ARTICLE

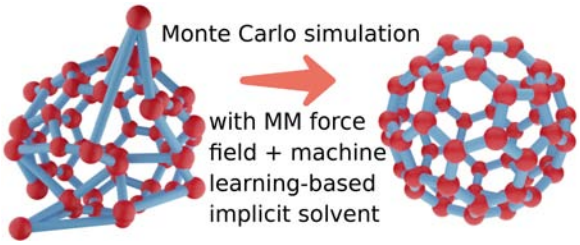
MLIMC: Machine Learning-Based Implicit-Solvent Monte Carlo†

Jiahui Chen[a], Weihua Geng[b*], Guo-Wei Wei[a,c*]

a. Department of Mathematics, Michigan State University, MI 48824, USA
b. Department of Mathematics, Southern Methodist University, Dallas, TX 75275, USA
c. Department of Biochemistry and Molecular Biology, Michigan State University, MI 48824, USA

(Dated: Received on September 2, 2021; Accepted on September 24, 2021)

Monte Carlo (MC) methods are important computational tools for molecular structure optimizations and predictions. When solvent effects are explicitly considered, MC methods become very expensive due to the large degree of freedom associated with the water molecules and mobile ions. Alternatively implicit-solvent MC can largely reduce the computational cost by applying a mean field approximation to solvent effects and meanwhile maintains the atomic detail of the target molecule. The two most popular implicit-solvent models are the Poisson-Boltzmann (PB) model and the Generalized Born (GB) model in a way such that the GB model is an approximation to the PB model but is much faster in simulation time. In this work, we develop a machine learning-based implicit-solvent Monte Carlo (MLIMC) method by combining the advantages of both implicit solvent models in accuracy and efficiency. Specifically, the MLIMC method uses a fast and accurate PB-based machine learning (PBML) scheme to compute the electrostatic solvation free energy at each step. We validate our MLIMC method by using a benzene-water system and a protein-water system. We show that the proposed MLIMC method has great advantages in speed and accuracy for molecular structure optimization and prediction.



Key words: Machine learning, Implicit-solvent Monte Carlo simulation, Poisson-Boltzmann equation, Electrostatics

I. INTRODUCTION

The determination of protein structures is of paramount importance for structural biology and macromolecular study. However, not all protein structures can be determined with available experimental techniques due to various limitations. Computational methods offer important alternative approaches for structural determination and optimization [1]. Indeed, molecular force field models and molecular dynamics [2–4] can generate time-resolved trajectories of protein folding and protein-ligand binding predictions as well as structural ensemble simulations [5]. In these simulations, mathematical models and numerical algorithms are imperative for achieving computational accuracy and efficiency. A large number of advanced algorithms have been developed to reduce the computational cost and improve the accuracy for biomolecular simulations [6–9]. A major difficulty of molecular dynamics is the long timescales associated with real molecular processes taking place in nature. Therefore, ignoring the requirement of having time-resolved trajectories of the molecular processes will immediately remove the difficulty. Indeed, it is sufficient for most studies to have a predicted representative ensemble of structures

†Part of Special Issue "John Z.H. Zhang Festschrift for celebrating his 60th birthday".
*Authors to whom correspondence should be addressed. E-mail: wgeng@mail.smu.edu, weig@msu.edu

for a given process. This representative prediction can be generated by Monte Carlo sampling [10].

Monte Carlo method is one of the most of popular approaches for biomolecular systems. Under physiological condition, biomolecules are immersed in and interact with surrounding water molecules and other possible co-factors. As such, Monte Carlo simulations of a biomolecule have to deal with a large number of solvent water molecules, which makes the simulations very expensive and sometimes, intractable. Additionally, in Monte Carlo simulations, the biomolecular conformation is subject to random perturbations [11]. These perturbations will inevitably result in the overlaps between the biomolecule and explicit solvent molecules, which leads to an unfavorable and non-representative structure. Implicit solvent models, such as Poisson-Boltzmann (PB) [12, 13], polarizable continuum [14, 15] and Generalized Born (GB) methods [16–19] are developed to overcome this challenge by taking a mean field approximation of water molecules and resulting in a dielectric continuum. The GB method is faster than PB methods but it only provides an approximation for electrostatic energies. PB methods, derived from fundamental physical theories [20, 21], offer more accurate electrostatic analysis. PB model has been applied to the calculations of protein-protein and protein-ligand binding energies [22], the pH value predictions of protonation and/or deprotonation states of titration sites [23], and drug design [24]. To seek for an accurate, efficient, and robust numerical solver, a large number of numerical methods have been developed for the PB model, including finite difference method (FDM) [25], finite element method (FEM) [26], and boundary element method (BEM) [7, 27]. Among this variety of numerical explorations, the FDM has the most enfranchisement such as Amber PBSA [28], Delphi [29], APBS [23, 26], MIBPB [6, 30–33], and CHARMM PBEQ [25]. Among them, MIBPB is the solely available second-order accurate method and has been used to calibrate the GB method in Amber [34], where PB methods are generally very expensive. In addition, the molecular surface involved in all the aforementioned method with corresponding software developed, such as ESES [35], Nanoshaper [36], and MSMS [37].

Over the past a few years, machine learning, including deep learning, has had tremendous success in science and engineering. Especially, convolutional neural networks have proved their ability to automatically extract features and recognize patterns from relatively simple but large datasets. Deep learning has a growing dominance in important applications such as handwriting recognition, speech recognition, and drug discovery [38–40]. Aided by the availability of quality databases, new algorithms, graphics processing unit (GPU), and high-performance computers, various machine learning approaches have been established in many classical computational problems such as solvation free energies, protein-ligand binding affinities, mutation impacts, toxicity, partition coefficients, protein B-factors, *etc.* [41–50]. Additionally, deep learning neural networks are also applied in computational protein design [51], stability changes of protein induced by mutations [52, 53], and calculations of protein energy [54, 55].

Recently, we developed a Poisson-Boltzmann based machine learning (PBML) model, which can compute the solvation free energy of macromolecules in the solvent with the GB speed and the PB accuracy [56]. We assume that all of the macromolecular electrostatic solvation free energies follow a probability distribution, which can be sampled by the PB model. Our idea is based on a representability hypothesis and a learning hypothesis. The representability hypothesis states that the solvation free energy of a molecule can be described by the features of atom interactions and their geometric relations in the solvent. Thus, we can construct feature vectors to characterize the molecular electrostatic distribution. In our learning hypothesis, we assume that a machine learning model can be trained based on training labels and corresponding features for a sufficiently large training set of molecules. Additionally, advanced machine learning algorithms can give accurate predictions of the electrostatic potential for a new molecule which has the same probability distribution with the training set. In our approach, training labels are computed from MIBPB and features are generated using multiscale weighted colored subgraphs [47].

In the present work, we apply our newly developed PBML model to compute molecular solvation free energies in the implicit-solvent Monte Carlo simulations, which typically require millions of samplings. The new machine learning-based implicit-solvent Monte Carlo model can guarantee the accuracy of the implicit-solvent Monte Carlo model while dramatically speeding up existing implicit-solvent Monte Carlo algorithms.

This manuscript is organized as follows. Section II gives a brief introduction of molecular force fields,

DOI:10.1063/1674-0068/cjcp2109150

Monte Carlo methods, and implicit solvent models. The PBML model is introduced in this section as well, which includes the Poisson-Boltzmann equation, Generalized Born model, and multiscale weighted colored subgraphs. Section III presents the results of structural predictions of benzene and the human hyperplastic discs protein (PDB: 1i2t) [57] in water. We demonstrate that the PBML model is more accurate and faster than commonly used PB solvers and thus, can significantly reduce the computational time of implicit-solvent Monte Carlo simulations. A summary is given in Section IV.

II. METHODS AND ALGORITHMS

In this section, we briefly review biomolecular force fields, the Monte Carlo methods, and implicit solvent models, followed by the Poisson-Boltzmann based machine learning model.

A. Biomolecular force fields

The quality of molecular simulations depends crucially on molecular force fields to offer a physical representation of molecular interactions and energy distributions. Molecular force fields typically describe molecular interactions in terms of classical molecular mechanics of atoms. The potential energies of atomic interactions are approximated by a set of mathematical functions, modeling the bonded and non-bonded components. These functions consist of a set of free coefficients, which are obtained by approximating either the results of elaborate quantum mechanical calculations, or experimental data. One of the advantages of biomolecular force field approach is its computational efficiency. The potential energy can be efficiently computed at the molecular level comparing to other methods, such as quantum mechanical approaches, which deal with electrons [58, 59]. Additionally, the forces in molecular dynamics can be evaluated analytically from molecular force fields.

A variety of molecular force fields have been developed for various purpose. In this work, we adopt the popular and simple Amber ff99SB force field [59]. The Amber force field for governing the potential energy consists of the following terms,

$$E=\sum_{\text{bonds}} k_b(r-r_0)^2+\sum_{\text{angles}} k_\theta(\theta-\theta_0)^2+\sum_{\text{dihedrals}} V_n[1+\cos(n\phi-\gamma)]+\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\left(\frac{A_{ij}}{R_{ij}^{12}}-\frac{B_{ij}}{R_{ij}^{6}}+\frac{q_iq_j}{\epsilon_1 R_{ij}}\right) \quad (1)$$

where k_b, k_θ, and V_n are force constants. Here, r, θ, and ϕ are bond length, angle, and dihedral angle with r_0, θ_0, and γ being optimal bond length, optimal angle, and proper dihedral angle, respectively. The first three terms in the energy expression describe the bonded energy of the molecular system. The last term represents the Lennard-Jones interactions and electrostatic interactions, where N is the number of atoms in the molecular system, R_{ij} is the distance between ith and jth atoms, A_{ij} and B_{ij} are Lennard-Jones parameters, q_i is the atom charge, and ϵ_1 is the dielectric constant.

B. Monte Carlo methods

In this session, we provide a brief introduction of the molecular dynamics and the Monte Carlo method. We start from statistical mechanics and show that the calculation of the physical property of a solute-solvent system using molecular dynamics is computationally expensive or even intractable [10]. Then, we introduce Metropolis's Monte Carlo method for biomolecular simulations [11].

The classical expression for the partition function Q of a solute-solvent system is

$$Q=c\int \mathrm{d}\mathbf{r}\mathrm{d}\mathbf{p}\exp\left[\frac{-\mathcal{H}(\mathbf{r},\mathbf{p})}{k_\text{B}T}\right] \quad (2)$$

where $\mathbf{r}=\{\mathbf{X},\mathbf{Y}\}$ stands for the atomic coordinates of a solute $\mathbf{X}$ and solvent $\mathbf{Y}$, $\mathbf{p}$ stands for the corresponding momenta, c is a physical constant as specified below, k_B is the Boltzmann constant and T is the temperature of the system. The function $\mathcal{H}(\mathbf{r},\mathbf{p})$ is the Hamiltonian of the system. It describes the total energy of an individual system as summation of the kinetic energy $\mathcal{K}$ and the potential energy E: $\mathcal{H}=\mathcal{K}+E$, where $\mathcal{K}$ is a quadratic function of the momenta. For a sys-

DOI:10.1063/1674-0068/cjcp2109150

tem of N identical atoms, one has c=$1/(h^{3N}N!)$ using the Planck constant h. Under the assumption that all of the other physical observables A of interest depend only on the positions, *i.e.*, A=$A(\mathbf{r})$, the integration over the momenta can be carried out analytically in a classical mechanical treatment. As a result, the expected value of a physical observable of interest is given by

$$\langle A\rangle = \frac{\int \mathrm{d}\mathbf{r} A(\mathbf{r})\exp[-\beta E(\mathbf{r})]}{\int \mathrm{d}\mathbf{r}\exp[-\beta E(\mathbf{r})]} \tag{3}$$

where β=$1/k_{\mathrm{B}}T$. Evaluating $\langle A\rangle$ requires numerical techniques, such as quadrature rules for the integration. Since each particle moves in a three dimensional (3D) space, the total number of degrees of freedom is $3N$ for a system of N atoms. If each dimension is integrated with a mesh size of m points, the total number of points for the integration is m^{3N}, which is computationally prohibitive.

The complexity in evaluating Eq.(3) can be significantly reduced by using the Monte Carlo sampling. Indeed, Metropolis *et al.* [11] suggested an efficient Monte Carlo scheme to approximate the ratio in Eq.(3). Let us denote the probability density function in finding a microstate in the canonical ensemble in a configuration $\mathbf{r}$ by

$$P(\mathbf{r}) = \frac{\exp[-\beta E(\mathbf{r})]}{\int \mathrm{d}\mathbf{r}\exp[-\beta E(\mathbf{r})]} \tag{4}$$

According to this probability function, we can perturb randomly selected points in the configuration. Hence, the number of points n_i generated per unit volume in the neighborhood of $\mathbf{r}$ is equal to $N_{\mathrm{mc}}\times P(\mathbf{r})$ for the average of $A(\mathbf{r})$, which is

$$\langle A\rangle \approx \frac{1}{N_{\mathrm{MC}}}\sum_{i=1}^{N_{MC}} n_i A(\mathbf{r}_i) \tag{5}$$

where N_{MC} is the total number running in Monte Carlo simulations. Eq.(5) shows that all states of ensemble contribute to the average equally. Therefore, Metropolis Monte Carlo method starts at a given configuration $\mathbf{r}_0$=$\{\mathbf{X}_0, \mathbf{Y}_0\}$ and next perturbs the configuration by a defined transformation with a new configuration $\mathbf{r}_1$=$\{\mathbf{X}_1, \mathbf{Y}_1\}$. The probability to accept the new configuration is

$$p_{\mathrm{acc}} = \min\{1, \exp[-\beta(E(\mathbf{r}_0) - E(\mathbf{r}_1))]\} \tag{6}$$

If the new configuration is rejected, the previous configuration is retained and the method repeats another random perturbation. This process iterates until the iteration number equals to a fixed number. It is shown that the structure in the system will approach the Boltzmann distribution, if the perturbations satisfy the condition

$$\pi(\mathbf{r}_i)p_{ij} = \pi(\mathbf{r}_j)p_{ji} \tag{7}$$

where $\pi(\mathbf{r}_i)$ is the probability of the system in configuration $\mathbf{r}_i$ and p_{ij} is the probability to perturb the configuration from state $\mathbf{r}_i$ to state $\mathbf{r}_j$ [11].

C. Implicit solvent models

Implicit solvent models are class of multiscale techniques for reducing the dimensionality of a solvent-solute system. They retain the crucial electrostatic interactions between a biomolecule and its solvent environment without modeling solvent molecules explicitly. A variety of two-scale implicit solvent models have been developed, such as the Poisson-Boltzmann (PB) model [13] and the generalized Born (GB) model [16–19]. One desirable application of implicit solvent models is the Monte Carlo simulations of biomolecule in solvent, which is relatively easy to implement. The basic derivation for molecular implicit solvent models relies on statistical mechanics. For more detail, the reader is referred to the literature [60]. Essentially, the molecular solvation free energy can be given by

$$\Delta G_{\mathrm{solv}} = \Delta G_{\mathrm{elec}} + \Delta G_{\mathrm{nonpol}} \tag{8}$$

where ΔG_{elec} represents the electrostatic contribution of the solvent-solute interaction, and $\Delta G_{\mathrm{nonpol}}$ denotes the nonpolar energy in the reversible work needed to insert a fixed configuration molecule into the solvent with all solute charges set to zero. Here $\Delta G_{\mathrm{nonpol}}$ is proportional to the solvent accessible surface area. The molecular solvation free energy is used in our implicit-solvent Monte Carlo method to represent solvent-solute interactions.

DOI:10.1063/1674-0068/cjcp2109150

D. Poisson-Boltzmann based machine learning (PBML) model

In this section, we briefly discuss the Poisson-Boltzmann based machine learning (PBML) model [56], which is applied to compute ΔG_{elec} in Eq.(8). Our PBML model involves three major components, *i.e.*, training labels, molecular features, and learning algorithms. Our training labels for a large training set of molecules are generated from solving the Poisson-Boltzmann (PB) equation. Our molecular features for both the training set and the test set constitute two parts, a GB part and a correction part. The latter is computed from multiscale weighted colored subgraphs [56].

1. The Poisson-Boltzmann (PB) model

The PB model considers the solute biomolecule with N_{c} fixed charges as the interior domain Ω_1, and the solvent, including free ions, as the exterior domain Ω_2. The interface Γ separates these two domains. The PB model is given as

$$-\nabla\cdot\epsilon(\mathbf{r})\nabla\phi(\mathbf{r})+\bar{\kappa}^2(\mathbf{r})\phi(\mathbf{r})=\sum_{k=1}^{N_{\text{c}}}q_k\delta(\mathbf{r}-\mathbf{r}_k) \tag{9}$$

For $\mathbf{r}\in\mathbb{R}^3$, $\phi(\mathbf{r})$ is the electrostatic potential, $\epsilon(\mathbf{r})$ dielectric constant is given by

$$\epsilon(\mathbf{r})=\begin{cases}\epsilon_1, & \mathbf{r}\in\Omega_1\\ \epsilon_2, & \mathbf{r}\in\Omega_2\end{cases} \tag{10}$$

In the PB model, $\bar{\kappa}$ is the screening parameter with the relation $\bar{\kappa}^2{=}\epsilon_2\kappa^2$ where κ is the inverse Debye length measuring the ionic effective length. To ensure the continuity of electrostatic potential and flux density across the interface Γ, the PB equation is associated with following interface conditions

$$\phi_1(\mathbf{r})=\phi_2(\mathbf{r}),\quad \epsilon_1\frac{\partial\phi_1(\mathbf{r})}{\partial\mathbf{n}}=\epsilon_2\frac{\partial\phi_2(\mathbf{r})}{\partial\mathbf{n}},\quad \mathbf{r}\in\Gamma \tag{11}$$

where ϕ_1 and ϕ_2 are electrostatic potential from the solute domain Ω_1 and the solvent domain Ω_2, and $\mathbf{n}$ is the outward unit normal vector on Γ.

The solvation free energy can be obtained from the PB model by

$$\Delta G_{\text{elec}}^{\text{PB}}=\frac{1}{2}\sum_{k=1}^{N_c}q_k(\phi(\mathbf{r_k})-\phi_{\mathbf{0}}(\mathbf{r_k})) \tag{12}$$

where $\phi_0(\mathbf{r_k})$ is the free space solution to the PB equation assuming no solvent-solute interface. To solve the PB equation, we apply the accurate and robust 2nd order MIBPB solver [6, 32] developed in our group, which applies rigorous treatment on geometric complexity, interface condition, and charge singularity. The $\Delta G_{\text{elec}}^{\text{PB}}$ results generated by MIBPB solver for a set of macromolecules are used as the training labels in the representability hypothesis.

2. The Generalized Born (GB) model

Having described the labels for our machine learning training, we discuss the molecular feature construction for both machine learning training and test, which involves the GB model. As a fast approximation to the PB model, the GB model computes the electrostatic solvation free energy by

$$\Delta G_{\text{elec}}^{\text{GB}}\approx\sum_{i,j}\Delta G_{ij}^{\text{GB}}=-\frac{1}{2}\Big(\frac{1}{\epsilon_1}-\frac{1}{\epsilon_2}\Big)\frac{1}{1+\alpha\beta}\sum_{i,j}q_iq_j\Big(\frac{1}{f_{ij}(r_{ij},R_i,R_j)}+\frac{\alpha\beta}{B}\Big) \tag{13}$$

where R_i is the effective Born radius for i-th atom, r_{ij} is the distance between atoms i and j, $\beta{=}\epsilon_1/\epsilon_2$, $\alpha{=}0.571412$, and B is the electrostatic size of the molecule. The function f_{ij} is given as

$$f_{ij}=\sqrt{r_{ij}^2+R_iR_j\exp\Big(-\frac{r_{ij}^2}{4R_iR_j}\Big)} \tag{14}$$

The effective Born radii R_i is calculated by the following boundary integral

$$R_i^{-1}=\Big(-\frac{1}{4\pi}\oint_\Gamma\frac{\mathbf{r}-\mathbf{r}_i}{|\mathbf{r}-\mathbf{r}_i|^6}\cdot\mathrm{d}\mathbf{S}\Big)^{1/3} \tag{15}$$

In Eq.(15), the MSMS package [61] is used to generate the triangulation discretization of the molecular surface

DOI:10.1063/1674-0068/cjcp2109150

for the numerical surface integral on Γ.

3. Multiscale weighted colored subgraphs

The weighted colored subgraph (WCS) use the notion $G(V, E)$ with vertices V and edges E to describe the atomic interactions in a protein of N atoms. The vertices is defined as

$$V = \{(\mathbf{r}_i, \alpha_i) | \mathbf{r}_i \in \mathbb{R}^3, \alpha_i \in \mathcal{C}, i = 1, 2, \ldots, N\} \quad (16)$$

where $\mathcal{C}$=$\{\mathrm{C}, \mathrm{N}, \mathrm{O}, \mathrm{S}, \mathrm{H}\}$ contains all the commonly occurring element types in a protein. Each vertex is an atom labeled by both its position $\mathbf{r}_i$ element type α_i, for i=1, $\cdots$ N.

The edge E relates the pairwise interactions, which are defined as a colored set $\mathcal{P}$=$\{\alpha\beta\}$ with $\alpha, \beta \in \mathcal{C}$. For $\mathcal{C}$ defined above, $\mathcal{P}$=CC, CN, CO, CS, CH, NN, NO, NS, NH, OO, OS, OH, SS, SH, HH and we define the partition of $\mathcal{P}$ as $\mathcal{P}_k$, k=1, 2,..., 15 such that $\mathcal{P}_1$=$\{$CC$\}$, $\mathcal{P}_2$=$\{$CN$\}$ and so on. The set of involved vertices $V_{\mathcal{P}_k}$ is a subset of V containing all atoms involved in forming the pair in $\mathcal{P}_k$. For instance, $\mathcal{P}_2$=$\{$CN$\}$ contains all carbon-nitrogen atom pairs and $V_{\mathcal{P}_2}$ contains all carbon and nitrogen atom vertices in the protein. Based on these configuration, all the edges for pairwise atomic interactions in the WCS description are defined by

$$E_{\mathcal{P}_k}^{\sigma,\tau,\zeta} = \{\Phi_{\tau,\zeta}^{\sigma}(\|\mathbf{r}_i - \mathbf{r}_j\|) \mid \alpha_i\beta_j \in \mathcal{P}_k; i = 1, 2, \ldots, N_\alpha, j = 1, 2, \ldots, N_\beta\} \quad (17)$$

where $\|\mathbf{r}_i - \mathbf{r}_j\|$ defines the Euclidean distance between ith and jth atoms, N_α and N_β are numbers of type α and β atoms, σ indicates the type of radial basic functions (*e.g.*, σ=L for Lorentz kernel, σ=E for exponential kernel), τ is a scale distance factor between two atoms and ζ is a parameter of power in the kernel (*i.e.*, ζ=κ for σ=E, ζ=ν for σ=L). In this model, we use generalized exponential functions

$$\Phi_{\tau,\kappa}^{\mathrm{E}} = \mathrm{e}^{-(\|\mathbf{r}_i - \mathbf{r}_j\|/\tau(r_i + r_j))^\kappa}, \quad \kappa > 0 \quad (18)$$

and generalized Lorentz functions

$$\Phi_{\tau,\nu}^{\mathrm{L}}(\|\mathbf{r}_i - \mathbf{r}_j\|) = \frac{1}{1 + (\|\mathbf{r}_i - \mathbf{r}_j\|/\tau(r_i + r_j))^\nu}, \quad \nu > 0 \quad (19)$$

where r_i and r_j are, respectively, the van der Waals radius of the ith and jth atoms. Finally, the features for describing the electrostatics interactions and geometric properties are expressed as

$$\mu^{k,\sigma,\tau,\zeta,w} = \sum_{i=1}^{N_\alpha}\sum_{j=1}^{N_\beta} w_{ij}\Phi_{\tau,\zeta}^{\sigma}(\|\mathbf{r}_i - \mathbf{r}_j\|), \quad \alpha_i\beta_j \in \mathcal{P}_k, \quad (20)$$

where w_{ij} is a weight function assigned to each atomic pair with w_{ij}=1 for atomic rigidity or w_{ij}=q_j for atomic charge. Since we have 15 options of the colored subsets $\mathcal{P}_k$, we can obtain corresponding 15 subgraph centralities $\mu^{k,\sigma,\tau,\zeta,w}$, for k=1,2,..., 15. By varying kernel parameters (σ, τ, ζ, w), one can achieve multiscale centralities for multiscale weighted colored subgraph (MWCS) [62], which can be the features.

With labels and features described above, we can construct the machine learning model to predict the solvation free energy of new macromolecules. Specifically, using MIBPB results as labels, and GB and MWCS results as features, we train gradient boosting decision trees (GBDTs) for the solvation free energy prediction.

III. RESULTS

In this section, we demonstrate the performance of the proposed MLIMC method numerically. First, we describe the Poisson-Boltzmann based machine learning (PBML) model for computing protein electrostatic solvation energies, followed by the illustration of the accuracy and efficiency of the model. The use of the PBML model for electrostatic interactions in the MC simulations is introduced. Our main idea is to replace time-consuming electrostatic calculations by using our PBML model. The efficiency of our new MLIMC model is also examined. Finally, we validate the proposed MLIMC method by two cases. Case one is a small molecule, benzene, with initial atom position randomly protruded. Our MLIMC method is used to reconstruct the benzene molecule in solvent. Case two is a relatively larger molecule, protein (PDB: 1i2t) with 61 amino acid

DOI:10.1063/1674-0068/cjcp2109150

residues. In this case, we stretch the last two residues of 1i2t using steered molecular dynamics and then we try to restore the equilibrium configuration by using the proposed MLIMC method. Both simulations are carried out at temperature of 27 °C, the dielectric constants are ϵ_1=1 in the molecule and ϵ_2=80 in the solvent, the MSMS [61] mesh density is set as 2, and the Debye-Huckel constant is set as κ=0.1257 Å^{-1}. There are three kernels used to generate features for machine learning, which are $(\mathrm{E}, 0.3, 2, 1)$, $(\mathrm{E}, 4.7, 2, q_j)$, and $(\mathrm{L}, 4.2, 5, 1)$.

To measure the performance, we use the root-mean-square deviation (RMSD) of atomic positions in length units (Å), defined as

$$\mathrm{RMSD}(\mathbf{v}, \mathbf{w}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[(v_{ix}-w_{ix})^2 + (v_{iy}-w_{iy})^2 + (v_{iz}-w_{iz})^2\right]} \tag{21}$$

where $\mathbf{v}, \mathbf{w} \in \mathbb{R}^{N\times 3}$ are vectors of positions of the N atoms at two different MC samplings. Moreover, we also present relative errors of the total energy measured by comparing the energy for a MC sampling E_{MC}, and the energy for the equilibrium state E_{SS} as

$$\mathrm{e_{ttl}} = \frac{|E_{\mathrm{SS}} - E_{\mathrm{MC}}|}{|E_{\mathrm{SS}}|} \times 100\% \tag{22}$$

We compute the RMSD and errors between Monte Carlo sampling results and the original molecular structure for every 100 Monte Carlo steps for both cases. The core code was written in C/C++ and a cython wrapper calling the core code for performing adds-on functions and applications. Our simulations are produced on a desktop with an i5 7500 CPU and 16GB memory.

A. PBML model

The MLPB model used in Monte Carlo simulation is a pre-trained model. The training set includes 3706 protein structures from the PDBbind v2015 refined set [63]. This refined set was selected from a general set of 14,620 protein-ligand complexes. A data pre-processing (*i.e.* adding force field parameters) is required before a PB solver can be used for electrostatics calculations. Though the PDBbind refined set consists of protein-ligand complexes, only protein structures are applied for calculations. These protein structures are adjusted by the protein preparation wizard utility of the Schrodinger 2015-2 Suite [64] with default parameters unless filling the missing side chains is required.

The training set covers a wide range of proteins in different sizes with atom numbers from 997 to 27,713. The current training set can be expanded to an even larger group of proteins. However, from our test, we conclude that expanding training set will not significantly improve the trained model, thus the size of the current training set is sufficiently large.

The purpose of PBML is to implement a machine learning predictor of PB electrostatic solvation free energies for various proteins efficiently and accurately without explicitly solving the PB equation. Gradient boosting decision tree method is selected for this supervised learning task because of its efficiency. The accuracy of the PBML model is maintained by the accurate electrostatic free energy of solvation as the label calculated by the MIBPB solver. Once a trained PBML model is obtained, the MIBPB solver will not be called anymore. Using the learned PBML model only requires calculating features on the prediction of electrostatic solvation free energies for new compounds, which is rapid.

B. Efficiency of the PBML model

FIG. 1 shows the results for computing solvation energy on 195 proteins from PDBbind v2015 core set [63] using PBML, Amber, and Dephi. The results are shown in terms of the average CPU time per protein versus the mean absolute percentage errors. From FIG. 1(a), we can see PBML is more accurate and much faster than standard PB solvers such as DelPhi and Amber PB. FIG. 1(b) gives more details by zooming into the region where CPU time is small to distinct the CPU time used by the PBML using different MSMS density.

We here add a few notes about how we improve the PBML model in addition to machine learning. We notice that in the energy and feature calculations, every term has a degree of freedom associated with the num-

DOI:10.1063/1674-0068/cjcp2109150

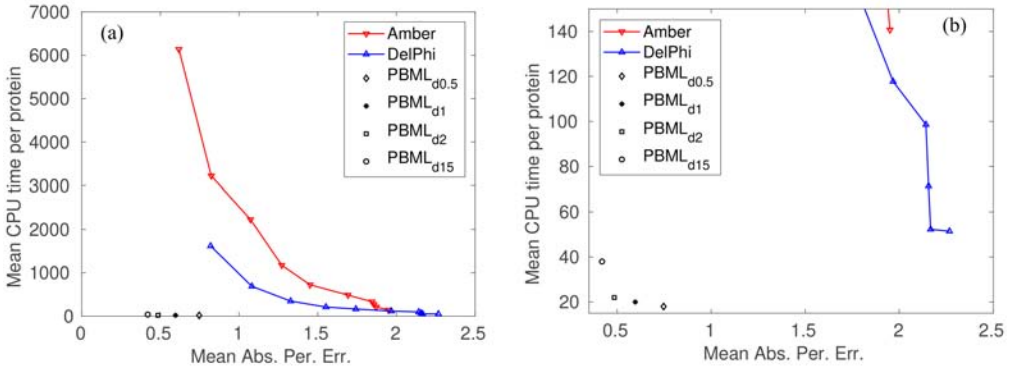


FIG. 1 Comparison of the mean CPU time (in unit of s) per protein and the mean absolute percentage errors of Amber, DelPhi and machine learning predictions of the electrostatic solvation free energies using the test set of 195 proteins. (a) Results of Amber and DelPhi were obtained at ten different mesh sizes from 0.2 Å to 1.1 Å; results of PBML were obtained at four MSMS densities (number of vertices per Å^2) at 15, 2, 1, and 0.5. (b) A zoom-in plot of the left plot for small CPU time.

TABLE I Average CPU time for one step MLIMC simulation using Amber, DelPhi and PBML for electrostatic solvation free energy on the 195 protein dataset. Results of Amber and DelPhi were obtained at 0.2 Å and 0.5 Å mesh sizes, and that from PBML uses mesh density 2. The average CPU time includes all computations needed for Monte Carlo evaluations. The PB error is obtained relative to the electrostatic solvation energy computed from MIBPB solver with grid size h=0.2 Å.

PB solver	CPU time/s		PB error/%	
	h=0.2 Å	h=0.5Å	h=0.2 Å	h=0.5Å
Amber	6136	1177	0.618	1.271
DelPhi	1621	214	0.819	1.552
PBML[a]	25		0.484	

[a] PBML uses mesh density of 2.

ber of atoms, except the computation of the effective Born radii R_i in Eq.(15), which depends on the number of surface triangles M. Since $M \gg N$, faster evaluating of Eq.(15) can significantly accelerate the entire Monte Carlo process. In our present implementation, instead of taking the integral in Eq.(15) on each triangle, we take the integral on a neighborhood of each vertex. This treatment nearly doubled the efficiency of the GB method since number of vertices is about half of number of triangles on the surface. In addition, applying a cut-off can also further improve the GB method.

C. MLIMC model

The assembling of MLIMC includes the implementation of empirical potential energy functions (except electrostatics) and the prediction of electrostatics for each step on Monte Carlo simulations. The conformation of the target protein is perturbed randomly on each step. The new conformation is directly accepted if it shows a lower energy or is accepted with a probability determined by the Boltzmann distribution if it shows a higher energy. As the MLPB model is pre-trained before simulations, the Monte Carlo simulation does not include the time for solving the PB equation, resulting in much reduced time for MLIMC simulations.

D. Efficiency of the MLIMC model

We show that the high efficiency of the MLPB model will significantly improve the efficiency of the MLIMC model.

Table I shows the mean CPU time of one Monte Carlo step and the mean absolute percentage errors of Amber, DelPhi and PBML predictions of the electrostatic solvation free energies of the 195 proteins. The mean CPU time for each protein includes the computations for the total energies, in which computing electrostatic is the dominant component.

Clearly, the machine learning method has the highest accuracy but the lowest CPU time. For the same accuracy level (<1%), the estimation of the mean CPU time for a one-million-step Monte Carlo simulation is 6.136×10^8 s, 1.621×10^8 s, and 2.5×10^6 s for using Amber, DelPhi and PBML, respectively. Even with compromised accuracy for DelPhi and Amber at gird size of 0.5 Å, the MLIMC with PBML will be 47 times faster than that with Amber and 8 times faster than that with DelPhi. Next we show some MC simulation results using MLIMC on the benzene molecule and the human

DOI:10.1063/1674-0068/cjcp2109150

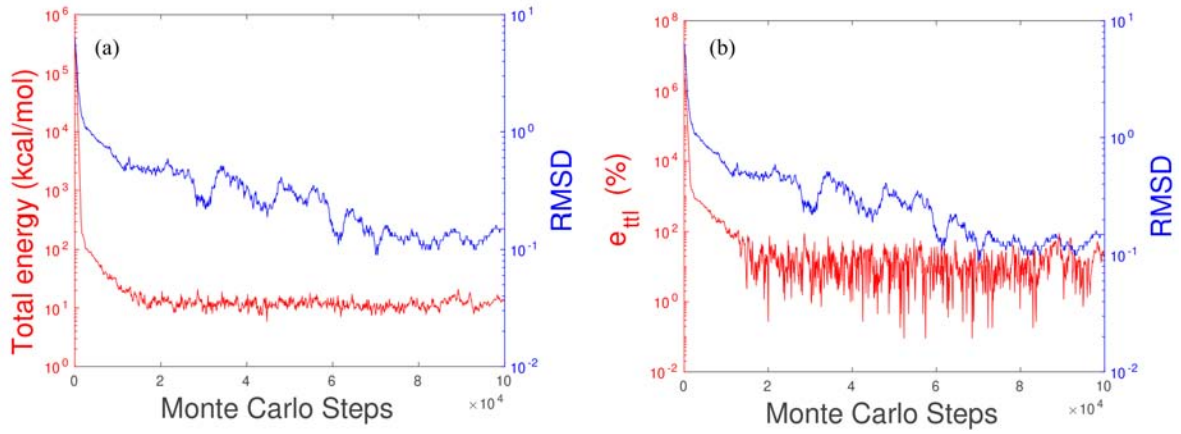


FIG. 2 MLIMC simulation of benzene in solvent. (a) The red curve is the total energy calculated by our implicit-solvent Monte Carlo model and the blue curve is the root mean square deviation of the atomic positions on each Monte Carlo step to the non-protruded one. (b) The red curve is the error of total energies e_{ttl} defined by Eq.(22) and the blue curve is the same RMSD as the left figure.

hyperplastic discs protein (PDB:1i2t).

E. Test case one: benzene molecule

Our first case is a Benzene molecule with some atomic position randomly perturbed. In detail, we fixed three atoms at equilibrium positions in order to have the prediction and the comparison structure in the same plane, and perturb the coordinates of the remained nine atoms in (ρ, θ, ϕ) directions by uniformly distributed random numbers in ([0, 10], [0, 2π], [0,π]). The initial RMSD is 6.42 Å as compared with the equilibrium position. We will try to perform a MC simulation on this perturbed molecule to see if the original steady status can be obtained. FIG. 2(a) shows the total energy and RMSD *vs.* MC steps, from which we can see that the total energy of benzene in solvent starts at 349123.61 kcal/mol and converges to the range of 5−15 kcal/mol after the first 20,000 MC steps. It stays in a convergent range for the rest MC steps. The RMSD initially is 6.42 Å and ends around 0.15 Å. It decreases rapidly as the total energy for the first 20,000 steps. After 20,000 steps, the total energy converges with only slightly oscillation, and the RMSD keeps the decreasing trend until it reaches around 0.15 Å when MC steps are greater than 70,000.

FIG. 2(b) shows errors and RMSD versus MC steps. Here we set E_{SS} in Eq.(22) to be 10.60 kcal/mol as the steady state energy for reference. The plot shows that the errors of total energy are very small for our MC simulation after 10,000 iterations. When the simulation structure is close to that of its equilibrium state, the RMSD is smaller than 1 Å and the errors stay in between 1% and 100%. Note since the total energy is a small number, a tiny perturbation causes a large error changing.

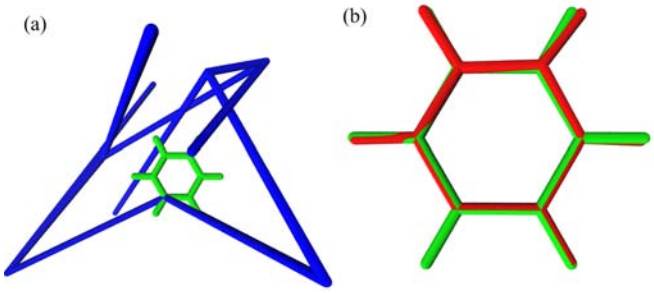


FIG. 3 Illustration of MLIMC simulations of a benzene molecular in solvent. (a) The blue structure is the randomly perturbed atom positions and the green one is the benzene structure in steady state. (b) The red one is the benzene structure after MLIMC sampling compared with the equilibrium structure in green. Pictures are produced with VMD [65].

Qualitatively, FIG. 3(a) shows that the benzene molecule with its initial perturbed structure is in blue and the equilibrium structure is in green. After the MC simulation, we receive the predicted structure in red as compared with the steady state structure in green as shown in FIG. 3(b). The total CPU time for 100,000 Monte Carlo steps is 643 s.

F. Test case two: protein (PDB: 1i2t)

The second MC test is on the human hyperplastic discs protein (PDB: 1i2t) with 61 residues. We first

DOI:10.1063/1674-0068/cjcp2109150

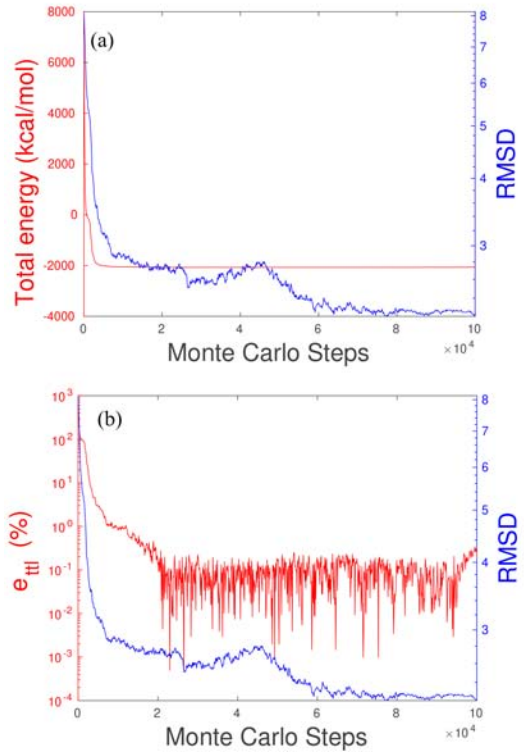


FIG. 4 MLIMC simulation of the protein (PDB: 1i2t) in solvent. (a) The red curve is the total energy calculated by implicit-solvent Monte Carlo model, the blue curve is the root mean square deviation of the atomic positions on each Monte Carlo step to the non-protruded one. (b) The red curve is the error of total energies e_{ttl} defined by Eq.(22), the blue curve is the same RMSD as the left figure.

stretch the last two residues of the original protein by a steered molecular dynamics. As a result, the stretched molecule has an initial RMSD of 8.14 Å. We apply our MLIMC for 100,000 steps, which takes 16,684 s in CPU time. FIG. 4(a) shows that the total energy of 7260.90 kcal/mol initially decays rapidly within the first 5000 Monte Carlo steps, then oscillates around −2070.00 kcal/mol. In the same plot, we can see the RMSD drops quickly in the first 10,000 MC steps, after then decays slowly with fluctuation for the next 40,000 MC steps, then decays steadily after 45,000 MC steps, and finally oscillates slightly around 2.2 Å after 60,000 MC steps. For the energy errors shown in FIG. 4 (b), relative to the total energy in the equilibrium of −2068.13 kcal/mol, the errors rapidly decays in the first 20,000 MC steps and then oscillate within 10% after that.

Similar to the benzene case, FIG. 5(a) qualitatively shows the perturbed structure in blue against the steady state structure in green for the first two residues and

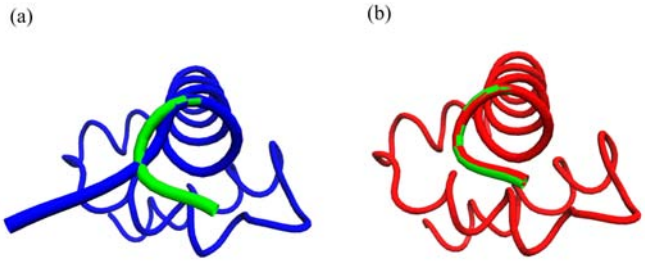


FIG. 5 Illustration of MLIMC simulations of the protein (PDB: 1i2t) [57] in solvent. (a) The blue structure is the perturbed protein structure generated with steered molecular dynamics and the green one is the original structure at the equilibrium state in solvent. (b) The red one is the predicted structure after MLIMC sampling compared with the original structure in green. Pictures are produced with VMD [65].

FIG. 5 (b) shows that the MLIMC structural prediction in red color after the MC simulation, which is very close to the steady state structure in green.

IV. CONCLUSION

Monte Carlo simulations are widely used in science and engineering for molecular structure optimization and prediction. In many situations, particularly biomolecular systems, the solute molecule is immersed in a water solvent and the full-scale explicit solvent Monte Carlo simulations are very expensive. Alternatively, implicit solvent Monte Carlo methods using either Poisson-Boltzmann (PB) model or generalized Born (GB) model for computing electrostatics can greatly reduce the degree of freedom. However, the accuracy reduction in GB model or the efficiency concerns in PB model hinders the wide application of implicit solvent Monte Carlo simulation. In this work, we introduce a machine learning-based implicit-solvent Monte Carlo (MLIMC) method for molecular structure optimization and prediction. A vital component of our MLIMC is the newly developed Poisson-Boltzmann based machine learning (PBML) model, which maintains the PB accuracy at the GB cost. We validate the proposed MLIMC method by simulating two molecular systems, randomly perturbed benzene structure and protein (PDB: 1i2t) structures modified by a steered molecular dynamics. Numerical experiments demonstrate that proposed MLIMC is efficient in predicting molecular structures at equilibrium. In a comparative analysis, we show that the MLIMC model has a great advantage on CPU time and accuracy over DelPhi and

DOI:10.1063/1674-0068/cjcp2109150

Amber PB based Monte Carlo methods. We believe this innovated PBML method can also disruptively change the current status of PB based molecular simulation involving molecular dynamics [66] and Monte Carlo. MLIMC provides accurate electrostatic solvation energy at each configuration of the target protein thus can be helpful in searching protein folding states as intermediate or final using MC based simulation. The resulting machine learning-based implicit molecular dynamics (MLIMD), together with the present MLIMC model, will have a vast variety of applications in molecular science, including drug design.

V. ACKNOWLEDGMENTS

This work was supported in part by NIH grant GM126189, NSF grants DMS-2052983, DMS-1761320, and IIS-1900473, NASA grant 80NSSC21M0023, Michigan Economic Development Corporation, MSU Foundation, Bristol-Myers Squibb 65109, and Pfizer. The work of WG was supported in part by NSF grants DMS-1819193 and DMS-2110922.

[1] G. W. Wei, Nature Machine Intelligence **1**, 336 (2019).
[2] B. J. Alder and T. E. Wainwright, J. Chem. Phys. **31**, 459 (1959).
[3] M. Karplus and J. Kuriyan, Proc. Nat. Acad. Sci. USA **102**, 6679 (2005).
[4] A. Rahman, Phys. Rev. **136**, A405 (1964).
[5] H. A. Scheraga, M. Khalili, and A. Liwo, Annu. Rev. Phys. Chem. **58**, 57 (2007).
[6] D. Chen, Z. Chen, C. Chen, W. H. Geng, and G. W. Wei, J. Comput. Chem. **32**, 657 (2011).
[7] W. H. Geng and R. Krasny, J. Comput. Phys. **247**, 62 (2013).
[8] C. Sagui and T. A. Darden. Annu. Rev. Biophys. Biomol. Struct. **28**, 155 (1999).
[9] G. Sutmann, P. Gibbon, T. Lippert, Forschungszentrum Jülich, (2011).
[10] D. Frenkel, NIC Series, **23**, 29 (2004).
[11] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. J. Chem. Phys. **21**, 1087 (1953).
[12] M. E. Davis and J. A. McCammon, Chem. Rev. **94**, 509 (1990).
[13] F. Fogolari, A. Brigo, and H. Molinari, J. Mol. Recogn. **15**, 377 (2002).
[14] M. Cossi, V. Barone, R. Cammi, and J. Tomasi, Chem. Phys. Lett. **255**, 327 (1996).
[15] J. Tomasi, B. Mennucci, and R. Cammi, Chem. Rev. **105**, 2999 (2005).
[16] B. N. Dominy and C. L. Brooks III, J. Phys. Chem. B **103**, 3765 (1999).
[17] J. Mongan, C. Simmerling, J. A. McCammon, D. A. Case, and A. Onufriev, J. Chem. Theory Comput. **3**, 159 (2007).
[18] A. Onufriev, D. A. Case, and D. Bashford, J. Comput. Chem. **23**, 1297 (2002).
[19] H. Tjong and H. X. Zhou, J. Chem. Phys. **126**, 195102 (2007).
[20] D. Beglov and B. Roux, J. Chem. Phys. **104**, 8678, (1996).
[21] A. Onufriev, D. Bashford, and D. A. Case, J. Phys. Chem. B **104**, 3712 (2000).
[22] D. D. Nguyen, B. Wang, and G. W. Wei, J. Computat. Chem. **38**, 94 (2017).
[23] E. Jurrus, D. Engel, K. Star, K. Monson, J. Brandi, L. Felberg, D. Brookes, L. Wilson, J. Chen, K. Liles, M. Chun, P. Li, D. Gohara, T. Dolinsky, R. Konecny, D. Koes, J. Nielsen, T. Head-Gordon, W. Geng, R. Krasny, G. W. Wei, M. Holst, J. McCammon, and N. Baker, Protein Sci. **27**, 112 (2018).
[24] E. Wang, H. Sun, J. Wang, Z. Wang, H. Liu, J. Z. Zhang, and T. Hou, Chem. Rev. **119**, 9478 (2019).
[25] S. Jo, M. Vargyas, J. Vasko-Szedlar, B. Roux, and W. Im, Nucleic Acids Res. **36**, W270 (2008).
[26] N. A. Baker, D. Sept, M. J. Holst, and J. A. Mccammon, IBM J. Res. Develop. **45**, 427 (2001).
[27] B. Lu, X. Cheng, J. Huang, and J. A. McCammon, Comput. Phys. Commun. **184**, 2618 (2013).
[28] J. Wang, C. H. Tan, Y. H. Tan, Q. Lu, and R. Luo. Commun. Comput. Phys. **3**, 1010 (2008).
[29] L. Li, C. Li, S. Sarkar, J. Zhang, S. Witham, Z. Zhang, L. Wang, N. Smith, M. Petukh, and E. Alexov, BMC biophys. **5**, 9 (2012).
[30] W. Geng, S. Yu, and G. W. Wei, J. Chem. Phys. **127**, 114106 (2007).
[31] W. Geng and S. Zhao, J. Comput. Phys. **351**, 25 (2017).
[32] D. D. Nguyen, B. Wang, and G. W. Wei, J. Comput. Chem. **38**, 941 (2017).
[33] Y. C. Zhou, M. Feig, and G. W. Wei, J. Comput. Chem. **29**, 87 (2008).
[34] N. Forouzesh, S. Izadi, and A. V. Onufriev, J. Chem. Inform. Model. **57**, 2505 (2017).
[35] B. Liu, B. Wang, R. Zhao, Y. Tong, and G. W. Wei, Eses: Software for Eulerian Solvent Excluded Surface, (2017).
[36] S. Decherchi and W. Rocchia, PloS one **8**, e59744, (2013).
[37] M. F. Sanner, A. J. Olson, and J.-C. Spehner, Biopolymers **38**, 305 (1996).
[38] T. B. Hughes, G. P. Miller, and S. J. Swamidass, ACS

Cent. Sci. **1**, 168 (2015).
[39] A. Lusci, G. Pollastri, and P. Baldi, J. Chem. Informa. Model. **53**, 1563 (2013).
[40] D. D. Nguyen, Z. Cang, K. Wu, M. Wang, Y. Cao, and G. W. Wei, J. Computer-Aided Mol. Design **33**, 71 (2019).
[41] T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P. M. Agapow, M. Zietz, M. M. Hoffman, W. Xie, G. L. Rosen, B. J. Lengerich, J. Israeli, J. Lanchantin, S. Woloszynek, A. E. Carpenter, A. Shrikumar, J. B. Xu, E. M Cofer, C. A. Lavender, S. C. Turaga, A. M. Alexandari, Z. Y. Lu, D. J. Harris, D. DeCaprio, Y. J. Qi, A. Kundaje, Y. F. Peng, L. K Wiley, M. H. S. Segler, S. M. Boca, S. J. Swamidass, A. Huang, A. Gitter, and C. S. Greene, J. Royal Soc. Interface **15**, 20170387 (2018).
[42] Z. X. Cang, L. Mu, and G. W. Wei, PLOS Comput. Bio. **14**, e1005929, (2018).
[43] Z. X. Cang and G. W. Wei, Int. J. Numer. Meth. Biomed. Eng. **34**, e2914 (2018).
[44] J. Jiménez, M. Skalic, G. Martínez-Rosell, and G. De Fabritiis, J. Chem. Inform. Model. **58**, 287 (2018).
[45] M. Karimi, D. Wu, Z. Wang, and Y. Shen, arXiv:1806.07537, (2018).
[46] A. Korotcov, V. Tkachenko, D. P. Russo, and S. Ekins, Mol. Pharma. **14**, 4462 (2017).
[47] D. D. Nguyen, T. Xiao, M. L. Wang, and G. W. Wei, J. Chem. Inform. Model. **57**, 1715 (2017).
[48] C. Wang and Y. Zhang, J. Comput. Chem. **38**, 169 (2017).
[49] K. Wu and G. W. Wei, J. Chem. Inform. Model. **58**, 520 (2018).
[50] K. Wu, Z. Zhao, R. Wang, and G. W. Wei, J. Comput. Chem. **39**, 1444 (2018).
[51] J. Wang, H. Cao, J. Z. Zhang, and Y. Qi, Sci. Reports, **8**, 1 (2018).
[52] Z. Cang and G. W. Wei, Bioinformatics **33**, 3549, (2017).
[53] H. Cao, J. Wang, L. He, Y. Qi, and J. Z. Zhang, J. Chem. Inform. Model. **59**, 1508 (2019).
[54] J. Chen, X. Xu, S. Liu, and D. H. Zhang, Phys. Chem. Chem. Phys. **20**, 9090 (2018).
[55] Z. Wang, Y. Han, J. Li, and X. He, J. Phys. Chem. B **124**, 3027 (2020).
[56] J. Chen, Y. Xu, Z. Cang, W. Geng, and G. W. Wei, *Preprint*, (2021).
[57] R. C. Deo, N. Sonenberg, and S. K. Burley, Proc. Natl. Acad. Sci. **98**, 4414 (2001).
[58] D. A. Case, T. E. Cheatham, T. Darden, H. Gohlke, R. Luo, K. M. Merz, A. Onufriev and R. J. C. Simmerling, B. Wang, and A. Woods, J. Comput. Chem. **26**, 1668 (2005).
[59] K. Lindorff-Larsen, S. Piana, K. Palmo, P. Maragakis, J. L. Klepeis, R. O. Dror, and D. E. Shaw, Proteins **78**, 1950 (2010).
[60] B. Roux and T. Simonson, Biophys. Chem. **28**, 155 (1999).
[61] MSMS. https://mgl.scripps.edu/people/sanner/html/msms_home.html.
[62] D. Bramer and G. W. Wei, J. Chem. Phys. **140**, 054103 (2018).
[63] Z. Liu, Y. Li, L. Han, J. Liu, Z. Zhao, W. Nie, Y. Liu, and R. Wang, Bioinformatics **31**, 405 (2015).
[64] S. LLC, *Schrödinger Release 2015-2, Schrödinger LLC*, New York, (2015).
[65] W. Humphrey, A. Dalke, and K. Schulten, J. Mol. Graph. **14**, 33 (1996).
[66] W. Geng and G. W. Wei, J. Comput. Phys. **230**, 435 (2011).

DOI:10.1063/1674-0068/cjcp2109150