. These potentials, developed using DeePMD-kit [220], faces a "short blanket" dilemma, indicating that the current DNN architecture struggles to capture the properties of water in various phases simultaneously without losing its superior accuracy in the liquid phase. This suggested that the fundamental physics is not understood by the DNN model in the research.

Chapter 2 of this dissertation introduces a novel sampling based search framework, Monte Carlo tree descent (MCTD), for BBO that integrates MCTS, enhanced Bayesian optimization, and local search strategies. Unlike traditional tree search algorithms that partition the state space, the introduced MCTD strategy emphasizes utilizing collected samples as tree nodes. This innovative framework combines stochastic search techniques and Gaussian Processes as local descent methods at each tree node, prioritizing rapid local optimization through strategic sampling to enchance sample efficiency. Additionally, it employs the power of MCTS to ensure the balance between exploitation of known data and exploration of new data.

Chapter 3 presents the algorithm “Monte Carlo Tree Search with Interval Bounds and Regional Estimation" (MCIR), designed to tackle non-convex optimization challenges. MCIR is distinguished by its use of numerical overapproximations of the objective as a measure of uncertainty and by learning about specific subregions within the state space through sampled estimates of first-order and second-order information. This approach avoids the usual fixed combinatorial patterns in growing the tree, aggressively zooms into the promising regions, and still balances exploration and exploitation.

Chapter 4 demonstrates an effective active learning strategy for selecting representative configurations to compile the training dataset necessary for training a DNN potential for water molecular simulations. The framework employs uncertainty and error estimation through Gaussian process regression to identify the most relevant configurations needed for an accurate representation of the energy landscape of the molecular system under exam. Furthermore, this framework is applied to Cs+–water system for the energy of its N-body system, essential for developing of Cs+-water many-body potential energy functions. The deployment of the framework leads to significantly smaller training sets than previously used in the development of the original PEFs, without loss of accuracy.

Chapter 5 illustrates a study exploring the possibility of creating a DNN potential for large-scale simulation of water across its phase diagram. The investigation reveals that this potential falls short in simultaneously providing an accurate description of water in its liquid phase and capturing vapor-liquid equilibrium properties. This shortcoming is attributed to the DNN potential’s inability in accurately modeling many-body interactions. An attempt to explicitly include information about many-body effects results in a revised DNN potential. While this potential successfully reproduces the vapor-liquid equilibrium properties, it loses the accuracy in the description of the liquid properties. These findings suggest a limitation of such DNN potentials in accurately predicting properties for state points that are not explicitly included in the training process.

Chapter 2

Monte Carlo Tree Method for Black-Box Optimization

The key to Black-Box Optimization is to efficiently search through input regions with potentially widely-varying numerical properties, to achieve low-regret descent and fast progress toward the optima. Monte Carlo Tree Search (MCTS) methods have recently been introduced to improve Bayesian optimization by computing better partitioning of the search space that balances exploration and exploitation. Extending this promising framework, we study how to further integrate sample-based descent for faster optimization. We design novel ways of expanding Monte Carlo search trees, with new descent methods at vertices that incorporate stochastic search and Gaussian Processes. We propose the corresponding rules for balancing progress and uncertainty, branch selection, tree expansion, and backpropagation. The designed search process puts more emphasis on sampling for faster descent and uses localized Gaussian Processes as auxiliary metrics for both exploitation and exploration. We show empirically that the proposed algorithms can outperform state-of-the-art methods on many challenging benchmark problems.

2.1 Introduction

Black-Box Optimization (BBO), also referred to as Derivative-free or Zeroth-order Optimization, considers objective functions that are not known analytically and can only be evaluated at various inputs, potentially at a high cost. The generality of the formulation makes

BBO broadly applicable to a wide range of challenging problems in machine learning [36, 98, 250] as well as many scientific and engineering problems [101, 214, 235]. BBO problems over compact domains are naturally NP-hard: in the worst case, we need to exhaustively search through the combinatorially-large number of local regions to find high-quality solutions. Thus, the goal of BBO algorithm design is to accelerate optimization progress with respect to the number of function evaluations.

Existing work on BBO can be categorized into model-based and model-free approaches. Most model-based approaches, typically in the framework of Bayesian Optimization [93, 164], involve learning a surrogate function from samples of the unknown function and optimizing the surrogate rather than the original function. For highly nonlinear functions with high-dimensional input spaces, such methods are known to be costly because of the need for global modeling of the objective functions. Various Bayesian optimization approaches utilize ensembles of local surrogate models [55] to improve performance. Model-free approaches include simulated annealing [79], cross-entropy methods [51], search gradient [177], as well as traditional direct search methods such as Nelder-Mead [60, 104]. The goal is to iteratively propose sampling distributions that can approach the optima. Such methods typically do not attempt to maintain global information about the objective and are challenged when the optimization landscape is highly non-convex [114]. In general, the lack of mechanisms for explicitly managing the search over the combinatorially-large number of local regions, in both standard model-based and model-free BBO methods, has been a major bottleneck of the field.

Recent advances in stochastic tree search methods [30, 191] offer new opportunities for balancing local search and modeling with more systematic global exploration in BBO problems. In particular, Monte Carlo Tree Search (MCTS) has recently been introduced for computing good partitioning of the search space for BBO [98, 138, 223]. These approaches adaptively divide the input space into regions, balancing exploitation and exploration, to only perform Bayesian optimization at local regions and create better model ensembles. However, because the focus is still on modeling the objective, the ability of MCTS to quickly expand deep branches into promising search regions has not been fully utilized. As a result, the curse of dimensionality can still quickly stall the search, while model-free descent methods may be able to make more progress if they are also guided by MCTS.

We propose a new design of MCTS methods for BBO, with more emphasis on sample-efficient local descent, which can benefit the most from balanced exploitation and exploration. We use Bayesian optimization and local modeling as auxiliary metrics for guiding the search tree construction. At each node in our search tree, we iteratively collect samples in the neighborhood of some anchor point and also maintain a local Gaussian Process (GP) model for the neighborhood. The samples are chosen using sampling-based descent such as Stochastic Three Points methods (STP) [22], and they are also used to train the local GP models. These local models provide surrogate objectives to propose future samples without querying the ground truth function, and they also provide uncertainty metrics for exploration steps. We name our overall approach Monte Carlo Tree Descent (MCTD), because of the focus on faster descent led by samples that are managed by tree search, rather than using MCTS for explicit space partitioning. We evaluate the proposed methods with experiments on challenging benchmarks such as nonlinear optimization benchmarks [111], policy search for MuJoCo locomotion tasks [210], and neural architecture search [52]. We compare our algorithm with state-of-the-art model-based [55] and MCTS-based methods [223], as well as model-free [76] and direct search methods [60]. We observe clear benefits in the proposed designs for improving efficiency, consistently outperforming existing methods on the tested benchmarks.

2.2 Related Work

Model-based methods.

Bayesian optimization [93, 188] typically uses Gaussian Processes to construct surrogate models of the objective functions [164], with samples selected by acquisition functions (e.g., confidence bounds, expected improvement, etc.) [57, 201]. Model-based methods are known to suffer from the curse of dimensionality as the problem dimensionality and sample sizes grow quickly [146]. Many approaches have been proposed to improve the scalability of Bayesian optimization methods in high-dimensional problems [61, 139, 174]. For instance, TuRBO is a state-of-the-art method that uses Thompson sampling with Expected Hypervolume Improvement (EHVI) [55]. It samples in local trust regions and adjusts the trust regions after each sampling iteration, which has shown major benefits in improving the efficiency of model-based approaches for BBO.

Model-free methods.

Model-free approaches focus on sampling inputs, either point-wise or population-based, that can incrementally approach optimal regions in the search space without explicitly maintaining models of the objective. Standard approaches include stochastic methods such as simulated annealing (SA) [79] and cross-entropy (CE) [51] and deterministic schemes such as Nelder-Mead (NM) [60]. These methods have been successfully applied to a wide range of problems but they typically do not aim for optimizing efficiency, i.e., reducing the number of evaluations [10]. They may still offer improvements faster than local methods that rely on gradient information [10, 104]. The Stochastic Three Points method [22] is a simple but effective way of direct search that compares function values at the base point, in one random direction, and in the opposite direction. Each step evaluates only two more points that lie in the opposite direction of the current point and moves towards the one with a better value. To improve sample efficiency, we attempt to combine The Stochastic Three Points method with model-based methods and carefully design the direction in which the method will try in each iteration.

Tree search methods.

Various tree-search methods have been proposed to improve partitioning of the search space in BBO, such as Deterministic and Simultaneous Optimistic Optimization (DOO and SOO) [138], and Hierarchical Optimistic Optimization (HOO) in [31]. Specifically, DOO divides up the search domain into partitions, each of which is represented by a point within it, assuming known Lipschitz constants for the objective function. SOO and HOO extend DDO to stochastic versions but are mostly applicable to low-dimensional problems because of the high cost involved in creating good partition cells. Voronoi Optimistic Optimization (VOO) [98] can be more efficient in high dimensions by combining Voronoi partitioning and tree search. LA-MCTS [223] introduces MCTS to manage the partitioning of the search space. It learns latent actions that define boundaries between good and bad regions in the search space and prioritizes the expansion of the search tree around the boundaries. When continuing with such splitting, it sets a sampling preferential on every node in the tree. In every iteration, the search tree is traversed from the root node to a leaf by selecting the highest approximated value based on Upper Confidence bounds applied to Trees (UCT) algorithm. The optimization is then performed from the subspace partition on the selected node. These methods successfully change the objective function modeling for global space to local regions. However, the partitioning of the state space, particularly when the space is a high dimension, becomes a very challenging problem. The tree becomes extremely large when the optimization attempts to learn with high accuracy in local regions.

2.3 Preliminaries

We consider the problem of minimizing an objective function $f(x) : \Omega \to \mathbb{R}$ where the domain $\Omega \subseteq \mathbb{R}^n$ is compact. We assume the ability to evaluate $f(x)$ for arbitrary $x \in \Omega$ but do not have information about the analytic form of the function or its derivatives.

Gaussian Processes (GP) is commonly used in Bayesian optimization and is also used in our work to construct a surrogate model for the local model-based optimization. For a finite collection of points $x_1, ..., x_k \in R^d$, GP constructs the mean vector μ_0 from the function f at each x_i, and the covariance matrix Σ_0 by a kernel at each pair of $(x_i, x_j), i, j = 1, 2, ...k$. With μ_0 and Σ_0 the prior distribution on f is:

$$f(x_1, \ldots x_k) \sim \mathcal{N}(\mu_0(x_1, \ldots x_k), \Sigma_0(x_1 \ldots x_k; x_1, \ldots x_k)) \tag{2.1}$$

For any new point x, we can use Bayes' rule to compute the conditional distribution of $f(x)$:

$$f(x|x_1,\ldots x_k) \sim \mathcal{N}(\mu_0(x_1,\ldots x_k,x),\Sigma_0(x_1,\ldots x_k,x;x_1,\ldots x_k,x)) \tag{2.2}$$

The Stochastic Three Points (STP) method is a model-free approach to BBO that uses only a small number of samples in each iteration to identify descent directions. At each time step t with a current sample x_t, it generates a set $D_t = \{x_t, x_t + s_t \cdot \alpha_t, x_t - s_t \cdot \alpha_t\}$ where s_t is a direction and $\alpha_t > 0$ is the step size at step t. When α_t is small enough, the relationship between $f(x_t + s_t \cdot \alpha_t)$, $f(x_t)$ and $f(x_t - s_t \cdot \alpha_t)$ is monotonically non-increasing or non-decreasing if the gradient of the function f is not zero in the direction of s_t. For the next step, $x_{t+1} = \text{argmin}_{x \in D_t} f(x)$.

In our method, the STP-based local descent optimization will identify the best direction s_t with an optimized step size α_t for improving its performance.

Monte Carlo Tree Search (MCTS) is a leading framework for balancing exploration and exploitation in sampling-based tree search. It consists of four main steps: Selection, Expansion, Simulation, and Backpropagation. During Selection, a search tree is traversed from the root node to a leaf node. This traversal is made by selecting the node with the highest value based on the UCT algorithm. For a node n_i, the UCT v is computed by:

$$v(n_i) = R_i/N_i + C \cdot \sqrt{2 \cdot \ln N_b/N_i} \tag{2.3}$$

in which R_i is the rewards on n_i; N_i and N_b denote the number of visits on n_i and its parent node n_b, respectively; C is a constant to balance between exploitation and exploration. At each branch node n_b, the child to select is the one with the highest v value among all of its immediate children. At Expansion, a new child node is then added to expand the tree. During Simulation, a random simulation is run from the new child node until the terminal node is reached, and the simulation reward is approximated. Finally, the simulation reward is backpropagated through the selected nodes to update the tree. In our approach, we construct our Monte Carlo tree by assigning every leaf node to one optimization process. During each step, we use a modified UCT algorithm to select the node on which the optimization is launched.

2.4 Monte Carlo Tree Descent

Our MCTD algorithm iteratively constructs a search tree over the domain of the objective function, and at each node of the tree we maintain a set of samples and a surrogate model learned from them. The balancing of exploration and exploitation takes into account several factors that will be explained in the subsequent sections. The overall algorithm is illustrated in Alg.1, and we refer Fig. 2.1 that provides a visual illustration of the process.

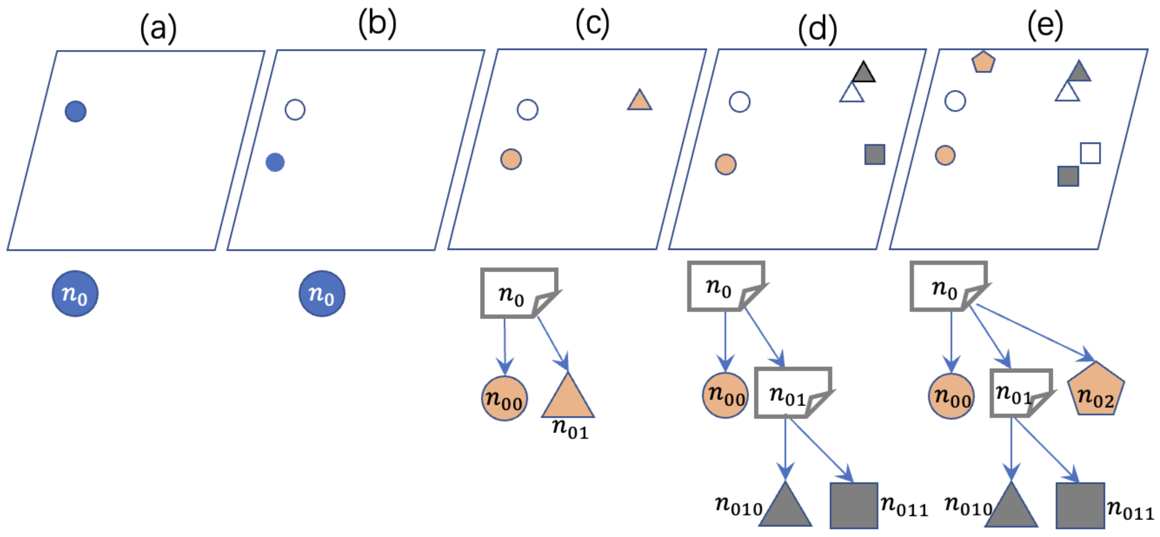


Figure 2.1. Steps in each iteration of the MCTD algorithm. Top: Illustration of the nodes in the input domain; Bottom: Illustration of the tree expansion. (a) The root n_0 begins with a random sample. (b) Optimization is carried out on the root n_0. (c) Leaf exploration on the root N^0 creates two nodes n_{00} and n_{01}. n_{00} starts from x_0^*, and n_{01} starts from a point distant from x_0^*. (d) Leaf exploration on the node n_{01}, generating two new node n_{010} and n_{011}. n_{010} starts from x_{01}^* while n_{011} starts at a point away from x_{01}^*. (e) Branch exploration at root n_0 creates a new child node n_{02}.

Algorithm 1. Monte Carlo Tree Descent (MCTD)

```
1: function MCTD(objective function: $f$, domain: $\Omega$)
2:     $x \leftarrow$ random sample in $\Omega$
3:     $n_0$.sample_set $\leftarrow (\mathbf{x}, \mathbf{y})$    $\triangleright$ root node
4:     for step = $1, \dots, t$ do
5:         $n \leftarrow$ Select($n_0$)
6:         Optimize($n$)
7:         Backup($n$)
8:     end for
9:     return $y_0^*$
10: end function
11:
12: function EXPAND(node: $n_i$)
13:     if $n_i$ is leaf then
14:         $n_{i0} \leftarrow n_i$ excluding parent/child
15:         $n_i$ child list $\leftarrow n_{i0}$
16:     end if
17:     $lv \leftarrow$ level of $n_i$
18:     $d \propto \exp(-lv)$
19:     $x \leftarrow$ random sample in $B(x_i^*, d)$
20:     $n_{im}$.sample_set$\leftarrow \{(x, f(x))\}$;
21:     $n_i$.children.append($n_{im}$)
22:     return $n_{im}$
23: end function
24:
25: function BACKUP(node: $n_i$)
26:     $n \leftarrow n_i$
27:     while $n$ has parent $n_p$ do
28:         Update $(x_p^*, y_p^*)$ with Eq.2.5
29:         Update $dy_p$ with Eq.2.6
30:         $n \leftarrow n_p$
31:     end while
32: end function
```

```
1: function SELECT(node: $n_i$)
2:     $n_b \leftarrow n_i$
3:     while $n_b$ has children do
4:         for child node $n_{bi}$ do
5:             Compute $v(n_{bi})$ by Eq.2.4
6:         end for
7:         Compute $v(n_{bx})$ by Eq.2.7
8:         if $\max_i(v(n_{bi})) < v(n_{bx})$ then
9:             return Expand($n_b$)
10:        end if
11:        $\hat{b} \leftarrow \mathrm{argmax}_i v(n_{bi})$
12:        $n_b \leftarrow n_{b,\hat{b}}$
13:    end while
14:    if $EP(n_b)$ in (2.8) is satisfied then
15:        $n_b \leftarrow$ Expand($n_b$)
16:    end if
17:    return $n_b$
18: end function
19:
20: function OPTIMIZE(node: $n_i$)
21:     $\alpha_D \leftarrow 1$
22:     if $|n_i$.sample_set$| >= NR$ then
23:         $\Theta \leftarrow$ GP model of $n_i$.sample_set
24:         $\alpha_D \leftarrow \alpha_D \cdot$ correlation length in $\Theta$
25:     else
26:         oracle $\Theta \leftarrow$ None
27:     end if
28:     Descend on $n_i$ by $\Theta$, $f$, $\alpha_D$ from $(x_i^*, y_i^*)$
29:     $n_i \leftarrow$ Bayesian Optimize from $\{(x,y)\}_i$
30:     Update $(x_i^*, y_i^*)$ and $dy_i$ by Eq.2.5 and 2.6
31:     return
32: end function
```

2.4.1 Overall Tree Search Strategy

We initialize our algorithm at a random sample in the domain of the objective function, and the sampled points create the root node of the entire search tree. Unlike standard MCTS that considers finite and discrete actions at each node, for BBO over the continuous domains we can not expand the infinitely-many possible next samples as child nodes of the root node.

Consequently, already at the root node, we need to decide between two choices. First, we could perform local descent on the current sample at this node. Second, we could explore a different region in the space by taking a sample that is far from the current one, which will act as a new anchor point that forms a new child node of the tree, which expands the tree. When multiple child nodes have been expanded at a node, there is the third option of going down the tree along the most promising branch, and then focusing the next steps of search from there.

Consequently, in each iteration of the algorithm, we perform three operations sequentially. First, we perform branch selection starting from the root node, and then either land at some existing node or create a new anchor sample and node, from which we will perform local descent.

2.4.2 Branch Selection

In every step, we pick a leaf for optimization. To balance exploration and exploitation, our algorithm uses UCT to determine the path between the root and the leaf, as shown in the function **SELECT** in Alg.1 line 1. We modified the UCT formula for fitting our MCTD algorithm. For each child node n_{bi} with the parent node n_b , its UCT $v(n_{bi})$ is given by:

$$v(n_{bi}) = -y^*_{bi} + C_d \cdot \sum_{j=1}^{J} dy^{-j}_{bi} + C_p \cdot \sqrt{\log N_b / N_{bi}} \tag{2.4}$$

Here, C_d is a weight factor controlling the importance of recent improvements during optimization, C_p is a hyper-parameter for the extent of exploration, N_b and N_{bi} are the number of visits to the branch node n_b and the child node n_{bi}, respectively. y^*_{bi} is the normalized current best function value in the sample set $\mathbf{S}_{bi} = \{(x,y)\}$ which stores the samples during optimization on node n_{bi}:

$$(x^*_{bi}, y^*_{bi}) = \operatorname{argmin}_y(x,y), (x,y) \in \mathbf{S}_{bi} \tag{2.5}$$

and dy^{-j}_{bi} is the normalized most recent j's improvement at n_{bi} after calling the objective function. When computing v, only the last J improvements are taken into account. For every call to the objective function during the optimization, we record the improvement in the function value from this call. If the value from this call is worse than the optimal value before the call, we set the improvement to zero; otherwise, we set the improvement as the absolute difference between the optimal value before and after the call. That is, for y^*_{bi} at the time step t as $y^*_{bi}(t)$,

$$dy^{-j}_{bi}(t) = \max(y^*_{bi}(t-j) - y^*_{bi}(t-j+1), 0) \tag{2.6}$$

We similarly integrate the tree expansion as the UCT algorithm. At a branch node n_b, in addition to examining the UCT of all its child nodes we add an artificial exploration node n_{bx} that has the UCT value $v(n_{bx})$ as following:

$$v(n_{bx}) = -\sum_i (y^*_{bi})/D_b + C'_p \cdot \sqrt{\log N_b} \tag{2.7}$$

where D_b is the number of children of the node n_b, C'_p is a hyper-parameter for the extent of exploration but may be different from C_p. This exploration node is to determine whether to optimize in a new domain because the existing children are not performing well enough. When the exploration node is selected, a new child node under the branch node is created and returned.

If the path selects a leaf that is not newly created, we need to determine whether it is worth optimizing on it. On a leaf node n_f, we expand the tree if the following condition is met:

$$EP(n_f) : -y^*_f + C''_d \cdot \sum_{j=1}^{J''} dy^{-j}_f < C''_p \cdot \sqrt{\log N_f} \tag{2.8}$$

Here, C''_d is a weight factor for recent last J'' improvements and may be different from C_d, C''_p is also a hyper-parameter for the extent of exploration different from C_p and C'_p. In the event the condition 2.8 is met, we will make a leaf expansion; otherwise, we descend on the selected leaf node n_f.

2.4.3 Tree Expansion

When we need to take an exploration step at a node, a new child node will be created. The new child node is created at a random point lying within some distance from the selected node. The minimum and maximum distances are set to 10% and 50% of the domain's dimensional length, with exponential decay according to the node level. After the newly created child node is placed, it will be immediately selected as the node for optimization at the current step. When the selected node to explore is a leaf node n_f, a new child node n_{f1} is created in the same way as above, making n_f a branch node. At this time, a new node n_{f0}, starting from the current best point at x_f^*, is also created as the child 0 of node n_f. This node n_{f0} inherits a batch of samples that are near its starting point x_f^*, as well as the latest improvement history on n_f. The reduced number of samples forces the inheriting node n_{f0} to focus on optimizing in the neighborhood of the starting point, while the newly expanded node n_{f1} is optimizing in a distant region. Thus, the tree grows a leaf node n_{f1} while maintaining the possibility of further exploiting around the best point found on n_f at node n_{f0}. These steps are in the function **EXPAND** in Alg. 1 line 12, and 3 subplots in Fig.2.1 show an example. As in Fig.2.1 (c), the expansion takes place on the root node n_0. The node n_{01} is a new node for exploration, placed distant from n_0. Node n_{00} starts from x_0^*. Similarly, in 2.1 (d), node n_{010} starts from x_{01}^*, and node n_{011} is placed away from n_{01}, but the distance between node n_{01} and n_{010} is much smaller than the distance between node n_0 and n_{01} at node creation. Fig. 2.1 (e) shows how a new leaf node is created.

2.4.4 Local Optimization.

In every iteration, we use the STP method to attempt local descent and also use TuRBO-1 [55] for local Bayesian optimization (BO). We tightly integrate the two methods. Samples obtained from local descent optimization are used to construct the surrogate GP regression model. The GP model not only serves as an oracle for the local descent optimization but also provides the correlation length according to which the local descent optimization scales its step sizes.

Local Descent.

We use the STP method with the following changes. In STP, the direction s_t at step t is usually selected from a sphere with uniform distribution in direct search. Instead, we use the surrogate GP regression model to identify the point with the highest expected improvement. The steps of local descent optimization are as follows:

1. Choose a node n_i by **SELECT**. If the number of samples exceeds some threshold, we train a Gaussian Process model that will be referred to as the oracle for this node.

2. Compute the step size α_t. In our case, we set α_t to be inversely proportional to the square root of the product of node visits N_i and the node level in the tree. We also rescale it according to the correlation length in the surrogate GP model when possible.

3. If the oracle is not available, get a random direction s_t, and use $s_t \cdot \alpha_t$ for checking ground truth.

4. If the oracle is available, generate multiple samples in the box with edge length equaling the step size α_t, and choose the best point. The direction to the best point is $s_t \cdot \alpha_t$.

5. Start one step of STP with the selected direction of $s_t \cdot \alpha_t$ by calling the objective function.

6. Depending on the optimization progress, we may further optimize the objective function along the same direction with tuned step sizes in a fine-grain descent approach.

The last step is used when the optimization comes to fine-tuned phase with small variations in samples, so one can set a function threshold from which the search applies the fine-grain descent approach.

Local Bayesian Optimization.

The TuRBO-1 [55] creates a hyper-rectangle Trust Region (TR) with volume L^N centered at the best sample. Afterward, it samples new candidates within the TR and queries the objective function for ground truth data. The length of L_i will either increase after successive "successes" or decrease after consecutive "failures". We changes TuRBO-1 in three ways to fit it into our algorithm: 1) TuRBO-1 begins with collected samples of the node. Consequently, TuRBO-1 is compelled to optimize from the vicinity of the collected sample. 2) The trust region length has been preserved on the same node, so the local BO can continue from the previous epoch. 3) We do not perform restarts for TuRBO-1 in order to avoid TuRBO-1 restarting from random samples.

2.4.5 Back Up

In the **BACKUP** function, we backpropagate the updated best score found at a leaf node and propagate it upwards to its parent nodes. This score update is important for informing future branch selections. This backup procedure is used in every step even if the best-found sample on the selected leaf node does not change after one iteration.

2.4.6 Implementation Details of Stochastic Three Point Approach

In Alg.2 , we can see the typical workflow of the stochastic three-point algorithm (STP).

Algorithm 2. Stochastic Three Point

```
while within Descent budget do
    dx ← random sampling
    X ← argmin(f(X), f(X + dx), f(X - dx))
end while
```

For our implementation, we have redesigned two versions of STP which are compatible with our search approach. In first place, the descent direction dx is not entirely random, but is sampled by Latin Hypercube Sampling within the domain of the step size. The length in each dimension is rescaled by the correlation length L from the local surrogate model, while maintaining the total length of dx: $dx = dx \cdot L \cdot ||dx||/||dx \cdot L||$. Further, the choice of dx is made by maximizing the expected improvement on the local surrogate model G: train $G(x), x \in$

collected samples at the node, and $dx = \text{argmin}_{dx} G(X + dx)$, where X is the best point at the node.

The first implementation is fundamentally similar to the typical STP as in Alg.3, with the exception that it continues to test along the same dx whenever it find a better value for $G(X + dx)$ or $G(X - dx)$:

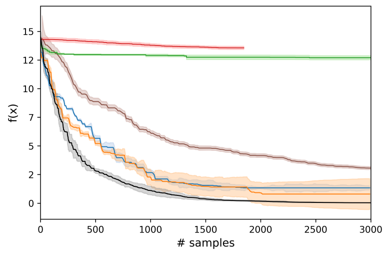
Algorithm 3. STP for local descent optimizer

```
$X \leftarrow$ best point at the node
Train surrogate model $G(x), x \in$ collected samples
**while** within Descent budget **do**
    $\mathscr{DX} \leftarrow$ Latin Sampling $\cdot$ Correlation Length $L$ $\cdot$ Step Size $\alpha$
    dx $\leftarrow \text{argmin}_{dx \in \mathscr{DX}} G(X + dx)$
    $k \leftarrow 0$
    **while** $G(X + (k+1) \cdot dx) < G(X + k \cdot dx)$ **do**
        $k \leftarrow k + 1$
    **end while**
    X $\leftarrow$ argmin( $f(X + dx), f(X)$ )
**end while**
```

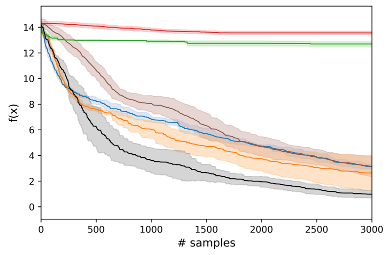
Algorithm 4. Fine-grained STP

```
$X \leftarrow$ best point at the node
Train surrogate model $G(x), x \in$ collected samples
**while** within Descent budget **do**
    $\mathscr{DX} \leftarrow$ Latin Sampling $\cdot$ Correlation Length $L$ $\cdot$ Step Size $\alpha$
    dx $\leftarrow \text{argmin}_{dx \in \mathscr{DX}} G(X + dx)$
    $k_0$, $k_-$, $k_+ \leftarrow$ 0.0, -1.0, +1.0
    **while** within computational budget **do**
        $k_0 \leftarrow \text{argmin}_{k \in (k_0, k_-, k_+)} G(X + k \cdot dx)$
        update $k_-$, $k_+$
    **end while**
    X $\leftarrow$ argmin( $f(X + k_0 \cdot dx), f(X)$ )
**end while**
```

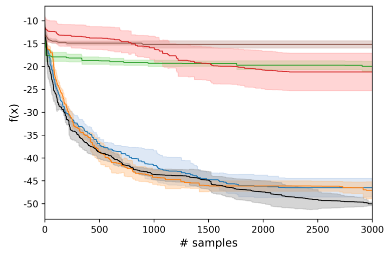
The second one differs from the first implementation by always trying to test two more points, which have never been tested, on either side of the current point. As an example, if X, $X + dx$, and $X - dx$ are currently being compared, and $X + dx$ is the best point of the three, the two points that will be tested in the next step are $X + 2 \cdot dx$ and $X + 0.5 \cdot dx$ (since $X + dx$ has



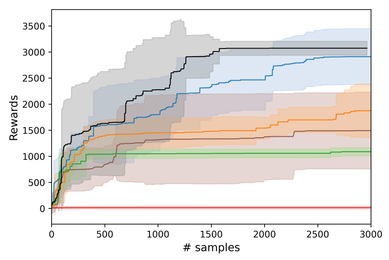
(a) Ackley-50d



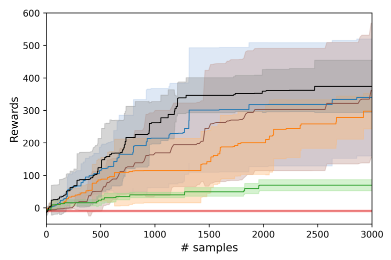
(b) Ackley-100d



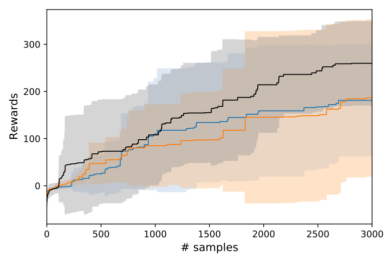
(c) Michalewicz-100d



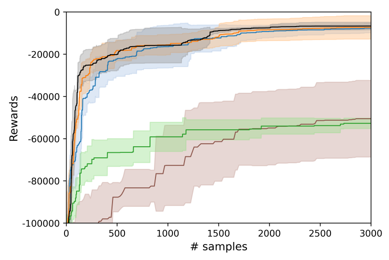
(d) Hopper-33d



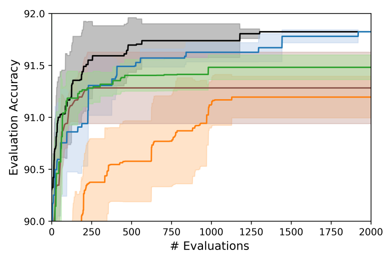
(e) Walker-102d



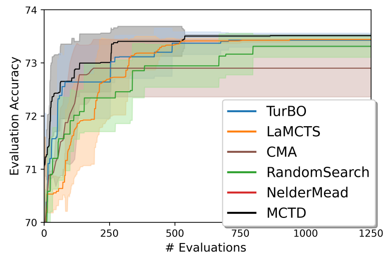
(f) Walker-204d



(g) HalfCheetah-102d



(h) CIFAR-10



(i) CIFAR-100

Figure 2.2. Overall performance of the baselines and MCTD. For Ackley and Michalewicz in (a), (b), and (c), the goal is to optimize for the lowest function values; in MuJoCo tasks (d), (e), (f), and (g), we aim to maximize the rewards; and for CIFAR-10 in (h) and CIFAR-100 in (i) we want to find the architecture with the highest accuracy as quickly as possible

already been tested and we need to select two more points in either side of $X + dx$ for evaluation). Should X be the best point among X, $X + dx$, and $X - dx$, the next round will be to test X, $X + 0.5 \cdot dx$, and $X - 0.5 \cdot dx$. The second version of the STP model, illustrated in Alg.4, may yield better results in fine-grain, however it is more computationally expensive. As a result, we switch to this fine-grained model when the function value drops below a threshold.

2.5 Experiments and Evaluation

2.5.1 Experiment Setup

Benchmarks

We use several standard benchmark sets for testing BBO algorithms, from three categories: synthetic functions for nonlinear optimization, reinforcement learning problems in MuJoCo locomotion environments, and optimization problems in Neural Architecture Search (NAS). Synthetic functions are widely-used in nonlinear optimization benchmarks [111]. These functions usually have numerous local minima, valleys, and ridges in their landscapes which is hard for normal optimization algorithms. MuJoCo locomotion environments [210] are popular for reinforcement learning tasks. NAS problems have practical significance, since many fields are using deep learning models, but implementing efficient neural networks requires a substantial amount of time and effort. We select multiple problems from each set, and their input dimensions range from 33d to 204d.

Synthetic Functions

We chose Ackley and Michalewicz from the synthetic function set in the nonlinear optimization benchmark [111]. Ackley is a function with multiple local minima, and Michalewicz has steep valleys and many ridges. We use Ackley-50d, Ackley-100d, and Michalewicz-100d as our benchmark.

MuJoCo Locomotion

For reinforcement learning problems from MuJoCo locomotion environments [210], we chose Hopper, Walker, and HalfCheetah for tests. Hopper has 3 dimensions in action space a and 11 in observation s. We choose a linear policy $a = Ws$ in which W is the weighting matrix to search for maximizing the reward, therefore, the search space for Hopper-33d is in the dimension of $3 \cdot 11 = 33$. Similarly, we set linear policies in both Walker-102d and HalfCheetah-102d. In addition to the above linear policy, we double the weighting matrix space dimension in Walker from 102 to 204, such that $a = W_1 s + W_2 s$ where W_1 and W_2 are matrices in the dimension of 102. In this case, the optimization problem is Walker-204d. Since our approach considers deterministic results, we set the noise scale to zero in all MuJoCo environments to avoid randomness in rewards.

Neural Architecture Search

For the NAS benchmark, we use two datasets CIFAR-10 and CIFAR-100 from NAS-Bench-201 [52]. Each network in the datasets consists of three stacks of searching cells, and each cell has six positions where one can select one type of layer from five different types: (1) zeroize, (2) skip connection, (3) 1-by-1 convolution, (4) 3-by-3 convolution, and (5) 3-by-3 average pooling layer. Overall, there are $5^6 = 15625$ different types of architectures, and each architecture is trained and evaluated on both CIFAR-10 and CIFAR-100. The accuracy of training and evaluation is recorded. To benchmark this set, we created the following functions in the real domain: we replace each of the five types of layers with an integer, and the real-valued input is rounded up to the nearest integer. The evaluation accuracy of the architecture is set as the function value. As an example, we set the input domain to $\{[0.5, 5.5]^6\}$, and $f([1.1]^6) = f([1]^6)$, where each $1-5$ corresponds to one type of the layers. It should be noted that in this method different inputs may refer to the same network architecture; therefore, the number of unique architectures examined is less than the number of functions called.

Baselines

We selected TuRBO [55] as one baseline from the BO algorithms. La-MCTS [223] is chosen as a major comparator since this algorithm also constructs trees in a similar manner. Moreover, CMA-ES [76] from the Evolutionary Algorithm category, Nelder-Mead [60] from Direct Search algorithms, as well as the Random Search algorithm are selected for comparison as baselines.

For CMA optimization, **fmin2** from the CMA-ES package [76] is used with its default parameters. We implement our own version of the Nelder-Mead algorithm as in [60], and set its expansion coefficient, contraction inside the simplex, contraction outside the simplex, and shrink coefficient as $2.0, 0.5, 0.5, and 0.5$, respectively. TuRBO [55] is initialized with 20 random samples selected using Latin Hypercube sampling, and its Automatic Relevance Determination (ARD) is set to **True**. For La-MCTS [223], we use different settings and include them in Tab.2.1, as well as our MCTD approach in 2.2. Benchmarks are made mainly on Google Colab with a Tesla P100 graphic card. Across all experiments, we set the number of evaluation calls to 3000.

2.5.2 Hyperparameters

In this chapter, we demonstrate the hyperparameters for various test functions in LaMCTS and MCDesent

LaMCTS

In LaMCTS, C_p is responsible for controlling the amount of exploration. Having a small C_p will make the search focus exclusively on the current best found value, but may result in being stuck at a local optimum. By contrast, a large C_p encourages LaMCTS to explore poor regions more frequently, but this can result in overexploration. The type of kernel and gamma determine the shape of the boundary drawn by the classifier in LaMCTS. Additionally, the leaf size determines the splitting threshold and the rate of tree growth. In all tests LaMCTS uses TuRBO-1 sampling method as default. All hyperparameters for LaMCTS are as listed below:

Table 2.1. Hyperparameters used in LaMCTS for each of the test functions

Functions	C_p	Leaf size	Num. of initial	Kernel type	Gamma type
Ackley-50d	1.	10	40	rbf	auto
Ackley-100d	1.	10	40	rbf	auto
Michalewicz-100d	10.	8.	40	rbf	auto
Hopper-33d	10	100	150	poly	auto
Walker-102d	20	10	40	poly	scale
Walker-204d	20	10	40	poly	scale
HalfCheetah-102d	20	10	40	poly	scale
CIFAR-10	10	8	10	poly	auto
CIFAR-100	10	8	10	poly	auto

MCTD

As part of our MCTD approach, we have several hyperparameters that can be tuned during tests. As a first step, we may allow a specific number of computational calls from both the local descent optimizer and the local BO optimizer to the objective function. Typically, the total number of calls allowed in a single iteration is either 30 or 40 in order to ensure that enough steps are performed by the optimizers in one iteration. One may, however, want to combine the two algorithms to maximize the benefits. Such a situation could be addressed by splitting the budget among a local descent optimizer and a local BO optimizer according to different ratios. Furthermore, in local descent, we can specify the step size α for the optimizer, and when to change over to fine-grained descent optimization. The step size in our algorithm is relative to the dimensional length of the test function, and we set to use the fine-grained descent optimizer when the best found value on the node is below a threshold. As a third point, the UCT of each node is determined by the equation

$$uct_i = -y_i^* + C_d \cdot \sum_{j=1}^{J} (dy_{i,-j}) + C_p \cdot \sqrt{\log n_{parent}/n_i} \tag{2.9}$$

, and one can adjust the weight of recent improvement C_d and the weight for exploration C_p. As a final step, we must decide when to expand the branch and the leaf node when selecting a path from the tree. To do this, we compute an additional UCT value that has the following setting: $y^{*'} = \sum(y_i^*)/N$, $C_d^{'} = 0$, and $C_p^{'} \neq C_p$ at every branch node, where N is the number of children at the branch node. This additional UCT value represents if the branch decides to explore in a new child domain, because the existing children are not performing well enough. And we apply the following criteria after selecting a leaf node in order to determine whether it is worth exploring or exploiting:

$$-y^* + C_d^{''} \cdot \sum_{j=1}^{J^{''}} dy_{-j} > C_p^{''} \cdot \sqrt{\log n_{leaf}} \tag{2.10}$$

To summarize, we have the budget ratio, step size at local descent, function value at which using fine-grained descent, C_d and C_p at computing node UCT, $C_p^{'}$ for branch exploration, and $C_d^{''}$ and $C_p^{''}$ for leaf exploration. The hyperparamters used for each test is as in Tab.2.2:

Table 2.2. Hyperparameters used in MCTD for each of the test functions

Functions	Bud. Rat.	α	Switch at $f(x)$	C_d	C_p	$C_p^{'}$	$C_d^{''}$	$C_p^{''}$
Ackley-50d	1:1	0.2	10	10	0.5	0.1	50	0.1
Ackley-100d	1:1	0.2	4	20	0.5	0.1	5	0.1
Michalewicz-100d	1:2	0.02	-30	50	1.	0.2	1	10
Hopper-33d	1:2	0.1	-1000	100	1	10	100	200
Walker-102d	1:2	0.01	-100	100	0.1	50	50	50
Walker-204d	1:2	0.01	-100	100	0.1	50	50	10
HalfCheetah-102d	1:2	0.01	35000	50	1	1000	100	10
CIFAR-10	1:4	0.5	5	50	1	1	100	10
CIFAR-100	1:4	0.5	5	50	1	1	100	10

2.5.3 Overall Performance

Evaluation Metrics

For each benchmark function, we run baselines and our algorithm by at least five different random seeds. Due to the limit on the computational power available to us, we set the number of calls to the objective function to 3000. Our study evaluates the best-found value at every step and computes the mean and standard deviation of all runs. As a result, we can compare the best-found value at the end of the run as well as the speed at which each algorithm is capable of reaching the most optimal result. There is a possibility that some algorithms will find the optimal value before 3000 calls, which will result in an early stop.

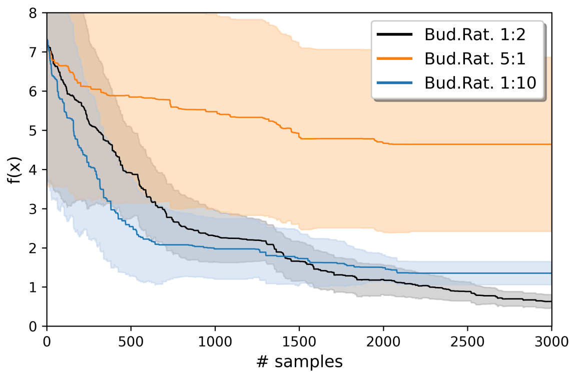
Efficiency

Fig. 2.2 illustrates the comparison between our model and baselines on benchmark sets. It was found that in general, random search, CMA, and NM methods performed poorly in these cases since they do not cooperate with any approach that may potentially improve the efficiency of the sample.

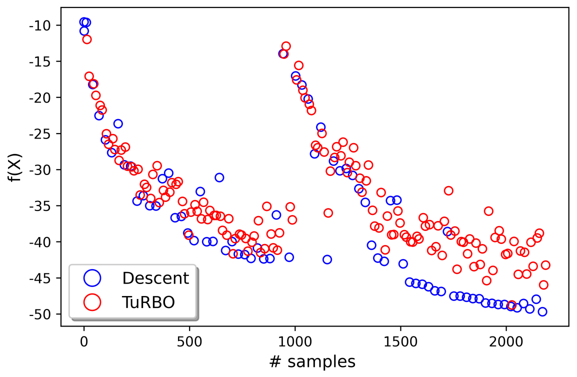
According to Fig. 2.2 (a), (b), and (c), MCTD significantly improves the speed of finding better results for the set of synthetic functions compared to TuRBO and La-MCTS. In particular, the Ackley synthetic function exhibits a noticeable improvement when we balance local optimization exploitation and state space exploration. Michalewicz is improved moderately through descent optimization, and MCTS helps improve the optimization consistently.

The Mujoco benchmark problems are very difficult for global optimization. Our approach is competitive with TuRBO and La-MCTS on this set and has moderate improvement over the average value on functions Hopper-33d, Walker-204d, and Cheetah-102d. In particular, the combination of local BO and local descent optimization speeds up the optimization during its early stages. It is, however, difficult to balance local exploitation and space exploration by picking the correct weights to bring recent improvements, exploration terms, and objective function values into the same order of magnitude. This is because we use the absolute value of the objective function that varies significantly at different optimization steps. In light of this, we see a large variation from different runs in this set, as in Fig.2.2 (d), (e), (f), and (g).

In CIFAR-10/CIFAR-100, MCTD reaches the optimal solution by a small number of samples, which is critical for NAS searches. The combination of descent and modeling approaches facilitates the search for the optimal solution more quickly than if only one method was used.



(a) Optimization by different budget ratio



(b) Queried value from local optimizers

Figure 2.3. Study of budget ratio and local optimization approaches in MCTD. (a) illustrates the optimization curves for Ackley-100d when the computational budget is divided between local descent and local BO in the ratio of 1:2, 5:1, and 1:10; (b) shows the values of Michalewicz-100d from local descent and local BO at each query.

Descent Optimizer and Bayesian Optimizer

We examine the performance of our approach when the computational budget is divided between a local descent model and a local Bayesian optimizer TuRBO. Fig. 2.3a illustrates the optimization history of Ackley-100d when budget ratios are 1:2, 5:1, and 1:10. It is demonstrated that a model with a high budget for local descent suffers from a low optimization rate. In contrast, the model with a high budget in the local Bayesian optimizer may have difficulty escaping the local optimal point.

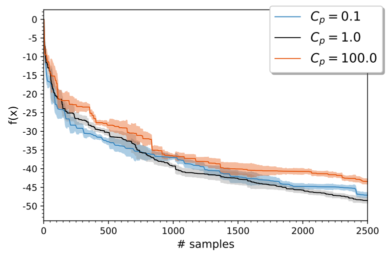
As shown in the case of Fig.2.3a, when we use the budget that emphasizes local descent (budget 5:1), the performance is less compared with that of emphasizing local BO (budget 1:20) in term of optimization speed. Based on the budget ratio for every function in Tab.2.2, it is generally advantageous to use at least the same (or even more) amount of computational budget on local BO as on local descent. This may be one challenge for the local descent approach, since this indicates that local descent may require local BO as the oracle when the function landscape is difficult, such as for the complicated functions in MoJoCo locomotion and non-continuous functions in NAS sets.

However, the local descent approach still proves beneficial despite these factors. From Fig.2.3b we can see the optimization improvement of the local BO becomes insignificant when the process is close to the local optimum. Local descent, on the other hand, can contribute steadily to the discovery of a superior solution. To conclude, using a balanced approach can yield better results than using each approach separately.

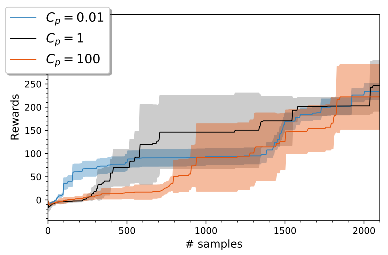
2.5.4 Best Found Value in Test Functions

On each of the functions tested, we present the best results using different methods and the fewest steps to achieve that result. Note for function Ackley-50d, Ackley-100d, and Michalewicz-100d we want to minimize the function value. In contrast, for MuJoCo tasks and NAS tests we want to find the highest reward or the highest accuracy.

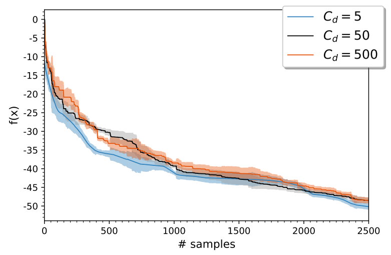
Tab.2.3 illustrates that our MCTD approach obtains the best value with relatively fewer



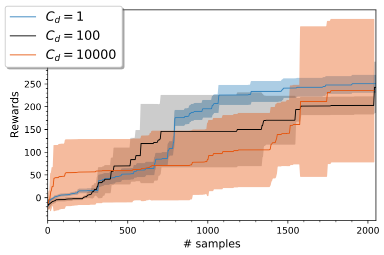
(a) Michalewicz-100d



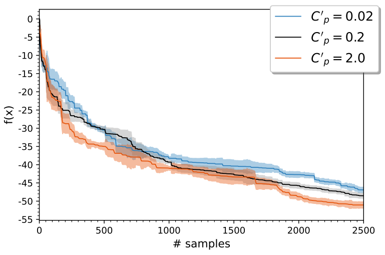
(b) Walker-102d



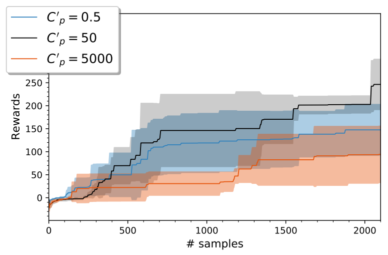
(c) Michalewicz-100d



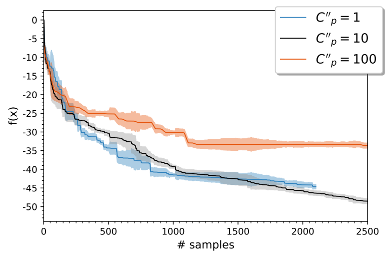
(d) Walker-102d



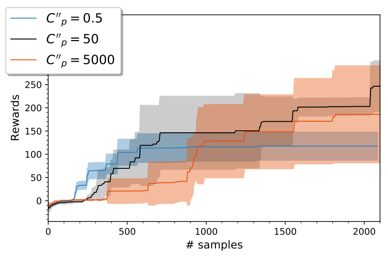
(e) Michalewicz-100d



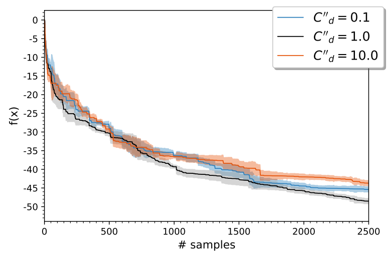
(f) Walker-102d



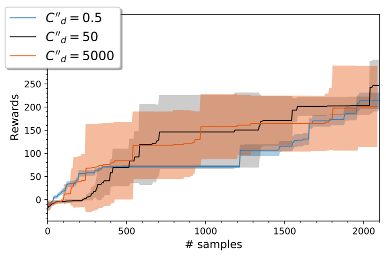
(g) Michalewicz-100d



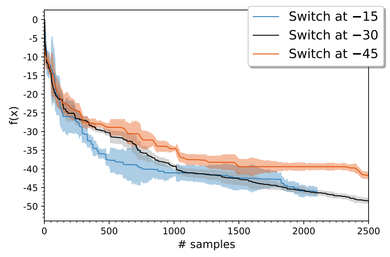
(h) Walker-102d



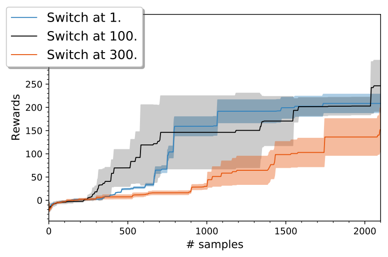
(i) Michalewicz-100d



(j) Walker-102d



(k) Michalewicz-100d



(l) Walker-102d

Figure 2.4. Ablation studies on MCTD hyper-parameters. Ablation studies on function Michalewicz-100d with hyper parameters (a) C_p, (c) C_d, (e) C'_p, (g) C''_p, (i) C''_d and (k) switching to fine-grain STP at function value; ablation studies on function Walker-102d with hyper parameters (b) C_p, (d) C_d, (f) C'_p, (h) C''_p, (j) C''_d and (l) switching to fine-grain STP at function value.

steps in most of the test cases out of five attempts. In particular, our approach performs reasonably well for cases where descent optimization can significantly improve the optimization performance.

Table 2.3. Best Found Value / earliest step toward reaching that value from all tested functions by MCTD, TuRBO, and LaMCTS. Bolded result is the best one among all tested methods.

Functions	MCTD	TuRBO	LaMCTS
Ackley-50d	**0.07/2342**	1.33/1889	0.80/2018
Ackley-100d	**0.29/2826**	2.81/2891	1.89/2971
Michalewicz-100d	**-51.13/2975**	-49.08/1776	-49.87/2945
Hopper-33d	3204/1890	**3397/2574**	2802/2858
Walker-102d	490/2472	**665/1316**	379/2056
Walker-204d	**993/2884**	862/1673	498/1830
HalfCheetah-102d	**-3268/2446**	-4679/2064	-4034/2970
CIFAR-10	**91.82/86**	91.82/1296	91.48/2724
CIFAR-100	**73.52/22**	73.52/80	73.49/2433

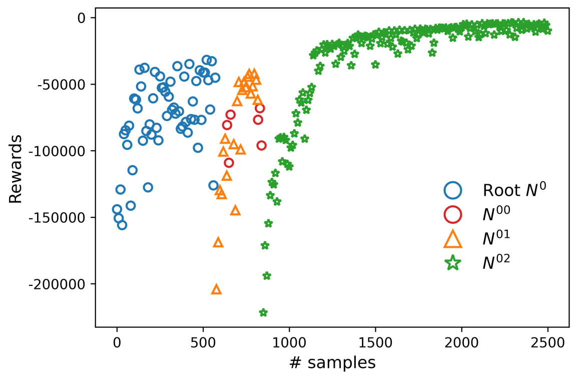
Table 2.4. Best Found Value / earliest step toward reaching that value from all tested functions by CMA, Nealder-Mead, and Random Search

Functions	CMA	Nealder-Mead	RandomSearch
Ackley-50d	0.13/3000	13.21/1225	12.32/1311
Ackley-100d	1.77/3401	13.34/1616	12.37/1326
Michalewicz-100d	-40.35/9015	-27.69/2017	-21.22/1759
Hopper-33d	3043/4128	67/4603	1220/193
Walker-102d	657/3264	-4/414	91/1957
Walker-204d	551/3386	-	-
HalfCheetah-102d	-22062/3145	-101228/2271	-50542/1145
CIFAR-10	91.70/198	-	91.56/458
CIFAR-100	73.52/135	-	73.52/338

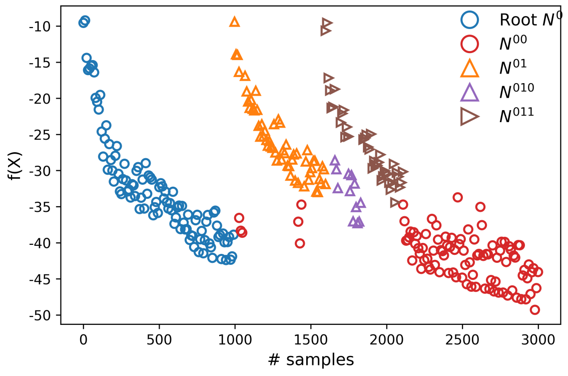
However, it should be noted that in some instances our approach leads to large variations between the different attempts. The adjustment of sample methods may provide one method for improving the descent optimization on those functions. Tab.2.4 shows the results from three other baselines.

2.5.5 Selection of Nodes

It is important to justify the expansion of the tree. Fig. 2.5 illustrates the nodes from which the query is made for the objective function. In Fig. 2.5a, the root node is optimized for the first 200 queries; however, no significant improvement is evident for the next 300 queries. At this point, the tree decides to expand, so it creates a new child node, N^{01}, and starts optimizing from this child. Nonetheless, the optimization is also stuck after 200 more queries. Therefore, our tree abandons to optimize in N^{01}, and adds a new child node named N^{02}. On N^{02}, the optimization procedure is significant, and a new best value is found. The tree in Fig. 2.5b attempts to optimize in the N^{01} and its new exploration child N^{011}, however, the improvements on these nodes are insignificant. Consequently, the tree decides to optimize from the root inherit node. In light of the newly gathered samples upon exploring N^{01} and N^{011}, optimization is able to proceed at the root inherit node. They demonstrate that the tuned tree model is capable of optimizing by selecting a correct node.



(a) Node queries on HalfCheetah-102d

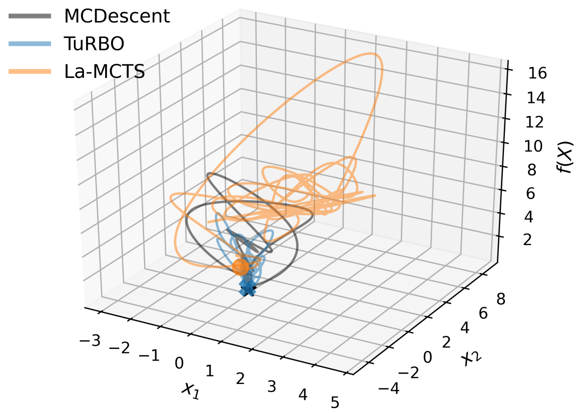


(b) Node queries on Michalewicz-100d

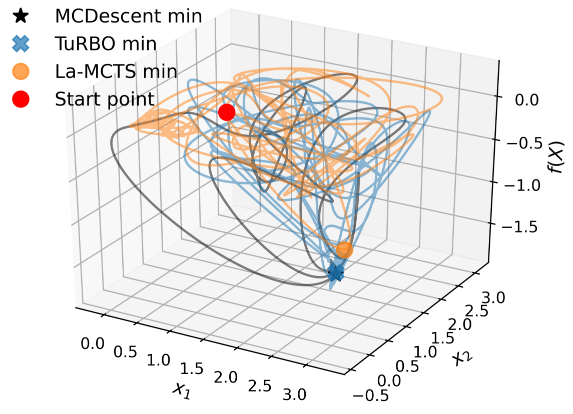
Figure 2.5. Illustration of nodes at queries to the objective function

2.5.6 Optimization route

Fig. 2.6 illustrates how MCTD, TuRBO, and LaMCTS optimize Ackley-2d and Michalewicz-2d in the first 30 samples after initialization. In both plots, LaMCTS explores a wide range of input domains, making it less likely to find a solution by a small number of calls. TuRBO locates efficiently the area where the optimal point is located in the beginning, however, its subsequent samples are diverse and fail to identify the global optimal solution. MCTD, on the other hand, samples much closer to the global optimal point and thus finds the solution more rapidly.



(a) Search path on Ackley-2d



(b) Search path on Michalewicz-2d

Figure 2.6. Search paths of different algorithms. The black star, the blue cross, and the orange circle indicate the best values found by MCTD, TuRBO, and LaMCTS, respectively; the red dot represents the starting point of all three methods.

2.5.7 Ablation Studies

We also perform ablation studies to understand the effect of the hyperparameters used in the algorithm, in three categories. The first category includes C_p' (the weight in uct_{exp}) C_p'', and C_d'' (the weights on leaf exploration in Eq.2.8) that control the expansion of the tree. The second set of values, C_d and C_p in Eq.2.4, balance local exploitation and space exploration. Lastly, the threshold value determines when fine-grain descent is required. We use the synthetic function Michalewicz-100d and the locomotion Walker-102d for the ablation study, and each case runs with at least 3 different seeds. Please note that hyperparameters in results may be different than those presented in Section 5.2. We found that a wise choice on C_p, C_p', and C_p'' is critical to improving performance, while C_d and C_d'' are less significant. The switching threshold value is highly dependent upon the objective function's properties.

State Space Exploration

The parameters C_d and C_p balance exploration and exploitation of the existing tree. As shown in Fig.2.4a, the moderate choice on C_p improves the overall performance slightly; however, this is not clearly observed in Fig.2.4b. From Fig.2.4c and 2.4d, we can see a variation in C_d may not help significant changes in the overall performance. Even so, we can observe a contribution from C_d and C_p: from Fig. 2.4b and Fig.2.4d, we can see that path selection with low values of C_d and C_p leads to little variation between runs since the path selection tends to select the node where the current best-known value resides, thus limiting the path selection.

Tree Expansion

The hyperparameters C'_p, C''_p and C''_d are important for expanding the current tree. As a result of setting the parameters C'_p, C''_p to large values and C''_d to a small value, it is likely that a new sibling leaf will be created at the selected node to explore the state space. Alternatively, the path will tend to select the node that has the current optimal value. Since a leaf always has zero children, even though C''_d and C''_p have the same functionality as C'_p, the criteria for tree exploration are different for branch and leaf nodes. According to Fig.2.4f and Fig.2.4h, it is evident that a good choice on C'_p and C''_p can improve the optimization performance by exploring new state space distant from the local optimal value. Conversely, when their values are set either too large (orange lines in Fig.2.4f and Fig.2.4g) or too small (blue line in Fig.2.4h), this would adversely affect the overall performance of the optimization process. The effect of C''_d is less noticeable. However, a small C''_d results in a small variation between different runs - a similar behavior as C_d.

Switching at Function Value

The fine-grain STP can be beneficial in certain cases, as the orange lines show in Fig.2.4k and Fig.2.4l. In these two lines, switching takes place at a late stage of optimization, which results in excessive use of normal STP. Generally, fine-grain STP can be used as soon as possible. However, in some experiments, the fine-grain STP exploits too much in a small neighborhood at an early stage of optimization and led to low-quality GP models.

2.6 Conclusion

In this paper, we proposed novel designs for using the MCTS framework in BBO problems, with more emphasis on sample-efficient local descent, instead of using MCTS for explicit space partitioning. We design new descent methods at vertices of the search tree that incorporate stochastic search and Gaussian Processes. The local models provide surrogate objectives to propose future samples without querying the ground truth function, and they also provide uncertainty metrics for exploration steps. We propose the corresponding rules for balancing progress and uncertainty, branch selection, tree expansion, and backpropagation. We evaluated the proposed methods on challenging benchmarks and observed clear benefits in improving the efficiency of BBO methods.

2.7 Acknowledgement

Chapter 2, in full, is a reprint of the material as it appears in "Monte Carlo Tree Descent for Black-Box Optimization," Y. Zhai, S. Gao. in *Advances in Neural Information Processing Systems*, 2022, 35: 12581-12593. The dissertation author is the primary investigator and author of this paper.

Chapter 3

Monte Carlo Tree Method for Non-convex Optimization

Standard approaches for global optimization of non-convex functions, such as branch-and-bound, maintain partition trees to systematically prune the domain. The tree size grows exponentially in the number of dimensions. We propose new sampling-based methods for non-convex optimization that adapts Monte Carlo Tree Search (MCTS) to improve efficiency. Instead of the standard use of visitation count in Upper Confidence Bounds, we utilize numerical overapproximations of the objective as an uncertainty metric, and also take into account of sampled estimates of first-order and second-order information. The Monte Carlo tree in our approach avoids the usual fixed combinatorial patterns in growing the tree, and aggressively zooms into the promising regions, while still balancing exploration and exploitation. We evaluate the proposed algorithms on high-dimensional non-convex optimization benchmarks against competitive baselines and analyze the effects of the hyper parameters.

3.1 Introduction

Non-convex global optimization problems are pervasive in engineering [39, 131], computer science [89, 119], and economics [12]. The problem is well-known to be NP-hard, and the practical challenge lies in distinguishing the global optimum from exponentially many potential local optima [89, 234].

Existing approaches to non-convex optimization can be largely categorized into sampling-based methods and tree-search methods. Sampling-based approaches, such as simulated annealing (SA) [79] and cross-entropy (CE) [51], explore the solution space through random sampling and guided search strategies with the minimal assumptions about the objective function. Sampling methods can be designed to asymptotically converge towards the global optimum, but suffer from the curse-of-dimensionality in practice. Tree search and interval-based optimization methods [71, 142] leverage various branch-and-bound techniques that maintain a partition tree over the domain to systematically prune the space towards global optima. Such algorithms typically employ rigorous techniques (e.g., linear relaxation [236] and interval arithmetic [6, 80]) for bounding the functions and systematically explore the solution space. The size of the search tree can quickly become exponential in the number of dimensions and is the major bottleneck for scaling up.

We propose an approach that combines the benefits of sampling-based and tree-based approaches as well as interval bounding and local optimization techniques, by taking advantage of the recent progress in Monte Carlo Tree Search (MCTS) methods. We assume that the analytic form of the objective function is known over a compact domain, so that we can use interval bounding [6] on the function value and its local first-order and second-order information in different parts of the MCTS design. A key feature of the Monte Carlo trees is that the growth of the tree is driven by samples rather than partitions, and hence the name *Sample-and-Bound*. By associating the analytic and estimated properties of adjustable neighborhoods around each sample, we design the MCTS algorithm to best balance exploration and exploitation based on the important numerical properties of the objective function. We evaluate the proposed algorithms against a wide range of existing algorithms and analyze the importance of various hyper parameters.

3.2 Related Work

Some classical approaches to global optimization explore the search space by sampling without explicitly building models of the objective function. Common techniques in this category include stochastic methods such as SA [79] and CE [51], as well as deterministic schemes like Nelder-Mead (NM) [60]. SA [79] uses a probability-driven search process to escape local minima. CE [51], on the other hand, is a technique that iteratively updates the probability distribution on the search space to look for optimal regions. NM method [60] is a deterministic sampling approach, which maintains a simplex within the search space and updates its vertexes based on evaluations at selected points.

Sampling-based approaches have recently been combined with tree search by building a search tree for the state space and pick only the most promising subspace to sample. Existing algorithms include Deterministic Optimistic Optimization (DOO) [138], Latent Action Monte Carlo Tree Search (LaMCTS) [223], and Monte Carlo Tree Descent (MCTD) [240]. DOO segments the search domain into sections, each represented by a point; and the new sample is carefully selected by choosing the most suitable section. LaMCTS method employs MCTS to manage search space partitioning. It learns latent actions to distinguish good and bad regions in the search space, and samples in the good partitions during its tree's expansion. MCTD assumes the objective function as black-box, utilizes a combination of sampling based approach and learning based approach for local optimization, and employes MCTS to select the best local optimization processes. Although these methods have adeptly laid out a comprehensive framework for navigating the search space, the task of identifying the most promising subspace from the sample data remains a challenge.

Another category of global optimization methods require the access to the formulation and rely on precise anticipation of objective function values within predefined regions. They employ the branch-and-bound algorithms in which they constitutes a robust framework that systematically partitions the solution space into more accessible sub-regions referred to as branches. The evaluation of each branch is made according to its potential to outperform the current optimal solution based on the bounding of objective function intervals specific to that branch. As the algorithm advances, it tactically prunes branches that can be definitively identified as incapable of providing a superior solution. The typical solvers for this category are BARON [176, 206] and Gurobi [71]. BARON [176, 206] is explicitly tailored to address non-convex global optimization problems by strategically exploring the solution space. Its purpose is to either uncover globally optimal solutions or provide verified lower bounds for the optimal objective value. It achieves this through accurate bounding of non-linear terms with several exceptions such as trigonometric functions and min/max functions. Gurobi [71] is a widely used commercial optimization solver famous for its proficiency in handling quadratic programming problems and various optimization scenarios.

3.3 Preliminary

Problem Formulation.

We consider the problem of minimizing an objective function $f(x) : \Omega \to \mathbb{R}$, where the domain $\Omega \subseteq \mathbb{R}^n$ is compact. In our approach, we assume that we have access to the analytical form of the function $f(x)$, enabling us to query its first-order derivative vector $G(x) = f'(x)$, the second-order partial derivative Hessian matrix $H(x) = f''(x)$, and evaluate the function value interval $f(B)$ over a specified input box $B \subseteq \Omega$.

Interval Arithmetic.

Interval computation is a mathematical and computational approach that operates on quantities and variables represented as intervals [3]. In this context, for a function f defined on an input box domain B, the value of $f(B)$ is expressed as an interval $[lb, ub]$, where for every x within B, the function value satisfies the inequality $lb \leq f(x) \leq ub$.

Monte Carlo Tree Search.

MCTS effectively balances exploration and exploitation based on the theory of multi-armed bandits. The MCTS framework consists of four main steps: Selection, Expansion, Simulation, and Backpropagation. During the Selection step, the search tree is traversed from the root node to a leaf node. The Upper Confidence Bound for Trees (UCT) value, defined as Eq. 3.1, is used to select the best child of a parent node:

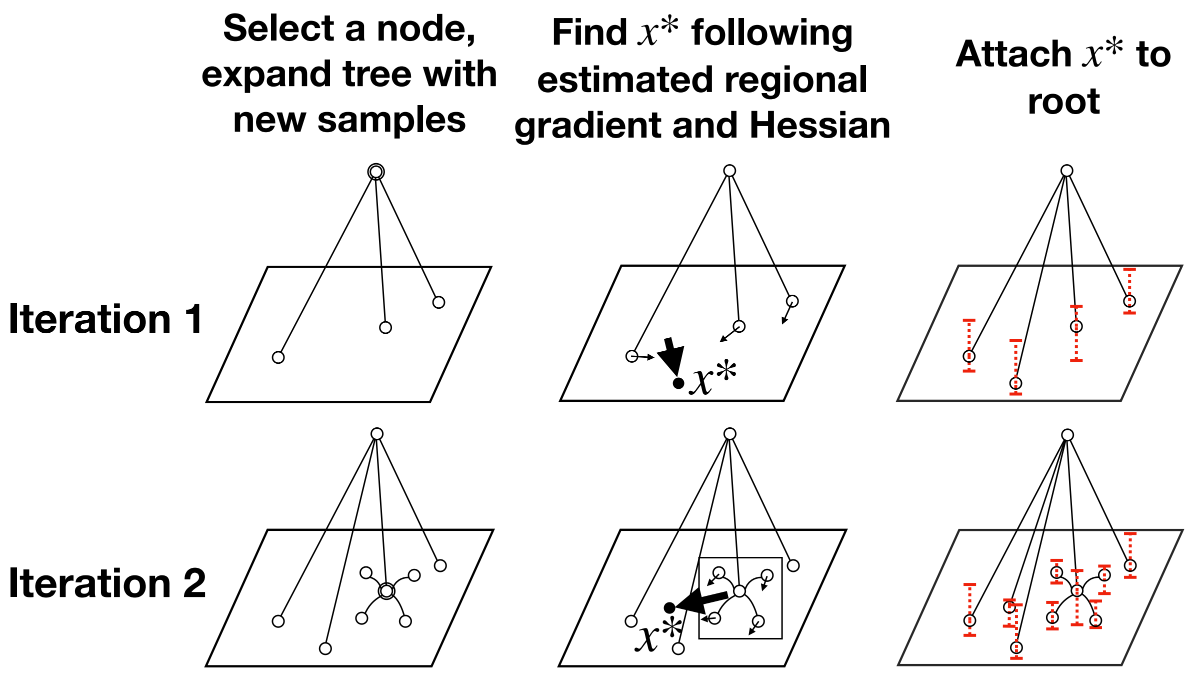
$$v(n_i) = \frac{R_i}{N_i} + C \cdot \sqrt{\frac{2 \cdot \ln(N_p)}{N_i}} \tag{3.1}$$

Here, R_i represents the rewards obtained on child node n_i, N_i is the number of visits to n_i, N_p is the number of visits to n_i's parent node n_p, and C is a constant that balances exploration and exploitation. From the root node, the algorithm recursively select the child node with the highest UCT value, until a leaf node is reached. During the Expansion step, a new child node is added to the selected leaf node. In the Simulation step, a random simulation is performed from the newly added child node until a terminal node is reached, and the simulation reward is estimated. Finally, in the Backpropagation step, the simulation reward is propagated backward from the expanded node to the root node, whose statistics are updated accordingly.

3.4 Monte Carlo Tree Search with Interval Bounds and Regional Estimation

Overview.

The pseudocode of MCIR is provided in Alg. 5, and a high-level visualization is depicted in Fig. 3.1. Our MCIR algorithm employs a search tree structure constructed based on collected samples of the objective function and follows a systematic searching approach in each iteration. Each node in the tree contains a batch of samples encompassed within a box domain associated with that node. In every iteration, we select a leaf node n_p using a modified UCT formula with function evaluation on the box, and we expand the selected node by adding new child nodes n_{ci}



Select a node, expand tree with new samples
Find x^* following estimated regional gradient and Hessian
Attach x^* to root
Iteration 1
Iteration 2
x^*
x^*

Figure 3.1. Steps in each iteration of the MCIR algorithm

generated from random sampling (Fig. 3.1 (a)). We also identify another new child node n^* by leveraging the regional estimation based on gradient and Hessian within the neighborhood of the selected node n_p (Fig. 3.1 (b)). The node n^* represents a node superior to the selected n_p, and we attach it with the root node. This attachment allows us to prioritize the search on this node, thereby accelerating the search process (Fig. 3.1 (c)). For each newly created node we run local optimization with limited steps to enhance the quality of the best-found sample on it.

Note that despite the special design of different parts of MCTS for the optimization context, the proposed algorithm ensures non-zero probability of sampling any neighborhood with positive measure in the input space. Consequently, MCIR is complete in the sense that it will eventually find an ε-neighborhood around the optimal value of any continuous function with arbitrarily small positive ε. As a result, MCIR will ultimately return a value that is within an ε range of the global optimal value of the objective for arbitrarily small positive ε.

Nomenclature

In this paragraph we summarize the symbols and terms used in MCIR (Alg.5).

Algorithm 5. Monte Carlo Tree Search with Interval Bounds and Regional Estimation (MCIR)

```
function MCIR(objective: $f$, domain: $\Omega$)
    $n_0 \leftarrow None$                ▷ root without sample
    $B_0 \leftarrow \Omega$, $lb_0 \leftarrow f(\Omega), V_0 \leftarrow V(B_0)$
    for step = $1, \ldots, t$ do
        $n_p \leftarrow$ Select($n_0$)
        EXPAND($n_p$)
        LEARN($n_p$)
        BACKUP($n_p$)
    end for
    return $y_0^*$
end function

function SELECT(node: $n$)
    $n_p \leftarrow n$
    while $n_p$ has children do
        for $n_{ci} \in n_p$.children do
            $u(n_{ci}) \leftarrow u(y_{ci}^*, lb_{ci}, V_{ci})$ by Eq. 3.2
        end for
        $j \leftarrow \text{argmax}_i u(n_{bi})$
        $n_p \leftarrow n_{cj}$
    end while
    return $n_p$
end function

function BACKUP(node: $n_p$)
    if $n_p$ has children then
        $\{n_{ci}\} \leftarrow n_p$.children
        $y_p^* = \min_i y_{ci}^*$
        $j = \text{argmin}_i lb_{ci}$
        $lb_p = lb_{cj}$
        $V_p = V_{cj}$
    end if
    BACKUP($n_p$.parent)
end function
```

```
function EXPAND(node: $n_p$)
    $B = B_p$
    while $B \neq \emptyset$ do
        $x_{ci} \leftarrow x \in B$
        $x_{ci} \leftarrow$ LocalOpt $(x_{ci}, B_{ci})$
        $n_{ci} \leftarrow x_{ci} \in B$
        $B_{ci} \leftarrow b \in B, x_{ci} \in b$
        $lb_{ci} \leftarrow f(B_{ci})$
        $V_{ci} \leftarrow V(B_{ci})$
        $B \leftarrow B \backslash B_{ci}$
        $n_p$.children.append($n_{ci}$)
    end while
end function

function LEARN(node: $n_p$)
    $n_{ci} \leftarrow \overline{n_p\text{.children}}$
    $\overline{H} \leftarrow \overline{Hessian(x_{ci})}, i = 1, \ldots$
    $j \leftarrow \text{argmin}_i(y_{bi})$
    $\overline{\{x'\} \leftarrow x}$, for $|x - x_{cj}| < \delta$
    $\overline{G} = \overline{grad(x')}$
    for $d = 1, \ldots, dims$ do
        if $\overline{H}_{dd} > 0$ then
            $x_d^* = x_{cj,d} - \overline{G_d}/\overline{H_{dd}}$
        else
            $x_d^* = x_{cj,d} - \overline{G_d}$
        end if
    end for
    $n^* \leftarrow x^*$
    $B^* \leftarrow B_{cj}$, centered at $x^*$
    $lb^* \leftarrow f(B^*), V^* \leftarrow V(B^*)$
    $x^* \leftarrow$ LocalOpt $(x^*, B^*)$
    $n_0$.children.append($n^*$)
end function
```

- Node n: we use character n to represent a node; subscripts are used to distinguish different node with its id: e.g., n_p is a parent node, and $\{n_{ci}\}$ are a set of child nodes, for $i = 1, \ldots$

- Best found sample (x, y^*): a node n is represented by its best-found sample (x, y^*), which is the sample with lowest objective function value found on it and its subtree. On node

n_p, we label the best found sample input vector as x_p, and corresponding function y^*_p. Similarly, for nodes $\{n_{ci}\}$, $i = 1, ...,$ their best found sample input vectors and function values are $\{x_{ci}\}$ and $\{y^*_{ci}\}$, respectively.

- Box B: for a node n, we assign a box $B \subseteq \Omega$ as the bounds in the input domain. We use B_p to indicate the box assigned to the node n_p.
- Function value lower bound lb: The function value interval on an input box B is computed as $f(B) = [lb(f(B)), ub(f(B))]$. We use lb_p to indicate $lb(f(B_p))$.
- V_p is used to denote the volume of the box B_p.
- N is the number of visits to the node
- C_{lb} is the weight factor that controls the importance of the function's lower bound in Equ. 3.2
- C_v is the weight factor associated with the volume of the box where the lower bound is identified.
- C_x is the hyper-parameter for the extent of visitation-based exploration.

Sub-domain Marking.

Samples are the primary information in each node of the search tree that our algorithm build. Around each sample, we mark up the subdomain around it that is considered at the node. The subdomain, typically a hyperbox, will be the focus of local search and optimization at the node, for determining the value of a node. We use the notation B_i to denote the box subdomain associated with the node n_i. In the first iteration, the root node n_{root} encompasses the entire search space, $B_{root} = \Omega$, with the function's lower bound on B_{root} denoted as $lb_{root} = lb(f(B_{root}))$, and its box volume V_{root} represented in logarithmic scale. For subsequent iterations, box B_i is assigned to a node n_i, while lb_i and V_i will be updated according to formulas to be described below. To compute the lower bound of the objective function within a specified input box domain efficiently using global interval bounding [142].

Path Selection.

The key to our design is a modified UCT formula that considers both exploration and exploitation. The pseudocode of this procedure can be found in the **SELECT** function in Alg. 5. For each child node n_{ci} with $i = 1, ...,$ and its parent node n_p, the UCT value $u(n_{ci})$ is determined by the following equation:

$$u(n_{ci}) = -y^*_{ci} - C_{lb} \cdot lb_{ci} - C_v \cdot V_{ci} + C_x \cdot \sqrt{\frac{\log N_p}{N_{ci}}} \tag{3.2}$$

In this formula, C_{lb}, C_v and C_x are weights for the function's lower bound, the volume of the box, and visitation-based exploration, respectively. The variables N_p and N_{ci} denote the number of visits to the parent node n_p and the child node n_{ci}. y^*_{ci} indicates the current best function value discovered on the node n_{ci}, and lb_{ci} corresponds to the lower bound of the function's interval value on the node n_{ci}. The term V_{ci} is the volume (in logarithmic scale) of the box where the lower bound is identified. It is worth noting that after the creation of new child nodes, the function lower bound lb_p and the box volume V_p on the parent node n_p can be updated, as detailed in the subsequent section.

This formulation takes into account the following factors to balance exploration and exploitation: (1) the best function value observed within the box domain, (2) the lower bound of the function value within the domain from interval computation, which reflects the potential best function value upon further exploitation, (3) the volume of the box where the lower bound is determined, related to the reliability of the function lower bound prediction, and (4) the frequency of node visitation. While we considered other ingredients - such as upper function value bound, or values from leveraging the function's analytical form - to put into the formula, the design in Eq. 3.2 turns out to be the most effective.

Utilizing Eq. 3.2, our algorithm tends to redirect its attention to probe alternative sub-domains when a local optimum is identified. When a box is tight enough, the variance of the objective function in the box is low, so the identified local optimum within the box lb_{ci} is relatively accurate.

If this lb_{ci} is close to the minimum of all other $lb_{ci'}$, it indicates a near-optimal solution has been identified. Conversely, if an $lb_{ci'}$ exists that is substantially lower than the current lb_{ci}, the search scheme leans towards selecting the node with the lower $lb_{ci'}$ value in the subsequent iteration due to the path selection criterion Eq. 3.2. In summary, Eq. 3.2 within our algorithm helps strike a balance between exploiting the neighbor region around the current best-found point and exploring other domains that might contain lower function values.

Tree Expansion.

In our algorithm, we utilize two steps to expand the tree effectively. The first step involves sampling within the box of the parent node and generating new child nodes based on these chosen samples. The second step emphasizes learning a high-quality sample point by leveraging both global Hessian and local gradients.

After selecting the leaf node, we proceed to exploit the function space by sampling and creating a cluster of child nodes within the corresponding box (**EXPAND** in Alg. 5). To ensure comprehensive coverage, we divide the box B_p into distinct subsets B_{ci} for each child node n_{ci}, satisfying $\cup\{B_{ci}\} = B_p$ and $B_{ci} \cap B_{cj} = \emptyset, i \neq j$. Additionally, local optimization may be applied to each individual child node n_{ci} to improve sample quality. When a child node n_{ci} is created, we ascertain its function lower bound lb_{ci} through interval propagation of the corresponding box B_{ci}: $lb_{ci} = lb(f(B_{ci}))$. Once the cluster of child nodes is created and their boxes fully cover the parent box B_p, we update the lower bound on the parent node $lb_p = \min(lb_{ci}) = lb_{cj}$ and the volume of the associated box $V_p = V_{cj}$, where $i = 1,..,j,...$. This update will be propagated to the root node.

Next, we learn a representative node n^* using the current set of samples n_{ci} from the selected node n_p, as outlined in Alg. 5 **LEARN**. This step is performed by computing the diagonal of the Hessian matrix, $diag(H)$, for each child node n_{ci}, and estimating the expected value. Furthermore, we collect the gradient information G around the best sample of $n_{ci}, i = 1,...$

and perform a step of Newton's method (or gradient decent when Newton's method is not conductive to minimization), starting from the best sample. The average Hessian, derived from the broad region within the box, represents the overall curvature characteristics of the box. By integrating locally-averaged gradient information, we can identify a sub-region within the box that is more likely to encompass a minimum. The learned representative node, denoted as n^*, is attached to the root node. Note that this attachment means the root node can have children nodes n_{ci} and n_{cj} where $B_{ci} \cap B_{cj} \neq \emptyset$. This step grants n^* higher priority in subsequent iterations. Such prioritization promises to guide the search toward a favorable region, thereby reducing unnecessary tree expansion and preserving tree manageability. Considering that this step may expand the tree's first level of children in every iteration, an extra step may be taken to evaluate the quality of the newly learned node and prune unnecessary ones.

Local Optimization.

Upon creating a child node, we have the option to conduct local optimization steps to improve the quality of the samples on the node. While this step is not obligatory, it offers a beneficial opportunity to refine the samples on each node. To ensure efficient execution, the number of optimization steps is typically kept at a low value, preventing over-exploitation of the immediate local neighborhood. Local optimization can utilize a variety of numerical optimization algorithms. Since the representative node has already been learned using second-order information, we make quasi-Newton methods such as L-BFGS-B [38, 246] the default choice for local optimization. To ensure computational efficiency, we limit the number of function evaluations during the local optimization. In most cases we cap the number of iterations at fewer than 50, as we do not want to overemphasize the choice of the local optimizer. It is worth mentioning that alternative local optimization algorithms can be employed based on specific requirements and preferences.

Backward Propagation.

After creating and locally optimizing children nodes n_{ci} and n^*, we back propagate three important values upwards as in Alg. 5 **BACKUP**, to enhance efficient exploration and decision-making in the subsequent steps.

Firstly, we update the best function value y^*_{ci} found on the child node n_{ci}, to the parent node n_p with y^*_p. This ensures that the parent node retains the most optimal function value discovered within its subtree. Secondly, we update the lower bound of the function interval value lb_p on the parent node n_p with $lb_p = \min(lb_{ci})$. Given that the newly created child nodes comprehensively cover the box of the parent node, this update provides more precise information guiding the search towards the global minimum. Lastly, we propagate the size of the box V_{ci} from which the lower bound of the function value originates: $V_p = V_{cj}$, where $j = \text{argmin}_i lb_{ci}$. This box size represents the uncertainty in the input search space concerning the approximated function interval value. The same propagation is applied to the node n^*, even though its parent is the root node.

3.5 Experiments

3.5.1 Benchmarks

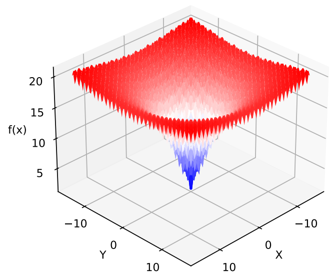
To evaluate the performance of our algorithms, our benchmark sets include three distinct categories: synthetic functions designed for nonlinear optimization, bound-constrained non-convex global optimization problems derived from real-world scenarios, and neural networks fitted for single valued functions. It is important to note that our approach relies on having access to the symbolic expression of the objective function and do not consider other relational constraints to the variables (e.g., "<="). As a result, benchmark sets that are commonly used for black-box optimization problems and constraint optimization problems are not applicable in our case.

Synthetic functions are widely-used in nonlinear optimization benchmarks [111]. These functions usually have numerous local minima, valleys, and ridges in their landscapes which is hard for normal optimization algorithms. In our tests, we choose three functions: Levy, Ackley, and Michalewicz, and examine our algorithm's performances on the functions in 50d, 100d, and 200d. For our evaluation of non-convex global optimization problems in various fields, we select bound-constrained problems from the collection presented in [160, 207] that do not involve any additional inequality or equality constraints. To strike a balance between computational resources and the complexity of the function landscapes, we specifically select functions with input dimensions between 30 and 1000, and ensure that the functions could be evaluated within a reasonable time, considering the computational cost of computing the gradient and Hessian. The chosen functions for our evaluation include Biggsbi1 (1000d), Harkerp (100d), and Watson (31d). It is worth noting that this set of test functions is also utilized in the development of BARON [206] and continues to be used in the latest version [176]. In addition to the aforementioned problems, we also explore the application of one-layer neural networks with ReLU activation functions fitted for specific objective functions. The nonlinearity introduced by activation functions and the partitioning of the input space pose challenges in finding the global minimum of neural networks. To assess the performance of our algorithm, we train neural networks with varying numbers of input dimensions and one layer of 16 hidden unite. We translate the entire network into an analytic expression form, enabling us to evaluate the algorithm's effectiveness in optimizing neural network models. We conduct our experiments on a local machine with Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz, 16G RAM, and NVIDIA GeForce GTX 1080 graphic card.

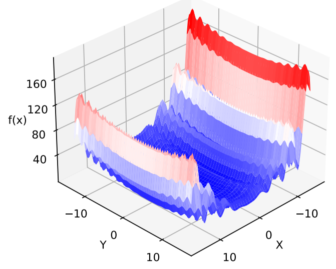
Test sets.

The test sets comprise functions from three different categorises: synthetic functions designed for nonlinear optimization, bounded-constrained non-convex global optimization problems derived from real-world scenarios, and neural networks fitting for single valued functions.

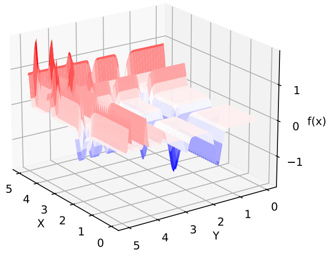
Synthetic functions [111] are widely-used in nonlinear optimization benchmarks . We choose three functions: **Levy**, **Ackley**, and **Michalewicz**, and examine our algorithms per-



(a) Ackley-2d



(b) Levy-2d



(c) Michalewicz-2d

Figure 3.2. Landscape of test functions: Ackley, Levy, and Michalewicz

formances on the functions in 50d, 100d, and 200d. The Levy function (Fig.3.2b) is known for its complex, multi-modal landscape, featuring numerous local optima separated by vast valleys. The Ackley function (Fig.3.2a) presents a rugged landscape with a prominent, flat region surrounding the global minimum. The Michalewicz function (Fig.3.2c) is characterized by its highly oscillatory and irregular landscape, characterized by numerous peaks and valleys.

We use three test functions from bounded-constrained non-convex problem sets for global optimization [160]: **Biggsbi1** (1000d), **Harkerp** (100d), and **Watson** (31d). Biggsbil1 is a function modeling the kinetics of a biochemical reaction. It has a global minimum that's surrounded by several local minima making it challenging for optimization algorithms. Harkerp is a mathematical problem in the field of economics, specifically in the area of profit maximization. This non-convex problem simulates a firm seeking to maximize profit through optimal pricing and advertising decisions, subject to market demand and cost constraints. Watson is a smooth, non-convex function known for its narrow, curved valley that leads to the global minimum.

Finally, we ventured into the realm of neural network optimization. Our methodology enabled us to transform a network with ReLU activations function into an analytic expression. This, however, introduces challenges due to the nonlinearity arising from ReLU activations and the partitioning of the input space. The global minimum of neural networks becomes elusive, especially due to the unpredictable error. Moreover, the node learning step in our algorithm can no longer use Hessian information because the Hessian of ReLU-activated neural network is zero. To balance computational cost and evaluation complexity, we trained the neural network to fit the Ackley and Michalewicz functions in 50d and 100d spaces with 16 hidden neurons in a single layer.

Baselines.

We select various sampling-based global optimization algorithms as baselines for our experiments, including: basinhopping [147], differential evolution [152, 202], dual annealing [233], direct [58], CMA [77], TuRBO [55], LaMCTS [223], and Gurobi [71]. It should be pointed out that some algorithms, including TuRBO and LaMCTS, are GPU-ready. However, due to the limitations of our computational resources, we refrain from using GPUs for the optimization process, except for tasks related to training and evaluating the neural network model. To compensate for the reduced performance from utilizing the CPU, we extend the timeout for TuRBO and LaMCTS to be five times of other baselines.

It is important to mention that we do not incorporate BARON [176] as one of our baseline methods, despite its renowned ability to efficiently bound boxes. The reason behind this decision lies in the fact that BARON can manage the functions present in their test sets during pre-processing, entirely eliminating the need to execute the optimization algorithm. For instances like Biggsbi1, Harkerp, and Watson, BARON can solve them instantly, requiring zero seconds and iterations. Moreover, BARON encounters challenges with certain function types, including but not limited to trigonometric functions and min/max functions [176]. These types of functions are prevalent in synthetic test function sets as well as function sets based on neural networks.

Another consideration is that Gurobi requires expertise and extra effort to achieve peak performance. While Gurobi stands out as an exceptionally efficient and versatile optimization solver, especially in the context of non-convex optimization problems, it comes with certain prerequisites. Its handling of non-linear terms, for instance, treats them as General Constraints,

Table 3.1. Hyperparameters used for MCIR

Functions	$\mathbf{C}_{lb}$	$\mathbf{C}_v$	$\mathbf{C}_x$	# CN	# LO
Ackley-50d	50.0	0.5	0.5	10	50
Ackley-100d	50.0	0.5	0.5	20	50
Ackley-200d	50.0	0.5	0.5	30	50
Levy-50d	50.0	0.5	0.5	10	50
Levy-100d	50.0	0.5	0.5	20	50
Levy-200d	50.0	0.5	0.5	30	50
Micha.-50d	50.0	0.5	0.5	10	20
Micha.-100d	50.0	0.5	0.5	20	50
Micha.-100d	50.0	0.5	0.5	30	50
Biggsbi1-1000d	50.0	0.5	0.5	10	10
Harkerp-100d	50.0	0.5	0.5	10	10
Harkerp-31d	50.0	0.5	0.5	10	10
NN-Ackley-50d	50.0	0.5	0.5	10	20
NN-Ackley-100d	50.0	0.5	0.5	20	50
NN-Micha.-50d	50.0	0.5	0.5	10	20
NN-Micha.-100d	50.0	0.5	0.5	20	50

which necessitates extra manual modification to the objective function expression, as outlined in [71]. This specific trait might limit our ability to deploy it on entire test sets.

Hyperparameters.

Tab.3.1 presents the hyperparameters utilized for benchmarking MCIR across various functions. In this context, the notation **# CN** corresponds to the quantity of appended child nodes during expansion. Notably, a higher count of child nodes contributes to an increased number of samples dedicated to exploiting the selected leaf node. Meanwhile, **# LO** denotes the number of samples permissible for the local optimizer to undertake in enhancing the sample quality on the node. We intentionally constrain this number to a low value, thereby preventing an excessive reliance on the local optimizer.

Hyperparameters for other baseline:

- **Basinhopping**: $niter = 1000$, $niter\_success = 1000$
- **Differential Evolution**: $maxiter = 1000000$

- **Dual Annealing**: *no local search=True*

- **Direct**: $maxiter = 100000$, $locally\_biased = False$

- **CMA**: use $fmin2$, with $\sigma = 1$

- **TuRBO**: $n\_init = 20$, $batch\_size = 40$, $use\_ard = True$, $n\_training\_steps = 50$

- **LaMCTS**: $n\_init = 50$, $C_p = 2.0$, $leaf\_size = 50$

Hyperparameters not specifically addressed above were subjected with their default values within the packages.

Metrics.

For each benchmark function, we conduct experiments using baseline algorithms and our proposed algorithm with five different random seeds. The time limits for the baselines are set to 2 hour. Due to CPU utilization, the limits for TuRBO and LaMCTS are extended to 10 hours. Throughout the experiments, we track the best-found function value until each step and compute the mean and standard deviation across all runs. This allows us to compare the final best-found values as well as the speed at which each algorithm converges to the optimal result.

3.5.2 Benchmark Results

Overall Performance.

Fig. 3.3 presents performance comparisons between the MCIR algorithm and baseline algorithms on the synthetic function benchmark set. For the Ackley function (Fig. 3.3 first row) and Levy function (Fig. 3.3 second row), CMA emerges as the top-performing algorithm, followed by MCIR and dual annealing. For the Michalewicz function (Fig. 3.3 third row), dual annealing and MCIR delivers similar best performance in terms of final optimization result, while CMA fails to optimize efficiently. Notably, Gurobi consistently completes its optimization with just one function evaluation call. It attains a best function value of 2.02 on Ackley (for three dimensions), 0.0 for Levy in both 50d and 100d, and -0.00016 for Levy-200d (an anomalous

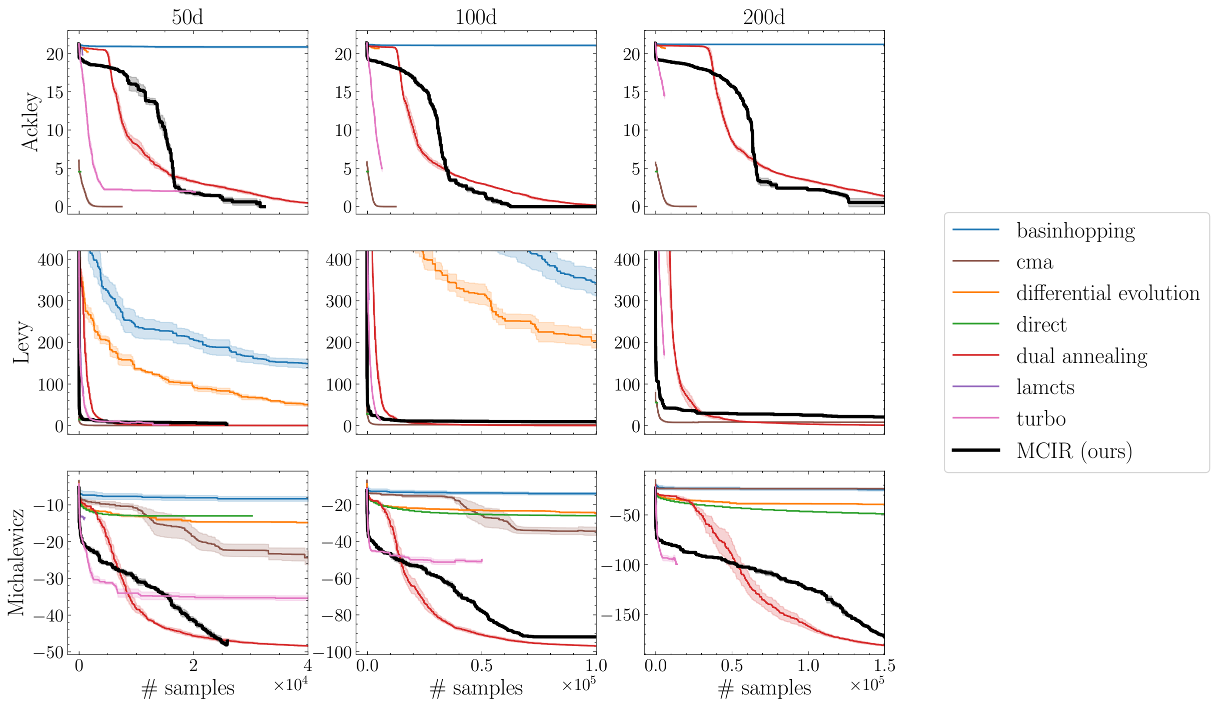


Figure 3.3. Overall performance of the baselines and MCIR on tested synthetic functions

value, because Levy function is greater or equal to 0.0). It also attains the lowest value amongst all algorithms on the Michalewicz function.

Fig. 3.4 offers an in-depth comparison between the MCIR algorithm and the baseline algorithms across benchmarks such as Biggsbi1, Harkerp, Watson, and Neural Networks. In the bound-constrained optimization problems (BCP) of Biggsbi1, Harkerp, and Watson (Fig. 3.4 first row), MCIR exhibits exemplary performance. It adeptly strikes a balance between exploring the search space and executing local optimization, culminating in the precise pinpointing of the global minimum from many suboptimal local minima derived from local optimization. CMA shows a performance comparable to MCIR on Harkerp and Watson, and direct algorithm mirrors MCIR's efficacy on Biggsbi1. Turning our attention to trained neural networks (Fig. 3.4 second row), CMA shines on Ackley-50d, yet MCIR continues to deliver impressive results. Notably, for Michalewicz-50d and Michalewicz-100d, MCIR outperforms all other baseline algorithms.

Upon a closer examination of the result curves, it becomes evident that the MCIR algorithm's optimization performance is both commendable and in line with our initial expectations.

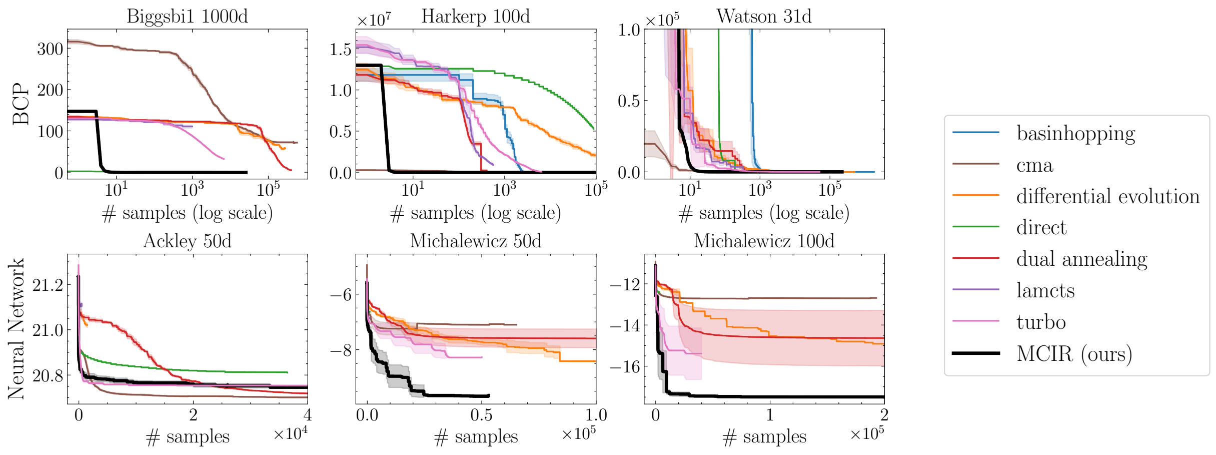


Figure 3.4. Overall performance of the baselines and MCIR on Biggsbi1, Harkerp, Watson, and Neural Networks

Detailed results.

In this section, we use the abbreviations as follows to denote different optimization algorithms:

- **BH**: Basinhopping
- **DE**: Differential Evolution
- **DIRECT**: DIRECT
- **DA**: Dual Annealing
- **CMA**: CMA
- **TuRBO**: TuRBO
- **LaMCTS**: LaMCTS
- **Gurobi**: the Gurobi solver
- **MCIR**: our algorithm, MCIR

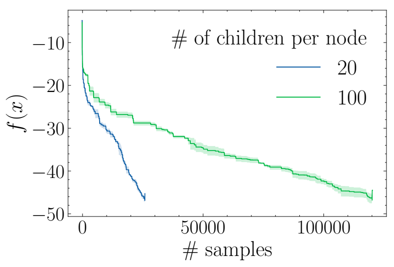
Tab.3.2, Tab.3.3, Tab.3.4 and Tab.3.5 provide detailed benchmark results. The column labeled $y^* \pm std$ represents the best-found objective function value. The column labeled **Time*** denotes the clock time, in seconds, at which the algorithm found the best point for the first time. The column labeled **# of Samples*** indicates the number of samples which have been evaluated when the algorithm reached the best-found point for the first time.

In the case of **Gurobi**, our testing was exclusively conducted on functions that could be precisely and accurately formulated. Consequently, its presence might be limited. Furthermore, in the cases where we carefully formulated the problems, Gurobi can solve the problem at pre-processing stage resulting in a zero seconds computational time and zero times of function evaluation calls.

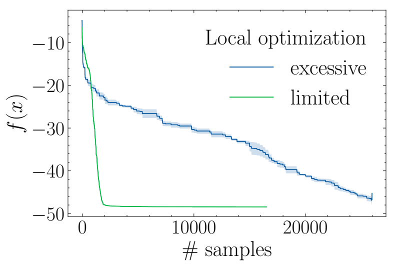
For **TuRBO** and **LaMCTS**, the number of function evaluations are still quite limited, even after we have increased the timeout as five times longer than other baselines. Particularly for **LaMCTS**, there are only hundreds of function evaluations after 5 hours of computational time. Therefore, it is hard to determine when they first encountered their best-found sample.

Ablation Study.

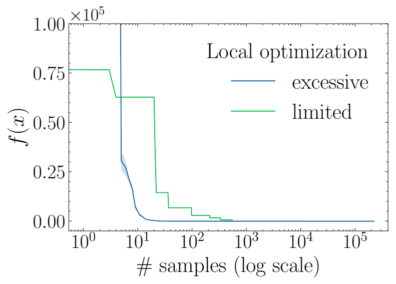
We conducted ablation studies to analyze the individual contributions of different components in our algorithm. Specifically, we investigated the influence of the number of random samples placed under each selected node on the Michalewicz-50d function (Fig. 3.5a), assessed the effectiveness of local optimization on Michalewicz-50d (Fig. 3.5b), and examined the effectiveness of local optimization on the Watson-31d (Fig. 3.5c). Furthermore, we tested the hyper parameters used in the UCT formula (Eq. 3.2) using the Ackley-50d function as depicted in Fig. 3.5d, 3.5e, and 3.5f. From Fig. 3.5a, we observed that the number of new nodes placed after selecting a leaf node should be kept at a moderate level. Overpopulating the same local region with new nodes does not significantly enhance performance due to the closeness of local optima. The significance of local optimization can be found in Fig. 3.5b and Fig. 3.5c. It can be concluded that local optimization can both improve and hinder the performance of the algorithm,



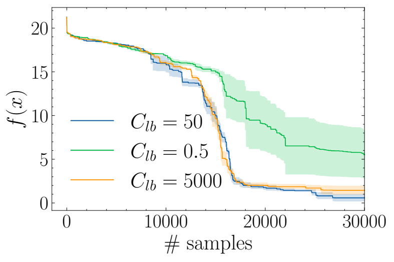
(a) Michalewicz-50d
number of nodes at expansion



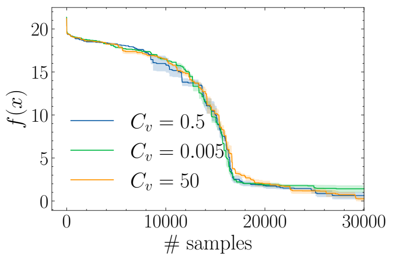
(b) Michalewicz-50d
number of local optimization steps



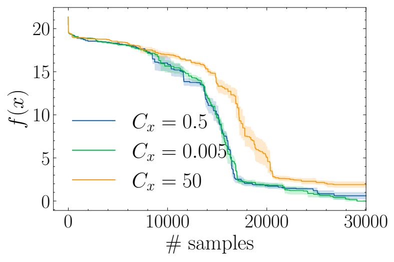
(c) Watson-31d
number of local optimization steps



(d) Ackley-50d, C_{lb}



(e) Ackley-50d, C_v



(f) Ackley-50d, C_x

Figure 3.5. Ablation studies for MCIR. Results are shown on function Michalewicz-50d for the number of children at expansion (a), the effectiveness of local optimization (b), on function Watson-31d for the effectiveness of local optimizaiton (c), and on function Ackley-50d for C_{lb} (d), C_v (e), and C_x (f).

as shown Fig. 3.5c and Fig. 3.5b, respectively. Therefore, setting a reasonable budget for the local optimizer is important. A local optimizer with stopping criterion such as improvement threshold could be more preferable over one with fixed number of iterations. Fig. 3.5d, Fig. 3.5e, and Fig. 3.5f demonstrate the importance of the lower bound of the function value in determining the best node for searching the global minimum. Additionally, the balance between exploiting nodes excessively and leaving nodes unexplored becomes evident. The size of the box where the lower bound originates is the least sensitive hyper parameter, as a smaller box increases the certainty of the function's lower bound but has less impact compared to the value of the function's lower bound itself.

We have noticed that sometimes the effectiveness of C_v is limited, but as C_v represents the confidence of the predicted objective function interval, the choice of C_v highly depends on the landscape of the objective function. For functions where the bounds are evaluated through rough approximation, C_v becomes important. In our benchmark tests, we select our hyper parameters C_{lb}, C_v, and C_x that perform consistently well over multiple tests without requiring too much fine-tuning. Most of the parameters are shared across all problems based on the dimension and the type of problems. In practice, one can start with the setup that provides the best performance in this paper, and fine tune to specific tasks based on observed function complexity and landscape.

3.6 Conclusion

We introduced a new approach to non-convex optimizations problems by leveraging analytic and sampling-based information in an MCTS framework, enabling efficient exploration and exploitation of the state space. Experiments results on standard benchmark problem sets demonstrated clear benefits of the proposed approach. Future work can focus on reducing the overhead of various numerical computation involved in the proposed algorithm and further optimizing the search tree.

3.7 Acknowledgement

Chapter 3, in full, is a reprint of the material as it appears in “Sample-and-Bound for Non-Convex Optimization,” Y. Zhai, Z. Qin, S. Gao. in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. The dissertation author is the primary investigator and author of this paper.

Table 3.2. Benchmark results for Ackley (50d, 100d and 200d) and Levy (50d and 100d)

Function	**Alg.**	$y^* \pm std$	**Time***	**# Samples** *
Ackley-50d	BH	20.8358 ± 0.1967	1.47 ± 0.62	12709 ± 5240
	DE	20.1856 ± 0.2135	0.45 ± 0.16	1171 ± 170
	DIRECT	4.5525 ± 0.0000	0.00	0
	DA	0.0011 ± 0.0001	110.75 ± 15.27	123591 ± 1364
	CMA	0.0000 ± 0.0000	5.68 ± 0.29	6792 ± 375
	TuRBO	2.0531 ± 0.0956		8022 ± 3248
	LaMCTS	20.5670 ± 0.3844		170 ± 220
	Gurobi	2.0217 ± 0.0000	0.00	0
	MCIR	0.6245 ± 0.7649	118.12 ± 15.51	23529 ± 3389
Ackley-100d	BH	20.9803 ± 0.1058	1.61 ± 1.04	14342 ± 9265
	DE	20.6655 ± 0.0568	0.82 ± 0.19	2328 ± 689
	DIRECT	4.5525 ± 0.0000	0.00 ± 0.00	0
	DA	0.0021 ± 0.0002	287.41 ± 19.83	265813 ± 4155
	CMA	0.0000 ± 0.0000	10.09 ± 0.20	12226 ± 244
	TuRBO	5.1832 ± 0.5763		6344 ± 105
	LaMCTS	20.8962 ± 0.0350		249 ± 136
	Gurobi	2.0217 ± 0.0000	0.00	0
	MCIR	0.0000 ± 0.0000	1033.78 ± 55.01	61351 ± 1469
Ackley-200d	BH	21.1346 ± 0.0700	2.29 ± 2.24	20183 ± 19168
	DE	20.6334 ± 0.1212	1.76 ± 0.25	4790 ± 976
	DIRECT	4.5525 ± 0.0000	0.00	0
	DA	0.0041 ± 0.0001	766.03 ± 51.73	546779 ± 7893
	CMA	0.0000 ± 0.0000	17.91 ± 1.08	23012 ± 1678
	TuRBO	14.2829 ± 0.4982		5695 ± 131
	LaMCTS	21.0537 ± 0.0283		223 ± 106
	Gurobi	2.0217 ± 0.0000	0.00	0
	MCIR	0.5135 ± 1.0270	6544.56 ± 1661.44	114651 ± 23828
Levy-50d	BH	114.9944 ± 18.8022	44.35 ± 4.47	98070 ± 12008
	DE	0.1373 ± 0.1733	288.61 ± 155.92	450636 ± 232094
	DIRECT	13.7759 ± 0.0000	0.20 ± 0.05	199
	DA	0.0000 ± 0.0000	86.62 ± 30.43	59073 ± 24203
	CMA	0.2149 ± 0.1561	10.94 ± 3.96	16597 ± 6516
	TuRBO	5.4087 ± 3.0669		8309 ± 3148
	LaMCTS	84.7584 ± 7.0483		648 ± 87
	MCIR	5.5889 ± 1.9306	168.77 ± 101.57	13469 ± 8078
Levy-100d	BH	248.9395 ± 32.5442	101.83 ± 4.58	206809 ± 16550
	DE	3.1676 ± 0.6679	791.58 ± 372.37	778528 ± 59597
	DIRECT	27.4119 ± 0.0000	0.28 ± 0.06	399
	DA	0.0001 ± 0.0002	211.72 ± 27.53	138905 ± 10619
	CMA	2.0102 ± 0.8746	18.74 ± 13.50	31149 ± 24441
	TuRBO	15.7115 ± 6.1465		5501 ± 177
	LaMCTS	326.9397 ± 38.7379		667 ± 43
	Gurobi	-0.0000 ± 0.0000	0.00	0
	MCIR	9.1970 ± 1.4018	1810.49 ± 1180.24	35611 ± 23348

Table 3.3. Benchmark results for Levy (200d) and Michalewicz (50d, 100d and 200d)

Function	**Alg.**	$y^* \pm std$	**Time***	**# Samples** *
Levy-200d	BH	660.9173 ± 51.2398	187.51 ± 19.31	377479 ± 35511
	DE	23.5931 ± 4.9629	1714.29 ± 598.96	1909534 ± 265067
	DIRECT	54.6839 ± 0.0000	0.52 ± 0.09	799
	DA	0.0005 ± 0.0005	639.01 ± 89.80	409497 ± 58671
	CMA	7.5793 ± 1.5815	28.14 ± 16.21	47511 ± 37708
	TuRBO	169.9204 ± 13.7658		5591 ± 108
	LaMCTS	1533.92 ± 111.02		504 ± 93
	Gurobi	-0.0002 ± 0.0000	0.00	0
	MCIR	20.5379 ± 4.1395	5388.18 ± 348.55	45079 ± 1564
Michalewicz-50d	BH	-8.8432 ± 0.8271	55.45 ± 76.27	209182 ± 185755
	DE	-20.1906 ± 1.0243	212.15 ± 93.79	751515 ± 644
	DIRECT	-13.0827 ± 0.0000	1.98 ± 0.07	25609
	DA	-49.5421 ± 0.0134	88.54 ± 7.28	127056 ± 2375
	CMA	-22.9280 ± 3.5200	26.95 ± 4.91	42784 ± 9507
	TuRBO	-35.3835 ± 1.1419		15915 ± 11657
	LaMCTS	-13.9431 ± 0.6567		463 ± 291
	Gurobi	-49.8521 ± 0.0000	0.00	0
	MCIR	-48.0423 ± 0.7511	177.66 ± 1.97	25418 ± 283
Michalewicz-100d	BH	-14.6945 ± 0.5995	15.15 ± 12.32	89181 ± 75509
	DE	-30.6034 ± 0.7476	702.97 ± 338.89	1503114 ± 1322
	DIRECT	-26.1498 ± 0.0000	35.85 ± 3.22	103813
	DA	-99.4230 ± 0.0207	228.55 ± 28.25	297807 ± 5504
	CMA	-34.9603 ± 3.7703	64.33 ± 10.81	111835 ± 21423
	TuRBO	-51.6695 ± 1.9650		27427 ± 9051
	LaMCTS	-23.8377 ± 1.1338		522 ± 197
	Gurobi	-100.5415 ± 0.0000	0.0	0
	MCIR	-91.9268 ± 1.0598	3518.56 ± 69.82	96317 ± 2286
Michalewicz-200d	BH	-27.3023 ± 1.6926	3394.23 ± 1876.30	1649919 ± 870987
	DE	-48.4502 ± 2.0750	1807.93 ± 365.10	3008602 ± 4315
	DIRECT	-51.1477 ± 0.0000	29.64 ± 6.27	205917
	DA	-199.1417 ± 0.0593	503.03 ± 60.18	766905 ± 11713
	CMA	-43.1354 ± 10.5304	155.91 ± 78.58	310636 ± 156979
	TuRBO	-96.4577 ± 2.6720		8502 ± 3394
	LaMCTS	-39.4668 ± 1.3005		355 ± 171
	Gurobi	-202.4635 ± 0.0000	0.00	0
	MCIR	-172.5783 ± 2.8443	7146.98 ± 167.62	51914 ± 856

Table 3.4. Benchmark results for bounded-constrained functions

Function	**Alg.**	$y^* \pm std$	**Time***	**# Samples** *
Biggsbi1-1000d	DE	56.1697 ± 3.8586	3555.39 ± 466.69	232118 ± 28533
	DIRECT	1.6592 ± 0.0000	4.60 ± 0.65	4001
	DA	7.5562 ± 2.9694	3615.67 ± 14.28	362556 ± 34168
	CMA	72.3690 ± 3.7933	166.55 ± 5.90	448919 ± 10571
	TuRBO	32.4614 ± 1.5262		6649 ± 22
	LaMCTS	110.3565 ± 6.6033		346 ± 129
	MCIR	1.0101 ± 0.0001	1986.40 ± 277.34	17584 ± 2360
Harkerp-100d	DE	-0.7887 ± 0.0738	2214.26 ± 1337.87	1112523 ± 514176
	DIRECT	5266895.8 ± 0.0	525.71 ± 955.15	86592
	DA	-0.6831 ± 0.1248	23.94 ± 30.71	15265 ± 60
	CMA	-0.3220 ± 0.2179	34.24 ± 27.89	61240 ± 50743
	TuRBO	91484.1 ± 18502.6		6251 ± 39
	LaMCTS	980949.0 ± 122103.2		520 ± 24
	MCIR	-0.9256 ± 0.0017	236.05 ± 60.83	13106 ± 3312
Watson-31d	DE	0.0010 ± 0.0006	228.30 ± 39.65	492335 ± 2941
	DIRECT	264.6848 ± 0.0000	0.90 ± 0.04	1849
	DA	0.1415 ± 0.1318	3.58 ± 0.54	7665 ± 102
	CMA	0.0000 ± 0.0000	26.68 ± 3.03	42245 ± 6335
	TuRBO	4.6289 ± 1.8125		16500 ± 14087
	LaMCTS	4109.8 ± 1959.8		97 ± 36
	MCIR	0.0023 ± 0.0018	487.88 ± 288.47	43682 ± 25967

Table 3.5. Benchmark results for NN functions

Function	**Alg.**	$y^* \pm std$	**Time***	**# Samples** *
NN-Ackley-50d	BH	21.1501 ± 0.1014	29.95 ± 22.82	17839 ± 14101
	DE	21.0200 ± 0.0181	2.09 ± 0.94	1056 ± 466
	DIRECT	20.8103 ± 0.0000	56.48 ± 1.82	36401
	DA	20.7037 ± 0.0009	195.30 ± 2.30	108967 ± 1559
	CMA	20.7056 ± 0.0030	17.88 ± 5.20	26648 ± 9448
	TuRBO	20.7524 ± 0.0023		27706 ± 18354
	LaMCTS	21.1075 ± 0.0250		150 ± 156
	MCIR	20.7481 ± 0.0152	598.52 ± 391.36	19031 ± 14205
NN-Ackley-100d	BH	21.2368 ± 0.0307	51.32 ± 26.29	30847 ± 15151
	DE	21.1453 ± 0.0080	3.49 ± 1.30	1723 ± 645
	DIRECT	21.0549 ± 0.0000	116.69 ± 6.78	72091
	DA	20.9845 ± 0.0010	389.19 ± 18.13	218588 ± 3637
	CMA	20.9828 ± 0.0020	53.63 ± 15.49	95970 ± 33576
	TuRBO	21.0412 ± 0.0030		21576 ± 13880
	LaMCTS	21.1735 ± 0.0111		122 ± 128
	MCIR	21.0162 ± 0.0139	2461 ± 427	39661 ± 2727
NN-Michalewicz-50d	DE	-7.8478 ± 0.6904	151.45 ± 108.62	79255 ± 57812
	DIRECT	-6.6457 ± 0.0000	1.48 ± 0.20	771
	DA	-7.6015 ± 0.6065	224.88 ± 19.00	129478 ± 12950
	CMA	-7.2868 ± 0.3838	25.40 ± 8.30	42098 ± 16777
	TuRBO	-8.2827 ± 0.0133		22526 ± 10821
	LaMCTS	-6.8803 ± 0.0271		755 ± 120
	MCIR	-9.6448 ± 0.0384	711.72 ± 463.97	25225 ± 16613
NN-Michalewicz-100d	DE	-12.8481 ± 1.1304	115.02 ± 171.16	57962 ± 86966
	DIRECT	-12.4153 ± 0.0000	3.93 ± 0.36	2391
	DA	-14.6488 ± 2.4041	430.69 ± 29.31	261728 ± 27258
	CMA	-12.7314 ± 0.0352	63.74 ± 24.53	120090 ± 55975
	TuRBO	-15.1034 ± 0.2637		26886 ± 18835
	LaMCTS	-12.2140 ± 0.1088		573 ± 135
	MCIR	-17.5062 ± 0.0068	6061 ± 4329	101634 ± 72926

Chapter 4

Active Learning in Developing Many-Body Potentials

The efficient selection of representative configurations that are used in high-level electronic structure calculations needed for the development of many-body molecular models poses a challenge to current data-driven approaches to molecular simulations. Here, we introduce an active learning (AL) framework for generating training sets corresponding to individual many-body contributions to the energy of a N-body system, which are required for the development of MB-nrg potential energy functions (PEFs). Our AL framework is based on uncertainty and error estimation, and uses Gaussian process regression (GPR) to identify the most relevant configurations that are needed for an accurate representation of the energy landscape of the molecular system under exam. Taking the Cs+–water system as a case study, we demonstrate that the application of our AL framework results in significantly smaller training sets than previously used in the development of the original MB-nrg PEF, without loss of accuracy. Considering the computational cost associated with high-level electronic structure calculations, our AL framework is particularly well-suited to the development of many-body PEFs, with chemical and spectroscopic accuracy, for molecular-level computer simulations from the gas to condensed phase.

4.1 Introduction

Computer simulations provide fundamental insights into the properties and behavior of molecular systems [95, 113, 227]. Since both accuracy and predictive ability of a molecular model are primarily limited by the computational cost associated with the model itself, developing cost-effective simulation approaches is key to studying increasingly more complex systems. It has recently become possible to perform molecular dynamics (MD) simulations of aqueous systems, from the gas to the condensed phase, retaining high accuracy in the description of the underlying molecular interactions [45]. This is achieved by employing many-body potential energy functions (PEFs) derived from high-level electronic structure data that are carried out on selected molecular configurations representative of the corresponding global many-body potential energy surfaces (PESs) [7–9, 11, 32, 126, 172, 225, 226]. An optimal approach to the development of many-body PEFs would require identifying a minimal pool of configurations that can guarantee an accurate description of the system under exam and, at the same time, computation time is not lost on calculations on redundant configurations describing similar regions of the many-body PES.

Efficient sampling of the configuration space is challenging due to the high dimensionality of the associated molecular configurations. In principle, a regular grid search would provide a homogeneous representation of all regions of the many-body PES. This approach, however, becomes unfeasible as the number of degrees of freedom increases. To reduce the size of the configuration space, it is common practice in the development of many-body PEFs to apply biases on the relative translations and rotations of the individual molecular species constituting the system under exam [7, 9, 11, 172]. Although of practical use, this approach can lead to redundant training sets containing several molecular configurations representing similar regions of the target many-body PES. While algorithms designed to remove geometrically similar configurations exist, it is not guaranteed that screening based on structural similarity is sufficient for identifying only configurations necessary for a faithful description of the target many-body PES.

The success of machine learning (ML) in many areas of molecular sciences (e.g., see Refs. [21, 37, 69, 75, 90, 112, 136, 144, 175, 185, 195, 205, 221, 232, 232, 242]) makes it a promising tool for efficiently screening large pools of molecular configurations for the development of many-body PEFs. Most common ML approaches rely on supervised learning, which, however, requires large set of known labeled data to train a model capable to accurately predict the labels of previously unseen data [4, 40, 106]. Active learning (AL) provides a potential solution to the need for constructing beforehand large training sets by interactively generating training configurations at runtime. AL schemes are thus particularly appealing when using large training sets is prohibitively expensive either because of the high cost associated with determining the data labels or because of the high computational cost of the training stage.

In this study, we investigate the application of AL to generating representative training sets of molecular configurations necessary for the development of many-body PEFs, with a specific focus on two-body (2B) and three-body (3B) contributions to the Cs^+–water interaction energies. Our AL framework consists of a finite pool of molecular configurations (i.e., $Cs^+(H_2O)$ dimers for the 2B pool and $Cs^+(H_2O)_2$ trimers for the 3B pool) whose energies are unknown, a training set with configurations selected from the pool, a predictive model (predictor) thirsting for the training set, and a learner that actively selects configurations from the pool. We assume that the size of the pool is beyond awareness of the learner and only a subset of the configurations (referred to as candidates) in the pool are available to the learner at each iteration. Through the application of our AL approach, we demonstrate that the size of the original pool of configurations used to develop the Cs^+–water MB-nrg PEF can be greatly reduced without compromising the accuracy with which the new MB-nrg PEFs describe Cs^+–water interactions, from small clusters to aqueous solutions.

4.2 Methods

4.2.1 MB-nrg potential energy functions

The total energy of a system containing N (atomic or molecular) monomers (“bodies"), can be rigorously expressed through the many-body expansion (MBE) of the energy [74],

$$V_N = \sum_{i}^{N} V_i^{1B} + \sum_{i<j}^{N} V_{ij}^{2B} + \sum_{i<j<k}^{N} V_{ijk}^{3B} + \ldots + V^{NB} \tag{4.1}$$

where the V_i^{1B} corresponds to the energy required to distort monomer i from its equilibrium geometry. Therefore, $V^{1B}(i) = 0$ for atomic monomers, and $V^{1B}(i) = E(i) - E_{eq}(i)$ for molecular monomers, where $E(i)$ and $E_{eq}(i)$ are the energies of monomer i in distorted and equilibrium geometries, respectively. All higher n-body (nB) interaction terms (V^{nB}) in Eq. 4.2.1 are defined recursively through

(4.2)

Within the MB-nrg framework, the water–water interactions are described by the MB-pol PEF [7, 9, 126], which has been shown to correctly reproduce the properties of water[148, 167] from small clusters in the gas phase [24, 29, 46, 48, 121, 169, 178, 179, 181, 212, 213, 217, 218], to bulk water [84, 128, 166, 204], the air/water interface [129, 135, 186, 203], and ice [133, 134, 156]. The interactions between Cs^+ ions and water molecules are described through the MBE of Eq. 4.2.1. Specifically, the Cs^+–water MB-nrg PEF includes explicit 2B Cs^+–H_2O and 3B Cs^+–$(H_2O)_2$ terms, with all higher-order interactions being implicitly taken into account through a classical many-body polarization term [172, 248]. The 2B term includes three contributions,

$$V^{2B} = V^{2B}_{short} + V^{2B}_{TTM} + V^{2B}_{disp} \tag{4.3}$$

where V^{2B}_{disp} is the 2B dispersion energy, and V^{2B}_{TTM} is the 2B classical polarization contribution described by a Thole-type model [35] . V^{2B}_{short} in Eq. 4.3 describes 2B short-range contributions represented by a 5th-degree permutationally invariant polynomial (PIP) in variables that are functions of the distances between the Cs$^+$ ion and each of the six sites of the MB-pol water molecule [172].

Similarly, the 3B term of the Cs$^+$–water MB-nrg PEF includes two contributions,

$$V^{3B} = V^{3B}_{short} + V^{3B}_{TTM} \tag{4.4}$$

where V^{3B}_{TTM} is the 3B classical polarization contribution described by the same Thole-type model as in V^{2B}_{TTM}, and V^{3B}_{short} describes 3B short-range contributions that are represented by a 4th-degree PIP in variables that are functions of the same distances as in V^{2B}_{short} [248]. The coefficients of both 2B and 3B PIPs were optimized using Tikhonov regression (also known as ridge regression)[209] to reproduce reference interaction energies obtained from high-level electronic structure calculations.

4.2.2 Interaction energies, fitting procedure, and MD simulations

The 2B and 3B reference energies were taken from Refs. 172 and 248 where MOLPRO (version 2015.1) was used to carry out electronic structure calculations at the coupled cluster level of theory using single, double and perturbative triple excitations, i.e., CCSD(T), the "gold standard" for chemical accuracy [168]. In Ref. 172, the 2B CCSD(T) energies were calculated in the complete basis set (CBS) limit that was achieved through a two-point extrapolation[67, 82] between the values obtained with the correlation-consistent polarized valence triple zeta (aug-cc-pVTZ for H,O, and cc-pwCVTZ for Cs$^+$) and quadruple zeta (aug-cc-pVQZ for H,O, and cc-pwCVQZ for Cs$^+$) basis sets [53, 81, 97, 231] . In Ref. 248, the 3B CCSD(T) energies were calculated using the aug-cc-pVTZ basis set for the O and H atoms, and the cc-pwCVTZ basis set for Cs$^+$, and were corrected for the basis set superposition error using the counterpoise method

[27]. In both 2B and 3B energy calculations, the ECP46MDF pseudopotential was used for the core electrons of Cs^+ [117].

The original 2B training set consisted of $Cs^+(H_2O)$ dimer configurations generated on a uniform spherical grid, with the Cs^+–O distance in the 1.6 - 8 Å range [172]. For the present study, dimer configurations with interaction energies larger than 100 kcal/mol were removed since they were found to be not necessary for representing $Cs^+(H_2O)$ configurations sampled in MD simulations at ambient conditions. The 2B pool was then further reduced to 13525 dimer configurations after randomly removing 1547 configurations for the 2B test set.

Due to the larger number of degrees of freedom, the original 3B training set was generated in Ref. 248 by extracting $Cs^+(H_2O)_2$ trimer configurations from MD simulations of a single Cs^+ ion in liquid water at 298.15 K. For the present study, the original 3B set of Ref. 248 was reduced to a 3B pool of 34441 configurations after randomly removing 4480 configurations for the 3B test set.

The MD simulations presented in Section 4.3.4 were carried out in the isobaric-isothermal (NPT) ensemble for a box containing a single Cs^+ ion and 277 H_2O molecules. The equations of motion were propagated using the velocity-Verlet algorithm with a timestep δt of 0.2 fs. The temperature of 298.15 K was controlled by Nosé-Hoover chains of 4 thermostats attached to each degree of freedom while the pressure of 1.0 atm was controlled following the algorithm described in Ref. 124. All MD simulations were carried out using an in-house software based on DL_POLY 2.0 [197].

4.2.3 Active learning

An AL framework based on uncertainty and error estimation was used to generate optimal 2B and 3B training sets with the goal of reducing the number of dimers and trimers necessary to develop Cs^+–water MB-nrg PEF, without compromising accuracy. The major difficulty faced by the active learner in generating optimal 2B and 3B training sets is represented by the need to determine the relevance of candidate dimer and trimer configurations before knowing the associated 2B and 3B energies [187]. It is apparent that the more accurate the active learner is, the more precise its assessment of a molecular configuration is. In addition, for efficiency purposes, the energy estimation made by the learner should be computationally inexpensive compared to the energy determination performed by the predictor.

In this context, Gaussian process regression (GPR) provides a general approach to assessing the relevance of a candidate configuration by accurately estimating the associated energy [165]. GPR implies a correlation between the unknown energies of the candidate configurations and the energies determined for configurations that are already in the training sets. The correlation is expressed by the covariance matrix between known and unknown values of the energies, with the elements of the covariance matrix being calculated by a kernel function. GPR assumes that both known and unknown energies are distributed according to a multidimensional Gaussian distribution and then uses the covariance matrix to predict the conditional probability distribution of the unknown energies given the known energies. The ability of GPR to interpolate between known energy values makes it a good model for local uncertainty prediction. It should be noted that a similar approach is exploited by Gaussian Approximation Potential (GAP) models that have been developed to represent interatomic interactions [15].

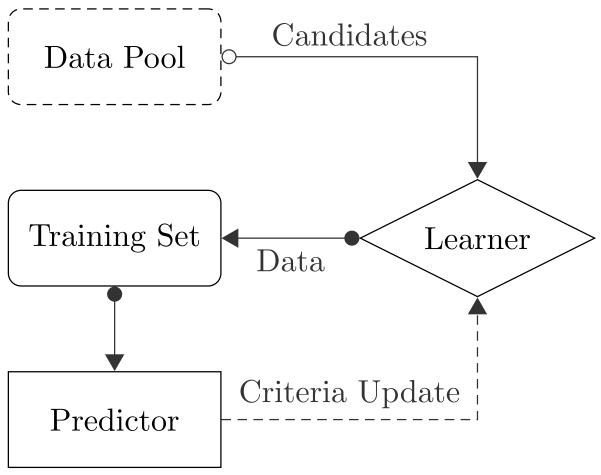


Figure 4.1. Schematic representation of the AL framework introduced in this study

Our AL framework, shown in Fig. 4.1, consists of a pool of an unknown number of molecular configurations, corresponding to $Cs^+(H_2O)$ dimers for the 2B pool and $Cs^+(H_2O)_2$ trimers for the 3B pool, a predictor, and a learner that, based on feedback from the predictor, selects configurations from the pool and adds them to the training set. The complete AL protocol is summarized below:

- At each iteration t, the pool S sends a subset of configurations with unknown energies ($C_t = \{x_j\}_t \subseteq S$) to the learner as training set candidates.
- Depending on the iteration index t, a training set T_t is formed:
  - At $t = 0$, all configurations in C_0 are added to the training set T_0 and their actual energies are determined.
  - For $t > 0$:
    - The training set T_{t-1} from the previous iteration is divided into clusters $\{\tau_{t-1,k}\}$ containing a fixed number of molecular configurations, independent of the training set size.
    - A cluster label k_j is predicted for each candidate configuration x_j in C_t (i.e., each candidate configuration x_j is assigned to one of the clusters $\{\tau_{t-1,k}\}$).
    - The uncertainty ΔE_j on the energy of the candidate configuration x_j is estimated as the GPR variance calculated for the entire cluster τ_k, $k = k_j$.
    - The error Err_j on the energy of the candidate configuration x_j is defined as the average error associated with the energies predicted by the model for all the configurations in the cluster τ_k, $k = k_j$.
    - A selection probability $P_t(x_j)$, proportional to the weighted sum of the energy uncertainty and the energy error, is assigned to each candidate configuration x_j in C_t,

$$P_t(x_j) \propto [w_{\Delta E} * \Delta E_j + w_{Err} * Err_j] \tag{4.5}$$

- $*$ A subset of configurations $\{\hat{x}_i\}_t \subseteq C_t$ is selected and, after determining the associated actual energies ε_i, added to the training set, $T_t = \{(\hat{x}_i, \varepsilon_i)\}_t \cup T_{t-1}$.

- The model M is trained on the training set T_t.

- The errors associated with the energies predicted by the model for all configurations in the training set T_t are updated

- The cycle is stopped when the gradient of the test error becomes lower than a predefined value.

The division into clusters $\{\tau_{t-1,k}\}$ of equal size reduces the computational cost associated with GPR, which typically scales as $O(n^3)$ [165] . Since a radial basis function (RBF) kernel, which is based on the L2 distance, is used to determine the similarity between two configurations, it follows that configurations close to the candidate configuration play a central role in the GPR process. The use of the RBF kernel function allows interpolation with GPR only between configurations that are in the same cluster as the candidate, which, in turn, helps reduce the computational cost without losing predictive accuracy. As shown in Eq. 4.5, the learner selects configurations based on the weighted sum of uncertainty and model error. This procedure ensures a balanced exploration of the configuration space, exploiting the decision-making process.

Different reduction methods that exploit either molecular features [86] or model diversity [196] have recently been proposed. While some are based on correlation estimation as in our AL framework, approaches solely based on molecular features lack the adaptability that arises from the constant feedback of the fitting model. In contrast, our AL framework improves its ability to select new structures as the process advances: a small subset (5%) of candidates is selected and added to the training set to improve the reliability of the learner, at each iteration. As our AL framework, approaches based on model diversity, such as the query by committee (QBC) methods of Ref. 196, share similar advantages over feature-based approaches. However, our AL framework differs from the QBC method of Ref. 196 in three main aspects. 1) Our AL framework does not assume any knowledge about the initial pool. In contrast, since the inclusion criterion used in Ref. 196 is chosen empirically, the resulting AL approach is system dependent. 2) Since the candidates in Ref. 196 are selected based on a preset value of the inclusion criterion, a balanced exploration of the configuration space is not guaranteed. Our AL framework instead assigns a probability to each configuration in the pool. This implies that there always exists the possibility to select low-probability candidates, which, consequently, guarantee a balanced exploration of the configuration space. 3) While the AL approach used in Ref. 196 only relies on the standard deviation calculated using the predictor model, our AL framework exploits both the training error calculated using the predictor model (i.e., the MB-nrg PEF) and the uncertainty calculated using the Gaussian process regression, which results in a performance improvement of the overall AL framework. In this context, it should be noted that, although our AL framework improves upon reduction methods that exploit either molecular features[86] or model diversity[196], a perfect training set reduction may still not be achieved due to the practical impossibility of achieving a perfect balance between exploration of completely new configurations and exploitation of configurations already in the training sets.

In this study, we used the KMeans module available in the Scikit-learn Python package, *version 0.21.3*, to cluster both the $Cs^+(H_2O)$ dimers and the $Cs^+(H_2O)_2$ trimers in the corresponding 2B and 3B training sets and the cluster size was fixed at 50 configurations. For GPR we used the class *GaussianProcessRegressor* and the *RBF* kernel available in the same Python package.

Both GPR and KMeans require a vector representation of the 2B and 3B structures in the high-dimensional configuration space. For this purpose, we used the many-body tensor representation (MBTR) of atomic environments [85]. MBTR defines a structural descriptor that is easily computable and well suited to calculate the kernels for both GPR and KMeans. The MBTR descriptor is constructed by storing the terms of the Coulomb matrix[175] associated with each pair of the N_e chemical elements constituting the molecular system of interest into an $N_e \times N_e \times d$ tensor, where d is the largest number of unique pairs of the same two chemical elements. The MBTR descriptor thus takes the form

$$f_k(x,\boldsymbol{z}) = \sum_{\boldsymbol{i}}^{N_a} w_k(\boldsymbol{i})\mathscr{D}(x, g_k(\boldsymbol{i})) \prod_{j=1}^{k} C_{z_j, Z_{i_j}}, \tag{4.6}$$

where $\boldsymbol{z} \in \mathbf{N}^k$ are atomic numbers, $\boldsymbol{i} = (i_1, \ldots, i_k) \in \{1, \ldots, N_a\}$ are index tuples, k runs over the number of atoms, $\mathscr{D}$ is a broadening function, C is the element correlation matrix, and g_k is a function that assigns a scalar to the k atoms based on a k-body physical feature. The MBTR descriptor is then discretized and rearranged in the form of a vector.

We used the Python package *qmmlpack* for the vector representation of the 2B and 3B configurations in their respective training sets. The broadening function $\mathscr{D}$ was chosen to be the normal distribution with $k = 2, 3$. The inverse of the distance, r^{-1}, and the angle, θ, were used as g_k for $k = 2, 3$, respectively. The number of bins and the width of the normal distribution were tuned to guarantee the efficiency of the MBTR calculations, without compromising accuracy.

4.3 Results

The results of our AL framework are presented in the following three subsections. First, we discuss the learning curves for the 2B and 3B energies, and comparisons are made between our AL framework and a generic approach based on a random selection (RS) of molecular configurations. Second, we introduce sketch-maps[41] of different 2B and 3B training sets generated through our AL framework and discuss the corresponding distributions of 2B and 3B energies. Third, we analyze the interaction and many-body energies of small water clusters as well as the Cs^+-oxygen radial distribution functions (RDFs) of liquid water calculated using different 2B and 3B training sets generated through our AL framework.

4.3.1 Learning curves of 2B and 3B energies

Figs. 4.2 and 4.3 show the learning curves for the 2B Cs^+–H_2O and 3B Cs^+–$(H_2O)_2$ energies, respectively, calculated for the training (left panels) and test (right panels) sets as a

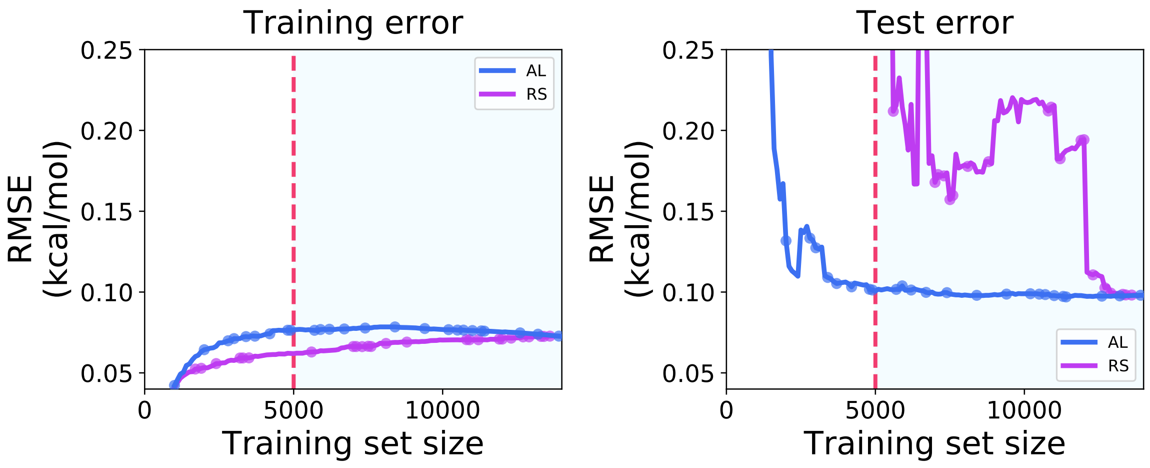


Figure 4.2. RMSEs (in kcal/mol) associated with the 2B training (left) and test (right) sets displayed as a function of the training set size. Blue and magenta curves correspond to AL and RS learning curves, respectively. The dashed line indicates the optimal training set size as determined in this study.

function of the training set size. Learning curves obtained using both our AL framework (blue) and RS approach (magenta) are shown.

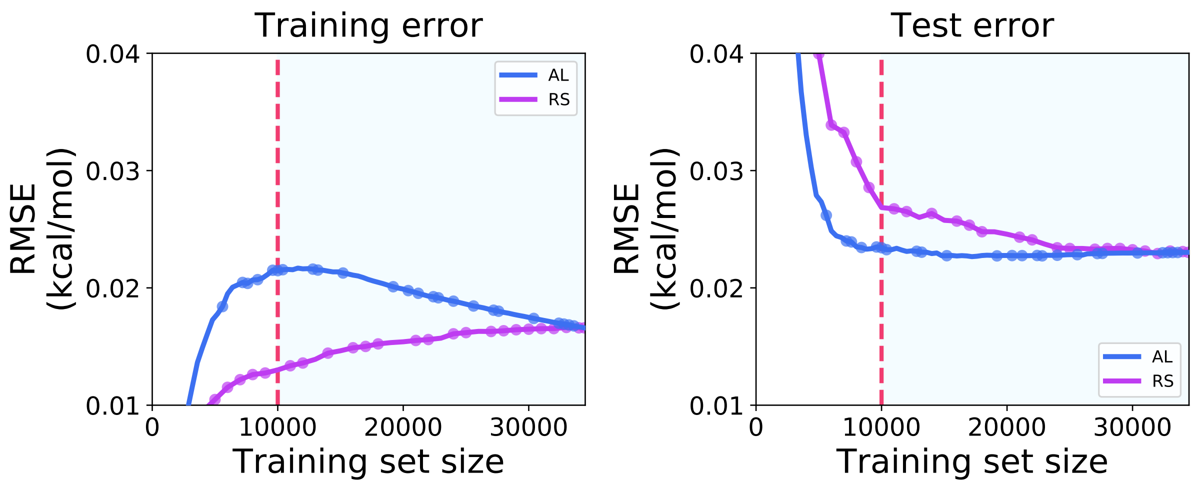


Figure 4.3. RMSEs (in kcal/mol) associated with the 3B training (left) and test (right) sets displayed as a function of the training set size. Blue and magenta curves correspond to AL and RS learning curves, respectively. The dashed line indicates the optimal training set size as determined in this study.

The training root-mean-square errors (RMSEs) associated with the RS approach increase monotonically as a function of the training set size for both 2B and 3B energies while the corresponding AL curves display steeper increases for smaller training sets, reach a maximum, and then decrease. The test RMSEs show different trends, with the curves obtained with our AL framework displaying a significantly faster decrease as a function of the training set size. Since our AL framework specifically targets configurations with higher uncertainties and neighborhood training errors, these configurations are selected more frequently by the learner and added to the training set. It follows that the configurations that are left in the pool after each iteration are associated with progressively smaller uncertainties and training errors. This implies that, when added to the training sets in subsequent iterations, these configurations necessarily lead to a decrease of the training RMSEs and only negligible variations in the test RMSEs as shown in in Figs. 4.2 and 4.3.

As a general rule, the simultaneous stabilization or decrease of the training error and the stabilization of the test error are good indicators of the convergence of the learning process [23, 66]. Therefore, based on the analysis of both training and test RMSEs obtained with our AL framework, cutoff values for the training set size could be chosen. The optimal numbers of configurations in the 2B and 3B training sets for the Cs^+–water MB-nrg PEFs were determined to be 5000 $Cs^+(H_2O)$ dimers and 10000 $Cs^+(H_2O)_2$ trimers, respectively.

4.3.2 Sketch-maps

Sketch-maps have been shown to be useful tools for representing high-dimensional configuration spaces with lower-dimensional projections that are easily interpretable in terms of well-defined structural features [41, 42, 141].

To provide structural insights into the composition of the 2B and 3B training sets, with varying sizes, obtained with our AL framework, MBTR was used to generate the sketch-maps shown in Figs. 4.4 and 4.5, respectively. Panel a) of Fig. 4.4 is a representation of the entire 2B pool projected onto a 2-dimensional space. Each point on the map corresponds to a $Cs^+(H_2O)$ dimer configuration and the associated color indicates the corresponding CCSD(T) reference 2B energy. Since the 2B pool was generated on a grid by varying the Cs^+-O distance and distorting the water molecule, these features are reflected in the resulting sketch-map where points cluster together, in an orderly fashion.

Panel b) of Fig. 4.4 shows a sketch-map of the energy differences between the reference 2B energies and the corresponding values predicted by the MB-nrg PEF trained on the full 2B pool (13525 configurations). This comparison shows that the the MB-nrg PEF provides an accurate description of the overall 2B energy landscape, with deviations larger than 0.5 kcal/mol only found for $Cs^+(H_2O)$ dimers with associated binding energies larger than 40 kcal/mol, and deviations on the order of 0.04 kcal/mol for $Cs^+(H_2O)$ dimer configurations with lower binding energies (less than 40 kcal/mol). It should be noted that dimer configurations with larger binding energies are unlikely to be visited in MD simulations at ambient conditions and are included in the 2B training sets to guarantee that the PIPs representing short-range interactions within the the

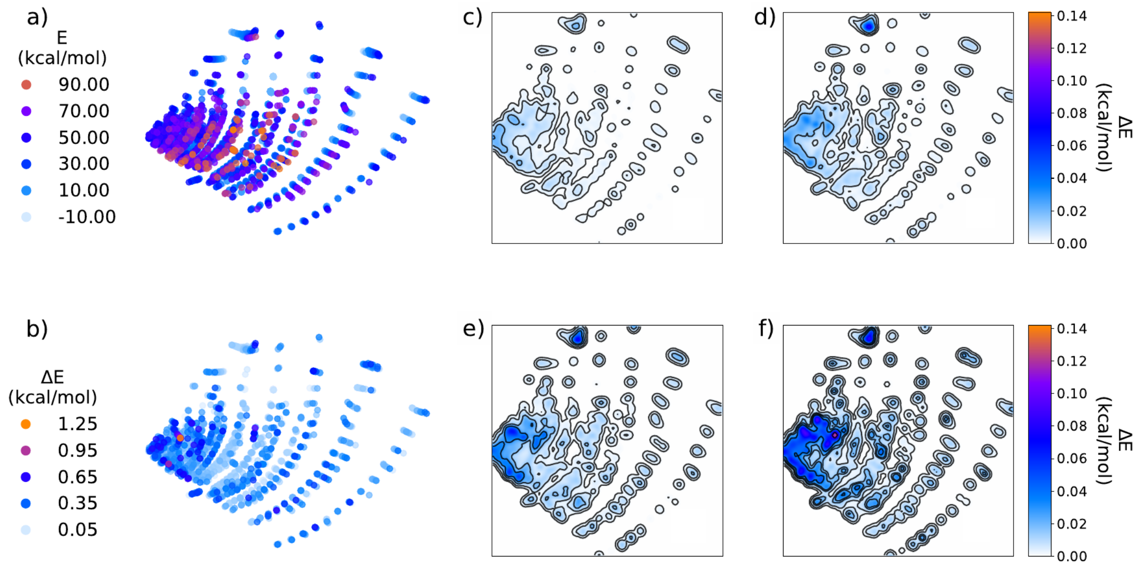


Figure 4.4. Sketch-maps of the 2B configurations. The map in in a) represents the reference CCSD(T) energies while the map in b) represents the difference, ΔE, between the reference CCSD(T) energies and the energies predicted by the MB-nrf PEF trained on the full pool of 2B configurations. The maps in c) to f) represent the difference, ΔE, between the energies predicted by the MB-nrg PEF trained on of the full training set and the corresponding values predicted by MB-nrg PEFs trained on the reduced-size training sets of 10000, 8000, 6000, and 4000 configurations generated using the AL framework, respectively.

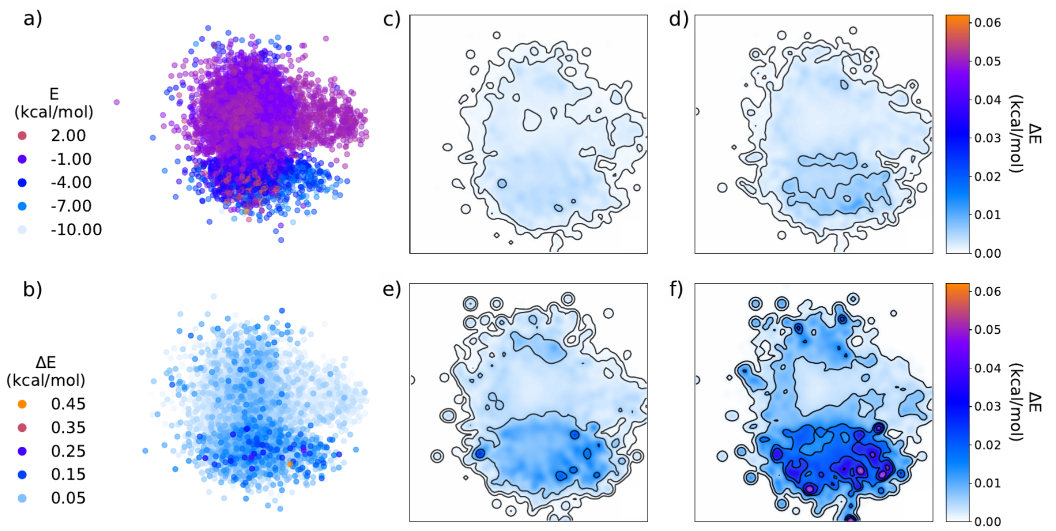


Figure 4.5. Sketch-maps of the 3B configurations. The map in in a) represents the reference CCSD(T) energies while the map in b) represents the difference, ΔE, between the reference CCSD(T) energies and the energies predicted by the MB-nrf PEF trained on the full pool of 2B configurations. The maps in c) to f) represent the difference, ΔE, between the energies predicted by the MB-nrg PEF trained on of the full training set and the corresponding values predicted by MB-nrg PEFs trained on the reduced-size training sets of 20000, 15000, 10000, and 5000 configurations generated using the AL framework, respectively.

MB-nrg PEF are well-behaved at short Cs^+–water distances. Panels c-f) show sketch-maps of the differences between 2B energies predicted by the MB-nrg PEF trained on the full 2B pool and the corresponding values predicted by MB-nrg PEFs trained on progressively smaller training sets containing 10000, 8000, 6000, 4000 configurations generated using our AL framework. As expected, systematically reducing the training set size introduces progressively larger errors, with training sets with fewer than 4000 dimer configurations leading to overfitting. This analysis shows that our AL framework allows for significantly reducing the original 2B Cs^+–H_2O training set without compromising the overall accuracy of the resulting MB-nrg PEF. In this context, it should be noted that the areas of the sketch-maps in panels c-f) that display larger deviations from the original MB-nrg PEF of Ref. 172, as the training set size decreases, correspond to dimer configurations for which the original MB-nrg PEF also shows larger deviations from the CCSD(T) reference data (panel b).

Similar conclusions can be drawn from the analysis of the sketch-maps of the 3B training sets shown in Fig. 4.5. Since the original 3B pool was generated by extracting $Cs^+(H_2O)_2$ trimers from MD simulations of a single Cs^+ ion in liquid water, the resulting sketch-map (panel a) displays a more uniform distribution in the 2-dimensional space compared to the corresponding sketch-map obtained for the 2B pool. Depending on the associated CCSD(T) reference 3B energies, trimer configurations broadly cluster in two areas, with the “dividing surface" being between -5.0 and -3.0 kcal/mol; this is highlighted by the sudden change in color in panel a). Also in this case, the original MB-nrg PEF closely reproduces the CCSD(T) reference 3B energies over the entire configuration space of the 3B pool, as shown in panel b). As for the 2B energies, progressively smaller training sets of 20000, 15000, 10000, 5000 configurations, generated using our AL framework and analyzed through the sketch-maps shown in panels c-f), lead to progressively larger deviations from the original MB-nrg PEF. It should be noted that, on average, the deviations remain smaller than 0.06 kcal/mol even for the smallest training set (5000 trimer configurations).

4.3.3 Clusters analysis

To assess the relative accuracy of the various training sets generated using our AL framework and determine how the associated differences in the representation of 2B and 3B energies affect the ability of the resulting MB-nrg PEFs to reproduce the properties of water from the gas to the condensed phase, deviations from the reference 2B and 3B energies of low-lying isomers of the $Cs^+(H_2O)_{n=1-3}$ clusters are analyzed in Fig. 4.6. This analysis is carried out for several MB-nrg PEFs generated using the minimal 2B and 3B training sets shown in Figs. 4.4 and 4.5 of 4000 dimer configurations and 5000 trimer configurations, respectively. Also shown for comparison are the deviations obtained with the same training sets generated from random selection. In all cases, the differences between the 2B and 3B energies predicted by the different MB-nrg PEFs are comparable for all clusters, and often smaller than the corresponding differences between the original MB-nrg PEF fitted to the full 2B and 3B training sets and

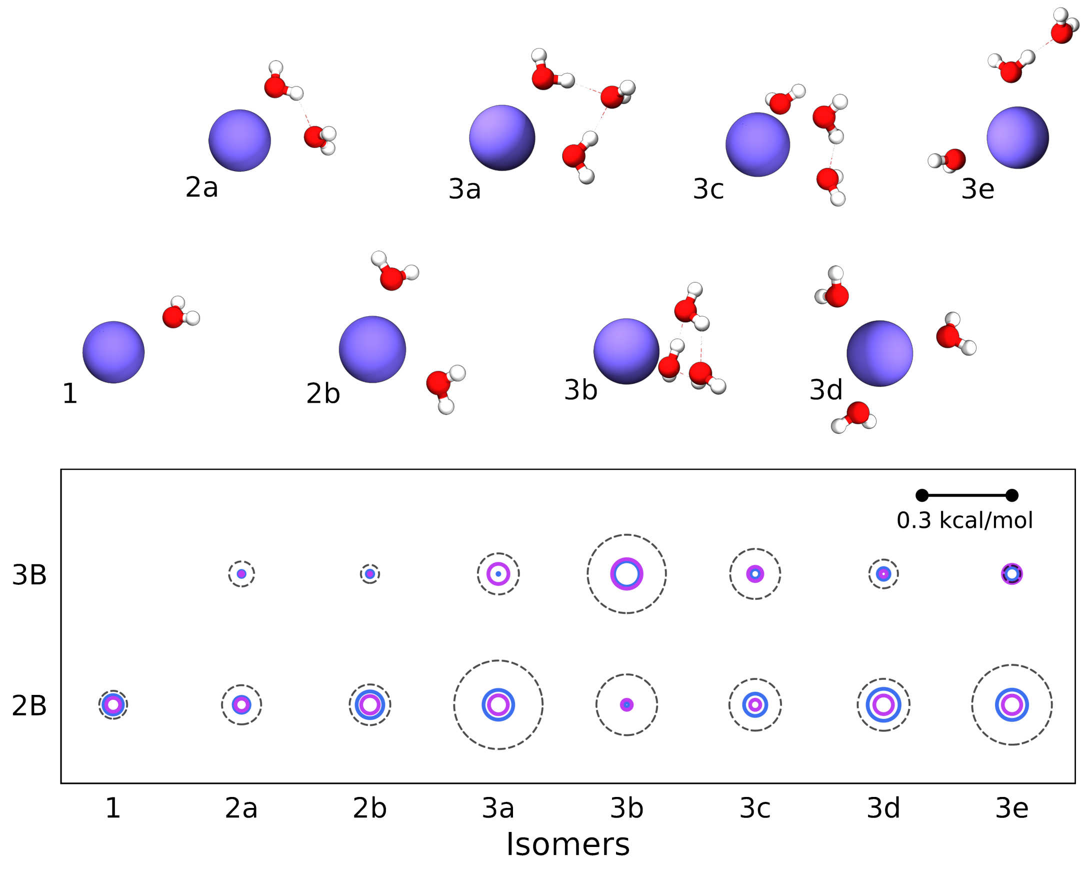


Figure 4.6. Schematic representation of the errors associated with the 2B and 3B energies of low-lying isomers of $Cs^+(H_2O)_{n=1-3}$ clusters. The dashed black circles represent the difference between the reference CCSD(T) energies and the corresponding values obtained with the MB-nrg PEF trained on the full 2B and 3B pools. The other solid circles represent the differences between the energies predicted by the MB-nrg PEF trained on the full 2B and 3B pools and the corresponding values predicted by MB-nrg PEFs trained on 4000 2B configurations and 5000 3B configurations, with blue and magenta corresponding to the AL and RS training sets, respectively.

the CCSD(T) reference data. This analysis thus indicates that the reduction of the training set sizes does not affect the ability of the resulting MB-nrg PEFs to correctly represent 2B and 3B energies in small water clusters. It should be noted that this is true for both families of MB-nrg PEFs derived from training sets generated through AL and RS. This similarity can be attributed to the intrinsic low dimensionality of the $Cs^+(H_2O)$ dimers and $Cs^+(H_2O)_2$ trimers that make up the corresponding 2B and 3B training sets, which allowed for extensive sampling of the relevant configurations for the development of the original training sets in Refs. 172 and 248 . However, while no appreciable differences exist in the performance of the two sets of MB-nrg PEFs, AL clearly provides a more efficient approach to the selection of the training set sizes as demonstrated by the significantly higher variability associated with the learning curves obtained with the RS approach. The efficiency of the AL framework thus becomes particular important when, differently from the present case of the Cs^+–water MB-nrg PEF, no prior information on training sets is provided. This aspect of our AL framework will be the subject of a forthcoming study.

4.3.4 Radial distribution functions

To investigate the effects of training set reduction on modeling the properties of bulk solutions, the Cs^+–O RDFs calculated using different MB-nrg PEFs obtained from fits to different combinations of 2B and 3B training sets generated using AL (left panels) and RS (right panels) are analyzed in Figs. 4.7 and 4.8. The effects of the 2B training set is first assessed in Fig. 4.7 by analyzing the performance of five MB-nrg PEFs generated by fitting the 2B term to 2B training sets of various sizes (full, 10000, 8000, 6000, and 4000 dimer configurations) while fitting the 3B term to the full 3B training set for training the 3B term (34441 trimer configurations). The resulting RDFs calculated from MD simulations with the resulting MB-nrg PEFs generated from both AL and RS training sets are shown in the top left and right panels of Fig.4.7, respectively. As discussed in more detail in Ref. 248, the Cs^+–O RDF displays a narrow peak, corresponding to the first hydration shell, at 3.16 Å, and a broader peak, corresponding to the second hydration shell, at $\sim$6 Å. No appreciable differences are found between the RDFs obtained using MB-nrg PEFs with progressively smaller 2B training sets. This is further evidenced by the curves shown in the bottom panels of Fig. 4.7 representing the differences between the RDFs calculated with each of the MB-nrg PEFs trained on reduced 2B training sets and the RDF calculated with the MB-nrg PEF trained on the full 2B training set.

Similarly, the effects of the 3B training set size reduction are investigated in Fig. 4.8

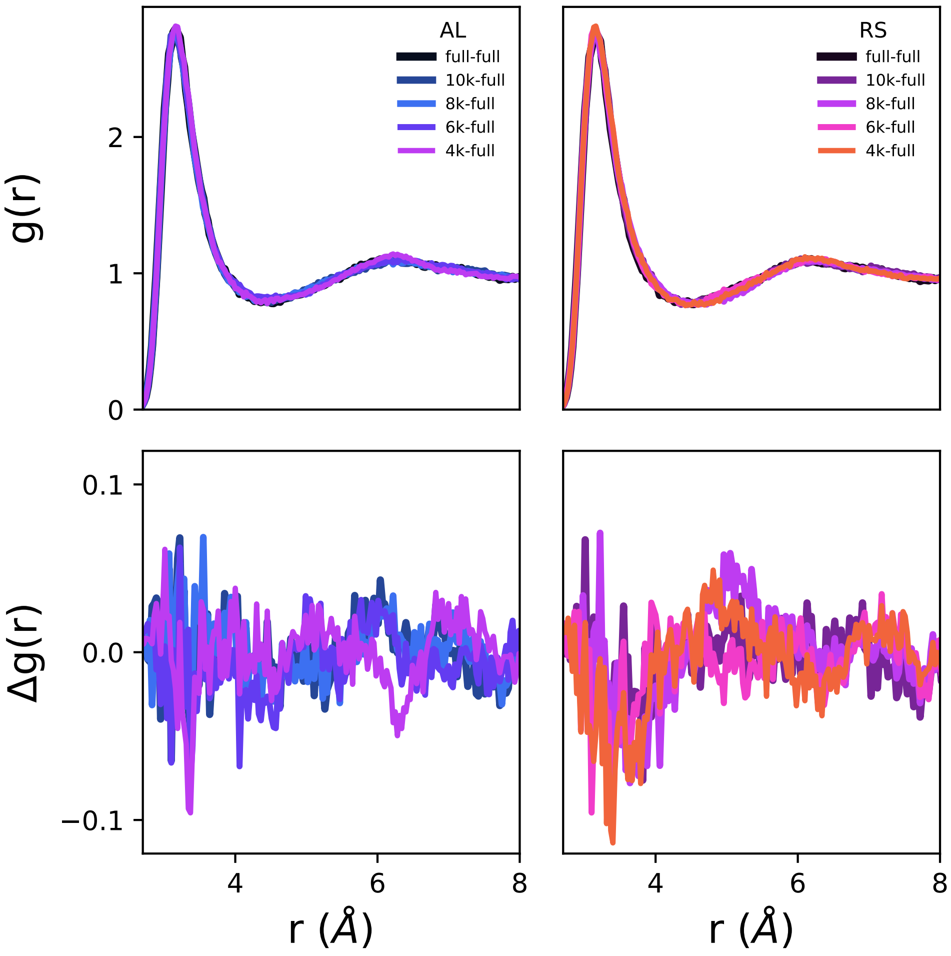


Figure 4.7. RDFs comparison between training by 2B sets from AL and RS. Top panels: Cs$^+$-O RDFs calculated from MD simulations with MB-nrg PEFs trained on progressively smaller 2B training sets (in the range of 10000-4000 dimer configurations) generated through AL (left) and RS (right), and the full 3B pool. Bottom panels: Differences between the RDF calculated with the MB-nrg PEF trained on the full 2B and 3B pool and the corresponding RDFs calculated with MB-nrg PEFs trained on the reduced-size AL (left) and RS (right) 2B training sets, and the full 3B pool.

through the analysis of five MB-nrg PEFs generated by fitting the 3B term to 3B training sets of various sizes (full, 20000, 15000, 10000, and 5000 trimer configurations) while fitting to the 2B term to the full 2B training set. In this case, reducing the 3B training set size to less than 10000 trimer configurations results in small differences in the Cs$^+$–water RDF for distances larger than 5.0 Å, which lead to a shift of the second hydration shell to slightly larger distances. However, as shown in the bottom panels of Fig. 4.8, these differences are barely noticeable and do not lead

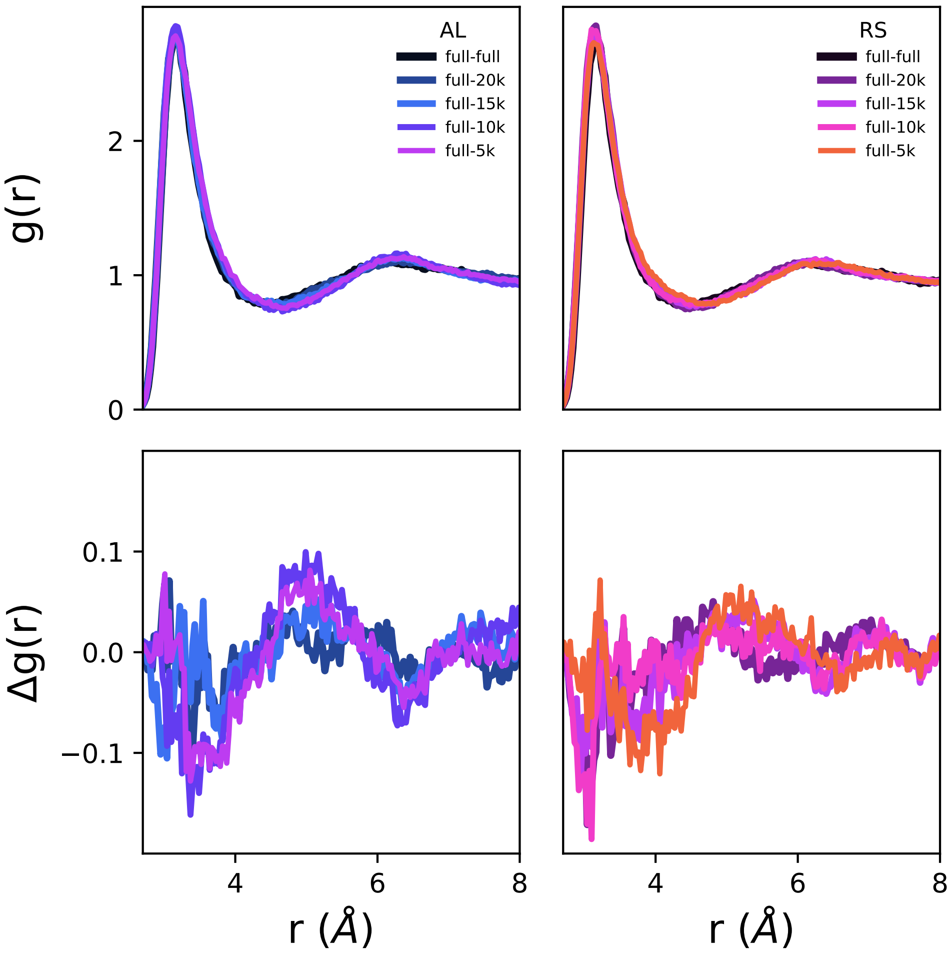


Figure 4.8. RDFs comparison between training by 3B sets from AL and RS. Top panels: Cs^+-O RDFs calculated from MD simulations with MB-nrg PEFs trained on progressively smaller 3B training sets (in the range of 20000-5000 trimer configurations) generated through AL (left) and RS (right), and the full 2B training set. Bottom panels: Differences between the RDF calculated with the MB-nrg PEF trained on the full 2B and 3B pool and the corresponding RDFs calculated with MB-nrg PEFs trained on the reduced-size AL (left) and RS (right) 3B training sets, and the full 2B pool.

to any qualitative change in the hydration structure of Cs^+ in liquid water.

Overall, the analysis of both cluster and bulk properties demonstrates that the application of our AL framework to the original pools of 2B and 3B configurations of Refs. 166 and 248, respectively, leads to significantly smaller training sets, without loss of accuracy, which, in turn, largely reduces the cost associated with the development of CCSD(T)-level MB-nrg PEFs.

4.4 Conclusions

In this study, we introduced an AL framework for generating representative training sets needed for the development of MB-nrg PEFs [11, 172] . Our AL framework is based on uncertainty and error estimation, and consists of a pool of an unknown number of molecular configurations, a predictor, and a learner that, based on feedback from the predictor, selects configurations from the pool and adds them to the training set. The selection process relies on Gaussian process regression and clustering of the configurations in the training set, which allows for efficiently identifying the most relevant configurations needed to accurately represent the target many-body PES.

The application of our AL framework to the development of a Cs^+–water MB-nrg PEF chosen as a case study led to significantly smaller training sets than those found necessary for the development of the original MB-nrg PEF. Analyses of the interaction and many-body energies calculated for small $Cs^+(H_2O)_n$ cluster as well as the Cs^+-oxygen RDF calculated from MD simulations of a single Cs^+ ion in water demonstrate that the new MB-nrg PEFs derived from the reduced-size training sets generated through AL display the same accuracy of the original MB-nrg PEF derived from the full 2B and 3B pools [172, 248].

Given the computational cost associated with reference CCSD(T) calculations of individual many-body energies, our AL framework is particularly well-suited to the development of many-body PEFs, with chemical and spectroscopic accuracy, which can then be used in MD simulations of the target molecular system, from the gas to the condensed phase.

4.5 Acknowledgement

Chapter 4, in full, is a reprint of the material as it appears in "Active learning of many-body configuration space: Application to the Cs+–water MB-nrg potential energy function as a case study," Y. Zhai, A. Caruso, S. Gao, F. Paesani, in *The Journal of Chemical Physics*, 2020, 152(14). The dissertation author is the primary investigator and author of this paper.

We thank Andreas Götz for helpful discussions on the selection of optimal training sets, Matthias Rupp for guidance on using the MBTR descriptor, and Giulio Imbalzano and Michele Ceriotti for suggestions on generating of the 2B and 3B sketch-maps. This research was supported by the National Science Foundation through grant no. ACI-1642336. All calculations and simulations used resources of the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by the National Science Foundation through grant no. ACI-1053575, under allocation TG-CHE110009, and the Triton Shared Computing Cluster (TSCC) at the San Diego Supercomputer Center (SDSC).

Chapter 5

A “Short Blanket” Dilemma for the Deep Neural Network-Based Many-Body Potentials

Deep neural network (DNN) potentials have recently gained popularity in computer simulations of a wide range of molecular systems, from liquids to materials. In this study, we explore the possibility of combining the computational efficiency of the DeePMD framework and the demonstrated accuracy of the MB-pol data-driven many-body potential to train a DNN potential for large-scale simulations of water across its phase diagram. We find that the DNN potential is able to reliably reproduce the MB-pol results for liquid water but provides a less accurate description of the vapor-liquid equilibrium properties. This shortcoming is traced back to the inability of the DNN potential to correctly represent many-body interactions. An attempt to explicitly include information about many-body effects results in a new DNN potential that exhibits the opposite performance, being able to correctly reproduce the MB-pol vapor-liquid equilibrium properties but losing accuracy in the description of the liquid properties. These results suggest that DeePMD-based DNN potentials are not able to correctly “learn” and, consequently, represent many-body interactions, which implies that DNN potentials may have limited ability to predict properties for state points that are not explicitly included in the training process. The computational efficiency of the DeePMD framework can still be exploited to train DNN potentials on data-driven many-body potentials, which can thus enable large-scale, “chemically accurate" simulations of various molecular systems, with the caveat that the target state points must have been adequately sampled by the reference data-driven many-body potential in order to guarantee a faithful representation of the associated properties.

5.1 Introduction

Molecular mechanics (MM) force fields (FFs) [116, 228] have been the workhorse in computational chemistry since the early days of Monte Carlo (MC)[130] and molecular dynamics (MD) simulations [2]. Continued progress in hardware technologies [68], accompanied by the development of more realistic representations of electrostatic interactions, has enabled not only molecular simulations of progressively larger systems but also the use of more rigorous polarizable FFs[45, 87, 92, 159, 216] that go beyond the pairwise additive approximation adopted by conventional fixed-charge FFs [125, 163, 215, 222].

At the same time, the development of efficient algorithms for correlated electronic structure methods, such as coupled cluster theory [1, 82, 102], has enabled routine calculations of interaction energies for molecular clusters with chemical accuracy [67, 78, 122]. This has led to the emergence of a new class of analytical potentials that quantitatively reproduce each individual term of the many-body expansion (MBE) of the energy[140] calculated using correlated electronic structure methods. When applied to aqueous systems[7–9, 11, 32–34, 47, 83, 126, 172, 224–226] and molecular fluids [170, 173, 238], these many-body (MB) potentials exhibit unprecedented accuracy, enabling predictive simulations from the gas to the condensed phases [148]. Concurrently, machine learning (ML) approaches have gained popularity in computational molecular sciences mainly due to the rapid evolution of GPU and TPU architectures [145]. In particular, potentials represented by deep neural networks (DNNs) derived from electronic structure data are routinely used to model various molecular systems, from clusters to liquids and materials [16–21, 43, 44, 64, 65, 72, 75, 88, 103, 123, 182–184, 194, 195, 211, 230, 239, 242, 251].

State-of-the-art MB and DNN approaches use regression algorithms to construct data-driven representations of the multidimensional energy landscape of the system of interest. This process involves generating representative training sets of reference data calculated at an appropriate level of theory. While MB potentials require tailored parametric functions for each term of the MBE, DNN potentials are usually trained on the total energy and forces of the entire system. By applying embedding schemes to construct low-dimensional descriptors of molecular environments, DNN potentials can compute the gradients required for the propagation of the equations of motion in MD simulations more efficiently than MB potentials [73]. On the other hand, since MB potentials only use information about small clusters, the corresponding training data can be calculated at a higher level of theory than DNN potentials. As a matter of fact, MB potentials are usually trained on reference energies calculated at the coupled cluster level of theory, including single, double, and perturbative triple excitations, i.e., CCSD(T), which is currently referred to as the "gold standard" for chemical accuracy [168]. Furthermore, by construction, the functional form of MB potentials allows for accurately representing all physical contributions to the interaction energies, including both short- and long-range many-body effects [25, 54, 149].

To account for long-range interactions, DNN potentials are often trained on condensed-phase configurations, which allows for modeling long-range effects either implicitly, by effectively encoding long-range contributions into short-range representations, or explicitly, by adding effective electrostatic terms [16, 73, 103, 237, 242, 243]. This implies that, due to the large number of molecules required to model condensed-phase systems, a lower level of theory than CCSD(T), usually density functional theory (DFT) [153], has to be used to retrieve the reference energies. In this context, it has recently been shown that the interplay between functional-driven and density-driven errors may impact the overall accuracy of DFT models and their transferability from gas-phase to condensed-phase systems [49, 50, 151, 192, 199, 219]. By construction, these limitations also affect the ability of DNN potentials derived from DFT reference data to "extrapolate" to thermodynamic state points different from those used in the training process

[109].

In this work, we investigate the possibility of integrating the best features of MB potentials (i.e., accuracy and transferability) and DNN potentials (i.e., speed and ease to use) into a computational framework that can enable large-scale MD simulations with chemical accuracy. To this end, we focus on the molecular modeling of water as a prototypical system that has posed several challenges since the early days of MC and MD simulations[13, 161] due to its rich phase behavior characterized by several anomalies [59]. As a representative state-of-the-art MB potential, we selected MB-pol[7, 9, 126] due to its demonstrated ability to correctly predict the properties of water across the entire phase diagram [167], including gas-phase clusters [29, 46, 169], liquid water [166], the vapor-liquid interface [129, 135, 137], and ice [132–134, 156]. MB-pol has also recently been used to predict structural and thermodynamic properties of supercooled water down to 200 K at 1 atm, which were found to be in excellent agreement with experimental data that are available above 225 K [62]. However, due to the associated computational cost, the MD simulations with MB-pol reported in Ref. 62 were limited in terms of both system's size (up to 512 water molecules) and sampling time (up to $\sim$130 ns). The prospect of developing a fast DNN potential trained on MB-pol simulation data, which retains the same accuracy of MB-pol across the entire phase diagram, is thus particularly appealing. This will enable large-scale simulations of water as a function of temperature and pressure, which will provide further insights into water's anomalous behavior and allow for full exploration of the so-called water's "no man's land" that has been proven difficult to probe experimentally [5, 99, 100, 107, 155, 200]. To this end, we selected DeePMD[73, 242, 243] as a representative, state-of-the-art framework for developing a DNN potential of water trained on the MB-pol simulations of Ref. 62. DeePMD-based DNN potentials have already been used in MD simulations of various molecular systems, including water [63, 198, 241, 244], ionic liquids [115], and metals [120, 143, 229], and enabled MD simulations with up to 10 billion atoms [70].

The article is organized as follows: In Section 5.2, we summarize the main features of the MB-pol potential (Section 5.2.1) and the DeePMD framework (Section 5.2.2). In Section 5.3.1, we first assess the ability of the DeePMD-based DNN potential to reproduce thermodynamic and structural properties of liquid water calculated with MB-pol from the boiling point down to deeply supercooled temperatures. We then use the DNN potential to characterize several vapor-liquid equilibrium properties as well as many-body dependent properties of gas-phase clusters. In Section 5.3.2, we introduce two other DeePMD-based potentials, DNN(VLE10) and DNN(VLE20), that are trained on an expanded training set that adds vapor-liquid configurations to the training set used to develop the DNN potential. The performance of both DNN(VLE10) and DNN(VLE20) potentials is assessed on the same structural and many-body-dependent properties used to assess the performance of the DNN potential. In Section 5.3.3, we introduce three DeePMD-based potentials, DNN(MB4), DNN(MB10), and DNN(MB20), that are trained to incorporate low-order many-body interactions, and assess their performance on the same structural and many-body dependent properties used in the assessment of the DNN potential. Lastly, in Section 5.4, we summarize our work and discuss possible future synergies between MB and DNN potentials.

5.2 Methods

5.2.1 MB-pol

Since the MB-pol potential of water has already been described in detail in the literature, we only overview here its salient features [7, 9, 126]. MB-pol was derived from the MBE that expresses the energy, E_N, of a system containing N (atomic or molecular) monomers as the sum of individual n-body energy contributions,

$$E_N(1,\ldots,N) = \sum_{i=1}^{N} \varepsilon^{1\mathrm{B}}(i) + \sum_{i=1}^{N}\sum_{j>i}^{N} \varepsilon^{2\mathrm{B}}(i,j) + \sum_{i=1}^{N}\sum_{j>i}^{N}\sum_{k>j>i}^{N} \varepsilon^{3\mathrm{B}}(i,j,k) + \cdots + \varepsilon^{\mathrm{NB}}(1,\ldots,N) \tag{5.1}$$

Here, $\varepsilon^{1\mathrm{B}}$ represents the distortion energy of an isolated monomer, such that $\varepsilon^{1\mathrm{B}}(i) = E(i) - E_{\mathrm{eq}}(i)$ where $E_{\mathrm{eq}}(i)$ is the energy of the i-th monomer in its equilibrium geometry. The n-body energies, $\varepsilon^{\mathrm{nB}}$, are defined recursively for $1 < n \leq N$ by the expression

$$\begin{aligned}\varepsilon^{\mathrm{nB}} = E_{\mathrm{n}}(1,\ldots,n) - \sum_{i=1}^{N}\varepsilon^{1\mathrm{B}}(i) - \sum_{i=1}^{N}\sum_{i<j}^{N}\varepsilon^{2\mathrm{B}}(i,j) - \sum_{i=1}^{N}\sum_{i<j}^{N}\sum_{i<j<k}\varepsilon^{3\mathrm{B}}(i,j,k) - \cdots \\ \cdots - \sum_{i<j<k<\ldots}^{N}\varepsilon^{(\mathrm{n}-1)\mathrm{B}}(i,j,k,\ldots,n-1)\end{aligned} \tag{5.2}$$

MB-pol approximates Eq. 5.2 as:

$$E_N(r_1,..,r_N) = \sum_{i=1}^{N}\varepsilon^{1\mathrm{B}}(i) + \sum_{i>j}^{N}\varepsilon^{2\mathrm{B}}(i,j) + \sum_{i>j>k}^{N}\varepsilon^{3\mathrm{B}}(i,j,k) + E_{\mathrm{POL}} \tag{5.3}$$

The one-body term ($\varepsilon^{1\mathrm{B}}$) is represented by the potential developed by Partridge and Schwenke [154]. The two-body term ($\varepsilon^{2\mathrm{B}}$) describes four distinct contributions: permanent electrostatics, dispersion, 2B polarization, and 2B short-range interactions. The three-body term ($\varepsilon^{3\mathrm{B}}$) describes two distinct contributions: 3B polarization and 3B short-range interactions. 2B and 3B short-range interactions are represented by 2B and 3B permutationally invariant polynomial (PIPs)[28] that were fitted in order for $\varepsilon_{2\mathrm{B}}$ and $\varepsilon_{3\mathrm{B}}$ to reproduce 2B and 3B energies calculated at the CCSD(T) level of theory in the complete basis set (CBS) limit [7, 9]. 2B and 3B polarization contributions are implicitly included in E_{POL} in Eq. 5.3 which represents classical many-body interactions at all orders through a polarization term. Further details of the MB-pol potential can be found in the original references [7, 9, 126].

5.2.2 DeePMD

The DeePMD framework reads atomic positions and associated atom types as input features [73]. Neighbor information for each atom i is extracted from the input feature using a predefined cutoff radius (r_c) and stored as the coordinate difference of each ij atom pair into $\mathscr{R}^i \in \mathbb{R}^{N_i \times 3}$, where N_i is the number of neighboring atoms. Each local feature is then mapped onto generalized coordinates $\tilde{\mathscr{R}}^i$ as outlined in Ref. 243. A local embedding matrix, $\mathscr{G}^i$, is applied to each local feature $\mathscr{R}^i$ in order to ensure rotation and permutation symmetry while preserving translation symmetry. The resulting encoded feature matrix $\mathscr{D}^i \in \mathbb{R}^{M_1 \times M_2}$ takes the form

$$\mathscr{D}^i = (\mathscr{G}^{i1})^T \tilde{\mathscr{R}}^i (\tilde{\mathscr{R}}^i)^T \mathscr{G}^{i2} \tag{5.4}$$

and is passed to a fully-connected feed-forward DNN that maps it onto an "atomic energy" E_i [243]. The total energy E is then calculated as the sum of all E_i, while the atomic forces F and virials Ξ are calculated from the derivative of the DNN with respect to the corresponding atomic positions. The DNN parameters are optimized by minimizing the loss function:

$$L(p_e, p_f, p_\xi) = \frac{p_e}{N}\Delta E^2 + \frac{p_f}{3N}\Delta F^2 + \frac{p_\xi}{9N}\Delta \Xi^2. \tag{5.5}$$

where p_e, p_f, p_ξ are weighting factors, and ΔE, ΔF, and $\Delta\Xi$ are the prediction errors for the reference energy, force, and virial values, respectively. The weighting factors p_e, p_f, p_ξ are adjusted as the training progresses in order to improve the quality of the fit.

The DeePMD-based DNN potentials presented in this study were developed with the Deep Potential Smooth Edition (DeepPot-SE) [243], following the procedure reported in Ref. 220, using 25, 50, and 100 neurons for the hidden embedding layers in the DeepPot-SE, while the submatrix of the embedding matrix uses 16 neurons. The distance cutoff was set to 6 Å, with a smoothing region of 0.5 Å. Each DNN potential is represented by a fully connected deep neural network with three layers of 240 neurons each.

Following Ref. 245, the training set for the DNN potential was constructed in an iterative fashion. Briefly, the training set comprises energies and forces of molecular configurations extracted from the MB-pol simulations of liquid water between 198 K and 368 K reported in Ref. 62 as well as additional configurations extracted from simulations carried out with three successive iterations of the DNN potential. The final training set includes 94770 configurations, each containing 256 molecules. All MB-pol reference data were computed using the MBX software package [150]. Additional details about the training set are discussed in the Supplementary Material. To account for variations in training, validation, and testing errors, four distinct potentials (hereafter referred to as seed 1, seed 2, seed 3, and seed 4, respectively) were trained using different random seeds. Only one of the four DNN potentials (seed 2) was then used in the MD simulations of liquid water and the vapor-liquid interface. Similarly, four distinct DeePMD-based potentials were trained on the expanded training sets containing vapor-liquid and cluster configurations as described in Sections 5.3.2 and 5.3.3, respectively. Fig. 5.1 shows the root-mean-square error (RMSE) curves of training and validation sets for the energies and forces per atom during the fitting process of the (seed 2) DNN potential. Overall, well-behaved learning curves are obtained for both quantities, with final errors of 0.01 kcal/mol and 1 kcal/mol Å on the energy and force validation errors, respectively. Similar errors have been reported for other state-of-the-art machine-learned potentials [158].

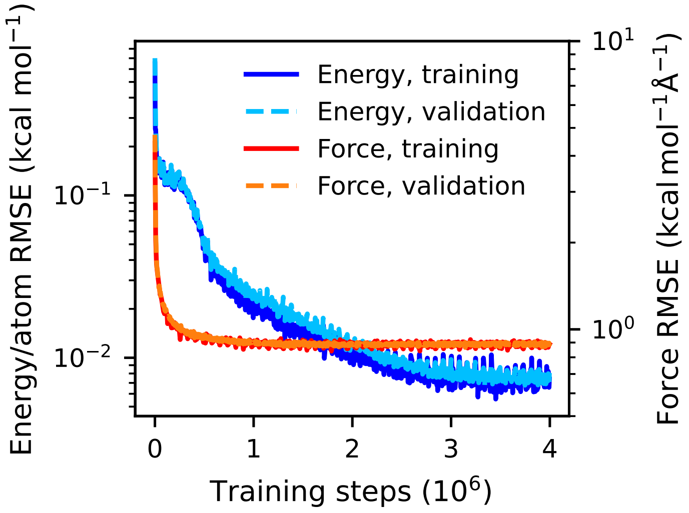


Figure 5.1. Variation of the DNN training and validation RMSEs per atom relative to the MB-pol values of the energy and force as a function of the number of training steps. For visual clarity, we show values averaged over 200 training steps.

5.2.3 Computational details

We performed two sets of MD simulations to determine the ability of the DNN potential to reproduce both bulk and interfacial properties of liquid water calculated with MB-pol. The first set of simulations was carried out for a box containing 256 water molecules in the isothermal-isobaric (*NPT*) ensemble at 1 atm and in the temperature range between 198 K to 368 K. The temperature was maintained using a global Nosé–Hoover thermostat chain of length 3 with a relaxation time of 0.05 ps, and the pressure was controlled by global Nosé–Hoover barostat with a relaxation time of 0.5 ps which was thermostatted by a Nosé–Hoover thermostat chain of length 3. At each temperature, the last frame of the MB-pol trajectories reported in Ref. 62 was used as the initial configuration for the *NPT* simulations with the DNN potential. The second set of simulations was carried out in the canonical (*NVT*) ensemble between 400 K and 575 K for a liquid slab of 512 water molecules in a box of dimensions 20 Å×20 Å×100 Å. The temperature was maintained using the same global Nosé–Hoover thermostat chain used in the *NVT* simulations. In both *NPT* and *NVT* simulations, the velocity-Verlet algorithm was used to propagate the equations of motion with a time step of 0.5 fs according to Ref. 189. All simulations were carried out using the DeePMD-kit[220] plugin for LAMMPS [208]. A complete set of input files is available on GitHub (see Data Availability).

Besides comparing the DNN and MB-pol radial distribution functions (RDFs), we also analyzed the ability of the DNN potential to describe the three-dimensional hydrogen-bond network by calculating the tetrahedral order parameter, q_{tet} [56],

$$q_{\text{tet}} = 1 - \frac{3}{8} \cdot \sum_{j=1}^{3} \sum_{k=j+1}^{4} \left(\cos(\psi_{jk}) + \frac{1}{3} \right) \tag{5.6}$$

Here, ψ_{jk} is the angle between the oxygen of the central water molecule and the oxygen atoms of two neighboring water molecules within a cutoff of 3.5 Å. When $q_{\text{tet}} = 1$, the water molecules are in a perfect tetrahedral arrangement, while $q_{\text{tet}} = 0$ represents the ideal gas limit.

In addition to the MD simulations for liquid water and the vapor-liquid interface, we also performed many-body decomposition analyses for two different sets of cluster structures. The first set consists of the first eight low-lying energy isomers of the water hexamer (Fig. 5.2), with geometries taken from Ref. 167. The hexamer occupies a special place in the description of

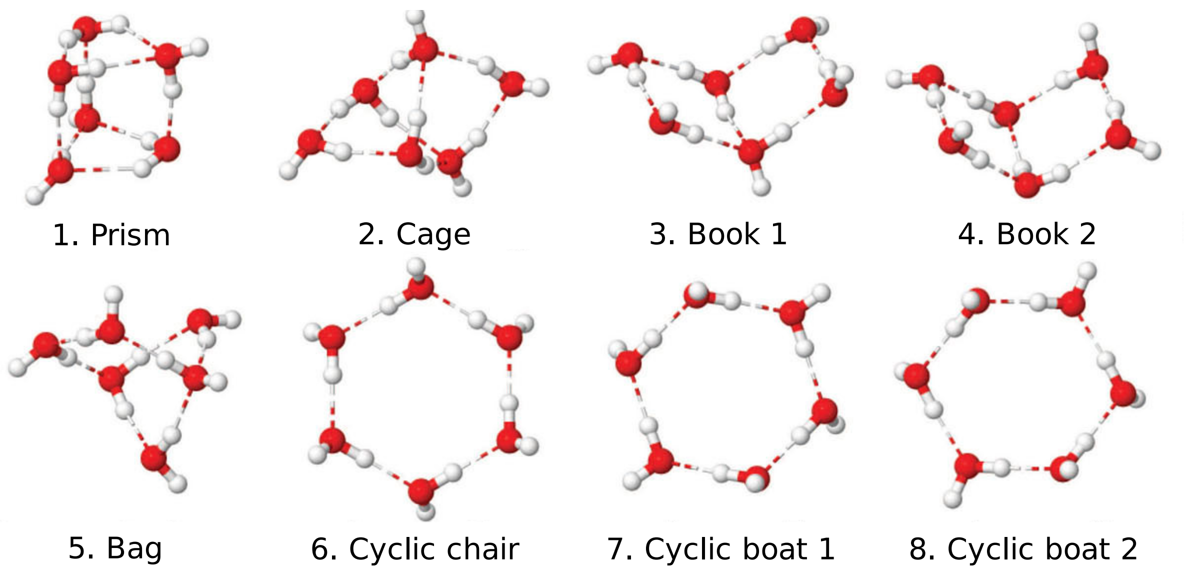


Figure 5.2. Structures of the first eight low-lying energy isomers of the water hexamer used in the analysis of interaction and many-body energies. The Cartesian coordinates of each isomer were taken from Ref. 167.

many-body interactions in water because it is the smallest cluster with low-lying isomers that display three-dimensional hydrogen-bonded arrangements similar to those found in liquid water and ice. The second set of clusters contains dimers and trimers extracted from the training sets used to fit the MB-pol 2B and 3B energy terms, respectively [7, 9].

5.3 Results

5.3.1 DNN potential

As a first step in assessing the ability of the DNN potential to reproduce MB-pol, we analyze various properties of liquid water. In Fig. 5.3a, we show the temperature dependence of the liquid density from 198 K to 368 K. In general, the DNN potential reproduces the MB-pol results over the entire temperature range, predicting similar temperatures for the density maximum and minimum. A more quantitative analysis indicates that the DNN potential underestimates the MB-pol density by $\sim 1\%$ in the $220-290$ K range while it predicts a slightly denser liquid as the temperature approaches the boiling point. The DNN curve also displays a less negative

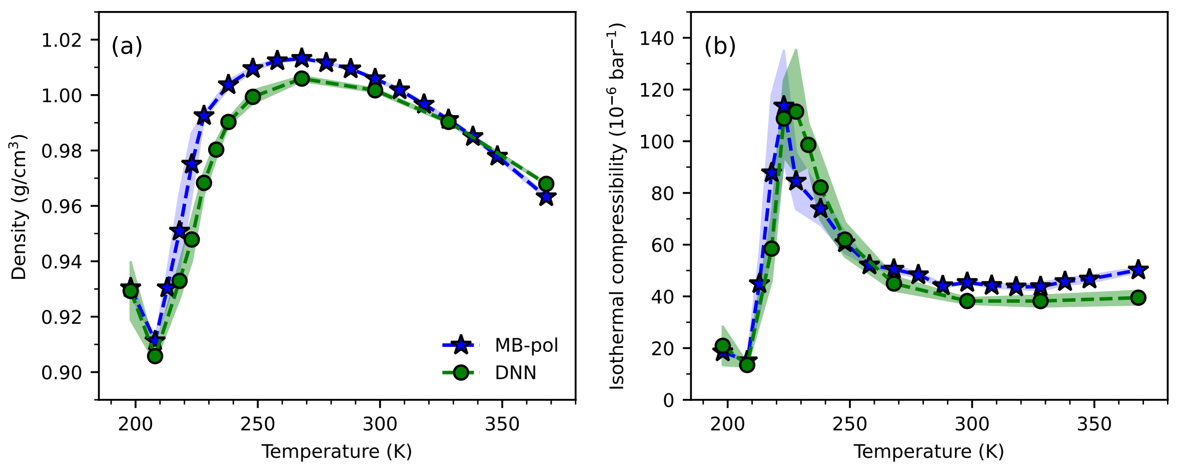


Figure 5.3. Temperature dependence of the density (a) and isothermal compressibility (b) calculated from the present NPT simulations carried out with the DNN potential at 1 atm (green) compared with the reference MB-pol values from Ref. 62 (blue). The associated shaded areas indicate 95% confidence intervals of the averages. Fig. S1 displays the time dependence of the densities calculated along the *NPT* trajectories carried out with the DNN potential at each temperature.

slope for temperatures above ∼320 K, which suggests that it is relatively more difficult for the DNN potential to reproduce MB-pol as the liquid properties become more gas-like. To put the present comparison between the MB-pol and DNN potentials in context, we note that the density of liquid water at 298 K predicted by an analogous DeePMD-based DNN potential trained on the SCAN functional was found to be ∼5% lower than the corresponding value calculated from the reference *ab initio* molecular dynamics (AIMD) simulations [157].

The comparison between the DNN and MB-pol values for the isothermal compressibility as a function of temperature is shown in Fig. 5.3b. Similar to the density, the DNN values are in agreement with the MB-pol reference data, reproducing the steep increase of the isothermal compressibility below 250 K and predicting a maximum at ∼230 K, which is ∼7 K higher than the temperature predicted by MB-pol [62]. As in the case of the liquid density, Fig. 5.3b also indicates that the ability of the DNN potential to reproduce MB-pol somewhat deteriorates as the temperature approaches the boiling point. In particular, the DNN potential predicts a nearly constant value of the isothermal compressibility above 300 K, with no indication of a distinct

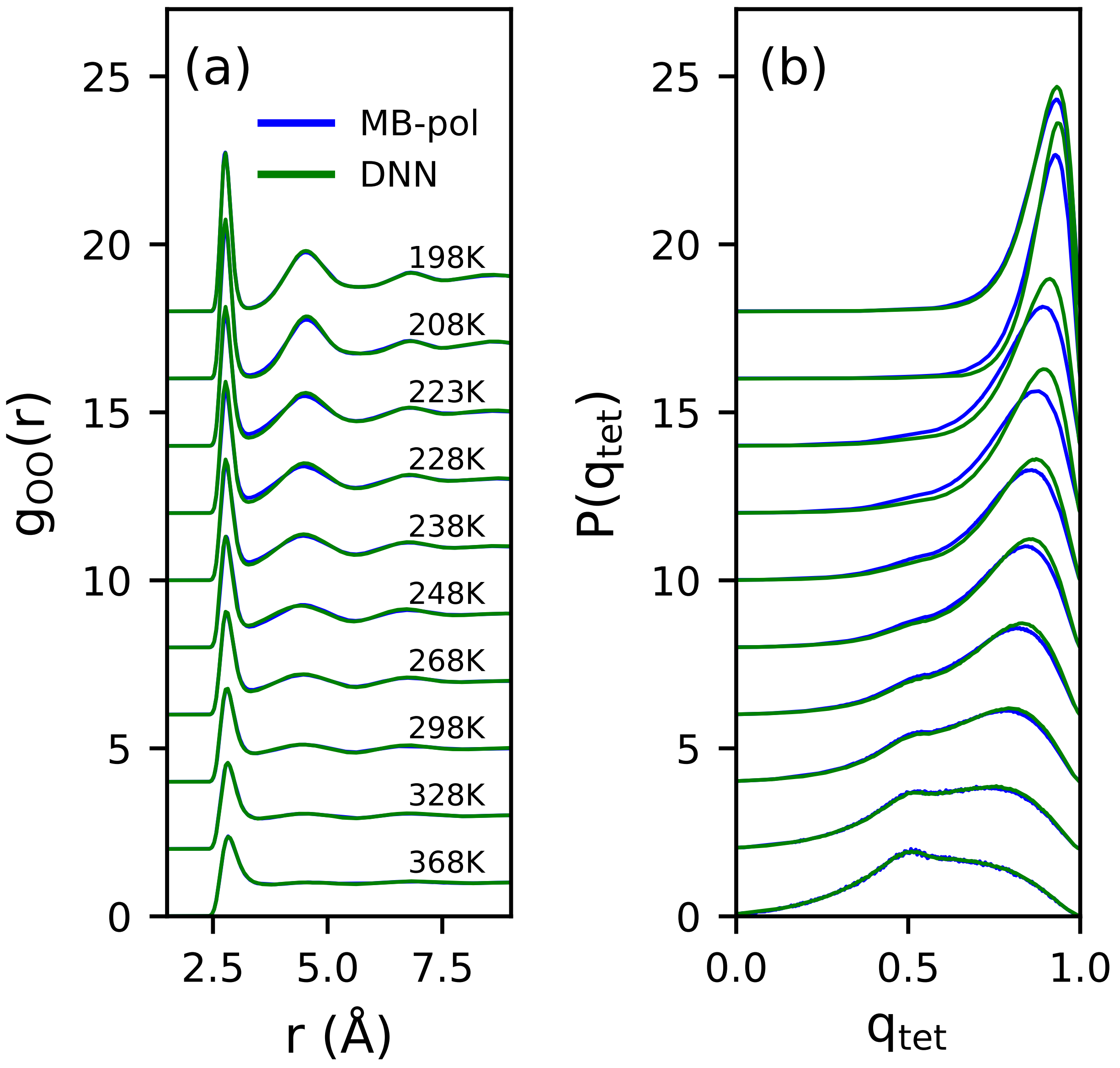


Figure 5.4. Oxygen-oxygen radial distribution functions and (b) tetrahedral order parameter distributions calculated from the present *NPT* simulations carried out with the DNN potential at 1 atm (green) compared with the reference MB-pol values from Ref. 62 (blue).

minimum that is instead found in both experiment[96, 100] and MB-pol simulations [62, 167].

A comparison between the structural properties of liquid water predicted by the DNN and MB-pol potentials between 198 K and 368 K at 1 atm is shown in Fig. 5.4. The oxygen-oxygen radial distribution functions (RDFs) calculated from the *NPT* simulations carried out with the two potentials (Fig. 5.4a) are nearly indistinguishable in the 238-368 K range. Small deviations are found at deeply supercooled temperatures, which become more apparent when analyzing the corresponding distributions of the tetrahedral order parameter in Fig. 5.4b. These deviations appear to be consistent with the shift of the isothermal compressibility maximum to a slightly higher temperature predicted by the DNN potential (Fig. 5.3b).

Previous studies demonstrated that MB-pol correctly predicts structural, thermodynamic, and spectroscopic properties of the vapor-liquid interface, including the surface tension, vapor pressure, vapor and liquid densities [137, 167], as well as sum-frequency generation spectra [129, 135]. To assess the ability of the DNN potential to reproduce properties that are not directly related to the MB-pol liquid configurations used during the training process, in Fig. 5.5, we analyze the surface tension and liquid-vapor equilibrium densities as a function of the temperature. These comparisons show that both surface tension and equilibrium densities predicted by the DNN potential deviate significantly from the corresponding MB-pol reference values as the temperature increases. Interestingly, while the liquid density predicted by the DNN potential decreases upon increasing temperature, in qualitative agreement with the expected physical behavior, the vapor density remains effectively constant over the entire temperature

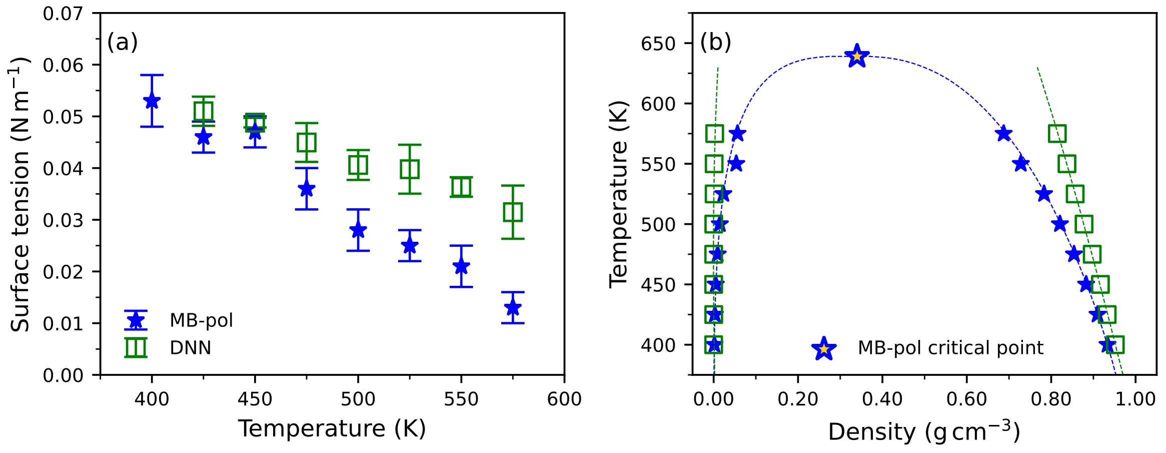


Figure 5.5. Surface tension (a) and vapor-liquid equilibrium densities (b) calculated from the present *NVT* simulations of a water slab carried out with the DNN potential (green) compared with the reference MB-pol values from Ref. 137 (blue).

range. Following Ref. 137, we estimated the critical temperature (T_c) and density (ρ_c) associated with DNN potential by fitting the vapor (ρ_v) and liquid (ρ_l) densities (Fig. 5.5) according to:

$$\frac{\rho_l + \rho_v}{2} = \rho_c + A(T_c - T) \tag{5.7}$$

$$\frac{\rho_l - \rho_v}{2} = \Delta\rho_0\,(1 - T/T_c)^{\beta} \tag{5.8}$$

Here, A and $\Delta\rho_0$ are system-specific parameters to be adjusted in the fitting and $\beta = 0.326$ is the critical exponent of the three-dimensional Ising model [249] . The DNN potential predicts $T_c = 857 \pm 17$ K and $\rho_c = 0.302 \pm 0.002$ g cm^{-3}, which are significantly different from the MB-pol values of $T_c = 639 \pm 14$ K and $\rho_c = 0.34 \pm 0.03$ g cm^{-3}. As a reference, the corresponding experimental values are $T_c = 647$ K and $\rho_c = 0.32$ g cm^{-3} [118].

In an attempt to rationalize the different performance of the DNN potential in reproducing bulk and interfacial properties calculated with MB-pol, we investigated the ability of the DNN potential to correctly describe many-body interactions. By construction, MB-pol quantitatively reproduces each term of the MBE (Eq. 5.1) calculated at the CCSD(T)/CBS level [7, 9]. In this context, we have shown that a correct representation of each individual n-body contribution to the interaction energies is required in order for a water model to be both accurate and transferable across different thermodynamic state points [45, 49, 109, 110, 127, 171].

Following previous studies [127, 167], in Fig. 5.6 we present a many-body decomposition analysis of the interaction energies of the first eight low-lying energy isomers of the hexamer cluster (Fig. 5.2). As mentioned in the Introduction, among water clusters, the hexamer occupies a special place because it is the smallest cluster with low-lying isomers that exhibit three-dimensional structures reminiscent of hydrogen-bonding arrangements found in liquid water and ice. In addition, the large number of isomers with similar interaction energies makes the hexamer the ideal benchmarking system for determining the accuracy of water models [45]. To provide a general perspective on DeePMD-based DNN potentials for water, in Fig. 5.6 we analyze

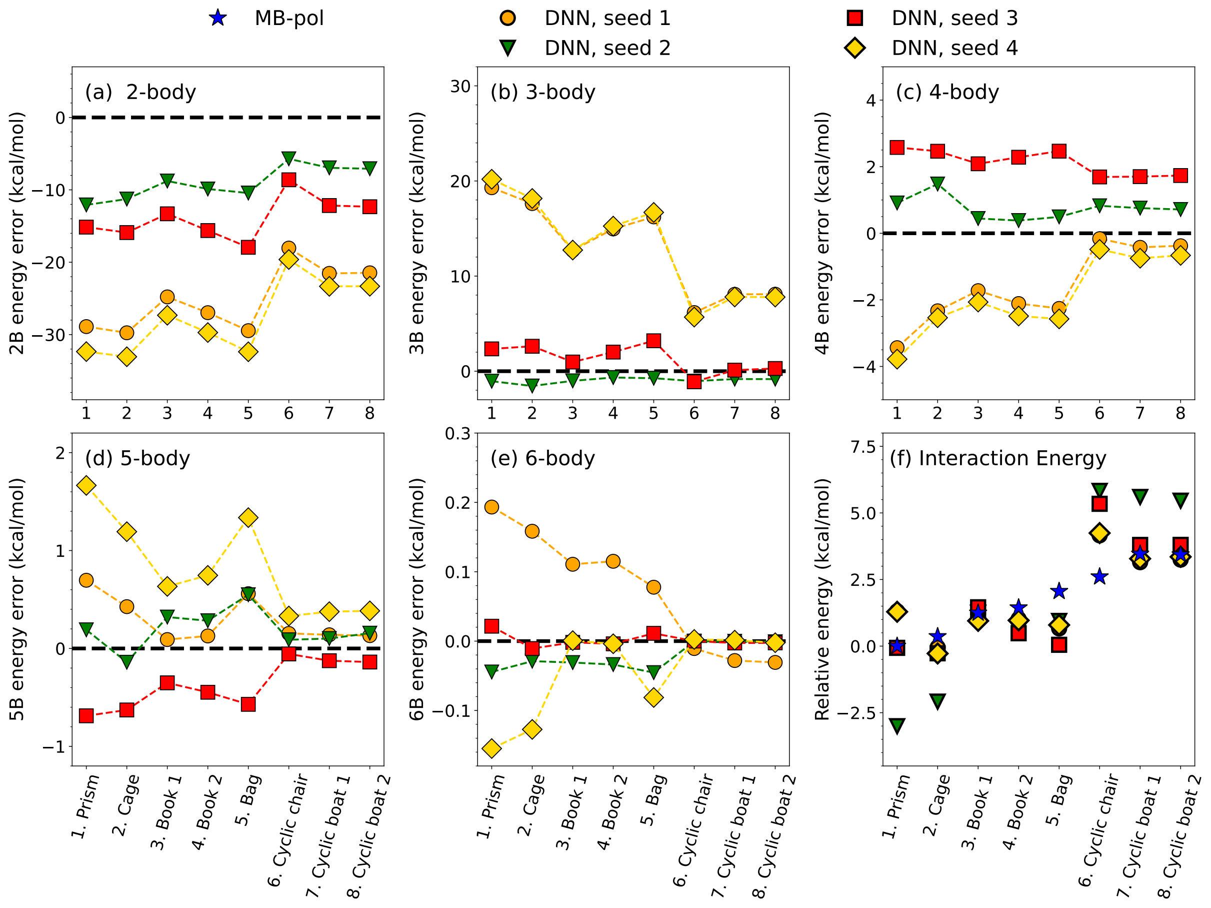


Figure 5.6. Many-body decomposition analysis for the eight low-lying energy isomers of the water hexamer (Fig. 5.2) calculated with four distinct DNN potentials trained on the same MB-pol training data using four different seeds to initialize the fitting process. Panels a) to e) show the errors associated with n-body energies ($n = 2-6$) calculated with the DNN potentials relative to the corresponding MB-pol values. Panel f) shows the errors associated with the interaction energies calculated with the DNN potentials relative to the corresponding MB-pol values. The DNN potential with seed 2 is used in the comparisons shown in Figs. 5.3-5.5.

the performance of four distinct DNN potentials trained on the same training set described in Section 5.2.2 but initialized using different random seeds, with seed 2 corresponding to the DNN potential used in Figs. 5.3-5.5.

All four DNN potentials provide statistically equivalent training, validation, and testing errors (see Tables S2 and S3 of the Supplementary Material). Fig. 5.6 shows that none of the four DNN potentials is capable of correctly reproducing individual n-body energies ($n = 2-6$) calculated with MB-pol, with significantly large errors, on the order of $10-20$ kcal/mol, for 2-

and 3-body contributions. Interestingly, these large errors compensate among different *n*-body contributions in such a way that, when the *n*-body energies are added together, they result in interaction energies that are in relatively better agreement with the MB-pol values than the individual *n*-body contributions. By definition, the interaction energies are calculated as the difference between the energy of the cluster and the sum of the 1-body energies of all six water molecules in the same distorted configurations as in the cluster. In this context, it should be noted that, besides MB-pol that reproduces the CCSD(T)/CBS reference energies of the hexamer isomers with chemical accuracy [167], several modern polarizable force fields predict *n*-body and interaction energies of water clusters with significantly higher accuracy than the four DNN potentials examined here [110]. Direct comparisons between *n*-body and interaction energies calculated with the four distinct DNN potentials and the corresponding MB-pol reference values are shown in Fig. S3. Importantly, Fig. S3 shows that, besides displaying large errors, some of the DNN potentials (i.e., seed 1 and seed 4) also predict physically incorrect many-body contributions (e.g., positive 3-body contributions), which indicates that, in their conventional implementation, DeePMD-based DNN potentials are not able to correctly disentangle individual many-body contributions to the interaction energy of a given water system. Importantly, the inclusion of long-range effects through a classical electrostatic term does not improve the description of many-body energies as shown in Figs. S4 and S5 of the Supplementary Material. It should be noted that this behavior is not specific to DeePMD-based DNN potentials but appears to be common to other neural network potentials. For example, Figs. S6 and S7 of the Supplementary Material show that similar behavior is exhibited by Nequip-based potentials[16] trained on MB-pol. Interestingly, the Nequip-based potentials demonstrate superior accuracy in predicting the interaction energies of the water clusters, but also exhibit larger error compensation among different *n*-body energies, with errors on 2- and 3-body energies being as large as $20 - 30$ kcal/mol.

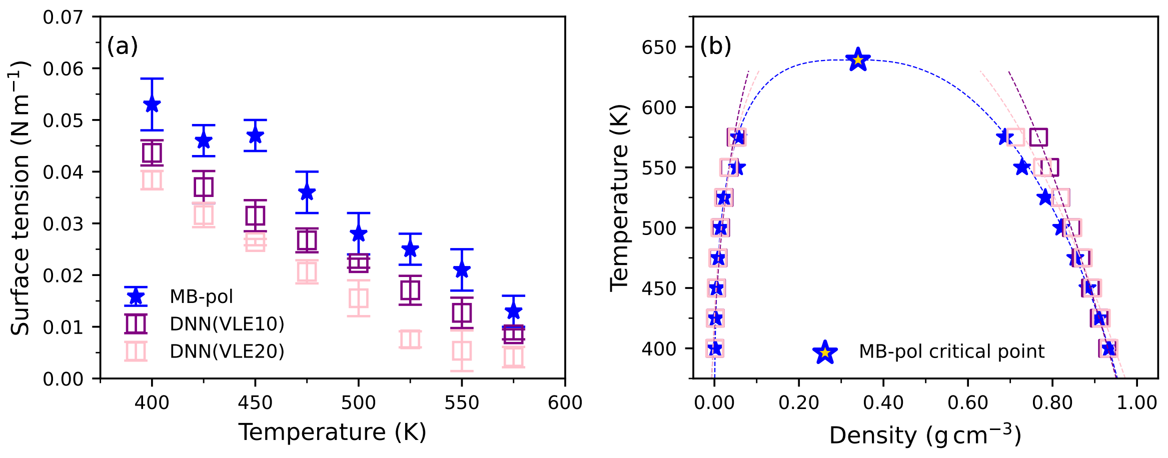


Figure 5.7. Surface tension (a) and vapor-liquid equilibrium densities (b) calculated from the present *NVT* simulations of a water slab carried out with the DNN(VLE10) (purple) and DNN(VLE20) (pink) compared with the reference MB-pol values from Ref. 137 (blue).

5.3.2 DNN(VLE) potential

In an attempt to improve the performance of the DNN potential on vapor-liquid equilibrium properties, we used active learning to incorporate vapor-liquid configurations extracted from simulations carried out with the DNN potential in the temperature range between 268 K and 575 K. At the end of the active learning process, 2412 were added to the training set. The expanded training set was then used to train two potentials, DNN(VLE10) and DNN(VLE20), with a 10% and 20% probability of selecting vapor-liquid configurations during training, respectively. Fig. 5.7 shows that adding vapor-liquid configurations leads to more accurate predictions of both surface tension and vapor-liquid equilibrium densities. In particular, compared to the results obtained with the DNN potential, the surface tension predicted by both DNN(VLE10) and DNN(VLE20) shows the same temperature dependence as determined by MB-pol, although a systematic deviation from the reference values is still observed at all temperatures. Similarly, adding vapor-liquid configurations to the training set improves the ability of the DNN(VLE10) and DNN(VLE20) potentials to describe the equilibrium densities of both vapor and liquid phases. While both DNN(VLE10) and DNN(VLE20) potentials quantitatively reproduce the

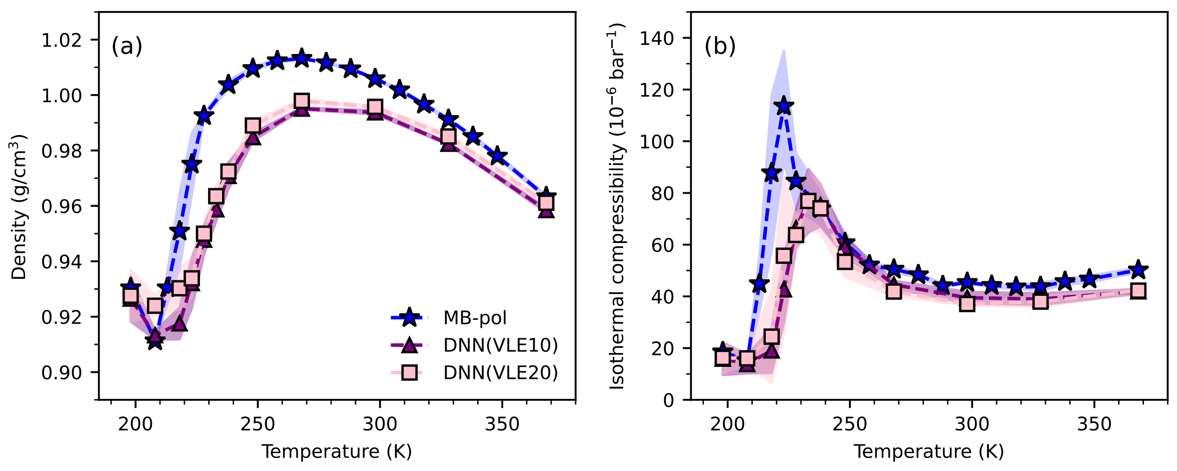


Figure 5.8. Temperature dependence of the density (a) and isothermal compressibility (b) calculated from *NPT* simulations carried out with the DNN(VLE10) (purple) and DNN(VLE20) (pink) potentials at 1 atm. Also shown for reference are the corresponding MB-pol values from Ref. 62 (blue). The associated shaded areas indicate 95% confidence intervals of the averages.

MB-pol vapor densities over the entire temperature range, the predicted liquid densities, however, increasingly deviate from the MB-pol reference values as the temperature increases. As a consequence, the critical point is still overestimated by both potentials, with DNN(VLE10) predicting $T_c = 718 \pm 4$ K and $\rho_c = 0.359 \pm 0.003$ g cm^{-3} and DNN(VLE20) predicting $T_c = 674 \pm 6$ K and $\rho_c = 0.347 \pm 0.005$ g cm^{-3}. Adding vapor-liquid configurations to the training set was reported to enable simulations of "water along its liquid/vapor coexistence line with unprecedented precision" [230]. Inspection of Fig. 3 of Ref. 230, however, indicates that relatively large deviations (similar to those found for DNN(VLE10) and DNN(VLE20) in Fig. 5.7) exist between the vapor and liquid densities predicted by the neural network potential used in the simulations and the corresponding reference RPBE-D3 values which, when extrapolated, lead to very different estimates for the critical point [180].

To assess the ability of DNN(VLE10) and DNN(VLE20) to reproduce properties that do not directly depend on the coexistence between vapor and liquid phases, we examined the performance of both potentials on the same bulk and cluster properties used in Section 5.3.1 to determine the accuracy of the DNN potential. Fig. 5.8 shows the temperature dependence of the density and isothermal compressibility of liquid water predicted by DNN(VLE10) and DNN(VLE20). While both potentials are able to qualitatively reproduce the MB-pol trends, a comparison with the DNN results reported in Fig. 5.3 indicates that the inclusion of vapor-liquid configurations to the training set deteriorates the ability of the DeePMD-based potentials to reproduce the bulk properties. This is further confirmed by the analyses of the liquid density, RDFs, and q_{tet} distributions shown for the DNN(VLE20) potential in Figs. S17 and S18 of the Supplementary Material.

Finally, Fig. 5.9 reports the many-body decomposition analysis of the interaction energies of the hexamer isomers (Fig. 5.2) carried out with the DNN(VLE20) potential. The corresponding analysis carried out with DNN(VLE10) is reported in the Supplementary Material in Fig. S8. As for the DNN potential, we used four different seeds to develop four distinct DNN(VLE20) potentials that were trained on the expanded MB-pol training set containing vapor-liquid configurations. Seed 4 corresponds to the DNN(VLE20) potential used in the comparisons shown in Figs. 5.7 and 5.8. As in the case of DNN in Fig. 5.6, none of DNN(VLE20) potentials is able to correctly reproduce the reference MB-pol many-body energies, with errors that are on the order of $\sim$10 kcal/mol for 2-, 3-, and 4-body energies. Similar poor performance on the many-body decomposition analysis is exhibited by the DNN(VLE10) potential in Fig. S8 of the Supplementary Material. Analyses analogous to those shown in Fig. S3 for the DNN potential are reported in Figs. S9 and S10 for the DNN(VLE10) and DNN(VLE20) potentials, which lead to similar conclusions, i.e., both DNN(VLE10) and DNN(VLE20) predict physically incorrect many-body energies.

The analyses presented in this section demonstrate that, while the description of vapor-liquid equilibrium properties can be improved by adding vapor-liquid configurations to the original DNN training set, this improvement is achieved at the cost of a less accurate representation of the bulk properties. Importantly, as in the case of the DNN potentials, the DNN(VLE10) and DNN(VLE20) potentials are unable to correctly capture the physics of many-body interactions in water.

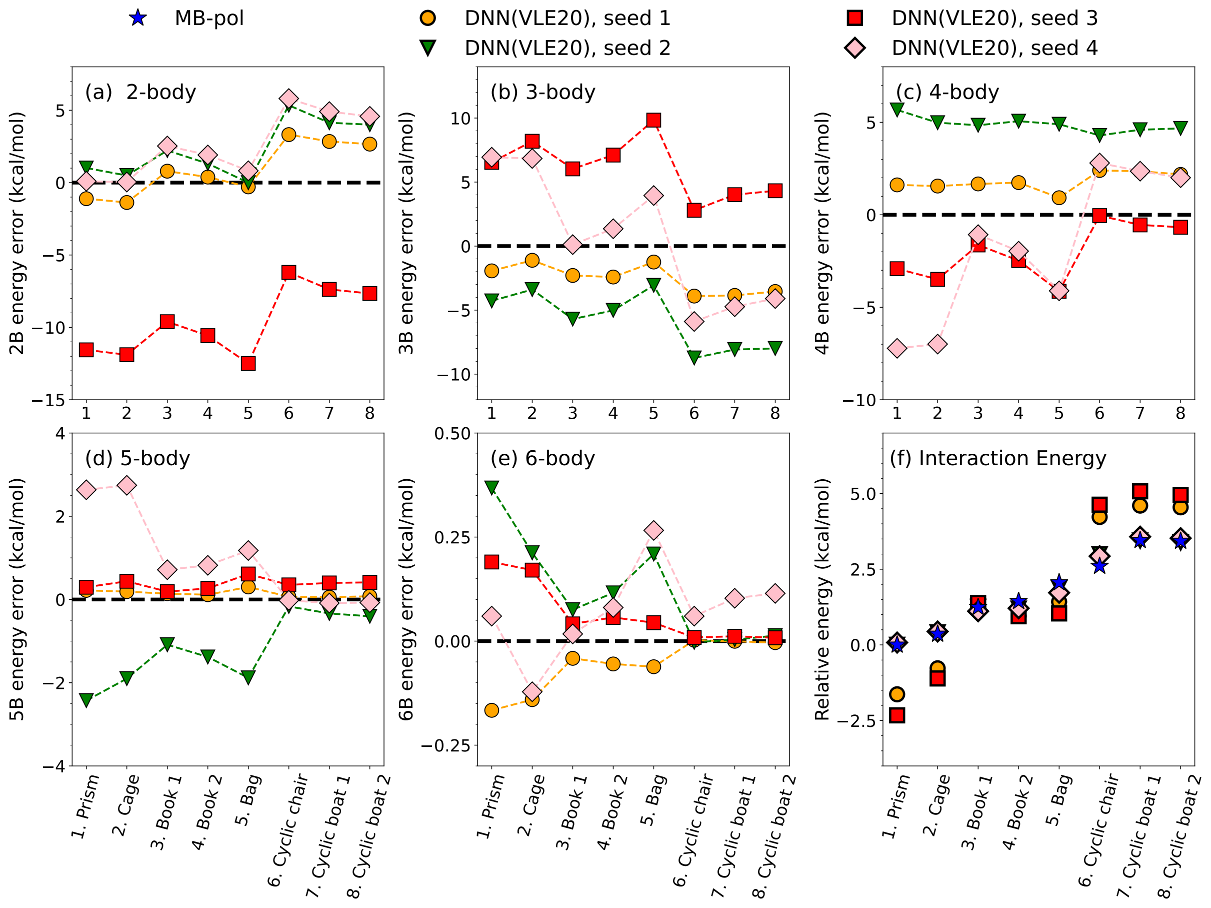


Figure 5.9. Many-body decomposition analysis for the eight low-lying energy isomers of the water hexamer (Fig. 5.2) calculated with four distinct DNN(VLE20) potentials that were trained on the expanded MB-pol training set containing vapor-liquid configurations using four different seeds to initialize the fitting process. Panels a) to e) show the errors associated with n-body energies ($n = 2 - 6$) calculated with the DNN(VLE20) potentials relative to the corresponding MB-pol values. Panel f) shows the errors associated with the interaction energies calculated with the DNN(VLE20) potentials relative to the corresponding MB-pol values. The DNN(VLE20) potential with seed 4 is used in the comparisons shown in Figs. 5.7 and 5.8.

5.3.3 DNN(MB) potential

Since the inability of a water model to correctly represent many-body contributions to the underlying molecular interactions appears to be correlated with the lack of transferability of the model across different thermodynamic state points [45], we investigated the possibility of "encoding" many-body effects in the DNN potentials within the DeePMD framework. To this end, we supplemented the original training set used for developing the DNN potentials discussed in Section 5.3.1 with a set of gas-phase clusters, including monomers, dimers, trimers, and tetramers which provides direct information about the low-order and most important terms (i.e., 1-body to 4-body terms) of the MBE in Eq. 5.1. We then used the expanded training set to train three different DeePMD-based potentials, referred to as DNN(MB4), DNN(MB10), and DNN(MB20), with 4%, 10%, and 20% probability of selecting gas-phase cluster configurations during the training process, respectively. Specific details about the composition of the extended training set are provided in Section S1 of the Supplementary Material.

Fig. 5.10 reports the same analysis reported in Fig. 5.6 for the DNN potential and shows the errors relative to the MB-pol reference values for *n*-body and interaction energies of the first eight isomers of the water hexamer (Fig. 5.2) calculated with four distinct versions of the DNN(MB20) potential which were trained on the same expanded training set but initialized with four distinct seeds. Seed 4 corresponds to the DNN(MB20) potential used in the comparisons shown in Figs. 11 and 12. Analogous analyses carried out with the DNN(MB4), and DNN(MB10) potentials are reported in Figs. S11 and S12 of the Supplementary Material, respectively. The addition of monomer, dimer, trimer, and tetramer configurations clearly allows the DNN(MB) potentials to become "aware" of the existence of distinct many-body contributions to the interactions energies, as demonstrated by the relatively smaller errors displayed by the DNN(MB20) 2-body, and 3-body energies compared to the corresponding errors associated with the DNN potentials in Fig. 5.6. Similar trends are observed in Figs. S13, S14, and S15 that show direct comparisons of individual many-body energies and interaction energies calculated with the DNN(MB4), DNN(MB10), and DNN(MB20) potentials, respectively. Additional analyses of the error distributions associated with 2-body and 3-body energies calculated for dimer and trimer configurations of the cluster training set are reported in Fig. S16 and demonstrate that the DNN(MB4), DNN(MB10), and DNN(MB20) potentials significantly improve on the DNN potential in the ability to represent many-body interactions in water. It should, however, be noted that the DNN(MB) potentials still rely on significant error compensation among individual *n*-body energy contributions to minimize the error on the interaction energies of the hexamer

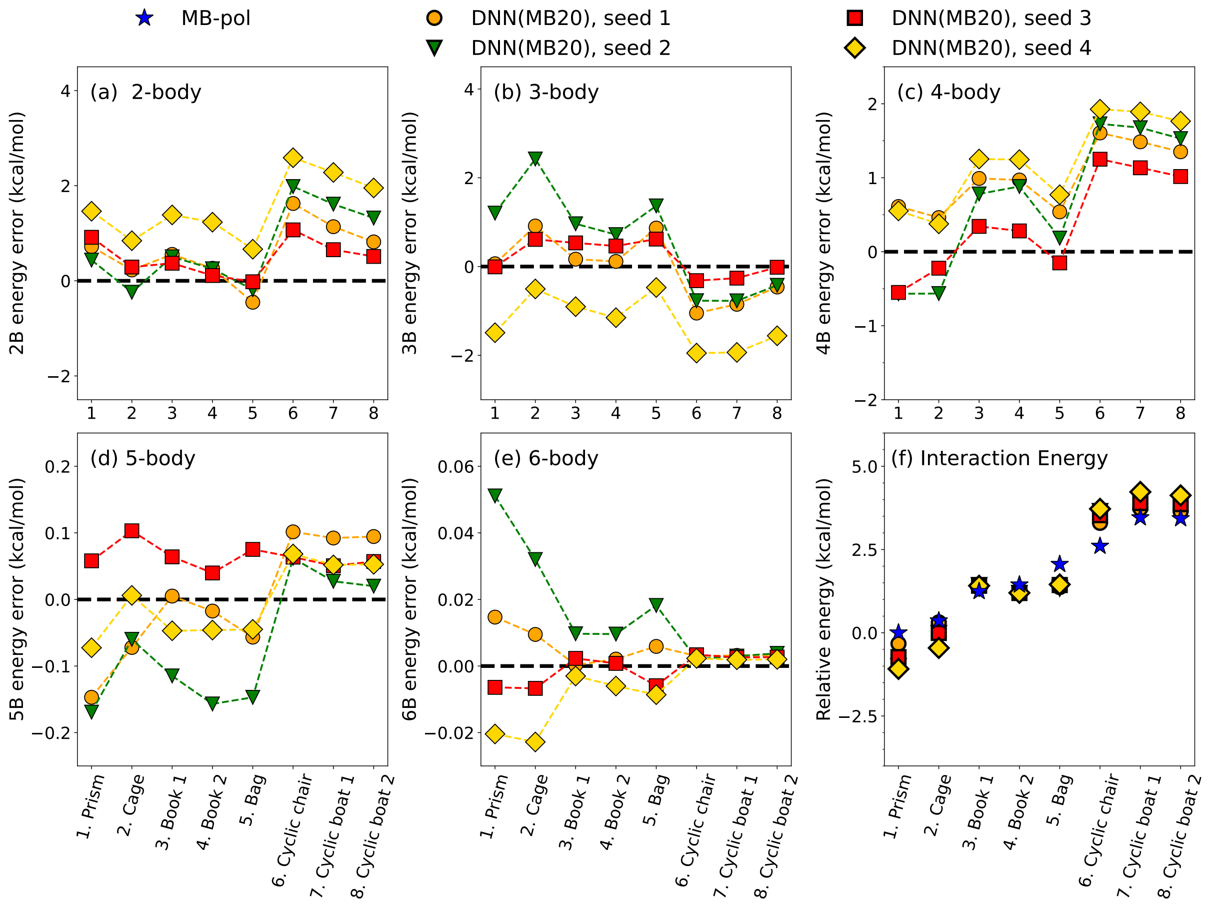


Figure 5.10. Many-body decomposition analysis for the eight low-lying energy isomers of the water hexamer (Fig. 5.2) calculated with four distinct DNN(MB20) potentials that were trained on the expanded MB-pol training set containing cluster configurations using four different seeds to initialize the fitting process. Panels a) to e) show the errors associated with n-body energies ($n = 2 - 6$) calculated with the DNN(MB20) potentials relative to the corresponding MB-pol values. Panel f) shows the errors associated with the interaction energies calculated with the DNN(MB20) potentials relative to the corresponding MB-pol values. The DNN(MB20) potential with seed 4 is used in the comparisons shown in Figs. 5.11 and 5.13.

isomers. The observed error compensation indicates that, as the DNN, DNN(VLE10), and DNN(VLE20) potentials, the DNN(MB) potentials are unable to “learn” that the interaction energy of a N-body system containing N water molecules is given by the sum of distinct n-body energy contributions (with $n = 2 - N$). To put things in perspective, the errors associated with the DNN(MB) predictions for each n-body energy contribution to the interaction energies of the hexamer isomers, in particular at the 2-body and 3-body levels, are still appreciably larger than those displayed by state-of-the-art polarizable force fields for water [110].

Having demonstrated that extending the training set by adding monomer, dimer, trimer, and tetramer configurations allows the DNN(MB) potentials to recover a more balanced representation of many-body interactions, we next assess the ability of the DNN(MB4), DNN(MB10), and DNN(MB20) potentials to reproduce vapor-liquid equilibrium properties that were poorly predicted by the DNN potentials (Fig. 5.5). Fig. 5.11 shows that all three DNN(MB) potentials more closely reproduce the MB-pol trends for the surface tension and the equilibrium densities of both vapor and liquid phases over the entire temperature range examined in this study than the DNN potential. The critical parameters predicted by the DNN(MB4), DNN(MB10), and DNN(MB20) potential are $T_c = 655 \pm 2$ K and $\rho_c = 0.325 \pm 0.002$ g cm^{-3}, $T_c = 605 \pm 10$ K and $\rho_c = 0.32 \pm 0.01$ g cm^{-3}, and $T_c = 660 \pm 6$ K and $\rho_c = 0.338 \pm 0.005$ g cm^{-3}, respectively, which are in better agreement with the MB-pol values ($T_c = 639 \pm 14$ K and $\rho_c = 0.34 \pm 0.03$ g cm^{-3}) than the results obtained not only with the DNN potential but also with the DNN(VLE10) and DNN(VLE20) potentials. The structural differences at the vapor-liquid equilibrium between DNN and DNN(MB20) are further highlighted in Fig. S17 which shows the density profiles predicted by the two potentials at different temperatures. In particular, the interface structure predicted by DNN(MB20) is significantly different from that predicted by DNN and in close agreement with the MB-pol results reported in Ref. 137.

Despite being able to provide more accurate estimates of the actual MB-pol critical point, Fig. 5.12 shows that the DNN(MB4), DNN(MB10), and DNN(MB20) potentials display a higher degree of variability in their predictions of the liquid density at high temperatures than the DNN(VLE10) and DNN(VLE20) potentials. This high variability at high temperatures, which is also displayed by the DNN potentials, can be traced back to the lack of explicit vapor-liquid configurations in the corresponding training sets, which, in turn, highlights the difficulty for DeePMD-based potentials to be transferable across different phases over a wide range of thermodynamic conditions.

The last question that remains to be addressed is whether the improved ability to rep-

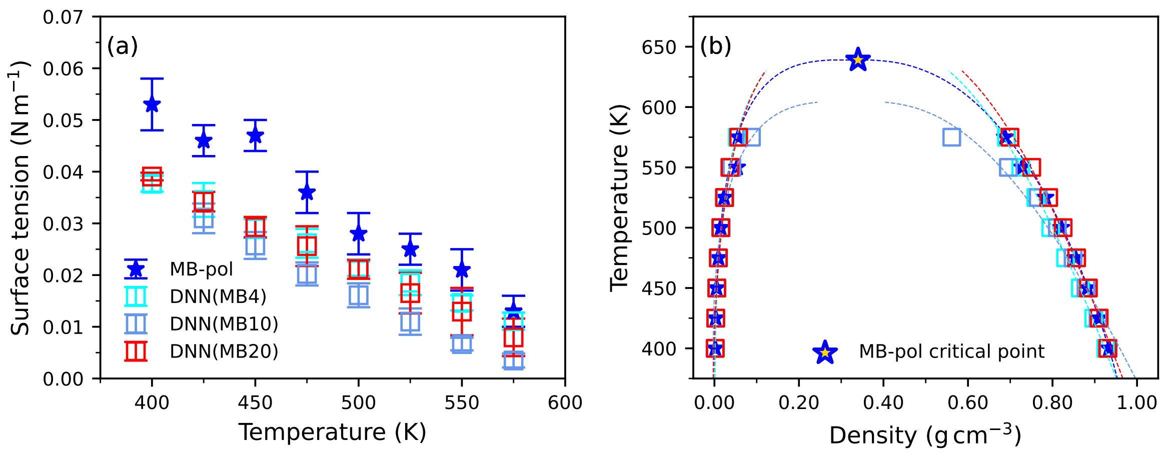


Figure 5.11. Surface tension (a) and vapor-liquid equilibrium densities (b) calculated from *NVT* simulations of a water slab carried out with the DNN(MB) potentials (cyan, light blue, red) compared with the reference MB-pol values from Ref. 137 (blue).

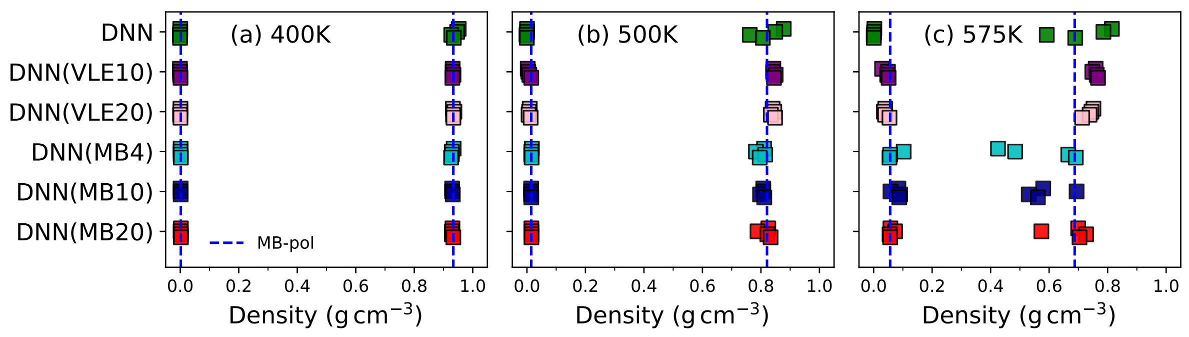


Figure 5.12. Vapor-liquid equilibrium densities calculated using the same four variants of each of the DNN, DNN(VLE), and DNN(MB) potentials used in the analyses of Fig. 5.6, 5.9, S8, 5.10, S11, and S12. The densities are compared with the reference MB-pol values from Ref. 137 (blue dashed line) at 400 K (panel a), 500 K (panel b), and 575 K (panel c).

resent many-body interactions and predict vapor-liquid equilibrium properties still allows the DNN(MB4), DNN(MB10), and DNN(MB20) potentials to accurately reproduce the liquid properties calculated with MB-pol. To this end, Fig. 5.13 shows comparisons between the temperature dependence of the density and isothermal compressibility calculated with the DNN(MB4), DNN(MB10), and DNN(MB20) potentials and the corresponding MB-pol reference values. The DNN(MB4), DNN(MB10), and DNN(MB20) potentials effectively predict indistinguishable (within statistical error) trends for both density and isothermal compressibility, which are in

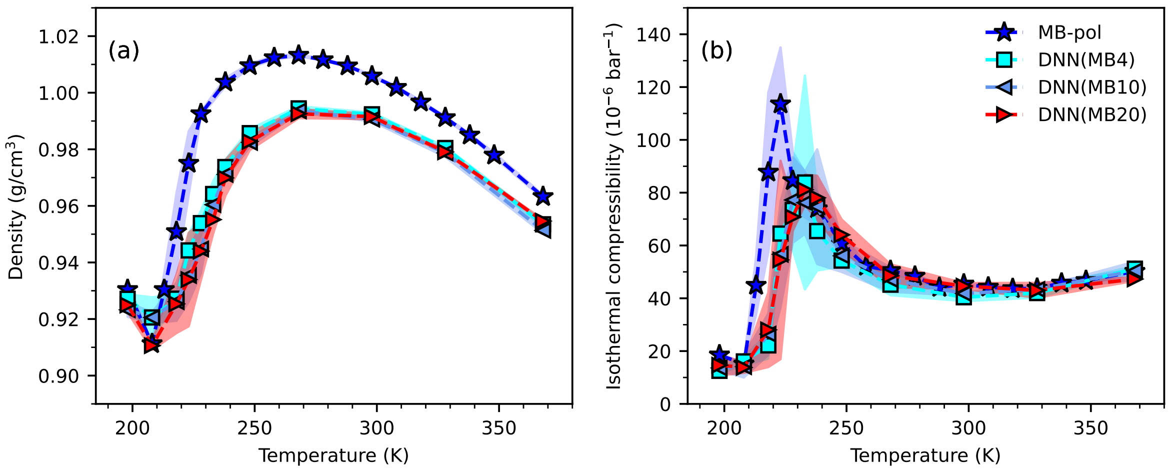


Figure 5.13. Temperature dependence of the density (a) and isothermal compressibility (b) calculated from *NPT* simulations carried out with the DNN(MB) potentials at 1 atm (cyan, light blue, red) compared with the reference MB-pol values from Ref. 62 (blue). The associated shaded areas indicate 95% confidence intervals of the averages. Fig. S2 show the density fluctuations along the *NPT* trajectories carried out with the DNN(MB20) potentials at each temperature.

qualitative agreement with the MB-pol reference values. Specifically, they correctly predict a minimum at ~300 K in the isothermal compressibility which is instead absent in the DNN, DNN(VLE10), and DNN(VLE20) potentials. This suggests that, by being able to more accurately represent many-body interactions, the DNN(MB4), DNN(MB10), and DNN(MB20) potentials display a higher degree of transferability to the gas phase at ambient conditions. As in the case of the DNN(VLE10) and DNN(VLE20) potentials, the comparison between the results of Fig. 5.13 and Fig. 5.3, however, indicates that the addition of configurations different from bulk configurations (in this case, monomer, dimer, trimer, and tetramer configurations) to the training set overall deteriorates the ability of the DNN(MB4), DNN(MB10), and DNN(MB20) potentials to reproduce bulk properties calculated with MB-pol. Despite these differences, the liquid structure predicted by the DNN(MB4), DNN(MB10), and DNN(MB20) potentials is in close agreement with that of MB-pol as demonstrated by the comparisons of the RDFs and q_{tet} distributions calculated with DNN(MB20) that are shown in Fig. S18 of the Supplementary Material.

5.4 Conclusion

In this study, we analyzed the performance and degree of transferability of a DeePMD-based DNN potential for water trained on MB-pol reference configurations extracted from MD simulations of liquid water carried out from 198 K to 368 K at 1 atm. We found that the DNN potential is able to reliably reproduce structural and thermodynamic properties of liquid water as predicted by MB-pol from the boiling point down to deeply supercooled temperatures. However, while MB-pol exhibits remarkable accuracy from the gas to the condensed phase, the DNN potential does not share the same high level of transferability across phases. In particular, we found that the DNN potential is not able to accurately describe vapor-liquid equilibrium properties. More importantly, a many-body decomposition analysis of the interaction energies of the hexamer isomers indicates that the DNN potential is not able to correctly "learn" many-body interactions and effectively relies on error compensation among individual many-body energy contributions to reproduce the interaction energy of a given N-body system containing N water molecules.

To improve the performance of the DNN potential on vapor-liquid equilibrium properties, we expanded the initial DNN training set of bulk configurations by adding configurations extracted from vapor-liquid equilibrium simulations carried out with MB-pol. While the new DNN(VLE10) and DNN(VLE20) potentials improve the description of the surface tension and equilibrium densities of both vapor and liquid phases, they predict less accurate bulk properties and are unable to correctly reproduce individual many-body contributions to the interaction energies.

In an attempt to explicitly encode many-body interactions onto the DNN potential, we also expanded the initial DNN training set by adding water monomer, dimer, trimer, and tetramer configurations, which provide direct information on the most important many-body contributions (i.e., 1-body to 4-body contributions) to the interaction energies in water systems. By improving the description of individual many-body contributions, the new DNN(MB4), DNN(MB10), and DNN(MB20) potentials are also able to reliably reproduce the vapor-liquid equilibrium properties predicted by MB-pol. We found, however, that all three potentials exhibit a high degree of variability in predicting the liquid density at high temperatures due to the lack of representative vapor-liquid configurations in their training sets, which limits their transferability over a wide range of thermodynamic conditions. Moreover, the improvement in the description of many-body interactions comes at the expense of a poorer representation of the liquid properties.

Although DeePMD-based potentials are intrinsically many-body in their functional form, our analyses show that they do not necessarily correctly represent the underlying many-body physics of the reference potentials. This suggests that some caution should be exercised when using DeePMD-based DNN potentials to predict thermodynamic properties for state points that are not explicitly and thoroughly included in the training sets. Although our study focuses on water, similar behavior is likely to be found in DeePMD-based DNN potentials for other molecular systems, including aqueous solutions as well as molecular fluids and solids. In this context, we hope that our results can stimulate further developments of new training procedures and neural network architectures capable of correctly capturing the physics of many-body interactions in molecular systems.

With this caveat in mind, the computational efficiency provided by the DeePMD framework suggests that large-scale CCSD(T)-level MD simulations are possible by training DeePMD-based DNN potentials on data-driven many-body potentials derived from the MBE calculated at the CCSD(T) level of theory, such as MB-pol. However, for this to hold, the thermodynamic state points of interest in the DNN simulations must be adequately represented in the training sets generated using the reference data-driven many-body potentials. This suggests that a DNN potential trained on an extensive training set, including molecular configurations extracted from MB-pol simulations carried out over a wide range of thermodynamic conditions, is well suited for exploring the rich phase diagram of water [26], particularly in the so-called “no man’s land” region at low temperature, which has been proven difficult to probe experimentally [99, 100, 107].

5.5 Acknowledgements

Chapter 5, in full, is a reprint of the material as it appears in “A “short blanket” dilemma for a state-of-the-art neural network potential for water: Reproducing properties or learning the underlying physics?” Y. Zhai, A. Caruso, S.L. Bore, Z. Luo, F. Paesani, in *The Journal of Chemical Physics*, 2023, 158(8). The dissertation author is the primary investigator and author of this paper.

We thank Maria Muniz, Athanassios Panagiotopoulos, and Vinicius Cruzeiro for stimulating discussions at the early stage of this research. This research was supported by the Air Force Office of Scientific Research under award FA9550-20-1-0351 and used computational resources of the Department of Defense High Performance Computing Modernization Program (HPCMP) as well as the Triton Shared Computing Cluster (TSCC) at the San Diego Supercomputer Center (SDSC). This work was supported in part by the UC Southern California Hub, with funding from the UC National Laboratories division of the University of California Office of the President.

Chapter 6

Conclusions

In this dissertation, we explore the application of Monte Carlo tree methods for non-linear optimizations, including black-box optimization and non-convex optimization, and the development of many-body potentials.

As shown from the above studies, Monte Carlo tree methods have been demonstrated to offer cost-effective solutions for navigating the complexities of state space. The introduced MCTD and MCIR frameworks utilize these sampling approaches combined with MCTS to address global optimization challenges. The MCTD framework emphasizes sample-efficiency at local descent, incorporating stochastic search and Gaussian Processes to generate surrogate objectives, and to provide uncertainty metrics for exploration. Conversely, MCIR applies numerical overapproximation of the objective function as heuristics and leverages first- and second-order information estimates to identify promising samples. These methodologies facilitate a deeper understanding of the state space via collected samples to guide the selection of new samples by balancing space exploitation and exploration.

In another work, an active learning framework is examined for its efficacy in creating training sets for PEFs for water molecule simulations. A case study on the development of a Cs^+-water PEF demonstrates the framework's ability to significantly reduce training set sizes without compromising the quality of the original PEF, highlighting the potential for efficient state space exploration with minimal information. We also analyzed the performance and degree of transferability of a DeePMD-based DNN potential for water trained on configurations extracted from MD simulations of liquid water. We found that the DNN potential is able to reliably reproduce structural and thermodynamic properties of liquid water from the boiling point down to deeply supercooled temperatures. However, the DNN potential does not share the same high level of transferability across phases compared to the reference.

Despite the advantages of Monte Carlo tree methods, challenges persist. A key concern lies in the tree management, particularly the need for effective and efficient pruning of branches to keep the tree at an optimal size. The challenge arises as the tree grows larger and the overhead associated with managing it becomes significant, requiring a pruning technique that can precisely identify the essential nodes.

Moreover, while samples can illustrate aspects of the state space, achieving a comprehensive evaluation of this space is particularly challenging in high-dimensional scenarios. This challenge is compounded by the risk of information loss with limited exploration, as the representativeness of a tree node can diminish in spaces that expand exponentially with dimensionality. The information on the node, which ideally reflects the landscape of the objective function, may not be as robust or informative in vast and complex spaces, highlighting a critical area for improvement in sampling methodologies.

Thirdly, the efficiency of optimization is closely tied to the choice of hyperparameters within the selected models and algorithms. Inappropriate hyperparameter settingscan negatively impact the quality of the optimization. Yet, determining the optimal set of hyperparameters is itself an optimization problem demanding thorough investigation. Consequently, methods that can analyze the characteristics of the function's landscape and adaptively choose suitable parameters in real time can significantly enhance the performance.

Finally, identifying the embedding features that govern molecular interactions continues to be a complex task in machine learning models. These features should encapsulate the fundamental physics governing the interaction between components, rather than merely fitting the data within a limited subset of the interaction space. The identification of such features requires in-depth studies in both the domains of physics and machine learning modeling.

Appendix A

A "Short Blanket" Dilemma for the Deep Neural Network-Based Many-Body Potentials

A.1 Details about the DNN potentials and training procedure

The initial training set consisted of potential energies and forces of configurations extracted from the MB-pol simulations reported in Ref. 62. The DNN potentials were then used in molecular dynamics (MD) simulations carried out in the isothermal-isobaric (NPT) ensemble at a pressure of 1 atm and temperatures between 198 and 368 K for a cubic box containing 256 molecules in periodic boundary conditions. Three successive iterations were performed to refine each DNN potential. At each iteration, new configurations were extracted from the MD simulations and added to the training set in an iterative process in order to enhance the stability of the DNN potential and overcome similar instabilities to those observed in Ref. 245. As in Ref. 245, the new configurations were filtered in order to retain only poorly predicted configurations with $\Delta F >= 5$ eV/A and $|F| < 200$ eV/A. Since well predicted configurations ($\Delta F < 5$ eV/A) and highly distorted configurations ($|F| >= 200$ eV/A or $|E| >= 50000$ eV) were found not to improve the quality of the resulting DNN potentials, they were disregarded. After three iterations, the final training set resulted in a total of 94,770 configurations, each containing 256 molecules.

As part of our effort to provide insights into how the model works with additional information from VLE simulations, we performed successive iterations to add additional configurations. Poorly predicted configurations from the VLE simulations were filtered analogously to DNN and added to the original dataset. Finally, 2412 VLE configurations were added to the original DNN training set and two new potentials were generated by sampling vapor-liquid configurations with probabilities of 10% and 20% during the training process. These two potentials are referred to as DNN(VLE10) and DNN(VLE20), respectively.

The DNN(MB) potentials use the same training set as the DNN potentials with the difference that additional monomer, dimer, trimer, and tetramer configurations were added in order to provide more direct information about the low-order terms of the many-body expansion of the energy. Specifically, the monomer configurations were taken from Ref. 108, while the dimers and trimer configurations were taken from the training sets that were used to fit the MB-pol 2-body [7] and 3-body [9] energy terms, respectively. The tetramer configurations were extracted from the *NPT* simulations carried out with MB-pol at 1 atm and 298 K in Ref. 62. All tetramer configurations with binding energies higher than 15 kcal/mol were filtered out. To ensure the correct weighting of gas-phase cluster configurations, we trained three distinct DNN(MB) potentials by sampling cluster configurations with overall probabilities of 4%, 10%, and 20%, respectively, during the training process (i.e., the monomer, dimer, trimer, and tetramer training sets were each sampled with a probability of 1%, 2.5%, and 5% during the training of the three DNN(MB) potentials, respectively). These three potentials are referred to as DNN(MB4), DNN(MB10), and DNN(MB20), respectively.

The composition of the sets is reported in Table A.1. The dataset was split into training, validation, and test sets with a ratio of 8:1:1. Each DNN potential was trained for 4 million epochs, with a learning rate starting at 0.0005 and decreasing linearly every 5000 epochs to 1.8×10^{-8}. The initial weighting factor p_e for the energy (Eq. 5 of the main text) was set to 0.2 and increased linearly to 1.0 during the training process. Similarly, the initial weighting factor

Table A.1. Training set composition.

Number of configurations						
MB-pol simulations	DNN 3 iterations	VLE	Monomers	Dimers	Trimers	Tetramers
35411	59359	2412	4955	20540	8478	40000

p_f for the forces (Eq. 5 of the main text) was set to 1000 and decreased to 1.0 at the end of the training process. Since we found that the inclusion of the virial in the training process did not lead to an improvement in the accuracy of the DNN potential, the associated weighting factor p_ξ (Eq. 5 of the main text) was set to zero.

All DNN potentials developed in this study have been trained starting from four different random seeds for which Table A.2 shows the corresponding errors on both training and validation sets, and Table A.3 shows the corresponding errors on the test sets.

Table A.2. Training and validation root-mean-square errors (RMSEs) for the different DNN, DNN(VLE), and DNN(MB) potentials developed in this study. All values are calculated on the last 200 epochs. The potentials used in the simulations are marked with an asterisk (*).

Potential	Energy (kcal mol^{-1}) Training	Energy (kcal mol^{-1}) Validation	Force (kcal mol^{-1}Å^{-1}) Training	Force (kcal mol^{-1}Å^{-1}) Validation
DNN, seed 1	0.009	0.012	0.962	0.932
DNN, seed 2*	0.010	0.012	0.957	0.931
DNN, seed 3	0.010	0.012	0.932	0.911
DNN, seed 4	0.009	0.012	0.973	0.942
DNN(VLE10), seed 1	0.006	0.009	0.900	0.875
DNN(VLE10), seed 2	0.007	0.009	0.916	0.866
DNN(VLE10), seed 3	0.008	0.009	0.879	0.875
DNN(VLE10), seed 4*	0.007	0.009	0.889	0.874
DNN(VLE20), seed 1	0.007	0.008	0.928	0.912
DNN(VLE20), seed 2	0.010	0.009	0.883	0.910
DNN(VLE20), seed 3	0.006	0.008	0.907	0.906
DNN(VLE20), seed 4*	0.007	0.008	0.936	0.915
DNN(MB4), seed 1	0.007	0.008	0.907	0.925
DNN(MB4), seed 2	0.007	0.007	0.914	0.935
DNN(MB4), seed 3	0.008	0.007	0.901	0.935
DNN(MB4), seed 4*	0.007	0.007	0.924	0.944
DNN(MB10), seed 1	0.010	0.008	0.890	0.945
DNN(MB10), seed 2	0.008	0.008	0.881	0.930
DNN(MB10), seed 3	0.010	0.008	0.890	0.947
DNN(MB10), seed 4*	0.009	0.008	0.884	0.958
DNN(MB20), seed 1	0.010	0.009	0.778	0.840
DNN(MB20), seed 2	0.010	0.010	0.829	0.889
DNN(MB20), seed 3*	0.011	0.010	0.844	0.912
DNN(MB20), seed 4	0.012	0.010	0.760	0.867

Table A.3. Root-mean-square errors (RMSEs) of the energy per atom for the different potentials. All values are calculated averaging over all test sets. The potentials used in the simulations are marked with an asterisk (*).

Potential	Energy (kcal mol^{-1})		
	MB-pol + DNN iter.	Mon. + Dim. + Trim. + Tetra.	VLE
DNN, seed 1	0.012	-	-
DNN, seed 2*	0.013	-	-
DNN, seed 3	0.013	-	-
DNN, seed 4	0.014	-	-
DNN(VLE10), seed 1	0.014	-	0.009
DNN(VLE10), seed 2	0.014	-	0.009
DNN(VLE10), seed 3	0.013	-	0.010
DNN(VLE10), seed 4*	0.015	-	0.009
DNN(VLE20), seed 1	0.015	-	0.009
DNN(VLE20), seed 2	0.012	-	0.009
DNN(VLE20), seed 3	0.013	-	0.009
DNN(VLE20), seed 4*	0.015	-	0.009
DNN(MB4), seed 1	0.012	0.014	-
DNN(MB4), seed 2	0.012	0.012	-
DNN(MB4), seed 3	0.012	0.010	-
DNN(MB4), seed 4*	0.012	0.010	-
DNN(MB10), seed 1	0.012	0.011	-
DNN(MB10), seed 2	0.012	0.010	-
DNN(MB10), seed 3	0.012	0.011	-
DNN(MB10), seed 4*	0.013	0.012	-
DNN(MB20), seed 1	0.016	0.010	-
DNN(MB20), seed 2	0.016	0.009	-
DNN(MB20), seed 3*	0.013	0.008	-
DNN(MB20), seed 4	0.015	0.008	-

A.2 Density fluctuations

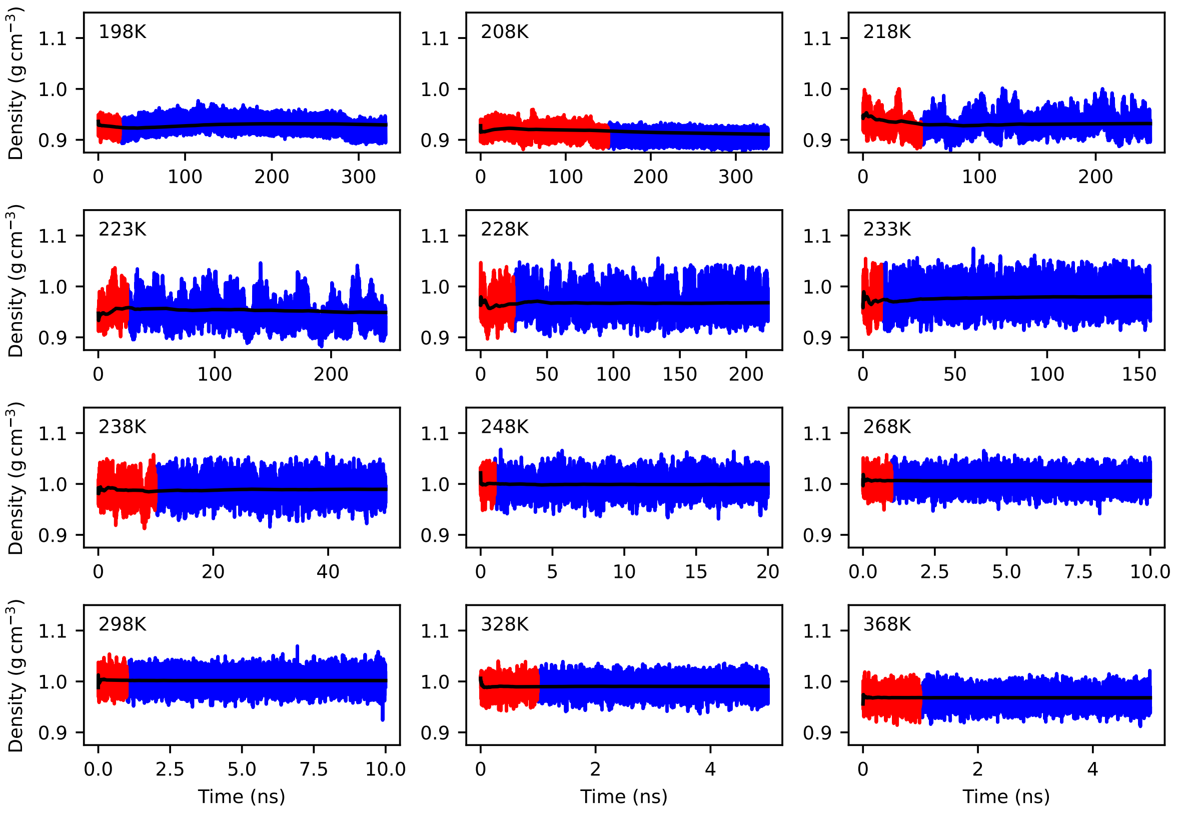


Figure A.1. Density of liquid water calculated from *NPT* simulations carried out with the DNN potential (seed 2) at 1 atm and different temperatures. The red interval corresponds to the equilibration time, the blue interval was used to calculate the average thermodynamic properties, and the black line corresponds to the cumulative average.

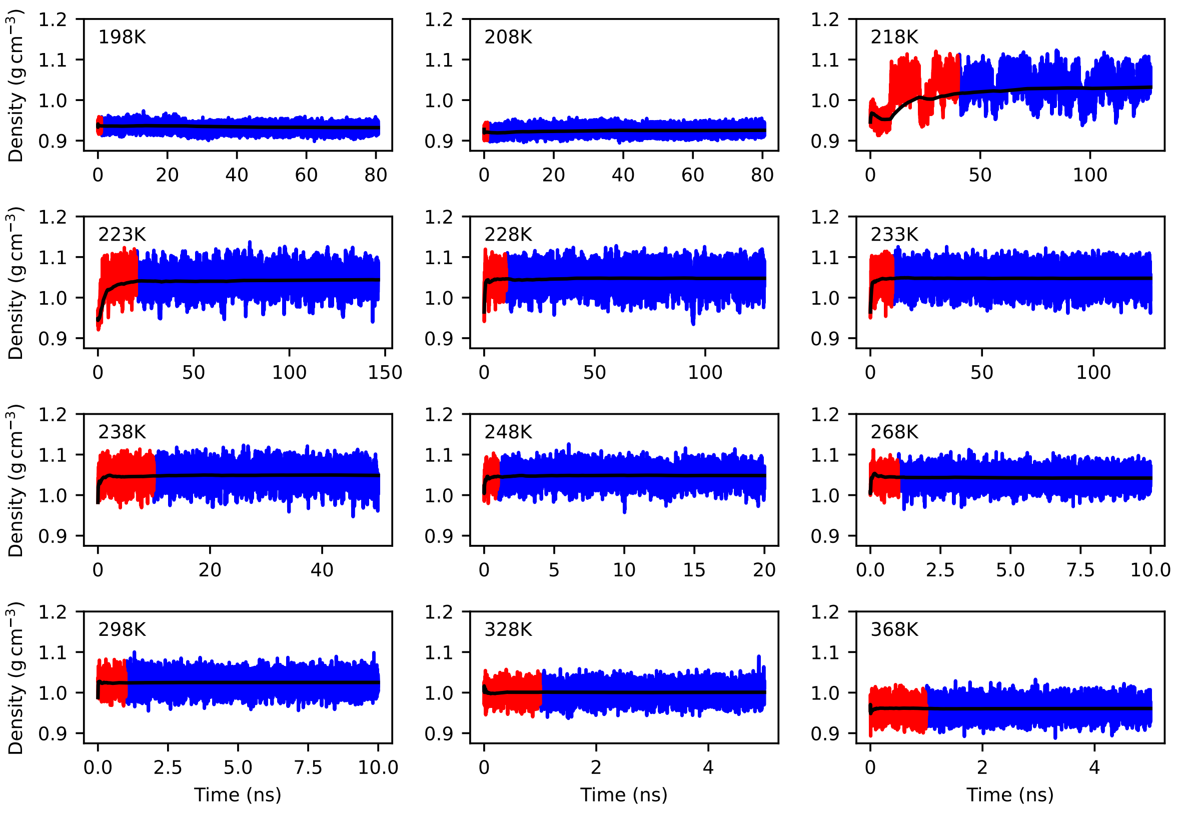


Figure A.2. Density of liquid water calculated from *NPT* simulations carried out with the DNN(MB20) potential (seed 3) at 1 atm and different temperatures. The red interval corresponds to the equilibration time, the blue interval was used to calculate the average thermodynamic properties, and the black line corresponds to the cumulative average.

A.3 Supplemental many-body analyses

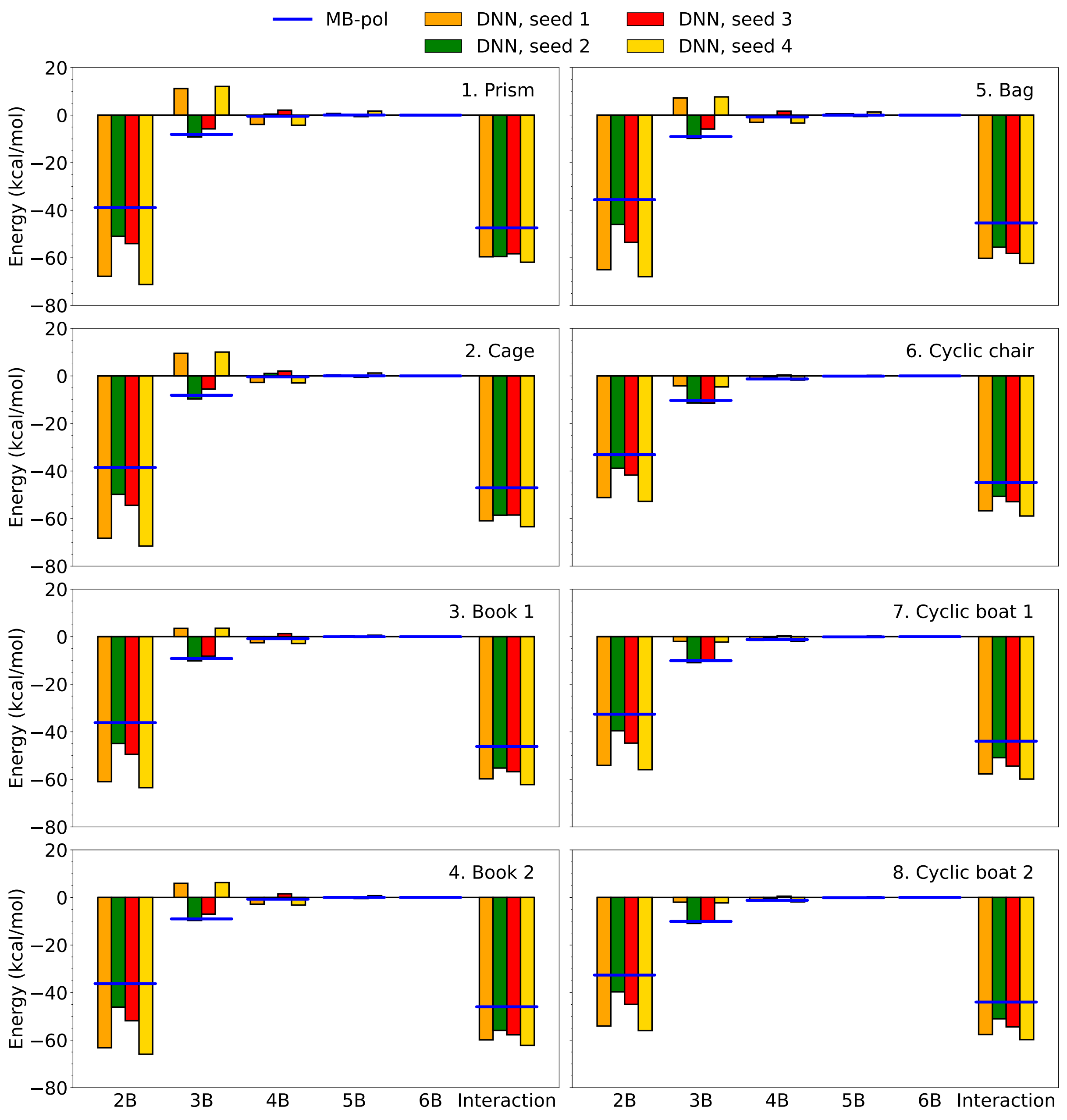


Figure A.3. Many-body energy decomposition for the first eight low-lying energy isomers of the water hexamer using DNN potentials. Individual n-body energies (nB, with $n = 2 - 6$) and interaction energies calculated with the four distinct DNN potentials are compared with the corresponding MB-pol reference values.

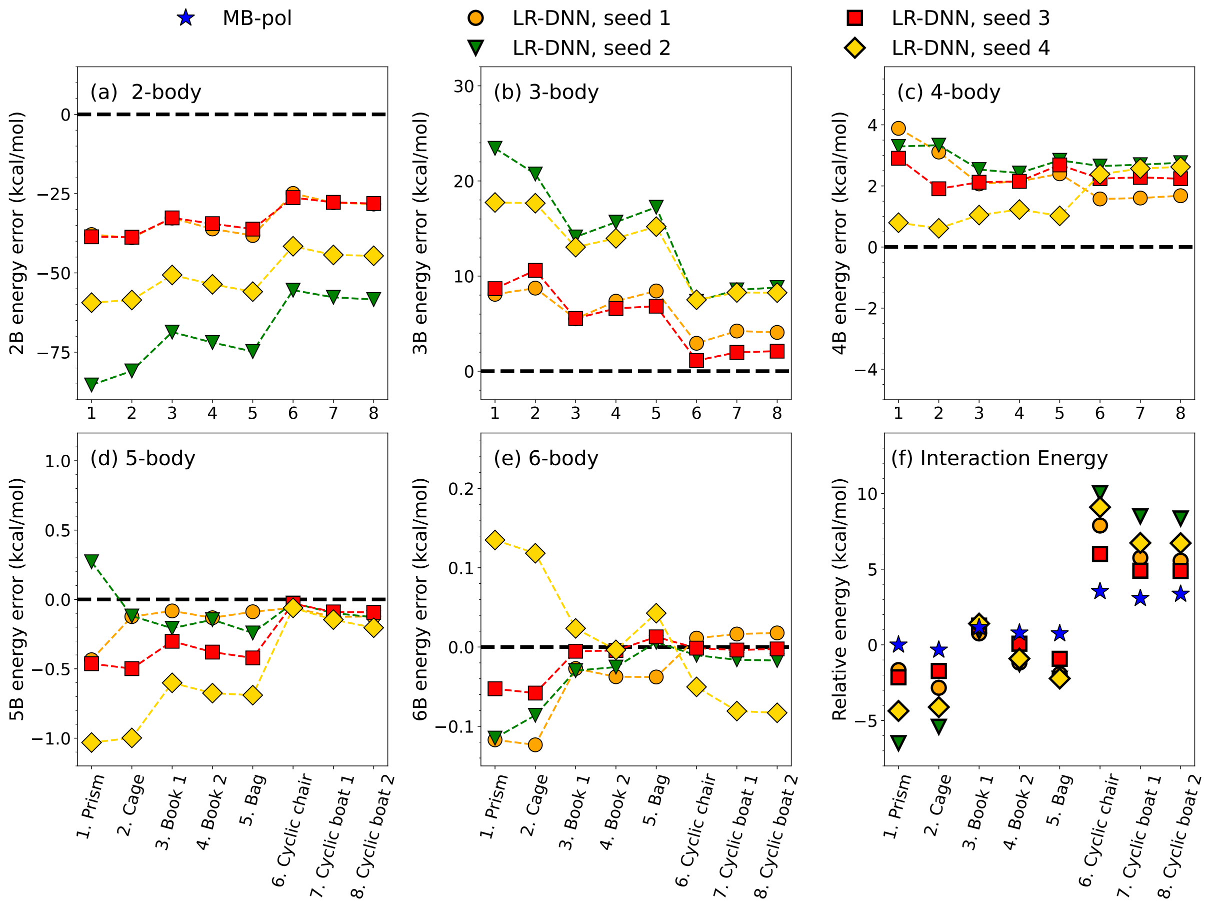


Figure A.4. Many-body decomposition analysis for the eight low-lying energy isomers of the water hexamer (Fig. 5) calculated with long-range corrected LR-DNN potentials trained on the same extended MB-pol training data. Panels a) to e) show the errors associated with n-body energies ($n = 2 - 6$) calculated with the LR-DNN potentials relative to the corresponding MB-pol values. Panel f) shows the errors associated with the interaction energies calculated with the LR-DNN potentials relative to the corresponding MB-pol values.

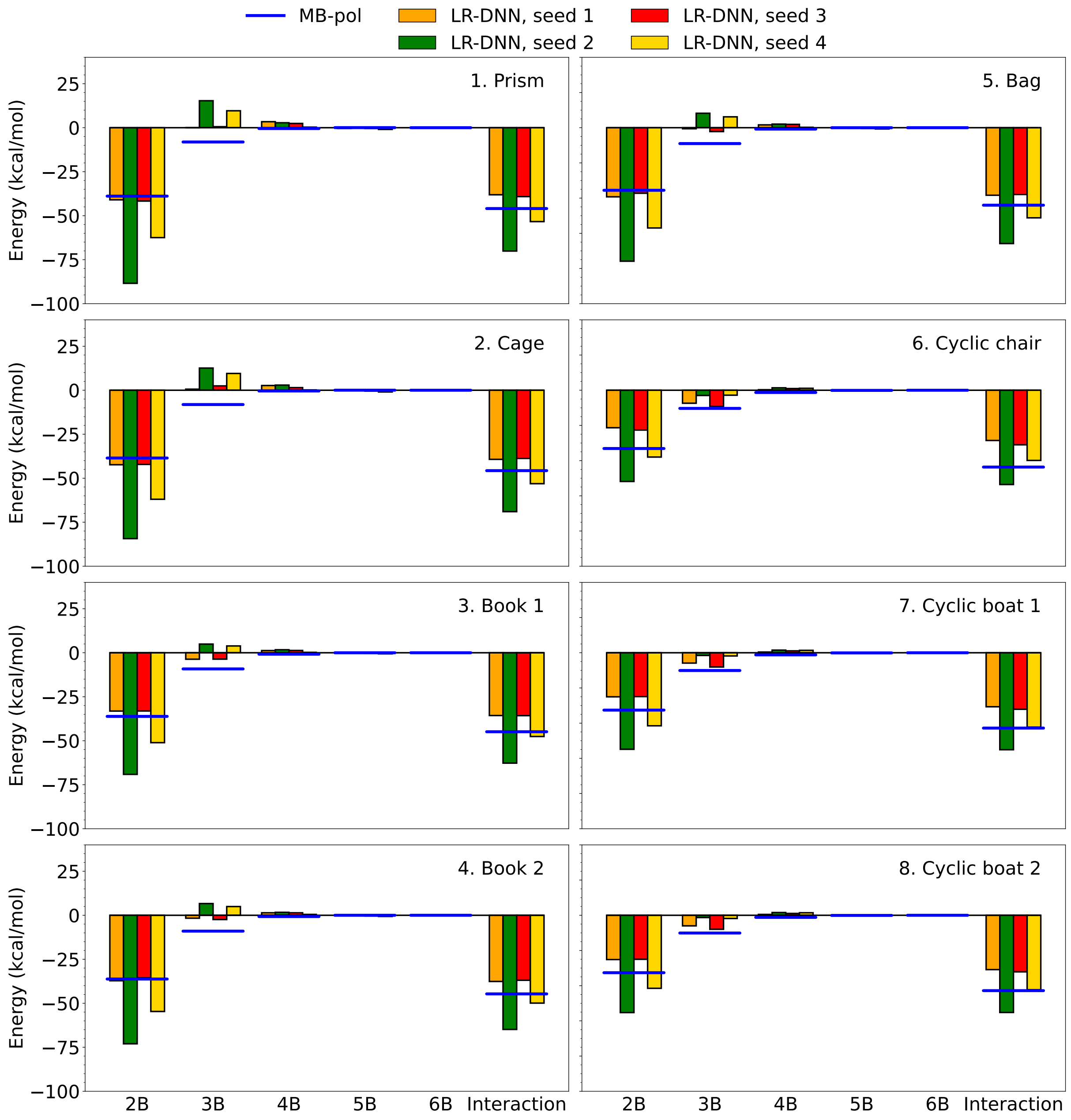


Figure A.5. Many-body energy decomposition for the first eight low-lying energy isomers of the water hexamer using LR-DNN potentials. Individual n-body energies (nB, with $n = 2 - 6$) and interaction energies calculated with the long-range corrected LR-DNN potentials are compared with the corresponding MB-pol reference values.

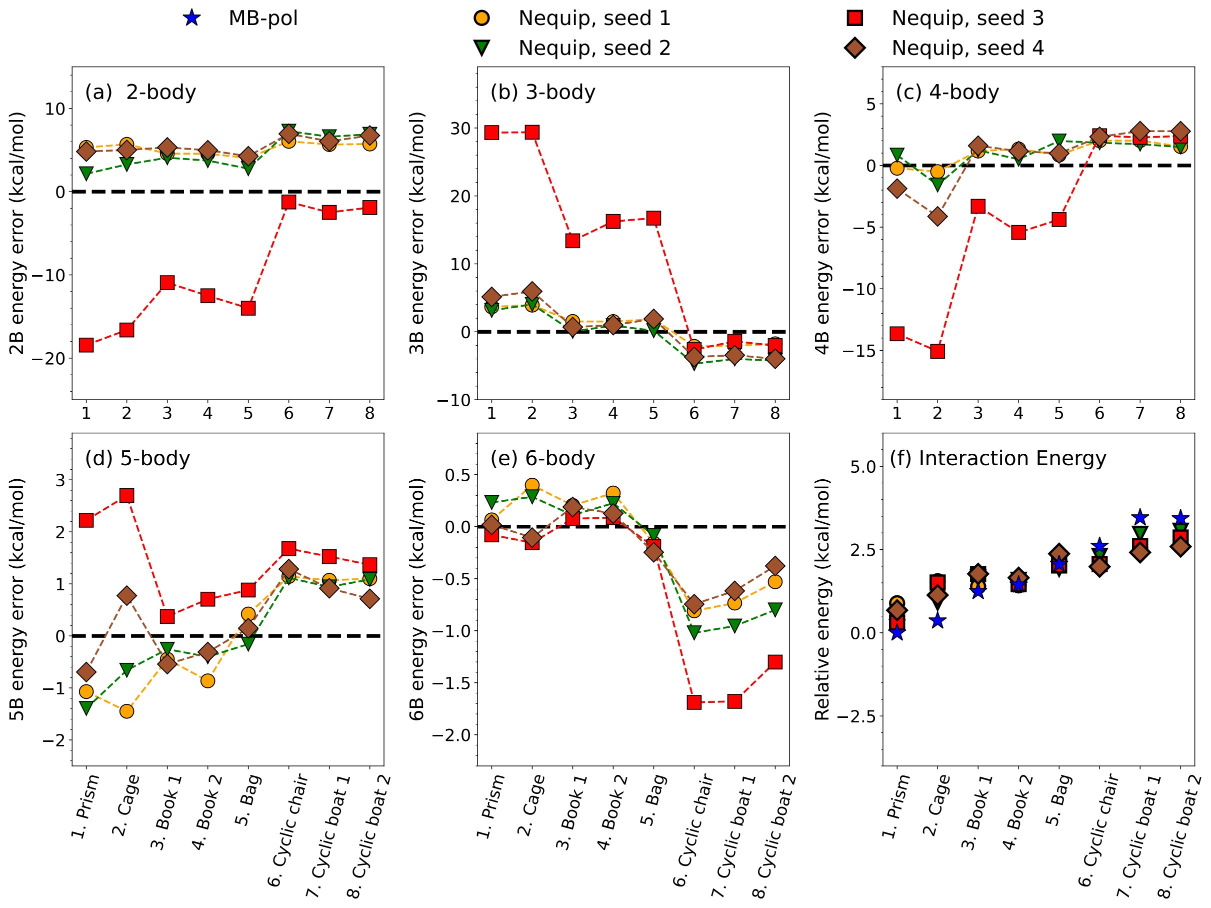


Figure A.6. Many-body decomposition analysis for the eight low-lying energy isomers of the water hexamer (Fig. 5) calculated with Nequip-based potentials trained on the same extended MB-pol training data. Panels a) to e) show the errors associated with n-body energies ($n = 2-6$) calculated with the Nequip-based potentials relative to the corresponding MB-pol values. Panel f) shows the errors associated with the interaction energies calculated with the Nequip-based potentials relative to the corresponding MB-pol values.

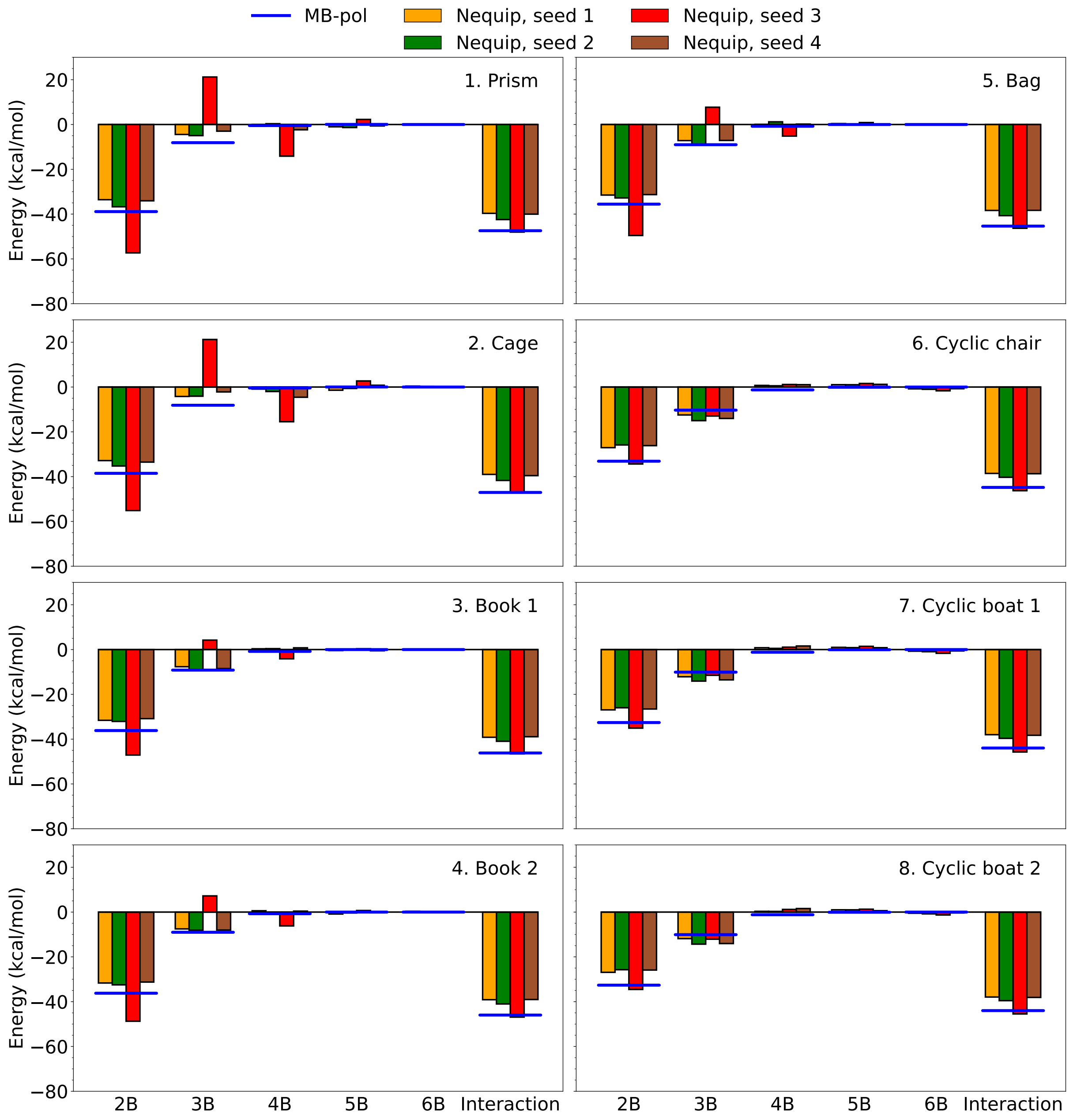


Figure A.7. Many-body energy decomposition for the first eight low-lying energy isomers of the water hexamer using Nequip potentials. Individual n-body energies (nB, with $n = 2-6$) and interaction energies calculated with the Nequip-based potentials are compared with the corresponding MB-pol reference values.

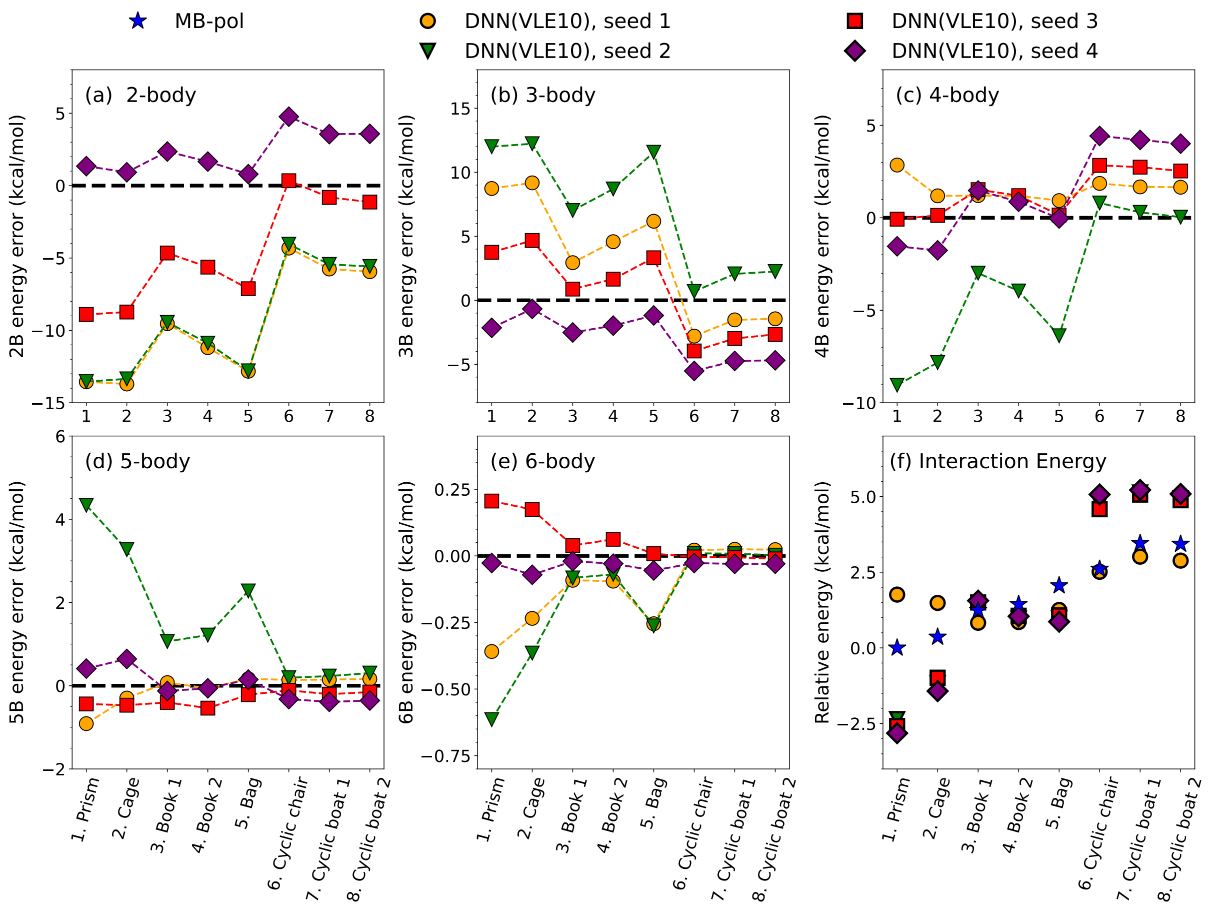


Figure A.8. Many-body decomposition analysis for the eight low-lying energy isomers of the water hexamer (Fig. 2 of the main text) calculated with four distinct DNN(VLE10) potentials that were trained on the expanded MB-pol training set containing vapor-liquid configurations using four different seeds to initialize the fitting process. Panels a) to e) show the errors associated with *n*-body energies ($n = 2-6$) calculated with the DNN(VLE10) potentials relative to the corresponding MB-pol values. Panel f) shows the errors associated with the interaction energies calculated with the DNN(VLE10) potentials relative to the corresponding MB-pol values. The DNN(VLE10) potential with seed 4 is used in the comparisons shown in Figs. 7 and 8 of the main text.

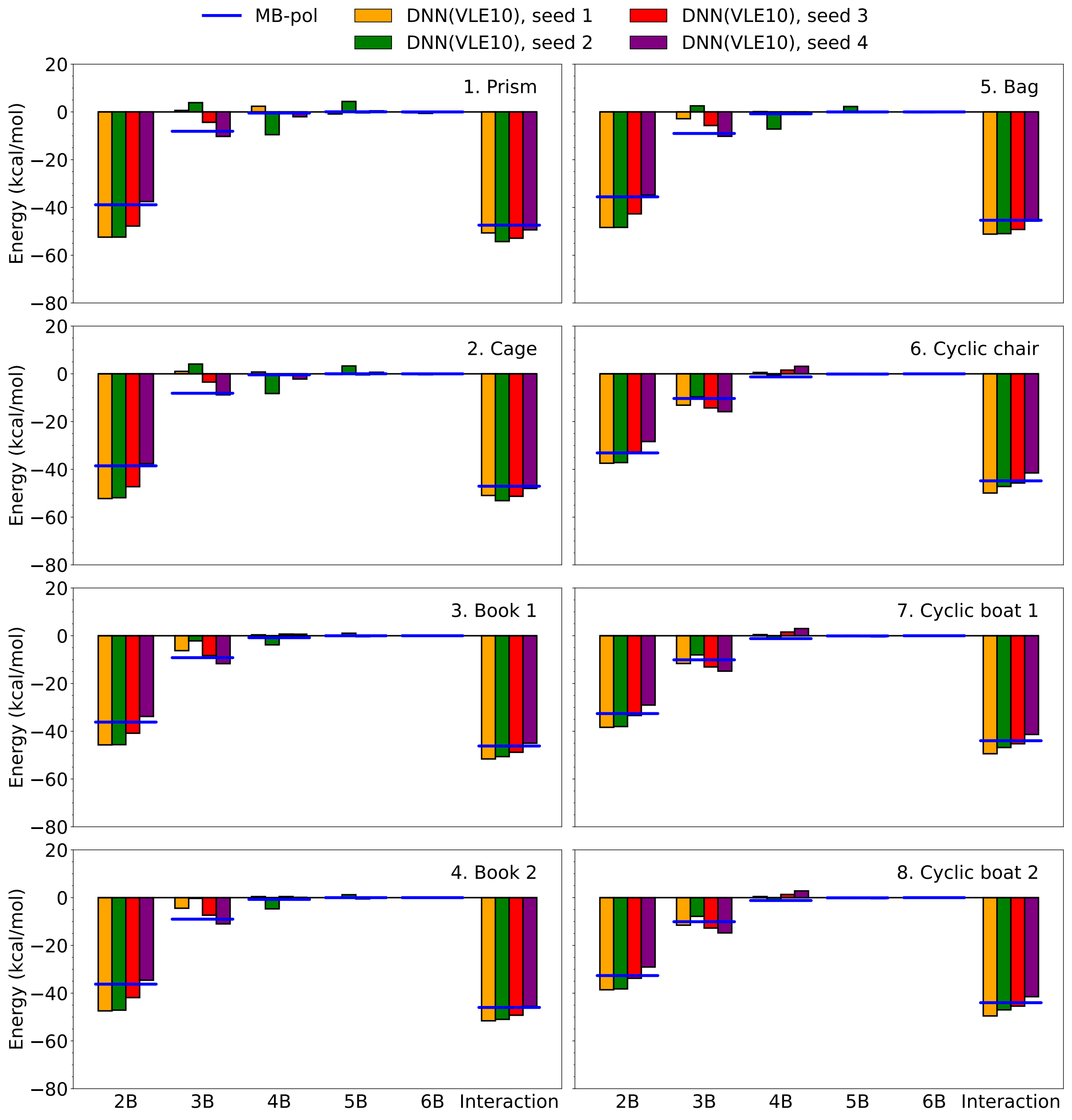


Figure A.9. Many-body energy decomposition for the first eight low-lying energy isomers of the water hexamer using DNN(VLE10) potentials. Individual n-body energies (nB, with $n = 2 - 6$) and interaction energies calculated with the four distinct DNN(VLE10) potentials are compared with the corresponding MB-pol reference values.

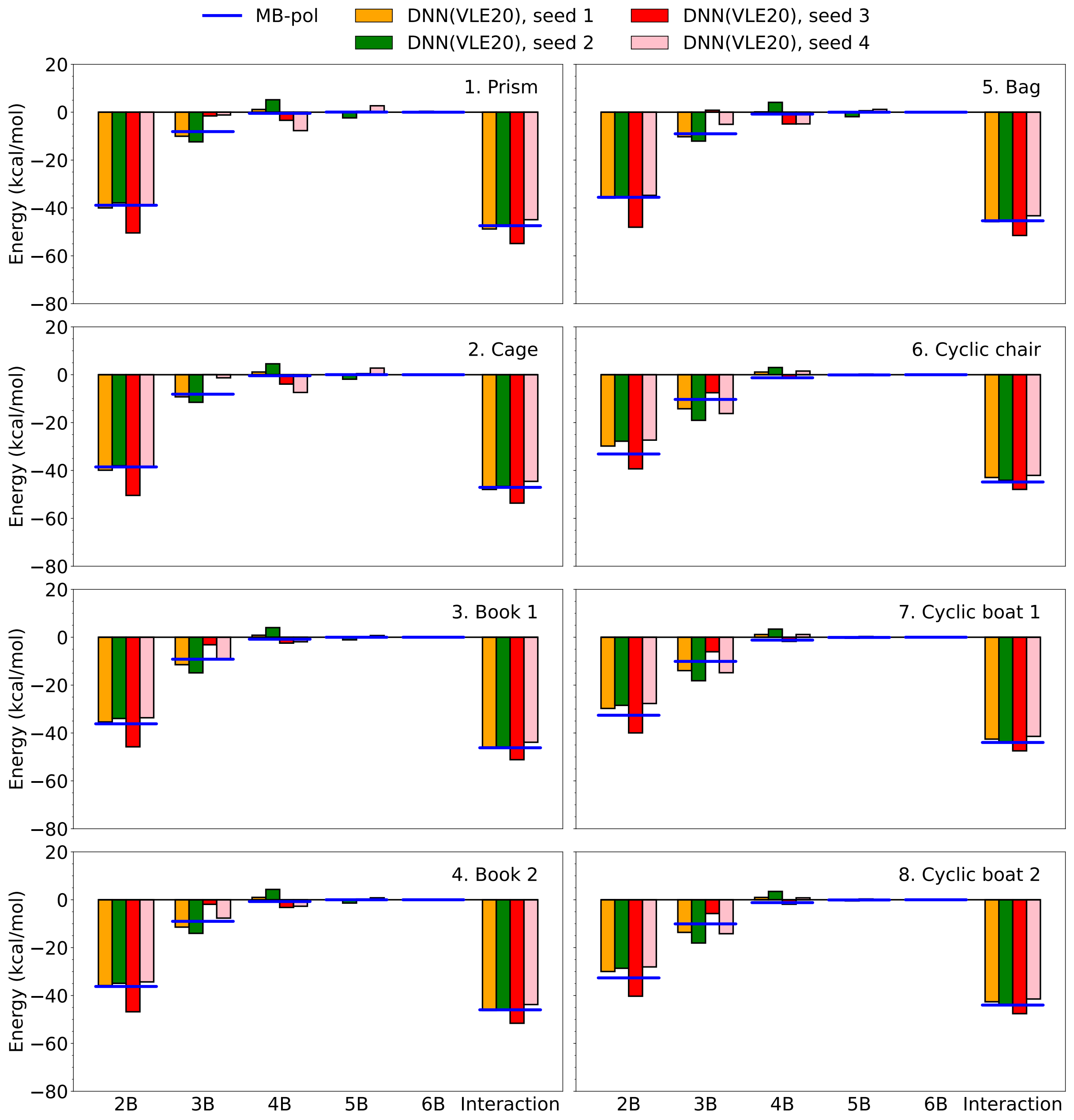


Figure A.10. Many-body energy decomposition for the first eight low-lying energy isomers of the water hexamer using DNN(VLE20) potentials. Individual *n*-body energies (*n*B, with $n = 2 - 6$) and interaction energies calculated with the four distinct DNN(VLE20) potentials are compared with the corresponding MB-pol reference values.

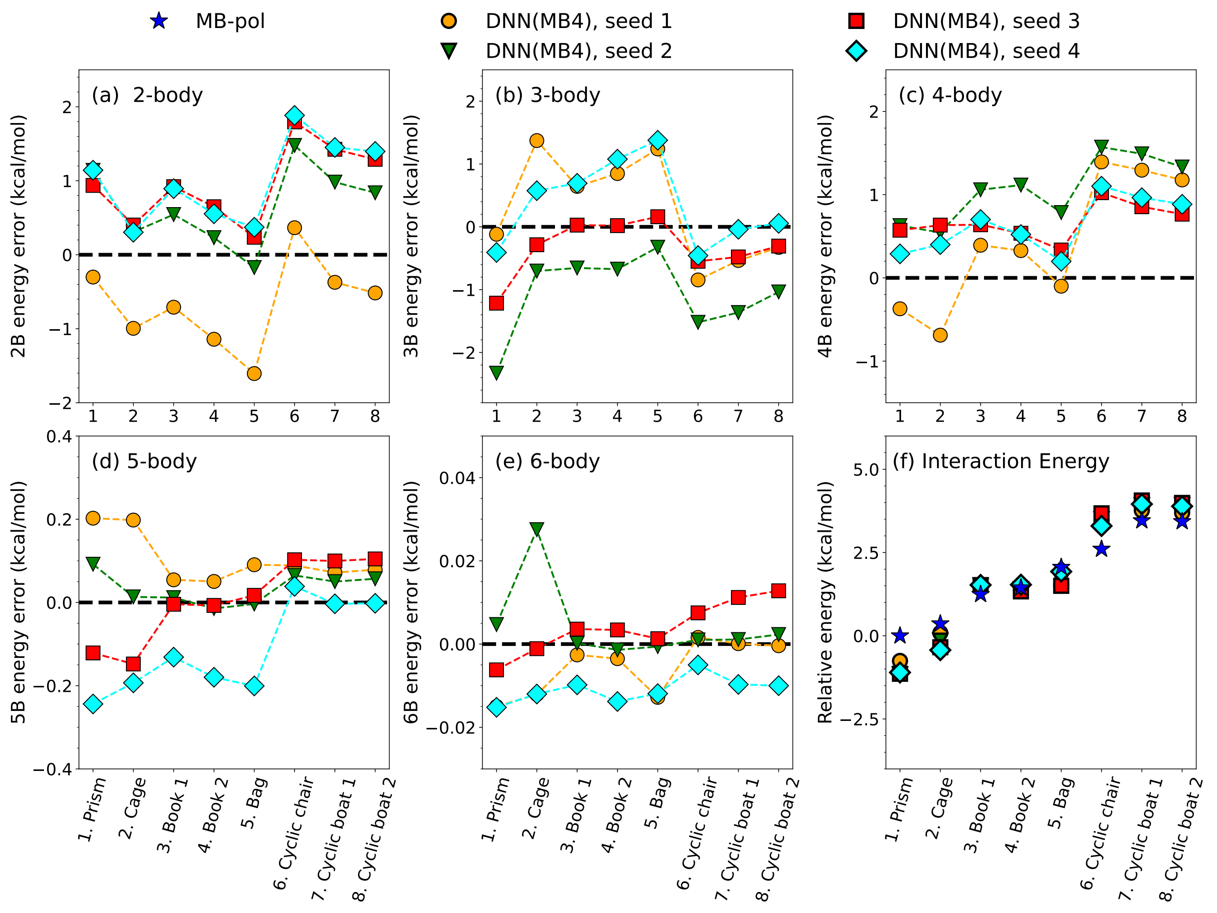


Figure A.11. Many-body decomposition analysis for the eight low-lying energy isomers of the water hexamer (Fig. 2 of the main text) calculated with four distinct DNN(MB4) potentials that were trained on the expanded MB-pol training set containing vapor-liquid configurations using four different seeds to initialize the fitting process. Panels a) to e) show the errors associated with n-body energies ($n = 2-6$) calculated with the DNN(MB4) potentials relative to the corresponding MB-pol values. Panel f) shows the errors associated with the interaction energies calculated with the DNN(MB4) potentials relative to the corresponding MB-pol values. The DNN(MB4) potential with seed 4 is used in the comparisons shown in Figs. 11 and 13 of the main text.

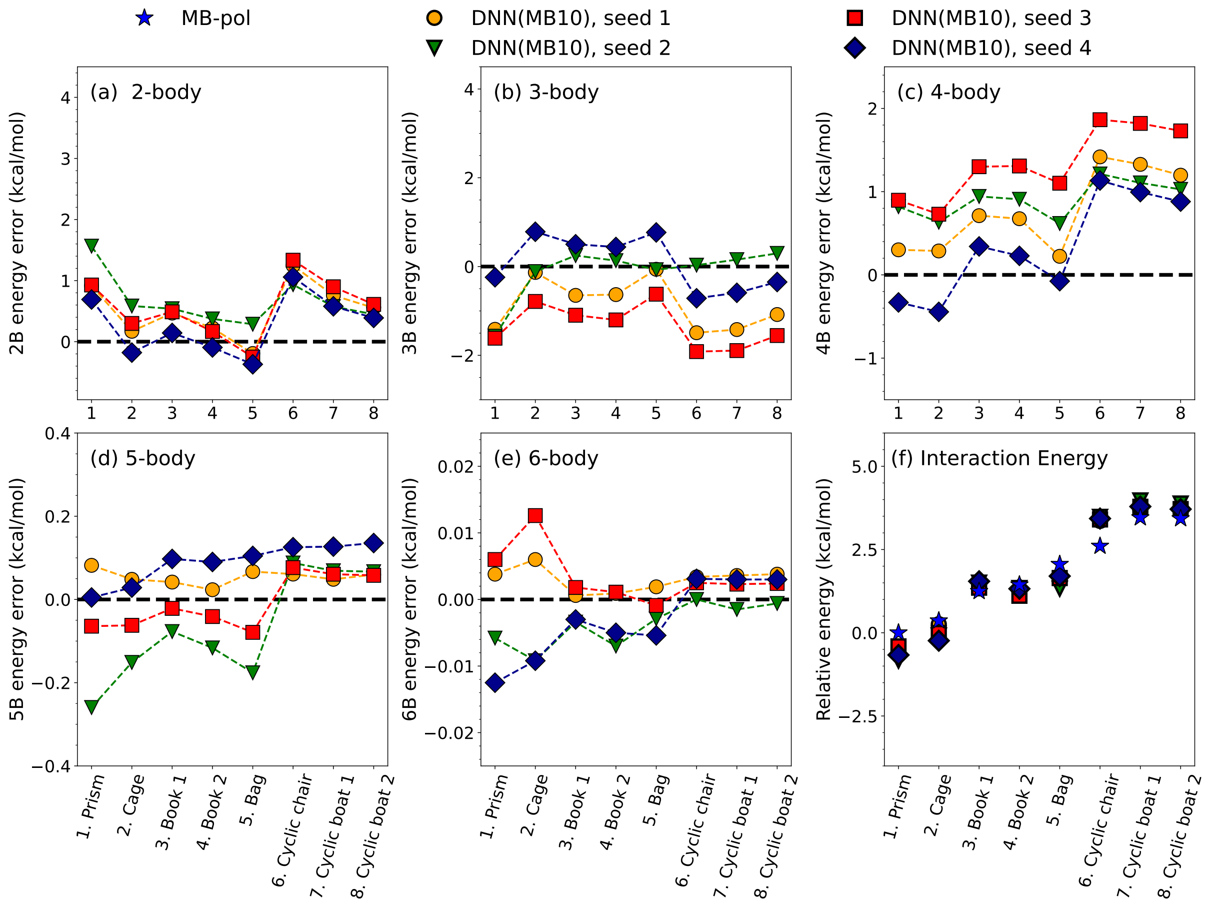


Figure A.12. Many-body decomposition analysis for the eight low-lying energy isomers of the water hexamer (Fig. 2 of the main text) calculated with four distinct DNN(MB10) potentials that were trained on the expanded MB-pol training set containing vapor-liquid configurations using four different seeds to initialize the fitting process. Panels a) to e) show the errors associated with n-body energies ($n = 2-6$) calculated with the DNN(MB10) potentials relative to the corresponding MB-pol values. Panel f) shows the errors associated with the interaction energies calculated with the DNN(MB10) potentials relative to the corresponding MB-pol values. The DNN(MB10) potential with seed 4 is used in the comparisons shown in Figs. 11 and 13 of the main text.

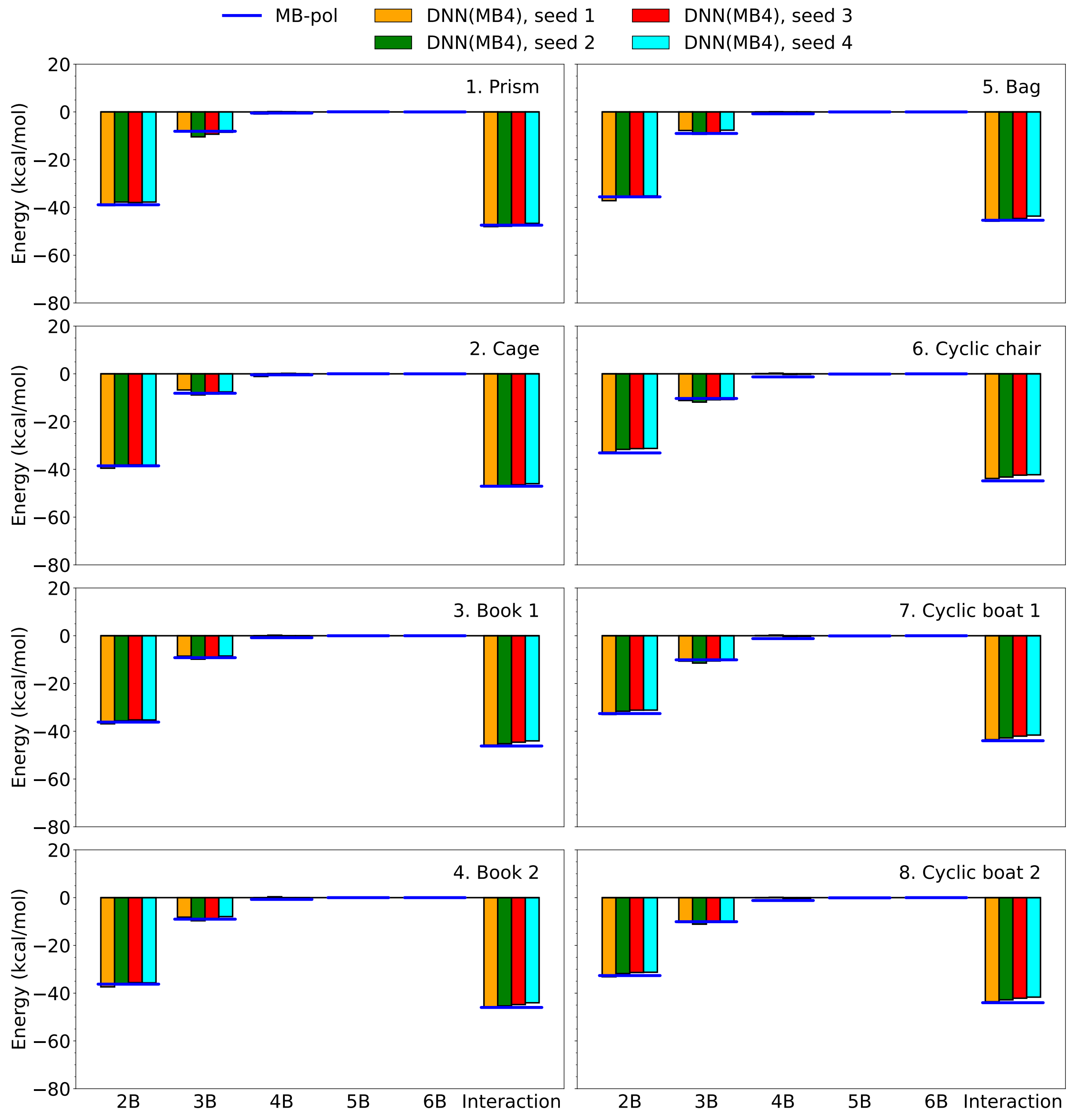


Figure A.13. Many-body energy decomposition for the first eight low-lying energy isomers of the water hexamer using DNN(MB4) potentials. Individual n-body energies (nB, with $n = 2-6$) and interaction energies calculated with the four distinct DNN(MB4) potentials are compared with the corresponding MB-pol reference values.

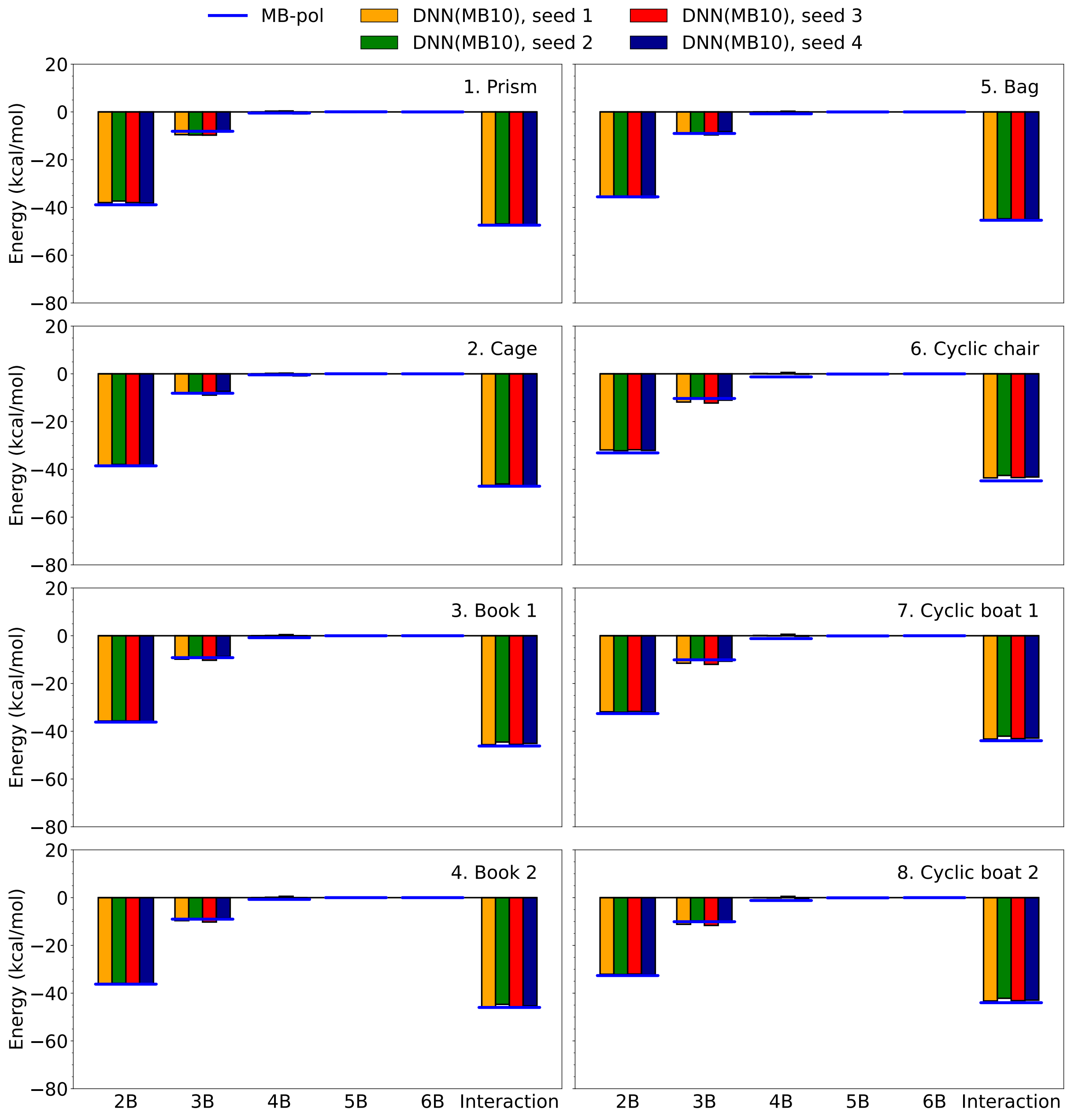


Figure A.14. Many-body energy decomposition for the first eight low-lying energy isomers of the water hexamer using DNN(MB10) potentials. Individual n-body energies (nB, with $n = 2-6$) and interaction energies calculated with the four distinct DNN(MB10) potentials are compared with the corresponding MB-pol reference values.

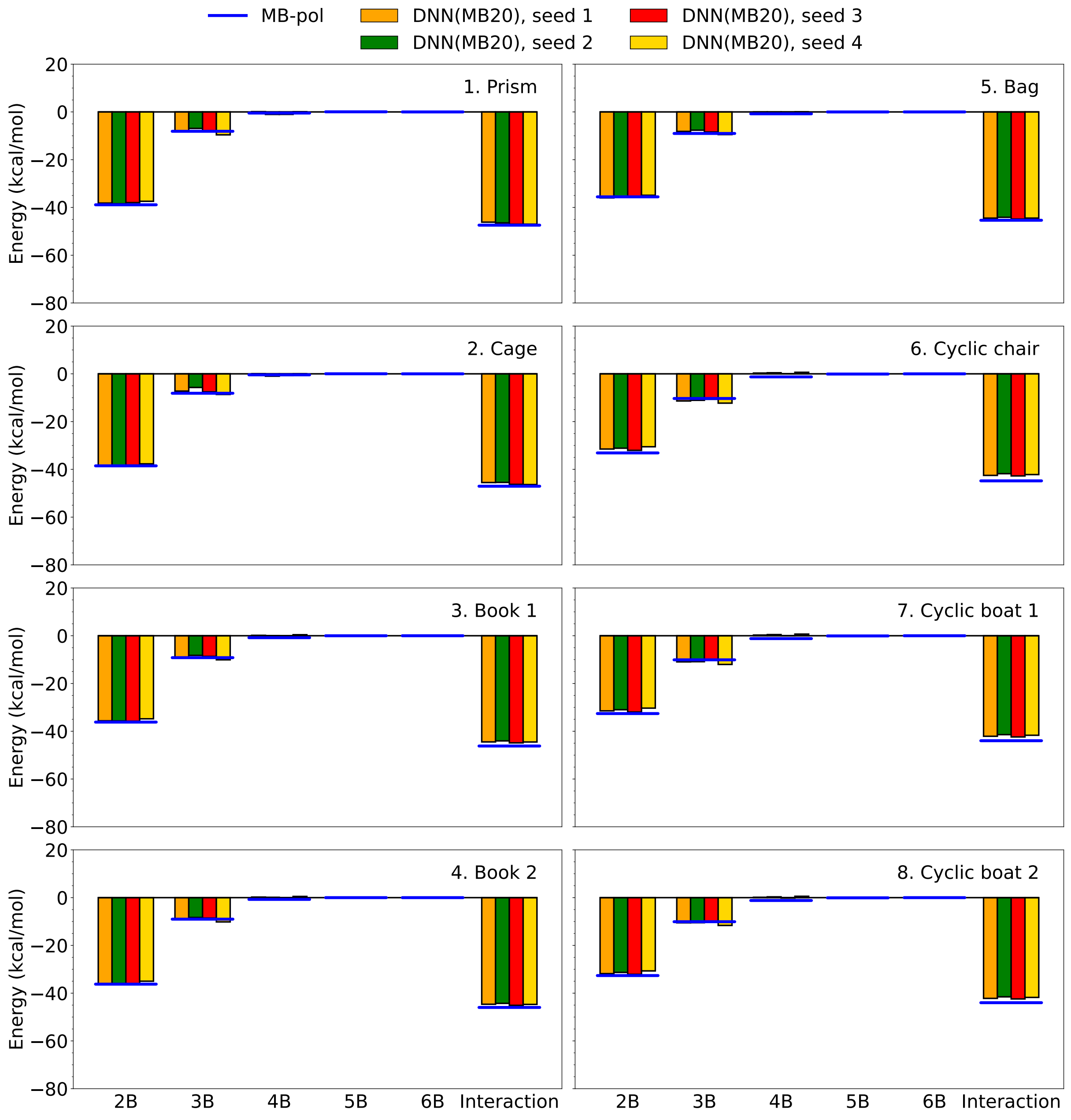


Figure A.15. Many-body energy decomposition for the first eight low-lying energy isomers of the water hexamer using DNN(MB20) potentials. Individual n-body energies (nB, with $n = 2-6$) and interaction energies calculated with the four distinct DNN(MB20) potentials are compared with the corresponding MB-pol reference values.

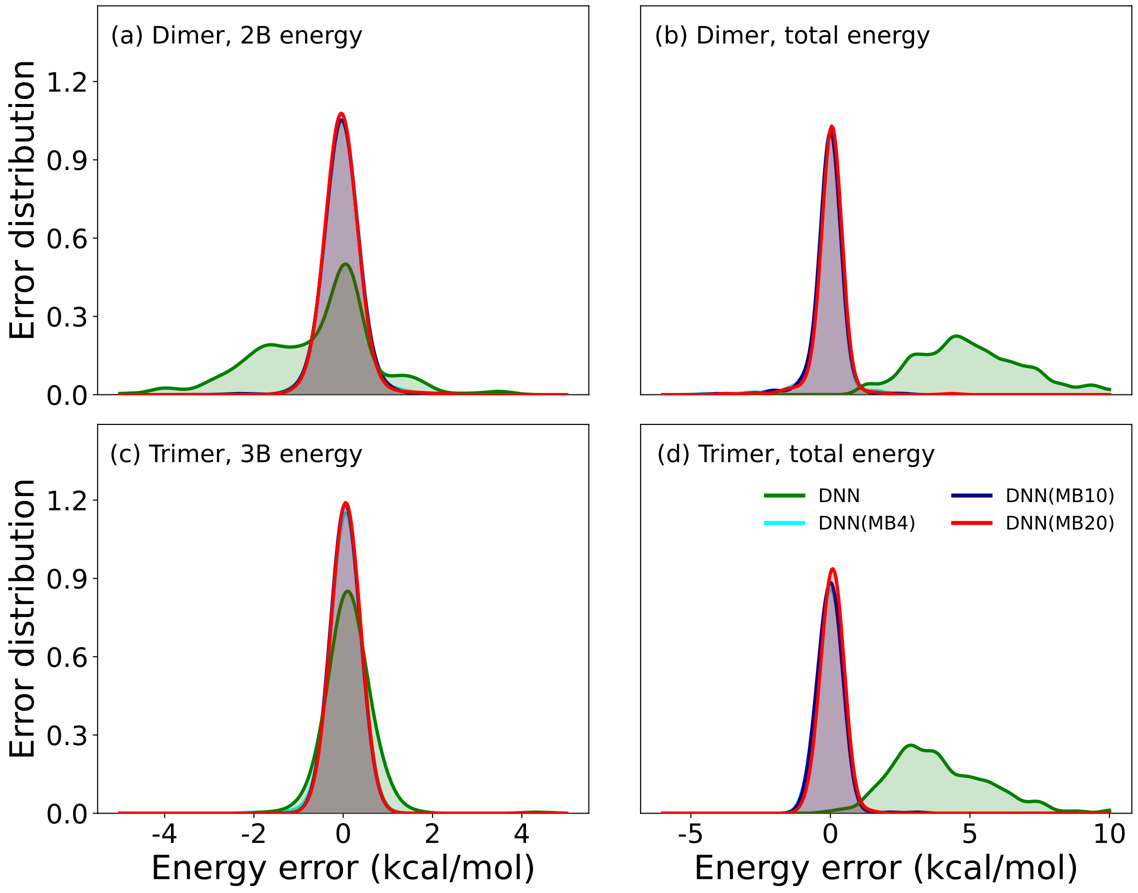


Figure A.16. Error distributions for 2-body (a) and 3-body (c) energies along with the total dimer (b) and trimer (d) energies calculated with the DNN, DNN(MB4), DNN(MB10), and DNN(MB20) potentials. Dimer and trimer configurations were taken from the MB-pol training sets introduced in Refs. 7 and 9, respectively.

A.4 Supplemental analysis of the vapor/liquid interface

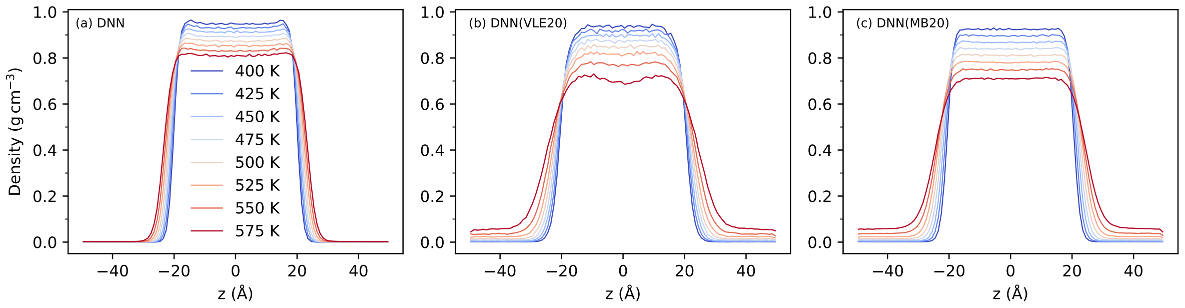


Figure A.17. Average density profiles obtained from the direct coexistence MD simulations carried out with the (a) DNN potential (seed 2), (b) DNN(VLE20) potential (seed 4), and (c) DNN(MB20) potential (seed 3). The thickness of the water slab is defined by the *z*-axis. Colors denote different temperatures as indicated.

A.5 Supplemental analysis of structural properties

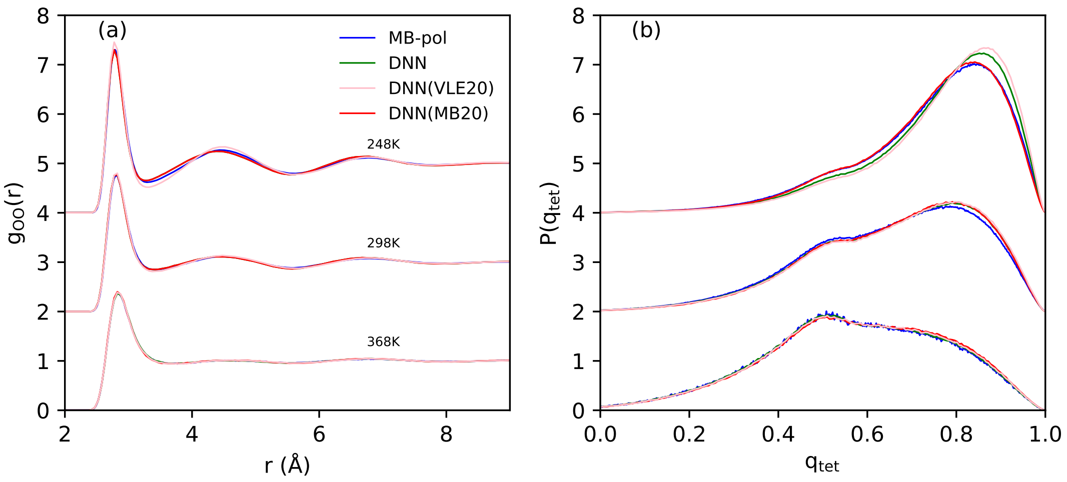


Figure A.18. (a) Oxygen-oxygen radial distribution function and (b) tetrahedral order parameter calculated from *NPT* simulations carried out at 1 atm and 298 K with the DNN potential (seed 2), DNN(VLE20) potential (seed 4), and DNN(MB20) potential (seed 3). Also shown as a reference is the corresponding MB-pol results reported in Ref. 62.

Bibliography

[1] Adler, T., Knizia, G., and Werner, H. 2007. A Simple and Efficient CCSD(T)-F12 Approximation. *J. Chem. Phys.*, 127(22): 221106–221106.

[2] Alder, B. J., and Wainwright, T. E. 1957. Phase Transition for a Hard Sphere System. *J. Chem. Phys.*, 27(5): 1208–1209.

[3] Alefeld, G., and Herzberger, J. 2012. *Introduction to Interval Computation*. Academic press.

[4] Almuallim, H., and Dietterich, T. G. 1991. Learning with Many Irrelevant Features. In *Proceedings of the Ninth National Conference on Artificial Intelligence - Volume 2*, AAAI'91, 547–552. AAAI Press. ISBN 0-262-51059-6.

[5] Angell, C., Sichina, W., and Oguni, M. 1982. Heat Capacity of Water at Extremes of Supercooling and Superheating. *J. Phys. Chem.*, 86(6): 998–1002.

[6] Araya, I., and Reyes, V. 2016. Interval Branch-and-Bound Algorithms for Optimization and Constraint Satisfaction: A Survey and Prospects. *Journal of Global Optimization*, 65: 837–866.

[7] Babin, V., Leforestier, C., and Paesani, F. 2013. Development of a "First Principles" Water Potential with Flexible Monomers: Dimer Potential Energy Surface, VRT Spectrum, and Second Virial Coefficient. *J. Chem. Theory Comput.*, 9(12): 5395–5403.

[8] Babin, V., Medders, G. R., and Paesani, F. 2012. Toward a Universal Water Model: First Principles Simulations from the Dimer to the Liquid Phase. *J. Phys. Chem. Lett.*, 3(24): 3765–3769.

[9] Babin, V., Medders, G. R., and Paesani, F. 2014. Development of a "First Principles" Water Potential with Flexible Monomers. II: Trimer Potential Energy Surface, Third Virial Coefficient, and Small Clusters. *J. Chem. Phys.*, 10(4): 1599–1607.

[10] Baeyens, E., Herreros, A., and Perán, J. R. 2016. A Direct Search Algorithm for Global Optimization. *Algorithms*, 9(2): 40.

[11] Bajaj, P., Götz, A. W., and Paesani, F. 2016. Toward Chemical Accuracy in the Description of Ion–Water Interactions Through Many-Body Representations. I. Halide–Water Dimer Potential Energy Surfaces. *J. Chem. Theory Comput.*, 12(6): 2698–2705.

[12] Bao, T. Q., and Mordukhovich, B. S. 2010. Set-Valued Optimization in Welfare Economics. *Advances in mathematical economics*, 113–153.

[13] Barker, J., and Watts, R. 1969. Structure of Water; A Monte Carlo Calculation. *Chem. Phys. Lett.*, 3(3): 144–145.

[14] Bartholomew-Biggs, M. 2008. *Nonlinear Optimization with Engineering Applications*, volume 19. Springer Science & Business Media.

[15] Bartók, A. P., Payne, M. C., Kondor, R., and Csányi, G. 2010. Gaussian Approximation Potentials: The Accuracy of Quantum Mechanics, without the Electrons. *Phys. Rev. Lett.*, 104: 136403.

[16] Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J. P., Kornbluth, M., Molinari, N., Smidt, T. E., and Kozinsky, B. 2022. E(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials. *Nat. Commun.*, 13(1): 1–11.

[17] Behler, J. 2011. Neural Network Potential-Energy Surfaces in Chemistry: A Tool for Large-Scale Simulations. *Phys. Chem. Chem. Phys*, 13(40): 17930–17955.

[18] Behler, J. 2014. Representing Potential Energy Surfaces by High-Dimensional Neural Network Potentials. *J. Phys. Condens. Matter*, 26(18): 183001.

[19] Behler, J. 2017. First Principles Neural Network Potentials for Reactive Simulations of Large Molecular and Condensed Systems. *Angew. Chem. Int. Ed.*, 56(42): 12828–12840.

[20] Behler, J. 2021. Four Generations of High-Dimensional Neural Network Potentials. *Chem. Rev.*, 121(16): 10037–10072.

[21] Behler, J., and Parrinello, M. 2007. Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces. *Phys. Rev. Lett.*, 98(14): 146401.

[22] Bergou, E. H., Gorbunov, E., and Richtárik, P. 2020. Stochastic Three Points Method for Unconstrained Smooth Minimization. *SIAM Journal on Optimization*, 30(4): 2726–2749.

[23] Bishop, C. M. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag. ISBN 0387310738.

[24] Bishop, K. P., and Roy, P.-N. 2018. Quantum Mechanical Free Energy Profiles with Post-quantization Restraints: Binding Free Energy of the Water Dimer over a Broad

Range of Temperatures. *J. Chem. Phys.*, 148(10): 102303.

[25] Bizzarro, B. B., Egan, C. K., and Paesani, F. 2019. Nature of Halide–Water Interactions: Insights from Many-Body Representations and Density Functional Theory. *J. Chem. Theory Comput.*, 15(5): 2983–2995.

[26] Bore, S. L., and Paesani, F. 2023. Quantum Phase Diagram of Water. ChemRxiv, https://doi.org/10.26434/chemrxiv-2023-kmmmz.

[27] Boys, S. F., and F., B. 1970. The Calculation of Small Molecular Interactions by the Differences of Separate Total Energies. Some Procedures with Reduced Errors. *Mol. Phys.*, 19(4): 553–566.

[28] Braams, B. J., and Bowman, J. M. 2009. Permutationally Invariant Potential Energy Surfaces in High Dimensionality. *Int. Rev. Phys. Chem.*, 28(4): 577–606.

[29] Brown, S. E., Götz, A. W., Cheng, X., Steele, R. P., Mandelshtam, V. A., and Paesani, F. 2017. Monitoring Water Clusters "Melt" through Vibrational Spectroscopy. *Journal of the American Chemical Society*, 139(20): 7082–7088.

[30] Browne, C., Powley, E. J., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Liebana, D. P., Samothrakis, S., and Colton, S. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Trans. Comput. Intellig. and AI in Games*, 4(1): 1–43.

[31] Bubeck, S., Munos, R., Stoltz, G., and Szepesvári, C. 2011. X-Armed Bandits. *Journal of Machine Learning Research*, 12(5).

[32] Bukowski, R., Szalewicz, K., Groenenboom, G. C., and Van der Avoird, A. 2007. Predictions of the Properties of Water from First Principles. *Science*, 315(5816): 1249–1252.

[33] Bukowski, R., Szalewicz, K., Groenenboom, G. C., and van der Avoird, A. 2008. Polarizable Interaction Potential for Water from Coupled Cluster Calculations. I. Analysis of Dimer Potential Energy Surface. *J. Chem. Phys.*, 128(9): 094313.

[34] Bukowski, R., Szalewicz, K., Groenenboom, G. C., and van der Avoird, A. 2008. Polarizable Interaction Potential for Water from Coupled Cluster Calculations. II. Applications to Dimer Spectra, Virial Coefficients, and Simulations of Liquid Water. *J. Chem. Phys.*, 128(9): 094314.

[35] Burnham, C. J., Anick, D. J., Mankoo, P. K., and Reiter, G. F. 2008. The Vibrational Proton Potential in Bulk Liquid Water and Ice. *J. Chem. Phys.*, 128(15): 154519.

[36] Buşoniu, L., Daniels, A., Munos, R., and Babuška, R. 2013. Optimistic Planning for Continuous-action Deterministic Systems. In *2013 IEEE Symposium on Adaptive Dynamic*

Programming and Reinforcement Learning (ADPRL), 69–76. IEEE.

[37] Butler, K. T., Davies, D. W., Cartwright, H., Isayev, O., and Walsh, A. 2018. Machine Learning for Molecular and Materials Science. *Nature*, 559: 547–555.

[38] Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. 1995. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM Journal on scientific computing*, 16(5): 1190–1208.

[39] Campi, M. C., Garatti, S., and Ramponi, F. A. 2015. Non-convex Scenario Optimization with Application to System Identification. In *2015 54th IEEE Conference on Decision and Control (CDC)*, 4023–4028. IEEE.

[40] Caruana, R., and Niculescu-Mizil, A. 2006. An Empirical Comparison of Supervised Learning Algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, 161–168. New York, NY, USA: ACM. ISBN 1-59593-383-2.

[41] Ceriotti, M., Tribello, G. A., and Parrinello, M. 2011. Simplifying the Representation of Complex Free-Energy Landscapes Using Sketch-Map. *Proc. Natl. Acad. Sci. U.S.A.*, 108(32): 13023–13028.

[42] Ceriotti, M., Tribello, G. A., and Parrinello, M. 2013. Demonstrating the Transferability and the Descriptive Power of Sketch-Map. *J. Chem. Theory. Comp.*, 9(3): 1521–1532.

[43] Cheng, B., Engel, E. A., Behler, J., Dellago, C., and Ceriotti, M. 2019. Ab Initio Thermodynamics of Liquid and Solid Water. *Proc. Natl. Acad. Sci. U.S.A.*, 116(4): 1110–1115.

[44] Chmiela, S., Tkatchenko, A., Sauceda, H. E., Poltavsky, I., Schütt, K. T., and Müller, K.-R. 2017. Machine Learning of Accurate Energy-Conserving Molecular Force Fields. *Sci. Adv.*, 3(5): e1603015.

[45] Cisneros, G. A., Wikfeldt, K. T., Ojamäe, L., Lu, J., Xu, Y., Torabifard, H., Bartok, A. P., Csanyi, G., Molinero, V., and Paesani, F. 2016. Modeling Molecular Interactions in Water: From Pairwise to Many-Body Potential Energy Functions. *Chem. Rev.*, 116(13): 7501–7528.

[46] Cole, W. T., Farrell, J. D., Wales, D. J., and Saykally, R. J. 2016. Structure and Torsional Dynamics of the Water Octamer from THz Laser Spectroscopy Near 215 μm. *Science*, 352(6290): 1194–1197.

[47] Cruzeiro, V. W. D., Lambros, E., Riera, M., Roy, R., Paesani, F., and Gotz, A. W. 2021. Highly Accurate Many-Body Potentials for Simulations of N_2O_5 in Water: Benchmarks, Development, and Validation. *J. Chem. Theory Comput.*, 17(7): 3931–3945.

[48] Cvitaš, M. T., and Richardson, J. O. 2020. Quantum Tunnelling Pathways of the Water Pentamer. *Phys. Chem. Chem. Phys.*

[49] Dasgupta, S., Lambros, E., Perdew, J. P., and Paesani, F. 2021. Elevating Density Functional Theory to Chemical Accuracy for Water Simulations through a Density-Corrected Many-Body Formalism. *Nat. Commun.*, 12(1): 1–12.

[50] Dasgupta, S., Shahi, C., Bhetwal, P., Perdew, J. P., and Paesani, F. 2022. How Good Is the Density-Corrected SCAN Functional for Neutral and Ionic Aqueous Systems, and What Is So Right about the Hartree–Fock Density? *J. Chem. Theory Comput.*, 18(8): 4745.

[51] De Boer, P.-T., Kroese, D. P., Mannor, S., and Rubinstein, R. Y. 2005. A Tutorial on the Cross-Entropy Method. *Annals of operations research*, 134(1): 19–67.

[52] Dong, X., and Yang, Y. 2020. Nas-bench-201: Extending the Scope of Reproducible Neural Architecture Search. *arXiv preprint arXiv:2001.00326*.

[53] Dunning, T. H. 1989. Gaussian Basis Sets for Use in Correlated Molecular Calculations. I. The Atoms Boron through Neon and Hydrogen. *J. Chem. Phys.*, 90(2): 1007–1023.

[54] Egan, C. K., Bizzarro, B. B., Riera, M., and Paesani, F. 2020. Nature of Alkali Ion–Water Interactions: Insights from Many-Body Representations and Density Functional Theory. II. *J. Chem. Theory Comput.*, 16(5): 3055–3072.

[55] Eriksson, D., Pearce, M., Gardner, J., Turner, R. D., and Poloczek, M. 2019. Scalable Global Optimization via Local Bayesian Optimization. In *Advances in Neural Information Processing Systems*, 5496–5507.

[56] Errington, J. R., and Debenedetti, P. G. 2001. Relationship Between Structural Order and the Anomalies of Liquid Water. *Nature*, 409(6818): 318–321.

[57] Frazier, P. I. 2018. A Tutorial on Bayesian Optimization. *arXiv preprint arXiv:1807.02811*.

[58] Gablonsky, J. M., and Kelley, C. T. 2000. A Locally-Biased Form of the DIRECT Algorithm. Technical report, North Carolina State University. Center for Research in Scientific Computation.

[59] Gallo, P., Amann-Winkel, K., Angell, C. A., Anisimov, M. A., Caupin, F., Chakravarty, C., Lascaris, E., Loerting, T., Panagiotopoulos, A. Z., Russo, J., Sellberg, J. A., Stanley, H. E., Tanaka, H., Vega, C., Xu, L., and Pettersson, L. G. M. 2016. Water: A Tale of Two Liquids. *Chem. Rev.*, 116(13): 7463–7500.

[60] Gao, F., and Han, L. 2012. Implementing the Nelder-Mead Simplex Algorithm with Adaptive Parameters. *Comput. Optim. Appl.*, 51(1): 259–277.

[61] Gardner, J., Guo, C., Weinberger, K., Garnett, R., and Grosse, R. 2017. Discovering and Exploiting Additive Structure for Bayesian Optimization. In *Artificial Intelligence and Statistics*, 1311–1319. PMLR.

[62] Gartner III, T. E., Hunter, K. M., Lambros, E., Caruso, A., Riera, M., Medders, G. R., Panagiotopoulos, A. Z., Debenedetti, P. G., and Paesani, F. 2022. Anomalies and Local Structure of Liquid Water from Boiling to the Supercooled Regime as Predicted by the Many-Body MB-Pol Model. *J. Phys. Chem*, 13(16): 3652–3658.

[63] Gartner III, T. E., Zhang, L., Piaggi, P. M., Car, R., Panagiotopoulos, A. Z., and Debenedetti, P. G. 2020. Signatures of a Liquid–Liquid Transition in an Ab Initio Deep Neural Network Model for Water. *Proc. Natl. Acad. Sci. U.S.A.*, 117(42): 26040–26046.

[64] Gkeka, P., Stoltz, G., Barati Farimani, A., Belkacemi, Z., Ceriotti, M., Chodera, J. D., Dinner, A. R., Ferguson, A. L., Maillet, J.-B., Minoux, H., Christine, P., Pietrucci, F., Silveira, A., Tkatchenko, A., Trstanova, Z., Wiewiora, R., and Lelièvre, T. 2020. Machine Learning Force Fields and Coarse-Grained Variables in Molecular Dynamics: Application to Materials and Biological Systems. *J. Chem. Theory Comput.*, 16(8): 4757–4775.

[65] Glick, Z. L., Metcalf, D. P., Koutsoukas, A., Spronk, S. A., Cheney, D. L., and Sherrill, C. D. 2020. AP-Net: An Atomic-Pairwise Neural Network for Smooth and Transferable Interaction Potentials. *J. Chem. Phys.*, 153(4): 044112.

[66] Goodfellow, I., Bengio, Y., and Courville, A. 2016. *Deep Learning*. The MIT Press. ISBN 0262035618.

[67] Góra, U., Podeszwa, R., Cencek, W., and Szalewicz, K. 2011. Interaction Energies of Large Clusters From Many-Body Expansion. *J. Chem. Phys.*, 135(22): 224102.

[68] Gordon, M. S., Barca, G., Leang, S. S., Poole, D., Rendell, A. P., Galvez Vallejo, J. L., and Westheimer, B. 2020. Novel Computer Architectures and Quantum Chemistry. *J. Phys. Chem. A*, 124(23): 4557–4582.

[69] Grisafi, A., Fabrizio, A., Meyer, B., Wilkins, D. M., Corminboeuf, C., and Ceriotti, M. 2018. Transferable Machine-Learning Model of the Electron Density. *ACS Cent. Sci.*, 5(1): 57–64.

[70] Guo, Z., Lu, D., Yan, Y., Hu, S., Liu, R., Tan, G., Sun, N., Jiang, W., Liu, L., Chen, Y., et al. 2022. Extending the Limit of Molecular Dynamics with Ab Initio Accuracy to 10 Billion Atoms. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 205–218.

[71] Gurobi Optimization, L. 2023. *Gurobi Optimizer Reference Manual*.

[72] Haghighatlari, M., and Hachmann, J. 2019. Advances of Machine Learning in Molecular Modeling and Simulation. *Curr. Opin. Chem. Eng.*, 23: 51–57.

[73] Han, J., Zhang, L., and Car, R. 2018. Deep Potential: A General Representation of a Many-Body Potential Energy Surface. *Commun. Comput. Phys.*, 23(3): 629–639.

[74] Hankins, D., Moskowitz, J. W., and Stillinger, F. H. 1970. Water Molecule Interactions. *J. Chem. Phys.*, 53(12): 4544–4554.

[75] Hansen, K., Biegler, F., Ramakrishnan, R., Pronobis, W., Von Lilienfeld, O. A., Müller, K.-R., and Tkatchenko, A. 2015. Machine Learning Predictions of Molecular Properties: Accurate Many-Body Potentials and Nonlocality in Chemical Space. *J. Phys. Chem. Lett*, 6(12): 2326–2331.

[76] Hansen, N., Akimoto, Y., and Baudis, P. 2019. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634.

[77] Hansen, N., yoshihikoueno, ARF1, Kadlecová, G., Nozawa, K., Rolshoven, L., Chan, M., Akimoto, Y., brieglhostis, and Brockhoff, D. 2023. CMA-ES/pycma: r3.3.0.

[78] Heindel, J. P., Herman, K. M., Apra, E., and Xantheas, S. S. 2021. Guest–Host Interactions in Clathrate Hydrates: Benchmark MP2 and CCSD (T)/CBS Binding Energies of CH_4, CO_2, and H_2S in $(H_2O)_{20}$ Cages. *J. Phys. Chem. Lett.*, 12(31): 7574–7582.

[79] Henderson, D., Jacobson, S. H., and Johnson, A. W. 2003. *The Theory and Practice of Simulated Annealing*, 287–319. Boston, MA: Springer US.

[80] Hickey, T., Ju, Q., and Van Emden, M. H. 2001. Interval Arithmetic: From Principles to Implementation. *Journal of the ACM (JACM)*, 48(5): 1038–1068.

[81] Hill, J. G., and Peterson, K. A. 2017. Gaussian Basis Sets for Use in Correlated Molecular Calculations. XI. Pseudopotential-Based and All-Electron Relativistic Basis Sets for Alkali Metal (K–Fr) and Alkaline Earth (Ca–Ra) Elements. *J. Chem. Phys.*, 147(24): 244106.

[82] Hill, J. G., Peterson, K. A., Knizia, G., and Werner, H.-J. 2009. Extrapolating MP2 and CCSD Explicitly Correlated Correlation Energies to the Complete Basis Set Limit With First and Second Row Correlation Consistent Basis Sets. *J. Chem. Phys.*, 131(19): 194105.

[83] Huang, X., Braams, B. J., and Bowman, J. M. 2006. Ab Initio Potential Energy and Dipole Moment Surfaces of $(H_2O)_2$. *J. Phys. Chem. A*, 110(2): 445–451.

[84] Hunter, K. M., Shakib, F. A., and Paesani, F. 2018. Disentangling Coupling Effects in the

Infrared Spectra of Liquid Water. *J. Phys. Chem. B*, 122(47): 10754–10761.

[85] Huo, H., and Rupp, M. 2017. Unified Representation for Machine Learning of Molecules and Crystals. *arXiv:1704.06439*.

[86] Imbalzano, G., Anelli, A., Giofré, D., Klees, S., Behler, J., and Ceriotti, M. 2018. Automatic Selection of Atomic Fingerprints and Reference Configurations for Machine-Learning Potentials. *J. Chem. Phys.*, 148(24): 241730.

[87] Inakollu, V. S., Geerke, D. P., Rowley, C. N., and Yu, H. 2020. Polarisable Force Fields: What Do They Add in Biomolecular Simulations? *Curr. Opin. Struct. Biol.*, 61: 182–190.

[88] Jacobson, L. D., Stevenson, J. M., Ramezanghorbani, F., Ghoreishi, D., Leswing, K., Harder, E. D., and Abel, R. 2022. Transferable Neural Network Potential Energy Surfaces for Closed-Shell Organic Molecules: Extension to Ions. *J. Chem. Theory Comput.*, 18(4): 2354–2366.

[89] Jain, P., Kar, P., et al. 2017. Non-convex Optimization for Machine Learning. *Foundations and Trends® in Machine Learning*, 10(3-4): 142–363.

[90] Janet, J. P., and Kulik, H. J. 2017. Predicting Electronic Structure Properties of Transition Metal Complexes with Neural Networks. *Chem. Sci.*, 8(7): 5137–5152.

[91] Jensen, F. 2017. *Introduction to Computational Chemistry*. John wiley & sons.

[92] Jing, Z., Liu, C., Cheng, S. Y., Qi, R., Walker, B. D., Piquemal, J.-P., and Ren, P. 2019. Polarizable Force Fields for Biomolecular Simulations: Recent Advances and Applications. *Annu. Rev. Biophys.*, 48: 371–394.

[93] Jones, D. R., Schonlau, M., and Welch, W. J. 1998. Efficient Global Optimization of Expensive Black-box Functions. *Journal of Global optimization*, 13(4): 455–492.

[94] Kamei, Y., Monden, A., Matsumoto, S., Kakimoto, T., and Matsumoto, K.-i. 2007. The Effects of Over and under Sampling on Fault-Prone Module Detection. In *First international symposium on empirical software engineering and measurement (ESEM 2007)*, 196–204. IEEE.

[95] Karplus, M. 2014. Development of Multiscale Models for Complex Chemical Systems: From H + H_2 to Biomolecules (Nobel Lecture). *Angew. Chem. Int. Ed.*, 53(38): 9992–10005.

[96] Kell, G. S. 1970. Isothermal Compressibility of Liquid Water at 1 atm. *J. Chem. Eng. Data*, 15(1): 119–122.

[97] Kendall, R. A., Dunning, T. H., and Harrison, R. J. 1992. Electron Affinities of the First-Row Atoms Revisited. Systematic Basis Sets and Wave Functions. *J. Chem. Phys.*, 96(9): 6796–6806.

[98] Kim, B., Lee, K., Lim, S., Kaelbling, L., and Lozano-Perez, T. 2020. Monte Carlo Tree Search in Continuous Spaces Using Voronoi Optimistic Optimization with Regret Bounds. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(06): 9916–9924.

[99] Kim, K. H., Amann-Winkel, K., Giovambattista, N., Späh, A., Perakis, F., Pathak, H., Parada, M. L., Yang, C., Mariedahl, D., Eklund, T., Lane, T. J., You, S., Jeong, S., Weston, M., Lee, J. H., Eom, I., Kim, M., Park, J., Chun, S. H., Poole, P. H., and Nilsson, A. 2020. Experimental Observation of the Liquid-Liquid Transition in Bulk Supercooled Water under Pressure. *Science*, 370(6519): 978–982.

[100] Kim, K. H., Späh, A., Pathak, H., Perakis, F., Mariedahl, D., Amann-Winkel, K., Sellberg, J. A., Lee, J. H., Kim, S., Park, J., Katayama, T., and Nilsson, A. 2017. Maxima in the Thermodynamic Response and Correlation Functions of Deeply Supercooled Water. *Science*, 358(6370): 1589–1593.

[101] Kimura, S., Ide, K., Kashihara, A., Kano, M., Hatakeyama, M., Masui, R., Nakagawa, N., Yokoyama, S., Kuramitsu, S., and Konagaya, A. 2005. Inference of S-system Models of Genetic Networks using a Cooperative Coevolutionary Algorithm. *Bioinformatics*, 21(7): 1154–1163.

[102] Knizia, G., Adler, T. B., and Werner, H.-J. 2009. Simplified CCSD(T)-F12 Methods: Theory and Benchmarks. *J. Chem. Phys.*, 130(5): 054104.

[103] Ko, T. W., Finkler, J. A., Goedecker, S., and Behler, J. 2021. A Fourth-Generation High-Dimensional Neural Network Potential with Accurate Electrostatics Including Non-Local Charge Transfer. *Nat. Commun.*, 12(1): 1–11.

[104] Kolda, T. G., Lewis, R. M., and Torczon, V. 2003. Optimization by Direct Search: New Perspectives on some Classical and Modern Methods. *SIAM review*, 45(3): 385–482.

[105] Kotsiantis, S., Kanellopoulos, D., Pintelas, P., et al. 2006. Handling Imbalanced Datasets: A Review. *GESTS international transactions on computer science and engineering*, 30(1): 25–36.

[106] Kotsiantis, S. B. 2007. Supervised Machine Learning: A Review of Classification Techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, 3–24. Amsterdam, The Netherlands, The Netherlands: IOS Press. ISBN 978-1-58603-780-2.

[107] Kringle, L., Thornley, W. A., Kay, B. D., and Kimmel, G. A. 2020. Reversible Structural Transformations in Supercooled Liquid Water from 135 to 245 K. *Science*, 369(6510): 1490–1492.

[108] Lambros, E., Dasgupta, S., Palos, E., Swee, S., Hu, J., and Paesani, F. 2021. General Many-Body Framework for Data-Driven Potentials with Arbitrary Quantum Mechanical Accuracy: Water as a Case Study. *J. Chem. Theory Comput.*, 17(9): 5635–5650.

[109] Lambros, E., Hu, J., and Paesani, F. 2021. Assessing the Accuracy of the SCAN Functional for Water through a Many-Body Analysis of the Adiabatic Connection Formula. *J. Chem. Theory Comput.*, 17(6): 3739–3749.

[110] Lambros, E., and Paesani, F. 2020. How Good Are Polarizable and Flexible Models for Water: Insights from a Many-Body Perspective. *J. Chem. Phys.*, 153(6): 060901.

[111] Lavezzi, G., Guye, K., and Ciarcià, M. 2022. Nonlinear Programming Solvers for Unconstrained and Constrained Optimization Problems: a Benchmark Analysis.

[112] Lee, E. Y., Fulan, B. M., Wong, G. C., and Ferguson, A. L. 2016. Mapping Membrane Activity in Undiscovered Peptide Sequence Space Using Machine Learning. *Proc. Natl. Acad. Sci. U.S.A.*, 113(48): 13588–13593.

[113] Levitt, M. 2014. Birth and Future of Multiscale Modeling for Macromolecular Systems (Nobel Lecture). *Angew. Chem. Int. Ed.*, 53(38): 10006–10018.

[114] Lewis, R. M., Torczon, V., and Trosset, M. W. 2000. Direct Search Methods: Then and Now. *Journal of computational and Applied Mathematics*, 124(1-2): 191–207.

[115] Liang, W., Lu, G., and Yu, J. 2021. Machine-Learning-Driven Simulations on Microstructure and Thermophysical Properties of $MgCl_2$–KCl Eutectic. *ACS Appl. Mater. Interfaces*, 13(3): 4034–4042.

[116] Lifson, S., and Warshel, A. 1968. Consistent Force Field for Calculations of Conformations, Vibrational Spectra, and Enthalpies of Cycloalkane and n-Alkane Molecules. *J. Chem. Phys.*, 49(11): 5116–5129.

[117] Lim, I. S., Schwerdtfeger, P., Metz, B., and Stoll, H. 2005. All-Electron and Relativistic Pseudopotential Studies for the Group 1 Element Polarizabilities from K to Element 119. *J. Chem. Phys.*, 122(10): 104103.

[118] Linstrom, P. J., and Mallard, W. G. 2001. The NIST Chemistry WebBook: A Chemical Data Resource on the Internet. *J. Chem. Eng. Data*, 46(5): 1059–1063.

[119] Liu, X., and Lu, P. 2014. Solving Nonconvex Optimal Control Problems by Convex

Optimization. *Journal of Guidance, Control, and Dynamics*, 37(3): 750–765.

[120] Lu, D., Wang, H., Chen, M., Lin, L., Car, R., Weinan, E., Jia, W., and Zhang, L. 2021. 86 PFLOPS Deep Potential Molecular Dynamics simulation of 100 million atoms with Ab Initio accuracy. *Comput. Phys. Commun.*, 259: 107624.

[121] Mallory, J. D., and Mandelshtam, V. A. 2016. Diffusion Monte Carlo Studies of MB-Pol (H2O) 2- 6 and (D2O) 2- 6 Clusters: Structures and Binding Energies. *J. Chem. Phys.*, 145(6): 064308.

[122] Manna, D., Kesharwani, M. K., Sylvetsky, N., and Martin, J. M. 2017. Conventional and Explicitly Correlated Ab Initio Benchmark Study on Water Clusters: Revision of the BEGDB and WATER27 Data Sets. *J. Chem. Theory Comput.*, 13(7): 3136–3152.

[123] Manzhos, S., and Carrington Jr., T. 2020. Neural Network Potential Energy Surfaces for Small Molecules and Reactions. *Chem. Rev.*, 121(16): 10187–10217.

[124] Martyna, G. J., Hughes, A., and Tuckerman, M. E. 1999. Molecular Dynamics Algorithms for Path Integrals at Constant Pressure. *J. Chem. Phys.*, 110(7): 3275–3290.

[125] Mayo, S. L., Olafson, B. D., and Goddard, W. A. 1990. DREIDING: A Generic Force Field for Molecular Simulations. *J. Phys. Chem.*, 94(26): 8897–8909.

[126] Medders, G. R., Babin, V., and Paesani, F. 2014. Development of a "First-Principles" Water Potential with Flexible Monomers. III. Liquid Phase Properties. *J. Chem. Theory Comput.*, 10(8): 2906–2910.

[127] Medders, G. R., Götz, A. W., Morales, M. A., Bajaj, P., and Paesani, F. 2015. On the Representation of Many-Body Interactions in Water. *J. Chem. Phys.*, 143(10): 104102.

[128] Medders, G. R., and Paesani, F. 2015. Infrared and Raman Spectroscopy of Liquid Water through "First-Principles" Many-Body Molecular Dynamics. *J. Chem. Theory Comput.*, 11(3): 1145–1154.

[129] Medders, G. R., and Paesani, F. 2016. Dissecting the Molecular Structure of the Air/Water Interface from Quantum Simulations of the Sum-Frequency Generation Spectrum. *J. Am. Chem. Soc.*, 138(11): 3912–3919.

[130] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. 1953. Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.*, 21(6): 1087–1092.

[131] Mistakidis, E. S., and Stavroulakis, G. E. 2013. *Nonconvex Optimization in Mechanics: Algorithms, Heuristics and Engineering Applications by the FEM*, volume 21. Springer

Science & Business Media.

[132] Moberg, D. R., Becker, D., Dierking, C. W., Zurheide, F., Bandow, B., Buck, U., Hudait, A., Molinero, V., Paesani, F., and Zeuch, T. 2019. The End of Ice I. *Proc. Natl. Acad. Sci. U.S.A.*, 116(49): 24413–24419.

[133] Moberg, D. R., Sharp, P. J., and Paesani, F. 2018. Molecular-Level Interpretation of Vibrational Spectra of Ordered Ice Phases. *J. Phys. Chem. B*, 122(46): 10572–10581.

[134] Moberg, D. R., Straight, S. C., Knight, C., and Paesani, F. 2017. Molecular Origin of the Vibrational Structure of Ice Ih. *J. Phys. Chem. Lett.*, 8(12): 2579–2583.

[135] Moberg, D. R., Straight, S. C., and Paesani, F. 2018. Temperature Dependence of the Air/Water Interface Revealed by Polarization Sensitive Sum-Frequency Generation Spectroscopy. *J. Phys. Chem. B*, 122(15): 4356–4365.

[136] Montavon, G., Rupp, M., Gobre, V., Vazquez-Mayagoitia, A., Hansen, K., Tkatchenko, A., Müller, K.-R., and Von Lilienfeld, O. A. 2013. Machine Learning of Molecular Electronic Properties in Chemical Compound Space. *New J. Phys.*, 15(9): 095003.

[137] Muniz, M. C., Gartner III, T. E., Riera, M., Knight, C., Yue, S., Paesani, F., and Panagiotopoulos, A. Z. 2021. Vapor–Liquid Equilibrium of Water with the MB-pol Many-Body Potential. *J. Chem. Phys.*, 154(21): 211103.

[138] Munos, R. 2011. Optimistic Optimization of a Deterministic Function without the Knowledge of its Smoothness. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc.

[139] Mutnỳ, M., and Krause, A. 2019. Efficient High Dimensional Bayesian Optimization with Additivity and Quadrature Fourier fFeatures. *Advances in Neural Information Processing Systems 31*, 9005–9016.

[140] Nesbet, R. 1969. Atomic Bethe-Goldstone Equations. *Adv. Chem. Phys.*, 1–34.

[141] Nguyen, T. T., Székely, E., Imbalzano, G., Behler, J., Csányi, G., Ceriotti, M., Götz, A. W., and Paesani, F. 2018. Comparison of Permutationally Invariant Polynomials, Neural Networks, and Gaussian Approximation Potentials in Representing Water Interactions through Many-Body Expansions. *J. Chem. Phys.*, 148(24): 241725.

[142] Ninin, J. 2016. Global Optimization Based on Contractor Programming: An Overview of the IBEX Library. In *Mathematical Aspects of Computer and Information Sciences: 6th International Conference, MACIS 2015, Berlin, Germany, November 11-13, 2015, Revised Selected Papers 6*, 555–559. Springer.

[143] Niu, H., Bonati, L., Piaggi, P. M., and Parrinello, M. 2020. Ab Initio Phase Diagram and Nucleation of Gallium. *Nat. Commun.*, 11(1): 1–9.

[144] Noé, F., Olsson, S., Köhler, J., and Wu, H. 2019. Boltzmann Generators: Sampling Equilibrium States of Many-Body Systems with Deep Learning. *Science*, 365(6457): eaaw1147.

[145] Noé, F., Tkatchenko, A., Müller, K.-R., and Clementi, C. 2020. Machine Learning for Molecular Simulation. *Annu. Rev. Phys. Chem.*, 71: 361–390.

[146] Oh, C., Gavves, E., and Welling, M. 2018. Bock: Bayesian Optimization with Cylindrical Kernels. In *International Conference on Machine Learning*, 3868–3877. PMLR.

[147] Olson, B., Hashmi, I., Molloy, K., and Shehu, A. 2012. Basin Hopping as a General and Versatile Optimization Framework for the Characterization of Biological Macromolecules. *Advances in Artificial Intelligence*, 2012: 3–3.

[148] Paesani, F. 2016. Getting the Right Answers for the Right Reasons: Toward Predictive Molecular Simulations of Water with Many-Body Potential Energy Functions. *Acc. Chem. Res.*, 49(9): 1844–1851.

[149] Paesani, F. 2020. Water: Many-Body Potential from First Principles (from the Gas to the Liquid Phase). In Andreoni, W., and Yip, S., eds., *Handbook of Materials Modeling: Methods: Theory and Modeling*, 635–660. Springer.

[150] Paesani Research Group. 2022. MBX: A Many-Body Energy and Force Calculator. available at https://github.com/paesanilab/MBX.

[151] Palos, E., Lambros, E., Swee, S., Hu, J., Dasgupta, S., and Paesani, F. 2022. Assessing the Interplay between Functional-Driven and Density-Driven Errors in DFT Models of Water. *J. Chem. Theory Comput.*, 18(6): 3410–3426.

[152] Pant, M., Zaheer, H., Garcia-Hernandez, L., Abraham, A., et al. 2020. Differential Evolution: A Review of More than Two Decades of Research. *Engineering Applications of Artificial Intelligence*, 90: 103479.

[153] Parr, R. G. 1980. Density Functional Theory of Atoms and Molecules. In *Horizons of Quantum Chemistry*, 5–15. Springer.

[154] Partridge, H., and Schwenke, D. W. 1997. The Determination of an Accurate Isotope Dependent Potential Energy Surface for Water from Extensive Ab Initio Calculations and Experimental Data. *J. Chem. Phys.*, 106(11): 4618–4639.

[155] Pathak, H., Späh, A., Esmaeildoost, N., Sellberg, J. A., Kim, K. H., Perakis, F., Amann-

Winkel, K., Ladd-Parada, M., Koliyadu, J., Lane, T. J., Yang, C., Lemke, H. T., Oggenfuss, A. R., Johnson, P. J. M., Deng, Y., Zerdane, S., Mankowsky, R., Beaud, P., and Nilsson, A. 2021. Enhancement and Maximum in the Isobaric Specific-Heat Capacity Measurements of Deeply Supercooled Water Using Ultrafast Calorimetry. *Proc. Natl. Acad. Sci. U.S.A.*, 118(6): e2018379118.

[156] Pham, C. H., Reddy, S. K., Chen, K., Knight, C., and Paesani, F. 2017. Many-Body Interactions in Ice. *J. Chem. Theory Comput.*, 13(4): 1778–1784.

[157] Piaggi, P. M., Panagiotopoulos, A. Z., Debenedetti, P. G., and Car, R. 2021. Phase Equilibrium of Water with Hexagonal and Cubic Ice Using the Scan Functional. *J. Chem. Theory Comput.*, 17(5): 3065–3077.

[158] Pinheiro, M., Ge, F., Ferré, N., Dral, P. O., and Barbatti, M. 2021. Choosing the Right Molecular Machine Learning Potential. *Chem. Sci.*, 12(43): 14396–14413.

[159] Ponder, J. W., Wu, C., Ren, P., Pande, V. S., Chodera, J. D., Schnieders, M. J., Haque, I., Mobley, D. L., Lambrecht, D. S., DiStasio Jr, R. A., Head-Gordon, M., Clark, G. N. I., Johnson, M. E., and Head-Gordon, T. 2010. Current Status of the AMOEBA Polarizable Force Field. *J. Phys. Chem. B*, 114(8): 2549–2564.

[160] Puranik, Y., and Sahinidis, N. V. 2017. Bounds Tightening Based on Optimality Conditions for Nonconvex Box-Constrained Optimization. *Journal of Global Optimization*, 67: 59–77.

[161] Rahman, A., and Stillinger, F. H. 1971. Molecular Dynamics Study of Liquid Water. *J. Chem. Phys.*, 55(7): 3336–3359.

[162] Rao, S. S. 2019. *Engineering Optimization: Theory and Practice*. John Wiley & Sons.

[163] Rappé, A. K., Casewit, C. J., Colwell, K. S., Goddard III, W. A., and Skiff, W. M. 1992. UFF, a Full Periodic Table Force Field for Molecular Mechanics and Molecular Dynamics Simulations. *J. Am. Chem. Soc.*, 114(25): 10024–10035.

[164] Rasmussen, C. E. 2003. Gaussian Processes in Machine Learning. In *Summer school on machine learning*, 63–71. Springer.

[165] Rasmussen, C. E., and Williams, C. K. I. 2006. *Gaussian Process for Machine Learning*. MIT Press. ISBN 026218253X.

[166] Reddy, S. K., Moberg, D. R., Straight, S. C., and Paesani, F. 2017. Temperature-Dependent Vibrational Spectra and Structure of Liquid Water from Classical and Quantum Simulations with the MB-pol Potential Energy Function. *J. Chem. Phys.*, 147(24): 244504.

[167] Reddy, S. K., Straight, S. C., Bajaj, P., Huy Pham, C., Riera, M., Moberg, D. R., Morales, M. A., Knight, C., Götz, A. W., and Paesani, F. 2016. On the Accuracy of the MB-pol Many-Body Potential for Water: Interaction Energies, Vibrational Frequencies, and Classical Thermodynamic and Dynamical Properties from Clusters to Liquid Water and Ice. *J. Chem. Phys.*, 145(19): 194504.

[168] Rezac, J., and Hobza, P. 2016. Benchmark Calculations of Interaction Energies in Noncovalent Complexes and Their Applications. *Chem. Rev.*, 116(9): 5038–5071.

[169] Richardson, J. O., Pérez, C., Lobsiger, S., Reid, A. A., Temelso, B., Shields, G. C., Kisiel, Z., Wales, D. J., Pate, B. H., and Althorpe, S. C. 2016. Concerted Hydrogen-Bond Breaking by Quantum Tunneling in the Water Hexamer Prism. *Science*, 351(6279): 1310–1313.

[170] Riera, M., Hirales, A., Ghosh, R., and Paesani, F. 2020. Data-Driven Many-Body Models with Chemical Accuracy for CH_4/H_2O Mixtures. *J. Phys. Chem. A*, 124(49): 11207–11221.

[171] Riera, M., Lambros, E., Nguyen, T. T., Götz, A. W., and Paesani, F. 2019. Low-Order Many-Body Interactions Determine the Local Structure of Liquid Water. *Chem. Sci.*, 10(35): 8211–8218.

[172] Riera, M., Mardirossian, N., Bajaj, P., Götz, A. W., and Paesani, F. 2017. Toward Chemical Accuracy in the Description of Ion–Water Interactions Through Many-Body Representations. Alkali-Water Dimer Potential Energy Surfaces. *J. Chem. Phys.*, 147(16): 161715.

[173] Riera, M., Yeh, E. P., and Paesani, F. 2020. Data-Driven Many-Body Models for Molecular Fluids: CO_2/H_2O Mixtures as a Case Study. *J. Chem. Theory Comput.*, 16(4): 2246–2257.

[174] Rolland, P., Scarlett, J., Bogunovic, I., and Cevher, V. 2018. High-dimensional Bayesian Optimization via Additive Models with Overlapping Groups. In *International conference on artificial intelligence and statistics*, 298–307. PMLR.

[175] Rupp, M., Tkatchenko, A., Müller, K.-R., and Von Lilienfeld, O. A. 2012. Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning. *Phys. Rev. Lett.*, 108(5): 058301.

[176] Sahinidis, N. 2023. *BARON: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual. The Optimization Firm, LLC, niksah@minlp.com.

[177] Salomon, R. 1998. Evolutionary Algorithms and Gradient Search: Similarities and Differences. *IEEE Transactions on Evolutionary Computation*, 2(2): 45–55.

[178] Samala, N. R., and Agmon, N. 2019. Temperature Dependence of Intramolecular Vibrational Bands in Small Water Clusters. *J. Phys. Chem. B*, 123(44): 9428–9442.

[179] Samala, N. R., and Agmon, N. 2019. Thermally Induced Hydrogen-Bond Rearrangements in Small Water Clusters and the Persistent Water Tetramer. *ACS Omega*.

[180] Schienbein, P., and Marx, D. 2017. Liquid–Vapor Phase Diagram of RPBE-D3 Water: Electronic Properties along the Coexistence Curve and in the Supercritical Phase. *J. Phys. Chem. B*, 122(13): 3318–3329.

[181] Schmidt, M., and Roy, P.-N. 2018. Path Integral Molecular Dynamic Simulation of Flexible Molecular Systems in Their Ground State: Application to the Water Dimer. *J. Chem Phys.*, 148(12): 124116.

[182] Schran, C., Behler, J., and Marx, D. 2019. Automated Fitting of Neural Network Potentials at Coupled Cluster Accuracy: Protonated Water Clusters as Testing Ground. *J. Chem. Theory Comput.*, 16(1): 88–99.

[183] Schran, C., Brezina, K., and Marsalek, O. 2020. Committee Neural Network Potentials Control Generalization Errors and Enable Active Learning. *J. Chem. Phys.*, 153(10): 104105.

[184] Schran, C., Thiemann, F. L., Rowe, P., Müller, E. A., Marsalek, O., and Michaelides, A. 2021. Machine Learning Potentials for Complex Aqueous Systems Made Simple. *Proc. Natl. Acad. Sci. U.S.A.*, 118(38): e2110077118.

[185] Segler, M. H., Preuss, M., and Waller, M. P. 2018. Planning Chemical Syntheses with Deep Neural Networks and Symbolic AI. *Nature*, 555(7698): 604–610.

[186] Sengupta, S., Moberg, D. R., Paesani, F., and Tyrode, E. 2018. Neat Water–Vapor Interface: Proton Continuum and the Nonresonant Background. *J. Phys. Chem. Lett.*, 9(23): 6744–6749.

[187] Settles, B. 2009. Active Learning Literature Survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

[188] Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and de Freitas, N. 2016. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1): 148–175.

[189] Shinoda, W., Shiga, M., and Mikami, M. 2004. Rapid Estimation of Elastic Constants by Molecular Dynamics Simulation under Constant Stress. *Phys. Rev. B*, 69(13): 134103.

[190] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G.,

Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *nature*, 529(7587): 484–489.

[191] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. 2017. Mastering the Game of Go without Human Knowledge. *Nature*, 550: 354–.

[192] Sim, E., Song, S., Vuckovic, S., and Burke, K. 2022. Improving Results by Improving Densities: Density-Corrected Density Functional Theory. *J. Am. Chem. Soc.*, 144(15): 6625–6639.

[193] Sioshansi, R., Conejo, A. J., et al. 2017. Optimization in Engineering. *Cham: Springer International Publishing*, 120.

[194] Smith, J. S., Isayev, O., and Roitberg, A. E. 2017. ANI-1, A Data Set of 20 Million Calculated Off-Equilibrium Conformations for Organic Molecules. *Sci. Data*, 4(1): 1–8.

[195] Smith, J. S., Isayev, O., and Roitberg, A. E. 2017. ANI-1: An Extensible Neural Network Potential with DFT Accuracy at Force Field Computational Cost. *Chem Sci.*, 8(4): 3192–3203.

[196] Smith, J. S., Nebgen, B., Lubbers, N., Isayev, O., and Roitberg, A. E. 2018. Less Is More: Sampling Chemical Space with Active Learning. *The Journal of chemical physics*, 148(24): 241733.

[197] Smith, W., and Forester, T. 1996. DL_POLY_2. 0: A General-Purpose Parallel Molecular Dynamics Simulation Package. *J. Mol. Graph.*, 14(3): 136–141.

[198] Sommers, G. M., Andrade, M. F. C., Zhang, L., Wang, H., and Car, R. 2020. Raman Spectrum and Polarizability of Liquid Water from Deep Neural Networks. *Phys. Chem. Chem. Phys.*, 22(19): 10592–10602.

[199] Song, S., Vuckovic, S., Sim, E., and Burke, K. 2022. Density-Corrected DFT Explained: Questions and Answers. *J. Chem. Theory Comput.*, 18(2): 817–827.

[200] Speedy, R., and Angell, C. 1976. Isothermal Compressibility of Supercooled Water and Evidence for a Thermodynamic Singularity at -45 ^{o}C. *J. Chem. Phys.*, 65(3): 851–858.

[201] Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. W. 2012. Information-Theoretic Regret Bounds for Gaussian Process Optimization in the Bandit Setting. *IEEE Transactions on Information Theory*, 58(5): 3250–3265.

[202] Storn, R., and Price, K. 1997. Differential Evolution-A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of global optimization*, 11(4): 341.

[203] Sun, S., Tang, F., Imoto, S., Moberg, D. R., Ohto, T., Paesani, F., Bonn, M., Backus, E. H., and Nagata, Y. 2018. Orientational Distribution of Free OH Groups of Interfacial Water Is Exponential. *Phys. Rev. Lett.*, 121(24): 246101.

[204] Sun, Z., Zheng, L., Chen, M., Klein, M. L., Paesani, F., and Wu, X. 2018. Electron-Hole Theory of the Effect of Quantum Nuclei on the X-ray Absorption Spectra of Liquid Water. *Phys. Rev. Lett.*, 121(13): 137401.

[205] Tallorin, L., Wang, J., Kim, W. E., Sahu, S., Kosa, N. M., Yang, P., Thompson, M., Gilson, M. K., Frazier, P. I., Burkart, M. D., et al. 2018. Discovering De Novo Peptide Substrates for Enzymes Using Machine Learning. *Nat. Commun.*, 9(1): 1–10.

[206] Tawarmalani, M., and Sahinidis, N. V. 2005. A Polyhedral Branch-and-Cut Approach to Global Optimization. *Mathematical Programming*, 103: 225–249.

[207] The Optimization Firm. 2023. NLP and MINLP Test Problems. https://minlp.com/nlp-and-minlp-test-problems. Accessed: Aug.10, 2023.

[208] Thompson, A. P., Aktulga, H. M., Berger, R., Bolintineanu, D. S., Brown, W. M., Crozier, P. S., in 't Veld, P. J., Kohlmeyer, A., Moore, S. G., Nguyen, T. D., Shan, R., Stevens, M. J., Tranchida, J., Trott, C., and Plimpton, S. J. 2022. LAMMPS - A Flexible Simulation Tool for Particle-Based Materials Modeling at the Atomic, Meso, and Continuum Scales. *Comput. Phys. Commun.*, 271: 108171.

[209] Tikhonov, A. N. 1963. Solution of Incorrectly Formulated Problems and the Regularization Method. *Soviet Math.*, 4: 1035–1038.

[210] Todorov, E., Erez, T., and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. IEEE.

[211] Unke, O. T., Chmiela, S., Sauceda, H. E., Gastegger, M., Poltavsky, I., Schütt, K. T., Tkatchenko, A., and Müller, K.-R. 2021. Machine Learning Force Fields. *Chem. Rev.*, 121(16): 10142–10186.

[212] Vaillant, C. L., and Cvitaš, M. T. 2018. Rotation-Tunneling Spectrum of the Water Dimer from Instanton Theory. *Phys. Chem. Chem. Phys.*, 20(42): 26809–26813.

[213] Vaillant, C. L., Wales, D. J., and Althorpe, S. C. 2018. Tunneling Splittings from Path-Integral Molecular Dynamics Using a Langevin Thermostat. *J. Chem. Phys.*, 148(23): 234102.

[214] Vanderplaats, G. 2002. Very Large Scale Optimization. In *8th Symposium on Multidisciplinary Analysis and Optimization*, 4809.

[215] Vanommeslaeghe, K., Hatcher, E., Acharya, C., Kundu, S., Zhong, S., Shim, J., Darian, E., Guvench, O., Lopes, P., Vorobyov, I., and MacKerell Jr, A. 2010. CHARMM General Force Field: A Force Field for Drug-Like Molecules Compatible with the CHARMM All-Atom Additive Biological Force Fields. *J. Comp. Chem.*, 31(4): 671–690.

[216] Vanommeslaeghe, K., and MacKerell Jr, A. 2015. CHARMM Additive and Polarizable Force Fields for Biophysics and Computer-Aided Drug Design. *Biochim. Biophys. Acta Gen. Subj.*, 1850(5): 861–871.

[217] Videla, P., Rossky, P., and Laria, D. 2016. Communication: Isotopic Effects on Tunneling Motions in the Water Trimer. *J. Chem. Phys.*, 144(6): 061101–061101.

[218] Videla, P. E., Rossky, P. J., and Laria, D. 2018. Isotopic Equilibria in Aqueous Clusters at Low Temperatures: Insights from the MB-Pol Many-Body Potential. *J. Chem. Phys.*, 148(8): 084303.

[219] Vuckovic, S., Song, S., Kozlowski, J., Sim, E., and Burke, K. 2019. Density Functional Analysis: The Theory of Density-Corrected DFT. *J. Chem. Theory Comput.*, 15(12): 6636–6646.

[220] Wang, H., Zhang, L., Han, J., and Weinan, E. 2018. DeePMD-kit: A Deep Learning Package for Many-Body Potential Energy Representation and Molecular Dynamics. *Comput. Phys. Commun.*, 228: 178–184.

[221] Wang, J., Olsson, S., Wehmeyer, C., Pérez, A., Charron, N. E., De Fabritiis, G., Noé, F., and Clementi, C. 2019. Machine Learning of Coarse-Grained Molecular Dynamics Force Fields. *ACS Cent. Sci.*, 5(5): 755–767.

[222] Wang, J., Wolf, R. M., Caldwell, J. W., Kollman, P. A., and Case, D. A. 2004. Development and Testing of a General AMBER Force Field. *J. Comp. Chem.*, 25(9): 1109–1213.

[223] Wang, L., Fonseca, R., and Tian, Y. 2020. Learning Search Space Partition for Black-Box Optimization Using Monte Carlo Tree Search. *Advances in Neural Information Processing Systems*, 33: 19511–19522.

[224] Wang, Y., and Bowman, J. M. 2011. Ab Initio Potential and Dipole Moment Surfaces for Water. II. Local-Monomer Calculations of the Infrared Spectra of Water Clusters. *J. Chem. Phys.*, 134(15): 154510.

[225] Wang, Y., Huang, X., Shepler, B. C., Braams, B. J., and Bowman, J. M. 2011. Flexible, Ab Initio Potential, and Dipole Moment Surfaces for Water. I. Tests and Applications for

Clusters up to the 22-Mer. *J. Chem. Phys.*, 134(9): 094509.

[226] Wang, Y., Shepler, B. C., Braams, B. J., and Bowman, J. M. 2009. Full-Dimensional, Ab Initio Potential Energy and Dipole Moment Surfaces for Water. *J. Chem. Phys.*, 131(5): 054511.

[227] Warshel, A. 2014. Multiscale Modeling of Biological Functions: From Enzymes to Molecular Machines (Nobel Lecture). *Angew. Chem. Int. Ed.*, 53(38): 10020–10031.

[228] Warshel, A., and Lifson, S. 1970. Consistent Force Field Calculations. II. Crystal Structures, Sublimation Energies, Molecular and Lattice Vibrations, Molecular Conformations, and Enthalpies of Alkanes. *J. Chem. Phys.*, 53(2): 582–594.

[229] Wen, T., Wang, R., Zhu, L., Zhang, L., Wang, H., Srolovitz, D. J., and Wu, Z. 2021. Specialising Neural Network Potentials for Accurate Properties and Application to the Mechanical Response of Titanium. *npj Comput. Mater.*, 7(1): 1–11.

[230] Wohlfahrt, O., Dellago, C., and Sega, M. 2020. Ab Initio Structure and Thermodynamics of the RPBE-D3 Water/Vapor Interface by Neural-Network Molecular Dynamics. *J. Chem. Phys.*, 153(14): 144710.

[231] Woon, D. E., and Dunning, T. H. 1995. Gaussian Basis Sets for Use in Correlated Molecular Calculations. V. Core-Valence Basis Sets for Boron through Neon. *J. Chem. Phys.*, 103(11): 4572–4585.

[232] Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. 2018. MoleculeNet: A Benchmark for Molecular Machine Learning. *Chem. Sci.*, 9(2): 513–530.

[233] Xiang, Y., Gubian, S., Suomela, B., and Hoeng, J. 2013. Generalized Simulated Annealing for Global Optimization: The GenSA Package. *R J.*, 5(1): 13.

[234] Yang, T. 2019. Advancing Non-convex and Constrained Learning: Challenges and Opportunities. *AI Matters*, 5(3): 29–39.

[235] Yang, Z., Sendhoff, B., Tang, K., and Yao, X. 2016. Target Shape Design Optimization by Evolving B-splines with Cooperative Coevolution. *Applied Soft Computing*, 48: 672–682.

[236] Yanover, C., Meltzer, T., Weiss, Y., Bennett, K. P., and Parrado-Hernández, E. 2006. Linear Programming Relaxations and Belief Propagation–An Empirical Study. *Journal of Machine Learning Research*, 7(9).

[237] Yue, S., Muniz, M. C., Calegari Andrade, M. F., Zhang, L., Car, R., and Panagiotopoulos, A. Z. 2021. When Do Short-Range Atomistic Machine-Learning Models Fall Short? *J.*

Chem. Phys., 154(3): 034111.

[238] Yue, S., Riera, M., Ghosh, R., Panagiotopoulos, A. Z., and Paesani, F. 2022. Transferability of Data-Driven, Many-Body Models for CO_2 Simulations in the Vapor and Liquid Phases. *J. Chem. Phys.*, 156(10): 104503.

[239] Zaverkin, V., Holzmüller, D., Schuldt, R., and Kästner, J. 2022. Predicting Properties of Periodic Systems from Cluster Data: A Case Study of Liquid Water. *J. Chem. Phys.*, 156(11): 114103.

[240] Zhai, Y., and Gao, S. 2022. Monte Carlo Tree Descent for Black-Box Optimization. In *Advances in Neural Information Processing Systems*.

[241] Zhang, C., Tang, F., Chen, M., Xu, J., Zhang, L., Qiu, D. Y., Perdew, J. P., Klein, M. L., and Wu, X. 2021. Modeling Liquid Water by Climbing up Jacob's Ladder in Density Functional Theory Facilitated by Using Deep Neural Network Potentials. *J. Phys. Chem. B*, 125(41): 11444–11456.

[242] Zhang, L., Han, J., Wang, H., Car, R., and Weinan, E. 2018. Deep Potential Molecular Dynamics: A Scalable Model with the Accuracy of Quantum Mechanics. *Phys. Rev. Lett.*, 120(14): 143001.

[243] Zhang, L., Han, J., Wang, H., Saidi, W., Car, R., and E, W. 2018. End-to-End Symmetry Preserving Inter-atomic Potential Energy Model for Finite and Extended Systems. *Adv. Neural Inf. Process. Syst.*, 31.

[244] Zhang, L., Wang, H., Car, R., and Weinan, E. 2021. Phase Diagram of a Deep Potential Water Model. *Phys. Rev. Lett.*, 126(23): 236001.

[245] Zhang, Y., Wang, H., Chen, W., Zeng, J., Zhang, L., Wang, H., and Weinan, E. 2020. DP-GEN: A Concurrent Learning Platform for the Generation of Reliable Deep Learning Based Potential Energy Models. *Comput. Phys. Commun.*, 253: 107206.

[246] Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. 1997. Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4): 550–560.

[247] Zhu, J., Wang, H., Yao, T., and Tsou, B. K. 2008. Active Learning with Sampling by Uncertainty and Density for Word Sense Disambiguation and Text Classification. In *22nd International Conference on Computational Linguistics, Coling 2008*, 1137–1144.

[248] Zhuang, D., Riera, M., Schenter, G. K., Fulton, J. L., and Paesani, F. 2019. Many-Body Effects Determine the Local Hydration Structure of Cs^+ in Solution. *J. Phys. Chem. Lett.*, 10(3): 406–412.

[249] Zinn-Justin, J. 2001. Precise Determination of Critical Exponents and Equation of State by Field Theory Methods. *Phys. Rep.*, 344(4-6): 159–178.

[250] Zoph, B., and Le, Q. V. 2016. Neural Architecture Search with Reinforcement Learning. *arXiv preprint arXiv:1611.01578*.

[251] Zubatiuk, T., and Isayev, O. 2021. Development of Multimodal Machine Learning Potentials: Toward a Physics-Aware Artificial Intelligence. *Acc. Chem. Res.*, 54(7): 1575–1585.