JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

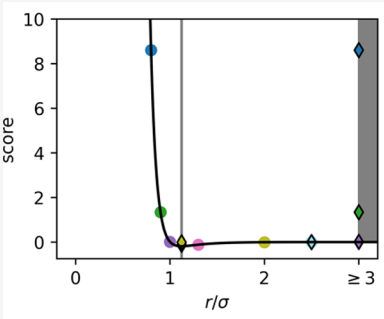
pubs.acs.org/jcim Article

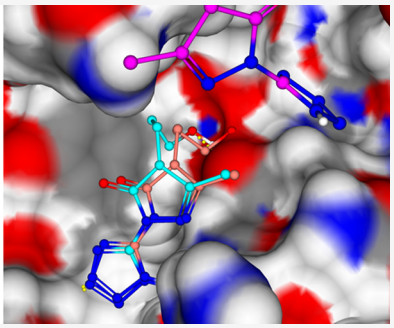
A Consistent Scheme for Gradient-Based Optimization of Protein–Ligand Poses

Florian Flachsenberg, Agnes Meyder, Kai Sommer, Patrick Penner, and Matthias Rarey*

Cite This: J. Chem. Inf. Model. 2020, 60, 6502–6522 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information





ABSTRACT: Scoring and numerical optimization of protein–ligand poses is an integral part of docking tools. Although many scoring functions exist, many of them are not continuously differentiable and they are rarely explicitly analyzed with respect to their numerical optimization behavior. Here, we present a consistent scheme for pose scoring and gradient-based pose optimization. It consists of a novel variant of the BFGS algorithm enabling step-length control, named LSL-BFGS (limited step length BFGS), and the empirical JAMDA scoring function designed for pose prediction and good numerical optimizability. The JAMDA scoring function shows a high pose prediction performance in the CASF-2016 docking power benchmark, top-ranking a pose with an RMSD of ≤2 Å in about 89% of the cases. The combination of JAMDA scoring with the LSL-BFGS algorithm shows a significantly higher optimization locality (i.e., no excessive movement of poses) than with the classical BFGS algorithm while retaining the characteristically low number of scoring function evaluations. The JAMDA scoring and optimization scheme is freely available for noncommercial use and academic research.

INTRODUCTION

Molecular docking is one of the main techniques in molecular modeling. The development of docking algorithms and tools is tightly connected to the development of protein–ligand scoring functions as their objective functions.[1,2]

The scoring function can be used at different stages of the docking process: It can be an integral part of the pose generation if a global optimization algorithm is used (e.g., PLANTS,[3,4] AutoDock Vina[5]), or it is used to select generated (intermediary) poses for further processing, e.g., in incremental construction (FlexX[6]) or exhaustive discrete enumeration (GLIDE[7]). In any case, the scoring function is used for the selection and ranking of the best poses from the potentially large list of candidates generated by the docking algorithm. In this scenario, protein–ligand scoring functions are used to identify energetically favorable protein–ligand complex geometries (also known as the pose prediction problem). Other uses of scoring functions are the identification of active compounds among nonbinders (virtual screening) and the prediction of binding affinities.[8,9]

A perfect scoring function that can accurately predict the binding affinity for any conformation and compound would obviously solve all problems simultaneously. However, real-world scoring functions all have their strengths and weaknesses and do not perform equally well for each task.[8] Consequently, the scoring problem in general remains unsolved (especially the estimation of the free energy of binding). In this paper, we focus on the use of scoring functions for the pose prediction task, i.e., the identification of near-native protein–ligand complex geometries and differentiation from those not seen experimentally (decoy poses).

A lot of scoring functions have been reported in the past, and active research is still ongoing. Existing functions are already very successful in redocking scenarios, i.e., in test cases where the conformation of the protein in its bound state is known. However, the behavior of scoring functions under numerical optimization has been rarely analyzed explicitly in

Received: September 19, 2020
Published: December 1, 2020

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502–6522

detail (see the work of Spitzmüller et al.[10] as an example), including questions such as, How much does a pose change when optimized? What influence do clashes with the protein have on the optimization behavior? How many iterations do different optimization algorithms need until convergence?

Generally, scoring functions can be classified into the categories of force field (e.g., the physics-based scoring function in DOCK[11,12]), knowledge-based (e.g., the PMF score[13]), empirical (e.g., the Böhm scoring function[14]), and machine-learning-based (e.g., $\Delta_{vina}RF_{20}$[15]) scoring functions. A detailed discussion of the numerous scoring concepts and functions is beyond the scope of this paper, and the reader is referred to relevant review articles.[1,9,16]

Nearly all docking methods locally optimize a small molecule within the binding pocket of a protein with respect to their scoring functions. The main motivations are the refinement of the poses generated by the preceding sampling or global optimization algorithm (e.g., AutoDock Vina[5] and PLANTS[3,4]) or the optimization before the final score is calculated in (re-)scoring applications (as is recommended by Cole et al.[17] and O'Boyle et al.[18]).

There are different ways to represent the molecule for optimization, i.e., to define the degrees of freedom. The most straightforward way is to directly use the Cartesian coordinates of the atoms of the molecule, resulting in $3N$ degrees of freedom (DOFs) for the optimization of a molecule with N atoms. This obviously requires the scoring function to include terms for bond angles and bond lengths.

Alternatively, the molecule can also be represented using internal coordinates (bond lengths, bond angles, and torsion angles). For the optimization of protein−ligand complexes, this can be combined with relative rigid-body movement implemented by three translational and three rotational DOFs. Using this representation, irrelevant DOFs can be frozen: Usually, the bond lengths and the bond angles are assumed to remain unchanged upon binding of the ligand to the protein. This has two advantages: It reduces the number of DOFs for the optimization to $6 + M$ for a molecule with M rotatable single bonds. Furthermore, it eliminates the need for the inclusion of the bond angle and bond length terms into the scoring function. This representation is regularly used for the optimization of scoring functions in docking[4,5,19−21] and will also be used throughout this paper.

Numerical optimization algorithms can be categorized into local and global optimizers. Most algorithms for local optimization have convergence guarantees; i.e., they are guaranteed to find a local optimum within a numerical tolerance bound. For global optimization problems, most often metaheuristics are used (e.g., Particle Swarm Optimization[22] or Simulated Annealing[23]) that do not have a convergence guarantee to a global optimum for general (nonconvex) problems.

In the following, we restrict our discussions to the local optimization algorithms. For the local optimization algorithm itself, there are two main choices: algorithms that use only the function value itself and algorithms that also use the function's derivatives. It is generally accepted that gradient-based numerical optimization algorithms are superior to gradient-free algorithms when it comes to the number of function evaluations; however, they assume that a continuous first derivative exists and can be calculated.

Derivatives of the optimized function can be calculated analytically or numerically. Analytical derivatives are the method of choice, and they can be obtained by symbolical differentiation (by hand or by a computer algebra system) or automatically[24] (generated from the program code). Sometimes, analytical derivatives exist for a scoring function; however, they are not continuous: in these cases, smoothing can be applied. One example is the smoothing of the Gehlhaar scoring function[25] by Fuhrmann et al.[20] Scoring functions sometimes contain terms based on surface complementarity or accessibility, e.g., the Böhm[14] or the HYDE[26] scoring function. These are especially challenging for analytical derivative calculation as surface terms are often inherently discrete and/or based on complex algorithms preventing the determination of (continuous) analytical derivatives.

The local pose optimization should be *stable* in the sense that small changes in the pose result in the same minimum after optimization and it should be *local*, meaning that a local minimum somehow close to the starting pose is selected. We would like to emphasize that we do not contradict the idea of Spitzmüller et al.[10] of including a global optimization step into the numerical pose optimization procedure to also find other close-by optima. We rather believe that for any method for pose optimization, the emphasis should be on *close-by* (as was also emphasized by Spitzmüller et al.[10]), i.e., that any optimization algorithm (may it be strictly local or a global heuristic) does not move the initial pose too far away. The motivation for this claim will become obvious in the Results and Discussion.

In summary, a function to score protein−ligand docking poses should follow two aims. First of all, the function must reflect the energy landscape of protein−ligand interactions, and second, it must be easily optimizable (i.e., it has an analytical and continuous gradient, and a stable and local optimization is possible). While the first aim is addressed by all developments, the second is usually only considered half-heartedly. Here, our goal is the development of an accurate, consistent, and stable scheme for the scoring and local optimization of small ligand poses for use in protein−ligand docking. To achieve this goal, we recognized that not only the scoring function has to be designed appropriately but also the optimization algorithm has to be adapted. We describe the novel empirical JAMDA scoring function and the new optimization algorithm LSL-BFGS (limited step length BFGS). Scoring functions for docking have to penalize intermolecular clash by steep flanks causing severe problems upon numerical optimization (especially regarding the abovementioned *locality*). Our LSL-BFGS algorithm is a variant of the well-known BFGS algorithm[27−31] addressing this issue resulting in a substantial increase in locality of the solutions.

METHODS

The JAMDA Scoring Function. The JAMDA scoring function was inspired by different empirical scoring functions, especially by the PLANTS scoring function,[4] the ChemScore[32,33] function, and also the original Böhm scoring function.[14] Great care was taken to model the individual contributions such that they represent the intermolecular interactions as accurately as possible and to build an easily optimizable scoring function by using twice continuously differentiable functions. The individual score contributions are terms often found in empirical scoring functions (e.g., hydrogen bonds, lipophilic interactions, etc.). The functional forms, however, are new. As it is typical for empirical scoring functions, the JAMDA scoring function is the weighted sum of

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

physics-inspired scoring contributions, and the weights are determined empirically.

The scoring terms are given in Table 1. The JAMDA scoring function is a linear combination of pairwise and torsional score

Table 1. Scoring Contributions[a]

score contribution	weight	symbol	functional form
hydrogen bond	w_{hb}	F_{hb}	*plateau3*
hydrogen bond clash	$w_{hb\text{-}clash} = 1$	$F_{hb\text{-}clash}$	*repulsive*
metal interaction	w_{metal}	F_{metal}	*plateau3*
metal clash	$w_{metal\text{-}clash} = 1$	$F_{metal\text{-}clash}$	*repulsive*
lipophilic	$w_{lipo} = 1$	F_{lipo}	*LJ-like*
vdW	w_{vdW}	F_{vdW}	*LJ-like*
general clash	$w_{clash} = 1$	F_{clash}	*repulsive*
intramolecular hydrogen bond clash	$w_{intra\text{-}hb\text{-}clash} = 1$	$F_{intra\text{-}hb\text{-}clash}$	*repulsive*
intramolecular lipophilic clash	$w_{intra\text{-}lipo\text{-}clash} = 1$	$F_{intra\text{-}lipo\text{-}clash}$	*repulsive*
intramolecular general clash	$w_{intra\text{-}clash} = 1$	$F_{intra\text{-}clash}$	*repulsive*
torsion	$w_{torsion}$	$F_{torsion}$	continuous torsion score
torsion conjugated	$w_{conjugated} = 1$	$F_{conjugated}$	*plateau2*

[a]For each scoring contribution, the symbol used for the weight and the scoring contribution as well as the functional form are given. The first seven contributions describe intermolecular interactions, and the last five describe intramolecular terms (intramolecular clash and torsion angle potentials).

contributions and is given in eq 1. The intermolecular terms are calculated for each atom pair where this score is applicable (see next section). For the evaluation of intermolecular pairwise score contributions, a *k*-d tree[34] is used for efficient neighborhood queries (implemented in the nanoflann library[35]). The intramolecular terms are only calculated for atom pairs that are more than three bonds apart in the molecular graph. The torsional terms are calculated only for rotatable bonds (acyclic single bonds), and the conjugated torsion term is calculated only for selected bonds (see Torsion Angles). The score contributions are only calculated for heavy atoms (attached hydrogens are handled implicitly in F_{hb}).

$$\begin{aligned} F_{JAMDA} = {} & w_{hb}\cdot F_{hb} + w_{hb\text{-}clash}\cdot F_{hb\text{-}clash} + w_{metal}\cdot F_{metal} \\ & + w_{metal\text{-}clash}\cdot F_{metal\text{-}clash} + w_{lipo}\cdot F_{lipo} + w_{vdW}\cdot F_{vdW} \\ & + w_{clash}\cdot F_{clash} + w_{intra\text{-}hb\text{-}clash}\cdot F_{intra\text{-}hb\text{-}clash} \\ & + w_{intra\text{-}lipo\text{-}clash}\cdot F_{intra\text{-}lipo\text{-}clash} + w_{intra\text{-}clash} \\ & \cdot F_{intra\text{-}clash} + w_{torsion}\cdot F_{torsion} + w_{conjugated}\cdot F_{conjugated} \end{aligned} \quad (1)$$

The following conventions are used throughout the paper: Capitalized score contributions (e.g., F_{hb}) describe the sum of all the individual contributions of that type. Individual contributions are given in lower case (e.g., f_{hb}). Heavy atoms of the pairwise interactions are denoted as *a* and *b*, respectively. Additional reference points (hydrogen atoms and lone pairs) are denoted as *r*. We define a function $\mathbf{c}(a)$ that maps the atoms and also the reference points to their coordinates $\mathbf{c}_i$. The function eq 2 gives the Euclidean distance between atoms *a* and *b*.

$$d(a, b) = \|\mathbf{c}(a) - \mathbf{c}(b)\| \quad (2)$$

Atom Types. Atoms are divided into seven mutually exclusive classes: hydrogen bond donor, hydrogen bond acceptor, hydrogen bond donor and acceptor, polar atom, and lipophilic atom (the difference between polar and lipophilic atoms is similar to the ChemScore function[32]) as well as metal atoms and other atoms. The rules for class membership are described in the Supporting Information Section S1.1. The atom classes determine which interaction potentials are applicable, and the applicable function contributions are given in Table S1. We define an indicator function $I_{pw}(a, b)$ for each pairwise interaction, which is equal to 1 if that pairwise interaction is applicable to the given pair of atoms or 0 otherwise. This indicator function is defined by Table S1. Similarly, an indicator function $I_b(\text{bond})$ is defined for each scoring contribution acting on torsion bonds. Note that we follow these conventions for a precise definition of the scoring function only. The implementation uses appropriate program logic and lists for higher efficiency.

Piecewise Polynomial Functions. All scoring contributions (except for the continuous torsion score, see Torsion Angles) are modeled using piecewise polynomial functions (splines). In order to model the desired curve shapes, we used polynomials of higher degrees and posed constraints not only on the function values but also on the first and second derivatives. The resulting system of equations was solved analytically for the polynomial's coefficients using Mathematica 11.0[36] or SymPy.[37] The results were processed and automatically translated into C++ code using SymPy.[37] An example for the definition of such a system of equations is given in Figure 1. The main advantage of obtaining an analytical solution is that expressions for the polynomial's coefficients as functions of the input parameters can be obtained. That way, the resulting piecewise polynomial function is tunable; i.e., the parameters (e.g., the optimum for the Lennard-Jones-like

(a)

$$g_{lj-like}(y_0, x_1, x_2, y_2, x_3, x) = \begin{cases} a_0 & x \le 0 \\ a_1x^5 + b_1x^4 + c_1x^3 + d_1x^2 + e_1x + g_1 & 0 < x \le x_1 \\ a_2x^3 + b_2x^2 + c_2x + d_2 & x_1 < x \le x_2 \\ a_3x^5 + b_3x^4 + c_3x^3 + d_3x^2 + e_3x + g_3 & x_2 < x \le x_3 \\ a_4 & x > x_3 \end{cases}$$

(b)

$$\begin{aligned} g_{lj-like}(y_0, x_1, x_2, y_2, x_3, 0) &= y_0 \\ g_{lj-like}(y_0, x_1, x_2, y_2, x_3, x_1) &= 0 \\ g_{lj-like}(y_0, x_1, x_2, y_2, x_3, x_2) &= y_2 \\ g'_{lj-like}(y_0, x_1, x_2, y_2, x_3, x_2) &= 0 \\ g''_{lj-like}(y_0, x_1, x_2, y_2, x_3, x_2) &= 0 \\ g_{lj-like}(y_0, x_1, x_2, y_2, x_3, x_3) &= 0 \end{aligned}$$

(c)

$$\begin{aligned} \lim_{x\to x_1^+} g_{lj-like}(y_0, x_1, x_2, y_2, x_3, x) &= \lim_{x\to x_1^-} g_{lj-like}(y_0, x_1, x_2, y_2, x_3, x) \\ \lim_{x\to x_1^+} g'_{lj-like}(y_0, x_1, x_2, y_2, x_3, x) &= \lim_{x\to x_1^-} g'_{lj-like}(y_0, x_1, x_2, y_2, x_3, x) \\ \lim_{x\to x_1^+} g''_{lj-like}(y_0, x_1, x_2, y_2, x_3, x) &= \lim_{x\to x_1^-} g''_{lj-like}(y_0, x_1, x_2, y_2, x_3, x) \end{aligned}$$

Figure 1. Definition of the piecewise polynomial Lennard-Jones-like potential. The function is given in panel (a). The system of polynomial equations for calculating the polynomial coefficients is built from the sampling points (b) and the continuity conditions ((c), only one example is listed for one transition between two polynomials). The input parameters are y_0 (maximum repulsion), x_1 (zero-crossing), x_2 (position of the minimum), y_2 (depth of the minimum), and x_3 (cutoff). Note that all the polynomial coefficients (a_0, a_1, etc.) are actually functions of the input parameters (i.e., $a_1(y_0, x_1, x_2, y_2, x_3)$); however, for reasons of clarity, this was left out of the formulas.

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

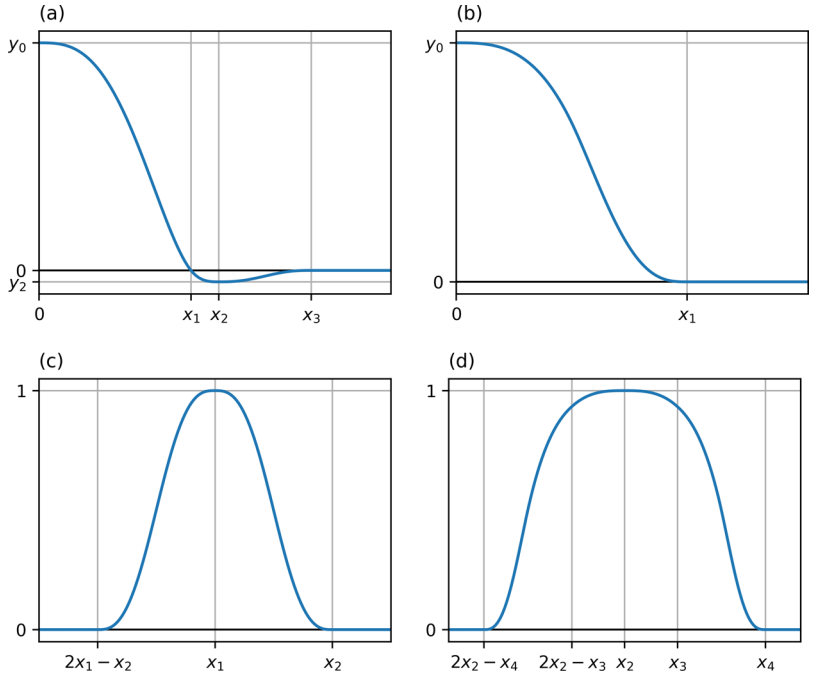


Figure 2. Basic functions used to model the interactions of the JAMDA score: (a) *LJ-like*, (b) *repulsive*, (c) two-parameter *plateau2*, and (d) three-parameter *plateau3*. Gray lines labeled with variable names denote the input parameters.

potential) can be adjusted. This can be used to easily reparameterize the scoring function, e.g., for soft docking[38,39] by reducing the zero-crossing but without changing the optimal distance and the cutoff value. We designed four different piecewise polynomial functions (*LJ-like*, *repulsive*, *plateau3*, *plateau2*) that are used for the different score contributions (Table 1). The *LJ-like* function is a piecewise polynomial approximation to the Lennard-Jones potential (defined in Figure 1 with analytical solutions given in the Supporting Information Section S1.3.1). The function $g_{LJ-like}(y_0, x_1, x_2, y_2, x_3, x)$ depends − in addition to x − on the five parameters y_0 (maximum repulsion), x_1 (zero-crossing), x_2 (minimum), y_2 (depth of minimum), and x_3 (cutoff value). It is shown in Figure 2a, and the constraints and free variables that build up the system of equations that is solved for the polynomial's coefficients are given in Figure 1.

The *repulsive* function is a repulsive term that is used for various clash contributions. The function $g_{repulsive}(y_0, x_1, x)$ depends − in addition to x − on the two parameters y_0 (maximum repulsion) and x_1 (cutoff value). It is shown in Figure 2b, and the details of its calculation are given in the Supporting Information Section S1.3.2. The *plateau2* function is a piecewise polynomial approximation to a plateau with two parameters. The function $g_{plateau2}(x_1, x_2, x)$ depends − in addition to x − on the two parameters x_1 (optimum) and x_2 (cutoff value) and is symmetric w.r.t. to x_1. It is shown in Figure 2c, and the details of its calculation are given in the Supporting Information Section S1.3.3. Finally, the *plateau3* function is a piecewise polynomial approximation to a plateau. The function $g_{plateau3}(x_2, x_3, x_4, x)$ depends − in addition to x − on the three parameters x_2 (optimum), x_3 (almost optimum), and x_4 (cutoff value) and is symmetric w.r.t. to x_2. It is shown in Figure 2d, and the details of its calculation are given in the Supporting Information Section S1.3.4. We have already used the *plateau3* function in the scoring function for our macrocycle conformer generator.[40] Note that the two polynomial approximations to plateaus were designed such that there is no actual plateau but always a nonzero slope (except for the maximum value).

Lipophilic Interactions. The pairwise function f_{lipo} in eq 4 is expressed in terms of $d(a, b)/\sigma(a, b)$, with $d(a, b)$ being the actual distance between two atoms and $\sigma(a, b)$ being the sum of van der Waals radii of the respective atoms. For carbon atoms with attached hydrogens, a united atom radius of 1.85 Å is assumed.

$$F_{lipo} = \sum_{a \in ligand} \sum_{b \in protein} I_{lipo}(a, b) \cdot f_{lipo}(a, b) \tag{3}$$

$$f_{lipo}(a, b) = g_{LJ-like}\left(20.0, 0.95, \sqrt[6]{2}, -0.01, 1.7, \frac{d(a, b)}{\sigma(a, b)}\right) \tag{4}$$

The function is parameterized as follows: For $d/\sigma = 0$, the maximum repulsive value was set to 20.0. The function is 0.0 (nonrepulsive) for $d/\sigma = 0.95$ (allowing little more clash than the 6-12 potential), and the minimum is reached at $d/\sigma = \sqrt[6]{2}$ (the same minimum as the 6-12 potential), and the depth of the minimum is arbitrarily set to −0.01 (all other contributions are weighted relative to the lipophilic interaction, see Scoring Function Weighting). Finally, at $d/\sigma = 1.7$, the function is again 0.0 along with the first two derivatives, which is the cutoff value for this contribution. The cutoff value was chosen for a similar curve shape to the 6-12 potential.

Other vdW Interactions. f_{vdW} were modeled the same way as lipophilic interactions but receive a different weighting term. The complete vdW score contribution F_{vdW} is the sum over the function f_{vdW} applied to all atom pairs with one hydrophobic and one nonhydrophobic (and nonmetal) atom and given in eqs 5 and 6.

$$F_{vdW} = \sum_{a \in ligand} \sum_{b \in protein} I_{vdW}(a, b) \cdot f_{vdW}(a, b) \tag{5}$$

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

$$f_{\mathrm{vdW}}(a, b) = g_{LJ-like}\left(20.0{,}0.95, \sqrt[6]{2}, -0.01{,}1.7, \frac{d(a, b)}{\sigma(a, b)}\right) \quad (6)$$

The purely hydrophobic vdW interactions f_{lipo} have been separated from the other vdW interactions (f_{vdW}) to allow for a separate weighting term. A similar term is also used in the PLANTS scoring function[4] where it is named *buried*. Korb et al.[4] emphasized the need for this term that favorably scores interactions between apolar and polar atoms. Here, we have made the same observation in preliminary experiments that this term is crucial for obtaining a satisfactory pose ranking performance (results not shown).

Hydrogen Bonds. Hydrogen bonds are the most prevalent type of strongly directed interaction in protein−ligand complexes. In the JAMDA scoring function, hydrogen bonds are modeled using the terms f_{hb} and $f_{\mathrm{hb\text{-}clash}}$. The total hydrogen bond score F_{hb} and hydrogen bond clash score $F_{\mathrm{hb\text{-}clash}}$ are both calculated as the sum over functions f_{hb} and $f_{\mathrm{hb\text{-}clash}}$, respectively, applied to all atom pairs where one atom can act as a donor and the other one as an acceptor. The term f_{hb} represents the attractive distance and angle-dependent term consisting of one distance (between the two heavy atoms) and two or three angles (see Figure 3). The attractive part is

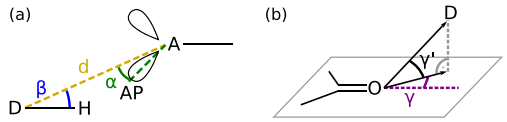


Figure 3. (a) Three parameters that are scored at a hydrogen bond (distance between donor and acceptor heavy atoms, angle at the donor, and angle at the acceptor). (b) The in-plane angle γ and the out-of-plane angle γ' measured at sp^2-hybridized oxygen atoms. This depiction is similar to Figure 2 in Nittinger et al.[42] For sp^2-hybridized oxygen atoms, the hydrogen bond score contains two angles at the acceptor (γ and γ') and therefore four parameters in total.

modeled as the product of multiple *plateau3* terms. Furthermore, instead of using the angle itself, the angle's cosine is used to obtain a continuously differentiable function.[41] The term $f_{\mathrm{hb\text{-}clash}}$ represents the repulsive term if the distance between the two heavy atoms gets too small. The hydrogen bond score must be split into two terms because the repulsive part must always be considered even if the attractive part is zeroed out by unfavorable angles.

We used the hydrogen bond model described by Nittinger et al.[42] (see also Böhm[43,44] and Klebe[45]). Briefly, each (potential) hydrogen bond can be described by a geometry resembling a spherical cone, a capped cone, or a spherical rectangle at the acceptor and donor.

The capped and spherical cones are both described by the distance between the heavy atoms and two angles (see Figure 3a). Consequently, the complete hydrogen bond can be described by four coordinates: the centers of the two heavy atoms (A and D in Figure 3a) as well as the hydrogen atom at the donor (H in Figure 3a) and an additional reference point (AP in Figure 3a), which usually is the position of a free electron pair (and will be called *acceptor interaction point* hereafter). The acceptor interaction point is placed at a 1 Å distance to the acceptor heavy atom A. cos α is given using eq 7 as $f_{\cos}(A, D, AP)$, and cos β is given as $f_{\cos}(D, A, H)$, and finally, the distance d in Figure 3a is given with eq 2.

$$f_{\cos}(a, b, r) = \frac{(\mathbf{c}(b) - \mathbf{c}(a))\cdot(\mathbf{c}(r) - \mathbf{c}(a))}{\|\mathbf{c}(b) - \mathbf{c}(a)\|\cdot\|\mathbf{c}(r) - \mathbf{c}(a)\|} \quad (7)$$

The modeling for the rectangular geometry is similar, except that it uses the two angles shown in Figure 3b and is described in more detail in the Supporting Information Section S1.6.

The optimal and acceptable angle and distance values in eq 11 derived from PDB statistics are based on Nittinger et al.[42] Note that the distance values are independent from the type of atom, but the angle values are not: Here, α_a^{opt} describes the optimal angle α for the given atom a (is it a water molecule? in a carboxylate?).

We can calculate the hydrogen bond score $f_{\mathrm{hb}}(a, b)$ for two atoms a and b with eq 11. Let $H(b)$ be the set of all hydrogen atoms of atom b and $AP(a)$ be the set of all acceptor interaction points of atom a. The factor 1.5 in the plateau parameters of eq 11 is used to account for the larger deviations from ideal values that are expected in docking poses.

If a has no acceptor interaction points or b has no hydrogen atoms, then the score will be zero. The plateaus in eq 11 are actually multiplied with further plateaus (not shown) that reduce the score to zero if any interatomic distance at the angle is close to 0 Å. This ensures the continuity of the function and its derivatives and has no influence on the score if no significant clashes are present. We have already described this technique before for our macrocycle conformation generator.[40]

$$F_{\mathrm{hb\text{-}clash}} = \sum_{a\in\mathrm{ligand}} \sum_{b\in\mathrm{protein}} I_{\mathrm{hb}}(a, b)\cdot f_{\mathrm{hb\text{-}clash}}(a, b) \quad (8)$$

$$f_{\mathrm{hb\text{-}clash}}(a, b) = g_{repulsive}\left(20.0{,}0.95\cdot\frac{2.9}{\sqrt[6]{2}}, d(a, b)\right) \quad (9)$$

$$F_{\mathrm{hb}} = \sum_{a\in\mathrm{ligand}} \sum_{b\in\mathrm{protein}} I_{\mathrm{hb}}(a, b)\cdot(f_{\mathrm{hb}}(a, b) + f_{\mathrm{hb}}(b, a)) \quad (10)$$

$$\begin{aligned} f_{\mathrm{hb}}(a, b) = \sum_{r\in AP(a)} \sum_{h\in H(b)} & g_{plateau3}(2.9{,}3.2{,}3.8, d(a, b))\cdot \\ & g_{plateau3}(0{,}1.5\cdot(\cos\alpha_a^{\mathrm{opt}} - \cos\alpha_a^{\mathrm{maxopt}}){,}1.5\cdot \\ & (\cos\alpha_a^{\mathrm{opt}} - \cos\alpha_a^{\mathrm{max}}), -f_{\cos}(a, b, r))\cdot \\ & g_{plateau3}(0{,}1.5\cdot(\cos\beta_b^{\mathrm{opt}} - \cos\beta_b^{\mathrm{maxopt}}){,}1.5\cdot \\ & (\cos\beta_b^{\mathrm{opt}} - \cos\beta_b^{\mathrm{max}}), -f_{\cos}(b, a, h)) \end{aligned} \quad (11)$$

Metal Interactions. Metal interactions are another type of directed interaction. Here, a very similar model to the hydrogen bonds is used. However, only the angles on the side of the metal acceptor are used. Modeling different metal coordination geometries is possible but neglected here for the sake of simplicity. The score contributions of metal interactions are given in eqs 12−15.

$$F_{\mathrm{metal\text{-}clash}} = \sum_{a\in\mathrm{ligand}} \sum_{b\in\mathrm{protein}} I_{\mathrm{metal}}(a, b)\cdot f_{\mathrm{metal\text{-}clash}}(a, b) \quad (12)$$

$$f_{\mathrm{metal\text{-}clash}}(a, b) = g_{repulsive}\left(20.0{,}0.95\cdot\frac{2.0}{\sqrt[6]{2}}, d(a, b)\right) \quad (13)$$

$$F_{\mathrm{metal}} = \sum_{a\in\mathrm{ligand}} \sum_{b\in\mathrm{protein}} I_{\mathrm{metal}}(a, b)\cdot f_{\mathrm{metal}}(a, b) \quad (14)$$

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

$$f_{\text{metal}}(a, b) = \sum_{r \in A(a)} g_{plateau3}(2.0,2.5,3.5, d(a, b)) \cdot g_{plateau3}(0,1.5 \cdot (\cos \alpha_a^{\text{opt}} - \cos \alpha_a^{\text{maxopt}}),1.5 \cdot (\cos \alpha_a^{\text{opt}} - \cos \alpha_a^{\text{max}}), -f_{\cos}(a, b, r)) \quad (15)$$

Other Clash Terms. Other clash terms are used to penalize intermolecular atomic penetrations that are purely repulsive and have not been described in the preceding paragraphs (e.g., between two donor atoms). Furthermore, they are also calculated intramolecularly to avoid the generation of self-clashing poses during the optimization procedure. For the intramolecular score terms, an additional indicator function eq 16 has been introduced that excludes atom pairs that are 1,4-connected (or closer). In eqs 17 and 18, intramolecular lipophilic clashes are given as an example. The choice of the softening factor 0.7 is the same as in our small molecule conformation generator Conformator.[40] The remaining clash terms are very similar and can be found in the Supporting Information Section S1.4. An important edge case is the clash between metal atoms and atoms covalently bound to hydrogen bond acceptors (e.g., the carbon atom of a carboxylate): The van der Waals radius of the metal atom is reduced by factor 0.5 for the clash term evaluation to polar/apolar atoms that are covalently bound to hydrogen bond acceptors. This is done to acknowledge the fact that these atoms may get considerably closer than the sum of their van der Waals radii.

$$I_{\text{topological}}(a, b) = \begin{cases} 1 & \text{if shortest-path } (a, b) \geq 4 \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

$$F_{\text{intra-lipo-clash}} = \frac{1}{2} \sum_{a \in \text{ligand}} \sum_{b \in \text{ligand}} I_{\text{topological}}(a, b) \cdot I_{\text{lipo}}(a, b) \cdot f_{\text{intra-lipo-clash}}(a, b) \quad (17)$$

$$f_{\text{intra-lipo-clash}}(a, b) = f_{\text{repulsive}}\left(20.0,0.7, \frac{d(a, b)}{\sigma(a, b)}\right) \quad (18)$$

Torsion Angles. Torsion angles are scored using a continuous torsion angle potential based on torsion angle peaks from a freely available torsion angle library derived from the CSD[46,47] that are smoothed by the periodic von Mises distribution. A torsion library consists of a sorted set of patterns describing the chemical environment around an acyclic bond. This torsion angle potential has already been used for macrocycle conformation generation in our small molecule conformation generator Conformator.[40] The indicator function $I_{\text{torsion}}(\text{bond})$ is 1 for acyclic single bonds (excluding terminal hydrogen bond donors/acceptors, e.g., hydroxy groups) and 0 for any other bond.

$$F_{\text{torsion}} = \sum_{\text{bond} \in \text{ligand}} I_{\text{torsion}}(\text{bond}) \cdot f_{\text{torsion}}(\text{bond}) \quad (19)$$

The continuous torsion score is based on a frequency histogram and does not represent relative energy differences of different torsion bonds (e.g., the score value for a rotation about a torsion bond between two sp^3-hybridized carbon atoms and about an amide bond is the same). To avoid unreasonable changes during optimization, for typical conjugated systems, an additional restriction has been implemented in the term $F_{\text{conjugated}}$.

The torsion bonds for the additional restriction $f_{\text{conjugated}}$ are selected based upon SMARTS expressions.[48] For ester derivatives ([*:1]-[#6X3:2](=[#8,#16,#7])-[#8X2H0:3]-[*:4]) and amide derivatives ([*:1]-[#6X3:2]-(=[#8,#16,#7])-[#7X3!H2:3]-[*:4]), the indicator function $I_{\text{conjugated}}(\text{bond})$ is 1 and the torsion angles are restricted to lie around 0 or 180° (with a tolerance of 5° in each direction) using a sum of *plateau2* functions (see Figure S8; this idea is similar to multidimensional ellipsoidal traps[19]). The scoring contribution is given in eqs 20 and 21.

$$F_{\text{conjugated}} = \sum_{\text{bond} \in \text{ligand}} I_{\text{torsion}}(\text{bond}) \cdot I_{\text{conjugated}}(\text{bond}) \cdot f_{\text{conjugated}}(\text{bond}) \quad (20)$$

$$f_{\text{conjugated}}(\text{bond}) = 20 \cdot (g_{plateau2}(-270°,-185°,\phi(\text{bond})) + g_{plateau2}(-90°,-5°,\phi(\text{bond})) + g_{plateau2}(90°,175°,\phi(\text{bond})) + g_{plateau2}(270°,355°,\phi(\text{bond})) \quad (21)$$

Further Design Choices. The JAMDA scoring function does not include an implicit solvent model; i.e., relevant water molecules must be present in the structure (or be predicted using, e.g., our tool WarPP[49]). We went without an implicit solvent model because our primary goal was the development of a scoring function that is efficient enough for docking. It would be an interesting extension of the JAMDA scoring function to integrate an efficient and differentiable implicit solvent model (e.g., the GB/SA model of Qiu et al.[50]) and assess its influence on the run time as well as the pose prediction performance.

We also did not include an electrostatics term as most electrostatic interactions are already described by the existing terms with the notable exception of interactions of charged atoms without involvement of metal or hydrogen bond interactions. Judging from the pose prediction performance (experiments 1−4), this does not seem to be a huge problem; however, these interactions could also be included.

Scoring Function Weighting. The JAMDA scoring function is the weighted sum of the different scoring contributions (eq 1). The weights of all clash contributions and the lipophilic term were set to 1 (see Table 1). The remaining four weights (w_{hb}, w_{metal}, w_{vdW}, w_{torsion}) were determined empirically. The JAMDA scoring function was parameterized with the docking task in mind, and consequently, the pose ranking performance was used as the objective function for the optimal selection of weights.

The calibration data was prepared as follows: For each complex in the calibration data set (see Datasets), the ligand was docked into the binding pocket using the TrixX algorithm[51,52] with max. 250 conformations generated with Conformator.[40] Here, the original ligand was also added to the set of conformations. The repulsive score (that is left uncalibrated, see Table 1) of the JAMDA scoring function was calculated for each pose, and poses with a score >1.0 were discarded. This was done to remove all the poses from the calibration dataset that are deemed to be too unfavorable by the JAMDA score. Similar to the procedure described by Su et

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

al.,[8] the poses were then sorted into bins depending on the RMSD to the crystal structure. We used 10 bins with a width of 1 Å covering the RMSD range of 0 − 10 Å. The poses within each bin were sorted increasingly by repulsive JAMDA score and clustered to a size of at most 100 poses using an adaptive sphere exclusion clustering algorithm.[40] Finally, the crystal ligand was always added to the poses. This resulted in a dataset where, for each target, at most 1001 poses (100 per RMSD to crystal structure bin and the crystal ligand) are provided. The weights of the different scoring contributions were sampled on a grid (see Table 2 for values), the resulting score for each pose was calculated, and the poses were ranked according to the score. We selected the weight parameter combination for which the number of targets with a top scored pose with an RMSD ≤2.0 Å to the crystal structure was maximal. That way, we calibrated the four weights of the scoring function for the pose prediction task, i.e., to separate poses close to the crystal structure from decoy poses. The weight ranges were chosen deliberately, i.e., the maximum value for w_{vdW} was set to 1, so it will not be weighted higher than the related $w_{lipophilic} = 1$. Furthermore, a minimum of $w_{vdW} = 0.05$ was chosen so that this term is guaranteed to be included because it is essential to controlling the torsion angles during optimization (the optimization is not part of the calibration itself). Values outside of the given range might give marginally better performance on the calibration dataset (see the Supporting Information Section S1.8).

Table 2. Parameters Used in the Grid Search for the Calibration of the JAMDA Scoring Function[a]

weight	values
w_{hb}	− 0.05, − **0.1**, − 0.2, − 0.3, − 0.4
w_{metal}	− 0.1, − **0.2**, − 0.3, − 0.4, − 0.6, − 0.8
w_{vdW}	0.1, 0.25, 0.5, 0.75, **1.0**
$w_{torsion}$	**0.05** , 0.1, 0.2

[a]Note that the values are positive for w_{vdW} because f_{vdW} uses the *LJ-like* function that already has the correct sign, and the values for $w_{torsion}$ are positive because this is a penalty term. The optimal parameter combination on the calibration dataset is marked in bold.

Molecular Representation. Small molecules are represented using three translational (T_x, T_y, T_z) and three rotational (R_1, R_2, R_3) degrees of freedom as well as rotatable torsion bonds (ϕ_i). Bonds within rings and double/triple bonds are treated as not rotatable. This representation − in the following referred to as rigid-body and torsion representation − has been used often in pose optimization[4,5,20,21] as it allows the freezing of noninteresting DOFs (e.g., bond lengths and bond angles) leading to less DOFs and simpler scoring functions. Abagyan et al.[19] used a different formulation with internal coordinates and virtual atoms that eventually also allows, among others, torsional flexibility and global rotation and translation.

Applying global translations to small molecules is straightforward, and they are added to the coordinates. The adjustment of torsion angles in a molecule has been described in detail by Schärfer et al.[46] and Friedrich et al.[40] (see Figure 4 as an example). Briefly, the molecule with its rotatable torsion bonds is represented as a directed rooted tree, and torsional transformations are applied only to atoms that are descendants of the torsion bond in the tree. This has two advantages for the optimization: First, for each torsion bond, it is clearly defined which atoms move if the torsion angle is changed and which atoms do not move. This is important for the calculation of derivatives. Second, the coordinates of the tree root are never changed by modifying torsion angles, and it is used as the center of global rotation.

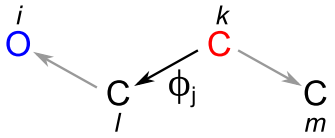


Figure 4. Representation of the torsion bonds of a 1-propanol molecule. The red carbon atom is the root of the tree, and its coordinates are never changed by modifying torsion angles. The torsion angle ϕ_j is defined by the four atoms (m, k, l, i). The blue marked oxygen atom is the only heavy atom whose coordinates are changed by modifying torsion angle ϕ_j.

There are different possibilities to represent global rotational DOFs. The classical way are Euler angles. These have the disadvantage of the gimbal lock phenomenon (see, e.g., Grassia[53] for a description of this well-studied problem). In the case of the gimbal lock, certain angle combinations lead to the loss of a rotational DOF. An interesting alternative is the use of the exponential map rotation that was described by Grassia[53] and applied to molecular systems by Fuhrmann et al.[20] and later Mirzaei et al.[54] The three rotational DOFs are represented by a vector $\mathbf{p} \in \mathbb{R}^3$ that is mapped into the unit quaternion space S^3. The quaternions can in turn be represented as rotation matrices. The advantage of this process is that, although the exponential map representation has singularities analogous to the gimbal lock, these singularities can be eliminated by keeping, e.g., $\| \mathbf{p} \| \leq \frac{3}{2}\pi$.[20] For details of this formulation, we refer to the respective publications.[20,53]

During optimization, this can be achieved by limiting the step lengths accordingly and by reparameterizing $\hat{\mathbf{p}} = \left(1 - \frac{2\pi}{\| \mathbf{p} \|}\right)\mathbf{p}$ if $\|\mathbf{p}\| \geq \pi$. In this way, the system stays away from the singularity and the derivatives also point away from it.[20,53]

Derivatives of Scoring Functions. An important prerequisite for the use of gradient-based optimization methods on a scoring function f is the calculation of the partial derivative $\frac{\partial f}{\partial P}$ with respect to each optimization parameter P. The gradient for the rigid-body and torsion molecular representation is given as eq 22.

$$\nabla f = \left(\frac{\partial f}{\partial T_x}, \frac{\partial f}{\partial T_y}, \frac{\partial f}{\partial T_z}, \frac{\partial f}{\partial R_1}, \frac{\partial f}{\partial R_2}, \frac{\partial f}{\partial R_3}, \frac{\partial f}{\partial \phi_1}, \ldots, \frac{\partial f}{\partial \phi_M} \right)^T \tag{22}$$

In general (and without further assumptions), no expression can be given for eq 22 without detailed knowledge of the scoring function. However, often, a protein−ligand scoring function can be decomposed according to eq 23 into a pairwise (coordinate dependent) and a torsional part (torsion angle dependent).[19] Abagyan et al.[19] actually identified another part termed *complex* that is not differentiable and therefore not used during minimization. Examples are the PLANTS scoring function,[4] the AutoDock Vina,[5] or the ChemScore[32] without the bond freezing contribution.

$$f = f_{pairwise} + f_{torsion} \tag{23}$$

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

Note that coordinate-dependent terms are not limited to atoms but can also cover virtual positions of free electron pairs as they appear, e.g., in a hydrogen bond definition.

By application of the chain rule, we can express $\frac{\partial f_{\text{pairwise}}}{\partial P}$ for a function with N coordinates c_1, ..., c_N as eq 24. This idea is central for the gradient-based optimization of any molecular system.[5,19−21,55]

$$\frac{\partial f_{\text{pairwise}}}{\partial P} = \sum_{i=1}^{N} \frac{\partial f_{\text{pairwise}}}{\partial \mathbf{c_i}} \frac{\partial \mathbf{c_i}}{\partial P} \tag{24}$$

This important result shows that we can calculate the derivative of the scoring function by calculating the derivative of the coordinates with respect to the optimization parameters $\frac{\partial \mathbf{c_i}}{\partial P}$ and the derivative of the scoring function w.r.t. the coordinates $\frac{\partial f_{\text{pairwise}}}{\partial \mathbf{c_i}}$. The latter can be reused for multiple parameter derivatives; i.e., $\frac{\partial f_{\text{pairwise}}}{\partial \mathbf{c_i}}$ should only be calculated once and not for each parameter P. We will give formulas for the derivatives of the coordinates w.r.t. the optimization parameters $\frac{\partial \mathbf{c_i}}{\partial P}$ in the following as these are independent from the actual scoring function; i.e., the result can be used for all different kinds of intermolecular scoring functions that can be decomposed according to eq 23. The derivatives of the scoring function w.r.t. the coordinates $\frac{\partial f_{\text{pairwise}}}{\partial \mathbf{c_i}}$ depend on the scoring function and are described in Derivatives of the JAMDA Scoring Function and the Supporting Information Section S1.7.

The derivative of the coordinates w.r.t. the translational parameters is trivial because the global translation changes the coordinate $\mathbf{c_i}$ linearly.

$$\frac{\partial \mathbf{c_i}}{\partial T_x} = (1,0,0)^T \tag{25}$$

$$\frac{\partial \mathbf{c_i}}{\partial T_y} = (0,1,0)^T \tag{26}$$

$$\frac{\partial \mathbf{c_i}}{\partial T_z} = (0,0,1)^T \tag{27}$$

The computation of torsional gradients is well established in the literature[20,55] and is given by eq 28. The following formula gives the derivative of the coordinate $\mathbf{c_i}$ if it is moved by changing the torsion bond ϕ_j defined by the two atoms/coordinates $\mathbf{c_k}$ and $\mathbf{c_l}$ (see Figure 4). As the torsion bonds are represented as a directed rooted tree, the torsion bond ($\mathbf{c_k}$, $\mathbf{c_l}$) is directed and $\mathbf{c_i}$ can be any atom that is a descendant of $\mathbf{c_l}$ in the tree.

$$\frac{\partial \mathbf{c_i}}{\partial \phi_j} = \frac{(\mathbf{c_k} - \mathbf{c_l})}{\|\mathbf{c_k} - \mathbf{c_l}\|} \times (\mathbf{c_i} - \mathbf{c_l}) \tag{28}$$

For the calculation of the derivatives of the coordinates w.r.t. the parameters of the exponential map $\mathbf{p}$ (used for global rotations), we refer the interested reader to the paper of Grassia.[53] These derivatives must be calculated after the adjustment of torsion angles but before the global rotation is applied. The reason for this is that the coordinate derivatives w.r.t. the exponential map parameters must be calculated on the coordinates before application of the global rotation.[20]

Derivatives of the JAMDA Scoring Function. For use in a gradient-based optimization algorithm, the derivatives of the pairwise part of the JAMDA scoring function w.r.t. the atomic coordinates and the derivative of the torsional part w.r.t. the torsion angles are required. All pairwise scoring contributions of JAMDA exclusively depend on the distances or angles between coordinates. Therefore, again, the chain rule can be used to calculate the derivatives of the individual scoring contributions w.r.t. to the atomic coordinates (for details, see the Supporting Information Section S1.7). The torsion terms exclusively depend on the torsion angles that are also the parameters of the optimization, and the derivatives can be directly used. The derivatives of the torsional term f_{torsion} w.r.t. to the torsional parameters are calculated from the derivatives of the von Mises distribution w.r.t. to the torsion angle.

The (L-)BFGS Algorithm. The (L-)BFGS algorithm is one of the frequently applied state-of-the-art algorithms for unconstrained optimization of a nonlinear function $f(\mathbf{x})$: $\mathbb{R}^n \rightarrow \mathbb{R}$ that is at least once continuously differentiable. The original BFGS algorithm was described independently by Broyden,[27,28] Fletcher,[29] Goldfarb,[30] and Shanno[31] and is named after the initials of the four authors. Two important modifications have been published since the original description: Nocedal[56] introduced the L-BFGS algorithm that is also suited for very high dimensional optimization problems.[57] Another important variant is the (L)-BFGS-B algorithm that handles simple box constraints.[58,59] Since a basic understanding of the BFGS algorithm is required, we summarize the algorithm in the following (a more detailed description can be found in the chapter on quasi-Newton methods by Nocedal and Wright[60]).

The BFGS algorithm is a quasi-Newton method that uses information of the function value, the first derivative (the gradient) and also the second derivative (the Hessian matrix). However, the Hessian matrix is not explicitly calculated, but an approximation is stored and updated within the BFGS algorithm. Each iteration of the BFGS algorithm consists of obtaining a new search direction (Where to go next?), a line search (How far to go in the new search direction?), and the update of the Hessian approximation.

Problems with the BFGS Algorithm. Despite the success and wide use of the BFGS algorithm, there is a problem with the BFGS algorithm for the optimization of molecular scoring functions: The algorithm may take such a large step that the ligand is moved outside of the binding pocket of the protein (see also Experiment 5: How Much Do Different Optimization Algorithms Change the Molecular System?). This tends to happen when there are intermolecular clashes.

An example is shown in Figure 5 where we have optimized a one-dimensional truncated and shifted-force 6-12 potential[61] from different starting positions with the L-BFGS implementation in the NLopt library.[62,63] Here, it becomes obvious that when optimization starts from some positions, then not the local optimum (that is also a global optimum) at $\sqrt[6]{2}$ is found but some stationary point far away ≥3.

For intermolecular scoring functions, this is especially problematic because of their characteristic properties: For short intermolecular atomic distances (clashes), the flanks of the scoring function are steep. This may lead to excessively large steps taken during optimization. Furthermore, after a

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

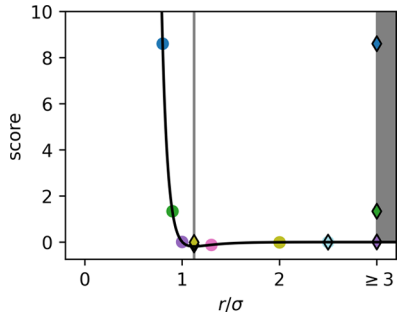


Figure 5. Optimization of a truncated and shifted-force 6-12 potential ($\varepsilon = 0.2$, truncated at $\frac{r}{\sigma} = 2.5$) with the L-BFGS implementation in NLopt.[62,63] The dots represent the starting positions of the optimization, and the same-colored diamonds represent the x-coordinates of the end points of the optimization (for the sake of clarity, the shown y-coordinate after optimization is the same as before). All results with an x-coordinate >3 were set to 3 for plotting. The vertical gray line at $\sqrt[6]{2}$ denotes the global minimum.

certain cutoff distance, the interaction energy is (almost) 0.0 along with its gradients. Therefore, a trivial − and completely undesired − stationary point for almost every intermolecular scoring function is far away from the protein. The combination of these two properties leads to the observed behavior: First, an excessively large step is taken that cannot be corrected because the system immediately ends up in a stationary point.

The effect becomes even more pronounced for the rigid-body and torsion molecular representation used in this paper because especially the values for the rotational gradients tend to be quite large. However, it can also be observed if only translation is performed (cf. Figure 5).

The problem of larger-than-expected movements during optimization especially for the rigid-body and torsion molecular representation has been described in the literature before. In DOCK 6,[12] different possibilities were explored to limit larger-than-expected ligand movements during simplex minimization. Among others, an RMSD restraint minimizer has been described to limit the ligand's movements during minimization. Mirzaei et al.[54] also commented on this issue during optimization with the BFGS algorithm. In case of too large global rotational movements at the first step, they propose to scale the diagonal elements of the initial estimate of the inverse Hessian matrix that correspond to the global rotational parameters. If it happens again, they reinitialize the inverse Hessian approximation to the identity matrix and restart the optimization.

A lot of other workarounds could be used to avoid excessive movements in the BFGS algorithm: Additional penalty terms that penalize movements from the ligand outside of the pocket (or scoring grid) can be introduced − an example for this is AutoDock Vina[5] (however, it is unclear if this was introduced in AutoDock Vina also for this reason). Linear constraints could be directly incorporated using, e.g., the L-BFGS-B algorithm,[58,59] or nonlinear constraints could be introduced (see Nocedal and Wright[64] for an overview). However, all constrained methods share the same disadvantages, namely, that they somehow restrict the search space as a whole while they do not prevent excessive ligand movements within the given bounds. Another commonly used technique in force field optimization is to resolve clashes first using gradient descent with a limitation of the step length (see below) and then switch to another algorithm. This approach has been used for force field optimization for more than 40 years.[65]

Instead of applying a workaround, we wanted to tackle this problem by adapting the BFGS algorithm itself (like Mirzaei et al.[54] did). However, our goal was to design a general algorithmic concept to include domain knowledge into the step length selection process within the BFGS algorithm. In the context of pose optimization, this would allow us to limit the BFGS algorithm in such a way that no atoms move more than, e.g., 0.5 Å in each step during the optimization. This would prevent excessive movement of the ligand during optimization and increase the locality of the solutions.

Limitation of the Step Length. This problem can be generally solved by limiting the step lengths, i.e., restrict how far the system is allowed to change in each iteration (*without* constraining the optimization).

A limitation of the step lengths is encountered often in numerical optimization. It is regularly done for the steepest descent algorithm (see, e.g., the reference manual of GROMACS 2019[66] or Niketić and Rasmussen[65] for examples in the context of force field minimization). Generally, for the steepest descent and conjugate gradient[67] algorithm, the steps are known to be poorly scaled and different strategies have been proposed.[68] Within the context of the BFGS algorithm, the need for a limitation of step lengths was recognized by Byrd et al.[58] for the (L-)BFGS-B algorithm. Here, a step length limitation is necessary to ensure that the solution remains within the box constraints. It was also employed by Fuhrmann et al.[20] during the optimization of molecules using the exponential map rotation representation to avoid the gimbal lock phenomenon (see also Molecular Representation). Press et al.[69] suggested a scaling of the direction vector to avoid excessive steps.

Interestingly, in the field of computational quantum chemistry, a complete set of distinct quasi-Newton algorithms evolved that center around the restriction of step lengths for improving and accelerating the convergence of quantum chemical geometry optimizations (see Hratchian and Schlegel[70] for an overview). Most of these algorithms require the definition of a trust radius that also defines the maximum allowed change per step during optimization.

As soon as a maximal step length is externally specified, additional domain knowledge is considered, i.e., how far the minimum approximately is away. However, for a given optimization system, it may not always be apparent which step length is appropriate: This is obviously the case when using the rigid-body and torsion representation used in this paper. The influence of the global rotation parameters depends on the shape of the molecule (is it spherical or rod-shaped?), and the influence of the torsion angles depends on their position within the molecule (or rather the tree structure).

Consequently, our goal was to design a BFGS variant that allows a more generic and also dynamic specification of the maximum allowed step length and avoids common pitfalls that result from restricting the step lengths.

Our new LSL-BFGS algorithm requires − in addition to the objective function $f(\mathbf{x})$ − the specification of an acceptance function

$$G(\mathbf{x}, \mathbf{y}): \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R} \tag{29}$$

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

$$G(\mathbf{x}, \mathbf{y}) = \begin{cases} > 0, & \text{if the change from } \mathbf{x} \text{ to } \mathbf{y} \\ & \text{is not acceptable} \\ \leq 0, & \text{otherwise} \end{cases} \quad (30)$$

that defines if the change from $\mathbf{x}$ to $\mathbf{y}$ is acceptable. The acceptance function $G(\mathbf{x}, \mathbf{y})$ is directly incorporated into the line search algorithm (see Algorithm S1). Additionally, if an analytical solution for the maximum step length is accessible (or for an upper bound), this can also be incorporated (see the Supporting Information Section S2.2). Further modifications of the BFGS algorithm to ensure convergence are given in the Supporting Information Section S2.3. In addition to the LSL-BFGS algorithm, we also implemented a variant of the BFGS algorithm that does not use the additional domain knowledge provided by $G(\mathbf{x}, \mathbf{y})$ but automatically scales the step lengths with a technique usually used in other optimization algorithms (see the Supporting Information Section S2.5 for details). This algorithm will be designated as LSLa-BFGS. For the sake of clarity, we will only provide results for the LSL-BFGS algorithm in the main paper and the results for the LSLa-BFGS algorithm in the Supporting Information.

In summary, our novel contribution is as follows: Instead of only allowing a simple specification of a maximum step length or the calculation of a maximum step length analytically from the search direction vector, we make the search for the maximum allowed step length part of the line search and therefore make the inclusion of arbitrary domain knowledge about reasonable step lengths (by means of the function $G(\mathbf{x}, \mathbf{y})$) possible. This means that the information "it is sensible to move any atom at most 0.5 Å per step" can be included directly into the BFGS algorithm *independently* from the molecular representation. We combined this with techniques described by Al-Baali et al.[71] to ensure convergence.

Note that the automatic selection of step lengths is actually a strength of the BFGS algorithm, substantially contributing to its efficiency. Therefore, we propose a restriction of the step lengths only in the case that the automatically selected ones (usually very early in the algorithm) are obviously wrong and very likely will miss a close-by minimum. However, this can be easily overdone by enforcing too small steps if the system is far away from a minimum.

RESULTS AND DISCUSSION

The LSL-BFGS algorithm was parameterized such that no heavy atom and no additional reference point (lone pair or hydrogen atom at a donor/acceptor atom) moves more than 0.5 Å in each iteration, unless noted otherwise (see the Supporting Information Section S2.4 for the used formula). The default parameters used for each optimization algorithm are given in the Supporting Information Section S3.2.

The JAMDA scoring function was evaluated alone as well as together with our new step length limiting LSL-BFGS optimization algorithm. The experiments were designed to answer the following questions related to quality, locality, stability, and efficiency:

1. Can the JAMDA scoring function discriminate between native and decoy poses?
2. Are the optima of the JAMDA scoring function in agreement with crystal structures?
3. Does the optimization of the JAMDA scoring function improve the RMSD of poses?
4. Does the optimization of the JAMDA scoring function improve the pose ranking?
5. How much do different optimization algorithms change the molecular system?
6. How many scoring function evaluations are necessary for different optimization algorithms?

Finally, we present the chemical elaboration of ligands as a practical use case for the LSL-BFGS algorithm with JAMDA.

While the LSL-BFGS algorithm itself was implemented as the independent LSLOpt C++ library, the scoring function with the derivative calculation and all the experiments were implemented using the NAOMI library.[72,73] All reported RMSD values in this paper are symmetry-corrected heavy-atom RMSDs.

Datasets. As the test dataset, we used the CASF-2016 benchmark.[8] The CASF-2016 benchmark dataset is based upon the PDBbind v2016[74] and contains data and scripts for the evaluation of scoring functions. As a calibration dataset, we used the PDBbind v2016[74] *refined* data set. The complexes and ligands also present in the CASF-2016 *core set* (that is our test dataset) were removed from the dataset. Duplicate complexes were identified by PDB code, and ligands present in both sets were determined using Mona[75,76] after canonizing the charge and tautomeric state and removing stereoinformation. Furthermore, we applied a drug-likeness filter based upon the findings of Ghose et al.[77] to the ligands using Mona.[75,76] Only those complexes were kept where the ligands have a number of heavy atoms between 20 and 70, a molecular weight between 160 Da and 480 Da, and a log P value between −0.4 and 5.6. This resulted in a subset of 943 protein−ligand complexes from the PDBbind *refined* set (PDB codes are given in the Supporting Information Section S4).

Experiment 1: Can the JAMDA Scoring Function Discriminate between Native and Decoy Poses? The key feature of a scoring function for pose prediction is its ability to discriminate between (near-)native and decoy poses. We evaluated the JAMDA scoring function on the CASF-2016 benchmark dataset to check how well the JAMDA score discriminates between native and decoy poses. The evaluation was performed according to the *docking power* protocol of the CASF-2016 benchmark.[8] We did not restrict the scoring to the binding site; i.e., the scoring was performed against the whole structure. Figure 6 reproduces the results for all the scoring functions from the original CASF-2016 benchmark and also includes results for the JAMDA scoring function. Note that the poses were not optimized with respect to JAMDA in this experiment. In this benchmark, the JAMDA score is among the best performing scoring functions for the pose ranking task. An RMSD of the top-ranked pose to crystal structure of ≤2 Å was achieved in 88.8% of the 285 targets. An example for a scoring failure with a top-ranked pose >2 Å is given in Figure 7: Here, the top-ranked pose has a slightly lower score than the crystal structure on rank two: The better hydrogen bonds of the native structure do not compensate for a small f_{clash} clash and a lower f_{vdW} contribution in the JAMDA scoring function.

The results in Figure 6 clearly show the ability of JAMDA to discriminate between native and decoy poses.

Experiment 2: Are the Optima of the JAMDA Scoring Function in Agreement with Crystal Structures? As a next step, we optimized the ligand in the crystal structures of the CASF-2016 *core set* with the newly implemented LSL-BFGS algorithm and recorded the RMSD value between the

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

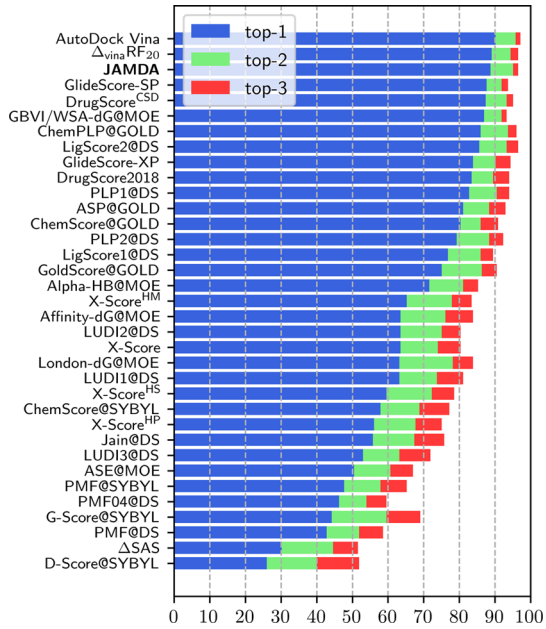


Figure 6. Results for the CASF-2016 dataset. The percentages of targets for which a pose with an RMSD of ≤2 Å is found on ranks one, two, and three are shown. Here, we recreated the plot from Figure 5 by Su et al.[8] The results for the JAMDA scoring function (marked in bold) were plotted together with the data for the other scoring functions from the CASF-2016 package.

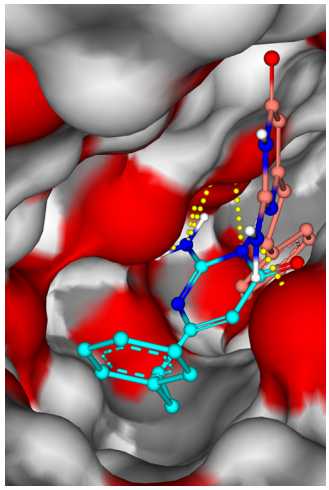


Figure 7. Example of a JAMDA scoring failure. The PDB code 2V00[78] is shown (endothiapepsin with a bound inhibitor). In blue, the second-ranked crystal structure (JAMDA score −1.254) is shown, and in light red, the top-ranked pose (decoy 325) is shown with an RMSD of 6.82 Å (JAMDA score −1.362). Hydrogen bonds are depicted as dashed yellow lines.

crystal structure and the optimized pose. The results are shown in Figure 8. Most structures (about 87%) are moved by less than 1 Å RMSD away from the crystal structure. Only a minority (about 1%) of the structures move by more than 2 Å RMSD.

We have analyzed the three examples with an RMSD >2 Å after optimization in detail: For the complex 3G2Z[79] (beta-lactamase CTX-M-9a with an inhibitor), the RMSD from the optimized structure to the crystal structure is 2.42 Å (Figure 9). A closer inspection reveals that the crystal structure contains the structurally relevant water molecule A526 that has

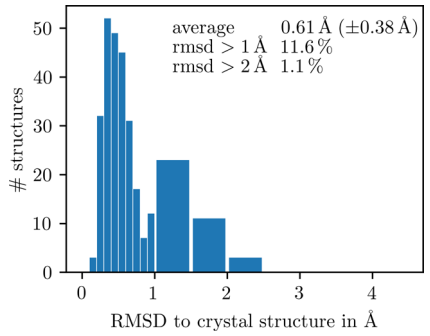


Figure 8. Histogram of the RMSD of the ligand after optimization with JAMDA scoring to the initial crystal structure is shown. The average RMSD and the percentage of structures with RMSDs >1 Å and >2 Å are given.

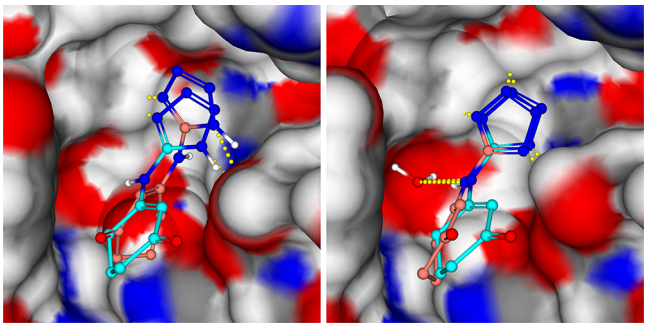


Figure 9. The complex 3G2Z[79] (beta-lactamase CTX-M-9a with an inhibitor) is shown before optimization (light blue) and after optimization (salmon). On the left side, the structure from the CASF-2016 dataset has been used. On the right side, the water molecule A526 has been added to the structure before optimization. The crystal structure and the optimized structure now make hydrogen bonds to the water molecule (dashed yellow lines). Note that executing Protoss[80] to correctly orient A526 also changes the protonation state of the tetrazole.

been removed during CASF-2016 preparation.[8] Readding this water molecule (and running Protoss[80] for the correct orientation of the water) results in a significantly lower RMSD of 0.88 Å after optimization (Figure 9). The situation is very similar for structure 1UTO[81] (bovine trypsin). Here, readding water molecules A2204 and A2210 (and running Protoss[80]) significantly lowers the RMSD after optimization from 2.32 down to 0.66 Å (Figure S10). For 1K1I[82] (bovine trypsin), the reason lies within crystal artifacts; i.e., the neighboring asymmetric units in the crystal are within interaction distance to the ligand (Figure S11). Including the neighboring asymmetric units leads to a much smaller RMSD of 0.76 Å to the crystal structure (instead of 2.18 Å).

Taken together, the three examples of an RMSD >2 Å after optimization can be attributed to missing crucial information in the given structures (water molecules and/or crystallographic symmetry).

We conclude that our JAMDA scoring function is generally in good agreement with crystal structures.

Experiment 3: Does the Optimization of the JAMDA Scoring Function Improve the RMSD of Poses? As our JAMDA scoring function has been developed for the pose prediction/docking task, it is important to analyze if the optimization of docking poses generally lowers the RMSD value and drives the pose into the direction of the crystal

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

structure. We compared the RMSD values of the decoy poses from the CASF-2016 dataset before and after optimization with the JAMDA score. This experiment has been suggested by Spitzmüller et al.[10] for the evaluation of their MiniMuDS optimizer. Note that the term *decoy pose* in this context (as used by the authors of the CASF benchmark) also includes poses close to the crystal structure. The results of this experiment are shown in Figure 10. For decoy poses with

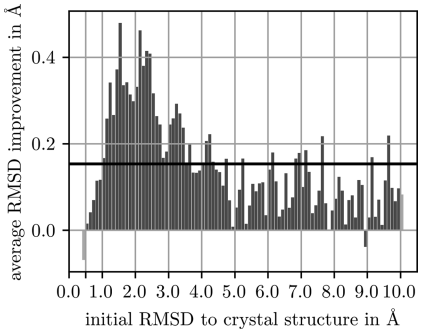


Figure 10. The mean improvement of RMSD values to the crystal structure by optimization of the decoy poses from the CASF-2016[8] benchmark are shown. The poses have been sorted into bins depending on their initial RMSDs (*before* optimization) to the crystal structure. The light gray bars pool the results for all poses with an RMSD of less than 0.5 Å or more than 10.0 Å, respectively. The black line represents the mean RMSD improvement.

initial RMSD values below 0.5 Å, the mean RMSD change is negative; i.e., the RMSD to the crystal structure is slightly increased. For all larger initial RMSD value bins (except one), a mean improvement of the RMSD values is indeed observed. This is especially true for the RMSD range between 1 Å and 3 Å that is important for the post-optimization of docking poses (these poses should be optimized as close to the crystal structure as possible).

Experiment 4: Does the Optimization of the JAMDA Scoring Function Improve the Pose Ranking? To check the influence of the numerical optimization on the pose ranking task, we performed the evaluation on the CASF-2016 dataset with optimization of the JAMDA score. It should be noted that the RMSDs have to be recalculated after the optimization, and therefore, the results are not obtained on the same poses as the other scoring functions in the CASF-2016 benchmark. To make up for potential differences in the RMSD calculation algorithms, we also recalculated the RMSDs for the decoy poses using our algorithm. The results for the JAMDA scoring function with and without optimization are shown in Figure 11. Interestingly, the pose prediction performance decreases significantly if local optimization is performed. To check if this observation is unique to our scoring function, we performed the same experiment with AutoDock Vina[5] by using the local optimization mode. We chose AutoDock Vina as a comparison because it shows a very high performance on this benchmark set, it provides a mode for local optimization, and finally because the source code is available. Parameters for AutoDock Vina experiments are given in the Supporting Information Section S3.1. Here, a similar observation is made. The local optimization decreases the pose ranking performance of the AutoDock Vina scoring (AutoDock Vina_{opt} in Figure 11), although not as much. This was further analyzed by removing the strict iteration limit in the local optimization

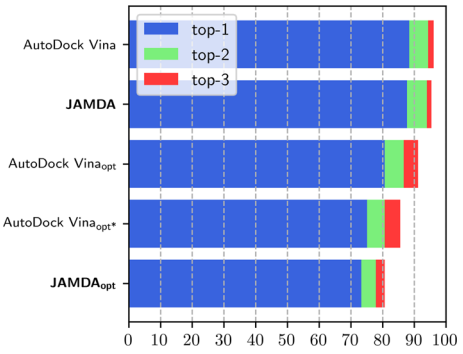


Figure 11. The results for JAMDA and AutoDock Vina with and without local optimization are shown. For AutoDock Vina, results for the default optimization (AutoDock Vina_{opt}) and the modified optimization without the strict iteration limit (AutoDock $\text{Vina}_{\text{opt*}}$) are given. The results without optimization use our RMSD calculation routine for better comparability and may show slight deviations from the results given in Figure 6.

routine of AutoDock Vina. By default, the number of BFGS iterations in the local optimization of AutoDock Vina is limited to $\frac{25 + \#\text{atoms}}{3}$. Interestingly, after removing the strict iteration limit, the pose ranking performance of AutoDock Vina (AutoDock $\text{Vina}_{\text{opt*}}$ in Figure 11) drops significantly. These results indicate that the observation of decreasing performance on the CASF-2016 docking power benchmark with local optimization also occurs for other scoring functions.

To analyze this observation further, we calculated the mean JAMDA score and the percentage of decoy poses in the CASF benchmark that receive unfavorable (positive) JAMDA scores before and after the optimization (Figure 12). Here, it

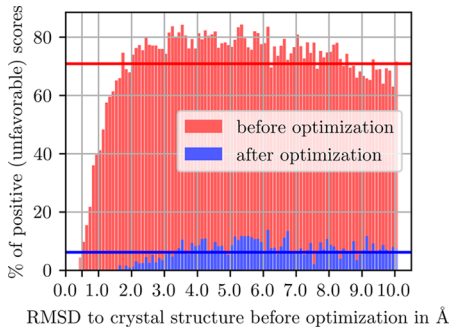


Figure 12. The percentages of positive (unfavorable) JAMDA scores are shown for the decoy poses of the CASF-2016 benchmark[8] before and after the optimization. The poses have been sorted into bins depending on their initial RMSD to crystal structure (as calculated using our RMSD algorithm). The lighter red/blue bars pool the results for all poses with an RMSD of less than 0.5 Å or more than 10.0 Å, respectively. The red and blue lines denote the mean positive score percentage before and after optimization, respectively.

becomes evident that the vast majority of decoy poses (especially with RMSD >1 Å) receives positive scores before the optimization (usually more than 70%). In contrast, only 2.5% of the native crystal structures receive positive scores before optimization. After optimization of decoy poses, the rate decreases significantly to often less than 10% of the structures. Similarly, the mean score value decreases after optimization (Figure S12).

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

A similar observation can be made for AutoDock Vina (Figure S13). These findings give a possible explanation for the observed performance drop in pose ranking: If a lot of poses with higher RMSDs have positive scores (but not the poses with lower RMSDs), then these higher RMSD poses are directly taken out of the competition for the top-ranked poses and poses with lower RMSDs are easily enriched on the lower ranks. After optimization, this effect is far less pronounced and a lot more poses receive favorable scores.

Taking these findings together, it is likely that the different clash definition or filtering of clashing poses of the docking tools used to generate the decoy poses in the CASF-2016 benchmark set[8] is (at least partly) responsible for the good performance of the JAMDA scoring function but also the AutoDock Vina scoring function in this benchmark. However, this may also hold true for any other scoring function in the CASF-2016 benchmark. It would be interesting to perform this experiment for all the other scoring functions in the CASF-2016 benchmark (i.e., perform a local optimization w.r.t. the scoring function − if possible for the given tool).

Since the use of force fields for pose minimization is common practice in modeling, we also evaluated the pose ranking performance (with and without optimization) for the OPLS_2005 force field[83] with the GB/SA solvation model[84] using Schrödinger's MacroModel[85] software. The detailed calculation protocol (including the necessary preprocessing steps) is given in the Supporting Information Section S3.4.1. The results are shown in Figure 13. Here, two important

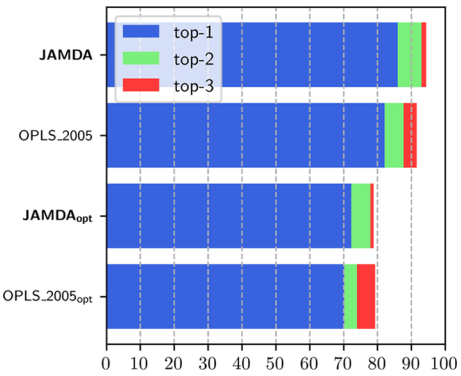


Figure 13. The results for JAMDA and the OPLS_2005 force field with the GB/SA solvation model with and without local optimization are shown. All results were obtained on the set of structures described in the Supporting Information Section S3.4.1. Therefore, the JAMDA results are different from the results in Figures 6 and 11.

observations can be made: First, the performance of the OPLS_2005 force field for pose ranking is similar to the performance of the JAMDA scoring function with the latter showing slightly better results. Second, also, the numerical optimization of the OPLS_2005 force field leads to a significant performance drop − similar to the observations for the JAMDA and the AutoDock Vina scoring function. It should be noted that also for the OPLS_2005 force field, only the ligand coordinates were optimized. It is possible that including the hydrogen atoms of the protein into the optimization might give better results. However, these experiments are beyond the scope of this paper.

All these observations again make a strong argument for performing local optimization before scoring. Using the unoptimized score may produce significant artifacts, as has been shown for our JAMDA scoring function, the established and widespread AutoDock Vina scoring function, and the OPLS_2005 force field. Performing numerical optimization before scoring reduces the presence of such artifacts, allowing for a more unbiased evaluation of the pose prediction performance.

Experiment 5: How Much Do Different Optimization Algorithms Change the Molecular System? The next experiment was designed to analyze the stability and locality of different optimization algorithms, namely, our novel LSL-BFGS, our BFGS implementation, the L-BFGS implementation in NLopt,[62,63] and the BOBYQA algorithm in NLopt.[62,86] It is checked how far a minimized starting structure can be deflected such that the optimization algorithm is still able to find the same minimum (or another close minimum) again and how often poses are moved unreasonably far away from the starting pose. Obviously, this is a property of the combination of scoring function and optimizer − the roughness of the energy landscape and peculiarities of the optimization algorithms will have a profound influence on the results. A simple example for the behavior of the optimization algorithms is given in Figure S17.

First, we optimized the crystal structures of the 285 protein−ligand complexes in the CASF-2016 *core set* with respect to the JAMDA scoring function using our LSL-BFGS algorithm to a high precision (see the Supporting Information Section S3.2 for the parameters used). Starting from the minimized structure, we increasingly deflected the structure by randomly rotating and translating the molecule and rotating around torsion bonds creating 313,785 starting structures in total. Clashing poses were not eliminated because this experiment should, among others, analyze the behavior of the optimization algorithms in the presence of clashes. The complete protein was used for scoring, not just the binding site.

The optimization procedure was repeated with all the deflected structures and different optimization algorithms, and the RMSD to the optimized crystal structure as well as to the starting point of the optimization (the deflected structure) was calculated.

While the stability and locality of the optimization algorithms are certainly related, we performed two slightly different evaluations. For the analysis of the stability, the RMSD to the minimized crystal structure is considered. In Figure 14a, the percentage of structures with a final RMSD up to 0.001 Å to the minimized crystal structure is shown. Up to an initial RMSD of 0.5 Å, the three evaluated gradient-based optimization algorithms are able to recover the starting structure to a high precision in almost all cases. After that, the percentage quickly drops, especially for the optimization algorithms other than LSL-BFGS. The latter is still able to recover the starting structure up to an initial RMSD of 2 Å in more than 50% of the cases. A similar behavior is observed in Figure 14b if a higher threshold of 0.1 Å is chosen as a successful recovery of the starting position. Here, also, the BOBYQA algorithm shows a good performance with an even higher reproduction rate of the starting pose than the LSL-BFGS algorithm for higher initial deflections. Because the starting structures were generated with a gradient-based optimization algorithm, the latter (higher) threshold of 0.1 Å is probably more fair for a comparison with gradient-free algorithms like BOBYQA in which the definition of the minima is different (because of missing access to the analytical

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

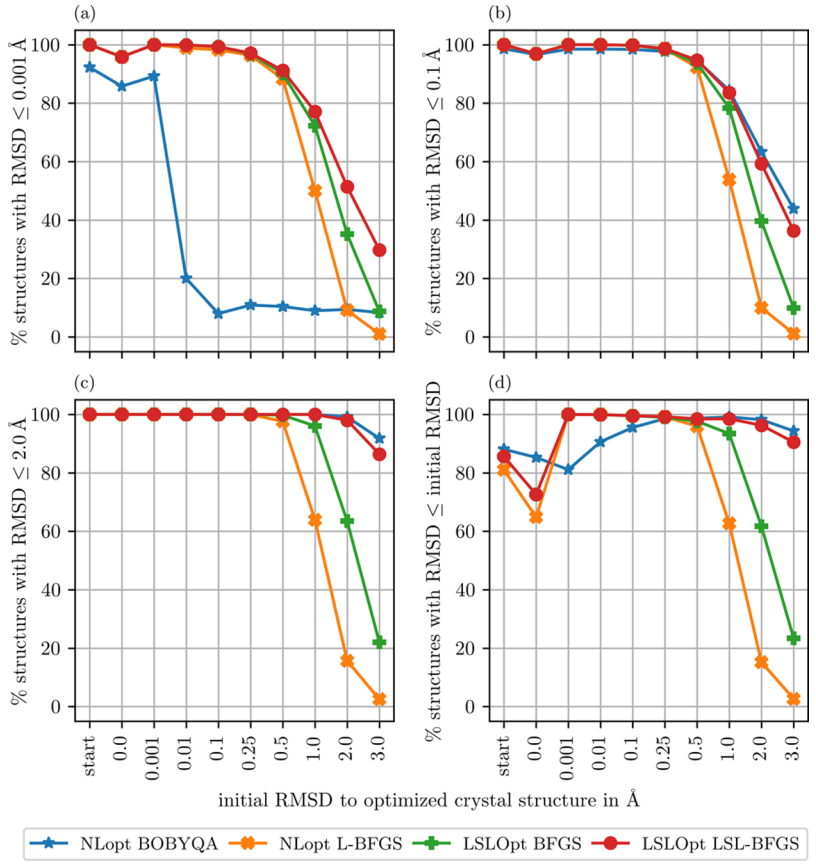


Figure 14. The percentages of structures with RMSDs to the optimized crystal structure of ≤0.001 Å (a), ≤0.1 Å (b), or ≤2.0 Å (c) are shown. Subplot (d) gives the percentage of structures that have a smaller RMSD after optimization than before. Optimization was performed w.r.t. JAMDA scoring. The results are given for four different optimization algorithms. The percentage is shown as a function of the binned initial RMSD of the structure. The difference between *start* and 0.0 is that the former contains only unmodified initial structures and the latter contains modified structures with a heavy-atom RMSD of 0.0 (e.g., with rotated H-bond donors).

gradient). Figure 14c bridges to the locality problem of some of the algorithms: While the LSL-BFGS and the BOBYQA algorithm are able to optimize most of the starting poses somewhere close (2 Å) to the optimized crystal structure, the NLopt L-BFGS and our BFGS implementation fail to do so.

Finally, also, the percentage of structures was determined where the RMSD after the optimization is lower than the RMSD before the optimization; i.e., the optimization drives the structure in the direction of the minimum (Figure 14d). For higher initial RMSD values (and more and more extensive clashes), the percentage of structures with lower RMSD after optimization drops quickly for the BFGS implementation but remains high for the LSL-BFGS algorithm and the BOBYQA algorithm. Interestingly, the percentage of structures with a reduced RMSD after optimization for the *start* bin (re-execution of the optimization algorithm) in Figure 14d is lower than that for the 0.001 Å or larger bins; meanwhile, especially for the gradient-based methods, one would expect a 100% reproduction of these poses. This can be attributed to the implementation of the termination criteria: The LSLOpt library terminates if the change in the function value is 0.0 for two successive iterations. If a new optimization run is started, then a few iterations may be performed before the algorithm terminates again. This may result in a very slight change of the pose (remember that in this data point, anything shows up that does not have an RMSD of *exactly* 0.0). However, judging by Figure 14a, these changes are less than 0.001 Å in terms of RMSD.

For the explicit analysis of the locality of the optimization algorithms, we considered the RMSD between the starting point and the end point of the optimization. Figure 15a–d shows the percentage of structures with an RMSD of the end point of the optimization to the starting point of the optimization up to 2 Å, 3 Å, 5 Å, and 10 Å, respectively. While the LSL-BFSG and the BOBYQA algorithm both show a high locality, i.e., they do not move the structure excessively during optimization, this is not true for our BFGS implementation and the L-BFGS implementation in NLopt. Especially, Figure 15c,d shows that these algorithms move clashing poses virtually outside of the binding pocket. Interestingly, the difference is even more prominent comparing our BFGS implementation to the L-BFGS implementation in the NLopt library.[62,63] This is probably due to the use of a line search that allows for even larger step lengths (similar to the line search of Moré and Thuente[87]).

These results indicate that the LSL-BFGS algorithm has exactly the intended effect that it preserves the locality of the solutions by considering additional domain knowledge (how far the atoms in the molecule are allowed to move in each step). Furthermore, the JAMDA scoring function shows a high stability; i.e., small changes in the pose can be reoptimized into the same (or a very close) local minimum with all tested optimization algorithms.

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

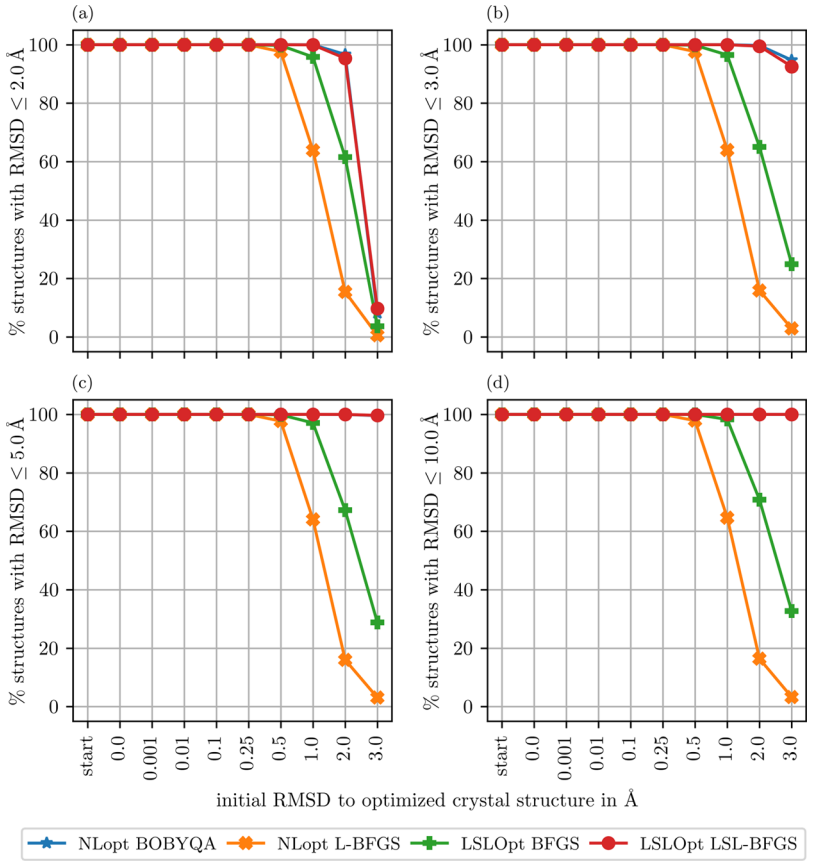


Figure 15. The percentages of structures with RMSDs to the start point of the optimization of ≤2 Å (a), ≤3 Å (b), ≤5 Å (c), or ≤10 Å (d) are shown. Optimization was performed w.r.t. JAMDA scoring. The results are given for four different optimization algorithms. The percentage is shown as a function of the binned initial RMSD of the structure. The difference between *start* and 0.0 is that the former contains only unmodified initial structures and the latter contains modified structures with a heavy-atom RMSD of 0.0 (e.g., with rotated H-bond donors).

We also investigated the success rate, i.e., how often the algorithm terminates successfully or how often the algorithm terminates with an error status. The results are given in Table 3. For the LSLOpt algorithms, all failures are due to the line search being unable to find a step length α that is larger than the given parameter α_{min}. This does not necessarily imply a total failure; for both algorithms from our LSLOpt library, the RMSD to the optimized crystal structure had already been reduced to values $<10^{-5}$ Å; however, optimization runs with irregular termination should naturally be taken with care. For the NLopt algorithms, such a detailed analysis cannot be performed because information on the error status is not propagated in detail.

Table 3. Number and Percentage of Failures for the Different Optimization Algorithms

algorithm	failures	failure percentage
NLopt BOBYQA	0	0%
NLopt L-BFGS	693	0.22%
LSLOpt BFGS	96	0.03%
LSLOpt LSL-BFGS	101	0.03%

Experiment 6: How Many Scoring Function Evaluations Are Necessary for Different Optimization Algorithms? Comparing the run time of different optimization algorithms with each other is notoriously hard. An excellent overview of problems and strategies for comparing optimization algorithms can be found by Beiranvand et al.[88] Here, we compared the number of function and gradient evaluations of our LSL-BFGS algorithm to the other optimization algorithms. This is an important evaluation because limiting the step length might have an unfavorable impact on the convergence rate and therefore ultimately on computational efficiency.

Besides the number of function/gradient evaluations, also, the run time of the optimization step was measured. Run times were obtained on a PC with an Intel Core i5-9500 CPU (3 GHz) and with 16 GB of RAM.

We employ two different strategies for comparing the number of function/gradient evaluations of different algorithms, one for comparing algorithms implemented in different libraries and another one for comparing algorithms from the same library.

Comparing Different Implementations. Since different optimization algorithms have different termination criteria, a direct comparison is usually not possible. Here, we used a variant of the so called *fixed-target* method[88,89] requiring that the optimization algorithms are run until the optimal solution is found to a given precision, e.g., until the objective function value $f_{stop} = f^* + \varepsilon$ has been reached. Obviously, in our benchmark setting, the optimal solution is unknown. In this case, the optimal function value f^* can be replaced by the best known function value f_{min}. This approach was suggested by different authors.[88,90]

However, here, also the roughness of the energy landscape has to be taken into account. As can be seen in Figure 14, for

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

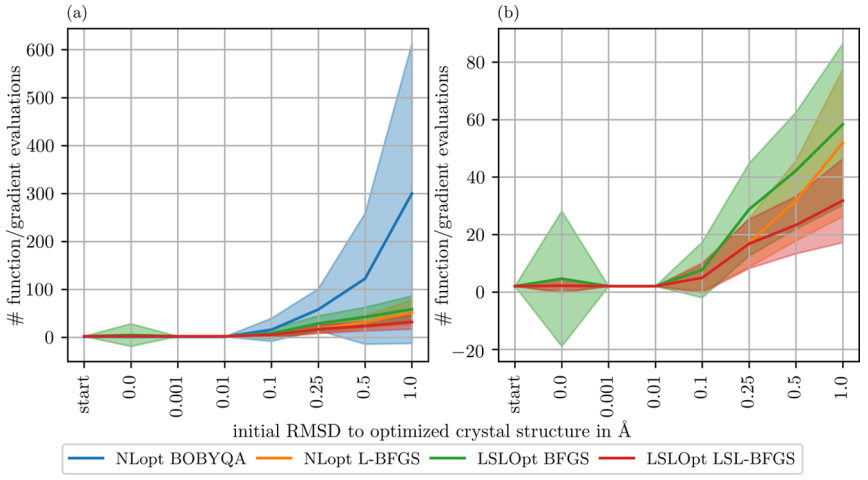


Figure 16. In panel (a), the mean number of function/gradient evaluations with standard deviation for the different optimization algorithms is shown. In panel (b), the same results are shown as in panel (a), but the BOBYQA was removed from the plot to make the difference between the BFGS-type algorithms clearer. Optimization was performed until the stop value f_{stop} was reached.

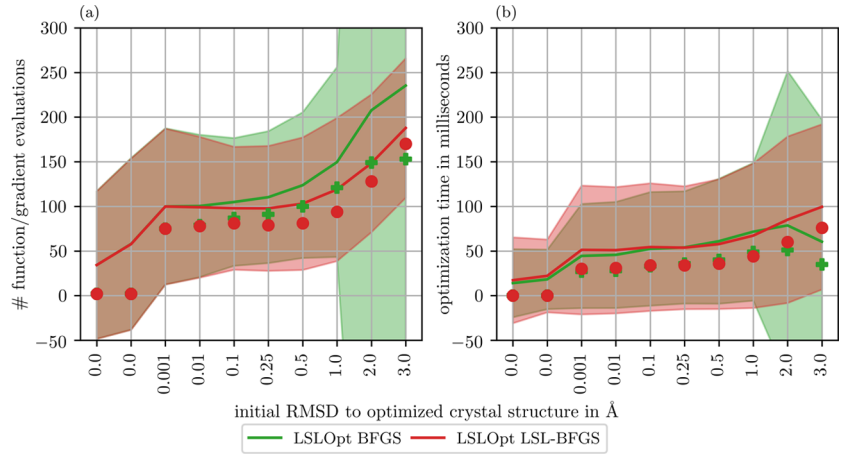


Figure 17. In panel (a), the mean number of function/gradient evaluations with standard deviation for the different optimization algorithms is shown. The markers denote the median values. In panel (b), the run time instead of the number of function/gradient evaluations is shown. Optimization was performed until convergence (see the Supporting Information Section S3.2).

initial RMSD values up to 1.0 Å, in most cases, a structure very close to the initially optimized structure can be found after optimization. Within this range, the score of the initially optimized structure is probably a good estimate for f_{min}. Outside of this range, however, the scoring function might also be optimized into completely different local minima (and rightfully so). We used the stop value $f_{stop} = f_{min} + 0.01$ for the optimization, which is a similar choice to Rios and Sahinidis[89] and corresponds roughly to a difference of 1% for typical JAMDA score values.

Other termination criteria were disabled as far as possible, except for a maximum number of 10,000 function/gradient evaluations and a small gradient tolerance. The latter is necessary to avoid numerical problems for the BFGS-type algorithms. Note that not all stopping criteria can be disabled. This was also recognized by Beiranvand et al.[88] One example is the L-BFGS implementation in the NLopt library where the value for the gradient tolerance stopping criterion is hard-coded. This problem cannot be circumvented but has to be kept in mind when interpreting the results.

As already discussed above, for a rough energy landscape, this leads to nonsuccessful runs; i.e., there are cases where no local minimum is found that has a score $\leq f_{stop}$. In these cases, the optimization terminates due to any other active termination criterion. For comparison, the number of function and gradient evaluations was only recorded if f_{stop} has been reached. Figure S18 shows the success rates, i.e., the percentage of optimization runs where f_{stop} is reached. These results are consistent with our discussions above that only for initial RMSD values up to 1 Å the iteration number can be compared in this experiment.

The number of function/gradient evaluations until reaching f_{stop} is shown in Figure 16. As expected, the number of function evaluations is largest for the derivative-free BOBYQA. There is less difference between the BFGS-type algorithms with the LSL-BFGS algorithm actually needing the least number of function/gradient evaluations. This indicates that limiting the step lengths does not increase the number of function/gradient evaluations but in contrast can help the algorithm to converge. The optimization run times shown in Figure S19 give a very similar picture. However, the difference between the gradient-free BOBYQA and the gradient-based BFGS algorithms is smaller. This is probably due to the slightly larger computational cost for evaluating the gradient than the function value.

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

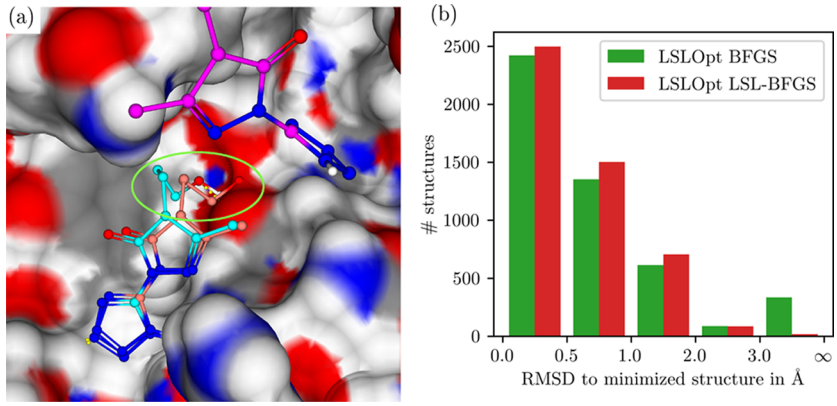


Figure 18. (a) The introduction of a hydroxymethyl group to the ligand 4-ethyl-5-methyl-2-(1*H*-tetrazol-5-yl)-1,2-dihydro-3*H*-pyrazol-3-one of AmpC beta-lactamase in PDB structure 3GR2[93] is shown as an example (see Figure S22 for a 2D depiction). In salmon, the modified structure is shown that clashes with the protein (marked with a green ellipse, JAMDA score 23.28), in blue, the LSL-BFGS-optimized structure is shown (RMSD 0.70 Å, JAMDA score −1.12), and in pink, the BFGS-optimized structure is shown (RMSD 14.25 Å, JAMDA score 0.08). Hydrogen bonds are indicated as yellow dashed lines. (b) Results from all binding sites and ligand modifications are combined into one histogram. The RMSDs of the structures with a hydroxymethyl substituent and after optimization to the optimized crystal structure are shown for the BFGS and the LSL-BFGS algorithm (both from our LSLOpt library).

Note that neither the number of function/gradient evaluations nor the run times in this experiment are necessarily representative for an optimization in practice because they only measure the time needed until a certain function value is reached (see experiment design above). In practice, these metrics heavily depend on the parameterization of the algorithm, e.g., which accuracy is requested. For our different BFGS implementations, this information can be obtained in the next section.

Comparing Different Algorithms from the Same Library. The approach described above does not make much sense for larger initial deflections as the recovered local minima differ considerably. However, for the LSL-BFGS and the BFGS algorithm from the LSLOpt library, the termination criteria are exactly the same. Therefore, a direct comparison of the number of steps until convergence can be performed. The number of function/gradient evaluations and the run time are shown in Figure 17. The difference from the experiment above is subtle but important: Instead of measuring the time until the function value gets sufficiently close to a minimum, here, the time to achieve convergence is obtained.

The mean number of function/gradient evaluations for the LSL-BFGS algorithm is always smaller or equal to the BFGS algorithm. The median number of function/gradient evaluations shows a similar behavior. Interestingly, the standard deviation of the number of function/gradient evaluations for the BFGS algorithm for high initial deflections is very large, and the run time median/mean values actually drop for the last 3 Å bin. Analyzing the values in detail shows that there are optimization runs with an excessively low or high number of function evaluations. In contrast, this never happens with the LSL-BFGS algorithm, making it very likely that this issue is related to the problem of excessive movements. There is almost no difference in the mean and median run times between the BFGS and the LSL-BFGS algorithm except for the last bin (Figure 17b).

In summary, no significant additional run time cost comes from using the LSL-BFGS algorithm (compared to the BFGS algorithm), making it possible to optimize a protein−ligand pose on average in less than 100 ms. This number, however, still depends on the chosen parameters of the optimization algorithm (see the Supporting Information Section S3.2); i.e., by requesting less accurate solutions, the process could be sped up.

Additionally, we tested different values for the limitation of the step lengths, i.e., instead of 0.5 Å, only 0.05 Å atom movement was allowed per step. The results are shown in Figure S21. For this smaller step length limitation, a significantly higher number of function/gradient evaluations is needed until reaching convergence if the optimization starts far away from the minimum. Therefore, if the value for the maximum allowed movement of atoms is chosen badly, the run time performance of the LSL-BFGS algorithm can be seriously hampered. This is hardly surprising, but it stresses the point that the step length limitations must generally be chosen with care.

Experiment 7: JAMDA and LSL-BFGS for Chemical Elaboration. The main use case of the LSL-BFGS algorithm is the optimization of potentially clashing poses (without resolving the clashes with another algorithm before). Here, we describe a practical use case for the LSL-BFGS algorithm that also serves as another evaluation scenario. The LSL-BFGS algorithm can be used together with the JAMDA scoring function for a chemical elaboration of a ligand within a protein binding pocket (e.g., introduction of a "magic methyl"[91]). Obviously, this approach is not suitable to model binding mode changes (see, e.g., Malhotra and Karanicolas[92]) as these would require global optimization (e.g., by docking).

We used the same dataset as in the experiments before (the CASF-2016 *core set*). In this experiment, we optimized the crystal structure of the ligand within the binding pocket with our LSL-BFGS algorithm and the JAMDA scoring function. Then we subsequently exchanged every carbon-bound hydrogen atom with a methyl, hydroxymethyl, or iodine substituent. An example is given in Figure S22. Note that all changes result in valid molecules with respect to our chemistry model;[72,73] however, molecules may be unstable or not synthetically accessible. These molecules were again optimized with the LSL-BFGS or BFGS algorithm, and the RMSD between the optimized crystal structure and the modified and optimized molecules was calculated; here, all heavy-atoms except for the

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

new substituent were considered. Furthermore, the change in the JAMDA score value was analyzed.

Results for the hydroxymethyl substituent are given in Figure 18. In Figure 18a, we show an example for the modification and optimization the ligand of PDB structure 3GR2.[93] Here, the effect of excessive movement by the BFGS algorithm − but not our LSL-BFGS − is evident. Figure 18b shows the results for the complete dataset (all binding sites) with a histogram over the RMSDs to the minimized crystal structure. For 6.9% of the structures, the BFGS algorithm moves the pose for more than 3 Å RMSD, and the LSL-BFGS algorithm does this for only 0.4% of the structures. The results for methyl and iodine substituents are given in Figures S24a and S25a. Introducing a methyl only results in 1.6% (BFGS) and 0.1% (LSL-BFGS) structures with an RMSD >3 Å; for an iodine substituent, the fraction increases to 7.1% (BFGS) and 1.1% (LSL-BFGS).

Besides the RMSD values, we also compared the JAMDA scores obtained after optimization with the BFGS algorithm and the LSL-BFGS algorithm. For this purpose, we analyzed the difference between the scores after optimization with the BFGS algorithm and the LSL-BFGS algorithm. For the hydroxymethyl substituent (Figure S23), the absolute difference in scores is at most 0.01 for 77.2% of the cases. The difference of JAMDA scores is less than −0.01 (the score after BFGS optimization is lower than that after LSL-BFGS optimization) in 9.1% of the cases and larger than 0.01 (the score after LSL-BFGS optimization is lower than that after BFGS optimization) in 13.7% of the cases. The mean score difference is 0.08. These results indicate that the score values after optimization with the BFGS algorithm and the LSL-BFGS algorithm are most often almost identical and otherwise tend to be lower for the LSL-BFGS algorithm more often. The results for the experiments with the other substituents show a similar trend (Figures S24b and S25).

CONCLUSIONS

We have presented a consistent scheme for the scoring and numerical optimization of protein−ligand binding poses. Integral parts are the LSL-BFGS optimization algorithm and the JAMDA scoring function. In combination, they allow stable numerical optimization with an efficient gradient-based optimizer that also prevents excessive ligand movements.

The LSL-BFGS algorithm allows for the inclusion of domain knowledge and thus avoids unreasonably large steps upon optimization. To the best of our knowledge, this is the first in-depth analysis and the first proposed algorithm for a generic, dynamic, and robust handling of step length restrictions in the BFGS algorithm. It has already been used in two different software tools developed by our group, namely, in Conformator[40] for the generation of macrocycle coordinates and in WarPP[49] for the post-optimization of placed water molecules.

The JAMDA scoring function is a novel empirical scoring function that has been built from easily optimizable individual terms, and it shows excellent pose prediction performance in the CASF-2016 docking power benchmark. Precise geometric hydrogen bond terms backed by PDB statistics[42] could be integrated in JAMDA. Furthermore, our newly developed piecewise polynomial functions, building the foundation of the JAMDA scoring function, allow a flexible reparameterization of the JAMDA scoring function for different use cases (e.g., soft docking). It should be noted that evaluations of the JAMDA scoring function are currently limited to the self-docking case (the CASF-2016 docking power benchmark). Further evaluations for the cross-docking scenario would be interesting; however, these require an appropriate benchmark dataset including results for other scoring functions.

In our experiments, we could show that the LSL-BFGS algorithm together with the JAMDA scoring function has exactly the intended effect of increasing the locality of solutions found during the numerical optimization. This makes any filtering of clashing poses or the use of different algorithms for clash resolving unnecessary because the LSL-BFGS algorithm can handle these gracefully. The JAMDA scoring function together with gradient-based optimization algorithms also shows excellent stability, recovering the local optimum (or a close-by optimum) in about 90% of the cases up to an initial perturbation of 0.5 Å RMSD.

An interesting advancement of the JAMDA scoring function would be the use of machine learning methods instead of a simple linear combination of terms while preserving the good optimizability of the scoring function. Machine learning methods for scoring have gained much interest over the last years.[9,94] However, while some of them were also evaluated for the pose prediction scenario,[15,95−98] to the best of our knowledge, none of them were actually used for pose generation or optimization in docking. Instead, poses generated by established docking tools were rescored. A notable exception is a convolutional neural network described by Ragoza et al. that was used for pose optimization.[99]

With a precise optimizer for molecular complexes in hand, further difficulties of comparing the performance of different scoring functions were revealed: The significant decrease of the pose prediction performance in the CASF-2016 benchmark of our JAMDA scoring function but also of the established AutoDock Vina scoring function with local optimization advises to take the results of all the other scoring functions with caution, too. Without local optimization, scoring functions capture small clashes in decoy structures overestimating the docking power in the benchmark. We expect this problem to be even more pronounced in case that machine learning is applied on docking poses and − especially if features are automatically extracted (e.g., in deep learning) − discriminative features can be obscured allowing bias in a dataset to go unnoticed (see a recent paper from Sieg et al.[100] for a detailed discussion of this matter).

On the other hand, we cannot arrive at a clear conclusion how this problem could be prevented in a dataset such as the CASF-2016 docking power benchmark. It is probably not possible for some of the scoring functions under investigation in the CASF-2016 benchmark to perform local optimization before scoring, and for the others, the influence of the optimization algorithm or its parameterization would have to be taken into account. One possibility would be a further refinement of the decoy poses in the dataset to remove conspicuous clashes and therefore diminish this bias − obviously at the risk of introducing other biases (e.g., hydrogen bond geometry) along the way. Nevertheless, it would be a very interesting experiment to establish a benchmarking pipeline that covers different aspects of scoring functions and their numerical optimization behavior (e.g., the experiments 1−6) and conduct these experiments not only for a variety of classical scoring functions but also different force fields − provided that numerical optimization is possible.

It is unclear if there are further implications of our observed pose prediction performance reduction due to numerical

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

optimization. The results suggest that numerical optimization of *all* poses should be avoided if there already is a way to reliably discriminate between decoy and near-native poses. However, we would like to emphasize that in the case of the CASF-2016 docking power benchmark, this discriminating feature is artifactual (more clashes in decoy poses). More in-depth investigations are advised before arriving at unambiguous conclusions.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jcim.0c01095.

Description of the atom classes of the JAMDA scoring function; applicability of scoring contributions to atom classes; analytical expressions for the polynomial coefficients of the basic functions *LJ-like*, *repulsive*, *plateau2*, and *plateau3*; a plot of the restriction term for conjugated torsion bonds; detailed description of rectangular hydrogen bond geometries; derivatives of the JAMDA scoring function; additional results for the calibration of the JAMDA scoring function. Description of the LSL-BFGS algorithm and the step length limitation function. Parameters used in the experiments and additional plots for the experiments. PDB codes in the calibration data set for the JAMDA scoring function (PDF)

■ AUTHOR INFORMATION

Corresponding Author

Matthias Rarey − *ZBH - Center for Bioinformatics, Universität Hamburg, 20146 Hamburg, Germany;* orcid.org/0000-0002-9553-6531; Email: rarey@zbh.uni-hamburg.de

Authors

Florian Flachsenberg − *ZBH - Center for Bioinformatics, Universität Hamburg, 20146 Hamburg, Germany;* orcid.org/0000-0001-7051-8719

Agnes Meyder − *ZBH - Center for Bioinformatics, Universität Hamburg, 20146 Hamburg, Germany;* orcid.org/0000-0001-8519-5780

Kai Sommer − *ZBH - Center for Bioinformatics, Universität Hamburg, 20146 Hamburg, Germany;* orcid.org/0000-0003-1866-8247

Patrick Penner − *ZBH - Center for Bioinformatics, Universität Hamburg, 20146 Hamburg, Germany;* orcid.org/0000-0003-4988-6183

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jcim.0c01095

Notes

The authors declare the following competing financial interest(s): M.R. declares a potential financial interest in the event that the JamdaScorer software is licensed for a fee to non-academic institutions in the future.

The command-line tool JamdaScorer for scoring and optimization of protein−ligand complexes with the JAMDA scoring function and the LSL-BFGS algorithm is freely available for noncommercial use and academic research within the NAOMI ChemBio Suite from https://uhh.de/naomi. We have implemented the LSL-BFGS algorithm together with limited step length variants of the L-BFGS, BFGS-B, and L-BFGS-B algorithms in the new LSLOpt C++ library that is used in this paper. We will describe the more technical aspects of the LSLOpt library in a later publication and release it as open-source software at that time.

■ ACKNOWLEDGMENTS

The authors thank the current and former members of the AMD group and the developers at the BioSolveIT for their contributions to the development of the NAOMI library. The authors thank Rainer Fährrolfes and Emanuel S. R. Ehmki for valuable discussions and Louis Bellmann and Robert Klein (Bayer CropScience) for careful reading of the manuscript.

■ REFERENCES

(1) Huang, S.-Y.; Grinter, S. Z.; Zou, X. Scoring functions and their evaluation methods for protein−ligand docking: recent advances and future directions. *Phys. Chem. Chem. Phys.* **2010**, *12*, 12899−12908.

(2) Kolodzik, A.; Schneider, N.; Rarey, M. In *Applied Chemoinformatics*; Engel, T., Gasteiger, J., Eds.; John Wiley & Sons, Ltd: Weinheim, 2018; Chapter 6.8, pp. 313−331.

(3) Korb, O.; Stützle, T.; Exner, T. E. An ant colony optimization approach to flexible protein−ligand docking. *Swarm Intell.* **2007**, *1*, 115−134.

(4) Korb, O.; Stützle, T.; Exner, T. E. Empirical Scoring Functions for Advanced Protein−Ligand Docking with PLANTS. *J. Chem. Inf. Model.* **2009**, *49*, 84−96.

(5) Trott, O.; Olson, A. J. AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J. Comput. Chem.* **2010**, *31*, 455−461.

(6) Rarey, M.; Kramer, B.; Lengauer, T.; Klebe, G. A Fast Flexible Docking Method using an Incremental Construction Algorithm. *J. Mol. Biol.* **1996**, *261*, 470−489.

(7) Friesner, R. A.; Banks, J. L.; Murphy, R. B.; Halgren, T. A.; Klicic, J. J.; Mainz, D. T.; Repasky, M. P.; Knoll, E. H.; Shelley, M.; Perry, J. K.; Shaw, D. E.; Francis, P.; Shenkin, P. S. Glide: A New Approach for Rapid, Accurate Docking and Scoring. 1. Method and Assessment of Docking Accuracy. *J. Med. Chem.* **2004**, *47*, 1739−1749.

(8) Su, M.; Yang, Q.; Du, Y.; Feng, G.; Liu, Z.; Li, Y.; Wang, R. Comparative Assessment of Scoring Functions: The CASF-2016 Update. *J. Chem. Inf. Model.* **2019**, *59*, 895−913.

(9) Shen, C.; Ding, J.; Wang, Z.; Cao, D.; Ding, X.; Hou, T. From machine learning to deep learning: Advances in scoring functions for protein−ligand docking. *WIREs Comput. Mol. Sci.* **2020**, *10*, e1429.

(10) Spitzmüller, A.; Velec, H. F. G.; Klebe, G. MiniMuDS: A New Optimizer using Knowledge-Based Potentials Improves Scoring of Docking Solutions. *J. Chem. Inf. Model.* **2011**, *51*, 1423−1430.

(11) Meng, E. C.; Shoichet, B. K.; Kuntz, I. D. Automated Docking with Grid-Based Energy Evaluation. *J. Comput. Chem.* **1992**, *13*, 505−524.

(12) Allen, W. J.; Balius, T. E.; Mukherjee, S.; Brozell, S. R.; Moustakas, D. T.; Lang, P. T.; Case, D. A.; Kuntz, I. D.; Rizzo, R. C. DOCK 6: Impact of New Features and Current Docking Performance. *J. Comput. Chem.* **2015**, *36*, 1132−1156.

(13) Muegge, I.; Martin, Y. C. A General and Fast Scoring Function for Protein−Ligand Interactions: A Simplified Potential Approach. *J. Med. Chem.* **1999**, *42*, 791−804.

(14) Böhm, H.-J. The development of a simple empirical scoring function to estimate the binding constant for a protein-ligand complex of known three-dimensional structure. *J. Comput.-Aided Mol. Des.* **1994**, *8*, 243−256.

(15) Wang, C.; Zhang, Y. Improving Scoring-Docking-Screening Powers of Protein−Ligand Scoring Functions using Random Forest. *J. Comput. Chem.* **2017**, *38*, 169−177.

(16) Li, J.; Fu, A.; Zhang, L. An Overview of Scoring Functions Used for Protein−Ligand Interactions in Molecular Docking. *Interdiscip. Sci.: Comput. Life Sci.* **2019**, *11*, 320−328.
(17) Cole, J. C.; Murray, C. W.; Nissink, J. W.; Taylor, R. D.; Taylor, R. Comparing Protein−Ligand Docking Programs Is Difficult. *Proteins* **2005**, *60*, 325−332.
(18) O'Boyle, N. M.; Liebeschuetz, J. W.; Cole, J. C. Testing Assumptions and Hypotheses for Rescoring Success in Protein−Ligand Docking. *J. Chem. Inf. Model.* **2009**, *49*, 1871−1878.
(19) Abagyan, R.; Totrov, M.; Kuznetsov, D. ICM − A New Method for Protein Modeling and Design: Applications to Docking and Structure Prediction from the Distorted Native Conformation. *J. Comput. Chem.* **1994**, *15*, 488−506.
(20) Fuhrmann, J.; Rurainski, A.; Lenhof, H.-P.; Neumann, D. A New Method for the Gradient-Based Optimization of Molecular Complexes. *J. Comput. Chem.* **2009**, *30*, 1371−1378.
(21) Mirzaei, H.; Zarbafian, S.; Villar, E.; Mottarella, S.; Beglov, D.; Vajda, S.; Paschalidis, I. C.; Vakili, P.; Kozakov, D. Energy Minimization on Manifolds for Docking Flexible Molecules. *J. Chem. Theory Comput.* **2015**, *11*, 1063−1076.
(22) Eberhart, R.; Kennedy, J. *A New Optimizer Using Particle Swarm Theory. MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*; 1995, pp. 39−43.
(23) Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671−680.
(24) Hoffmann, P. H. W. A Hitchhiker's Guide to Automatic Differentiation. *Numer. Algorithms* **2016**, *72*, 775−811.
(25) Gehlhaar, D. K.; Verkhivker, G. M.; Rejto, P. A.; Sherman, C. J.; Fogel, D. R.; Fogel, L. J.; Freer, S. T. Molecular recognition of the inhibitor AG-1343 by HIV-1 protease: conformationally flexible docking by evolutionary programming. *Chem. Biol.* **1995**, *2*, 317−324.
(26) Schneider, N.; Lange, G.; Hindle, S.; Klein, R.; Rarey, M. A consistent description of HYdrogen bond and DEhydration energies in protein−ligand complexes: Methods behind the HYDE scoring function. *J. Comput.-Aided Mol. Des.* **2013**, *27*, 15−29.
(27) Broyden, C. G. The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. *J. Inst. Math. Its Appl.* **1970**, *6*, 76−90.
(28) Broyden, C. G. The Convergence of a Class of Double-rank Minimization Algorithms: 2. The New Algorithm. *J. Inst. Math. Its Appl.* **1970**, *6*, 222−231.
(29) Fletcher, R. A new approach to variable metric algorithms. *Comput. J.* **1970**, *13*, 317−322.
(30) Goldfarb, D. A Family of Variable-Metric Methods Derived by Variational Means. *Math. Comput.* **1970**, *24*, 23−26.
(31) Shanno, D. F. Conditioning of Quasi-Newton Methods for Function Minimization. *Math. Comput.* **1970**, *24*, 647−656.
(32) Eldridge, M. D.; Murray, C. W.; Auton, T. R.; Paolini, G. V.; Mee, R. P. Empirical scoring functions: I. The development of a fast empirical scoring function to estimate the binding affinity of ligands in receptor complexes. *J. Comput.-Aided Mol. Des.* **1997**, *11*, 425−445.
(33) Baxter, C. A.; Murray, C. W.; Clark, D. E.; Westhead, D. R.; Eldridge, M. D. Flexible Docking Using Tabu Search and an Empirical Estimate of Binding Affinity. *Proteins: Struct., Funct., Bioinf.* **1998**, *33*, 367−382.
(34) Bentley, J. L. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* **1975**, *18*, 509−517.
(35) Blanco, J. L.; Rai, P. K. *nanoflann: a C++ header-only fork of FLANN, a library for Nearest Neighbor (NN) with KD-trees*; 2014 https://github.com/jlblancoc/nanoflann.
(36) *Mathematica*; Version 11.0, Wolfram Research, Inc.: Champaign, IL, 2016.
(37) Meurer, A.; Smith, C. P.; Paprocki, M.; Čertík, O.; Kirpichev, S. B.; Rocklin, M.; Kumar, A.; Ivanov, S.; Moore, J. K.; Singh, S.; Rathnayake, T.; Vig, S.; Granger, B. E.; Muller, R. P.; Bonazzi, F.; Gupta, H.; Vats, S.; Johansson, F.; Pedregosa, F.; Curry, M. J.; Terrel, A. R.; Roučka, Š.; Saboo, A.; Fernando, I.; Kulal, S.; Cimrman, R.; Scopatz, A. SymPy: symbolic computing in Python. *PeerJ Comput. Sci.* **2017**, *3*, No. e103.
(38) Jiang, F.; Kim, S.-H. "Soft docking": Matching of Molecular Surface Cubes. *J. Mol. Biol.* **1991**, *219*, 79−102.
(39) Ferrari, A. M.; Wei, B. Q.; Costantino, L.; Shoichet, B. K. Soft Docking and Multiple Receptor Conformations in Virtual Screening. *J. Med. Chem.* **2004**, *47*, 5076−5084.
(40) Friedrich, N.-O.; Flachsenberg, F.; Meyder, A.; Sommer, K.; Kirchmair, J.; Rarey, M. Conformator: A Novel Method for the Generation of Conformer Ensembles. *J. Chem. Inf. Model.* **2019**, *59*, 731−742.
(41) Brooks, B. R.; Bruccoleri, R. E.; Olafson, B. D.; States, D. J.; Swaminathan, S.; Karplus, M. CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations. *J. Comput. Chem.* **1983**, *4*, 187−217.
(42) Nittinger, E.; Inhester, T.; Bietz, S.; Meyder, A.; Schomburg, K. T.; Lange, G.; Klein, R.; Rarey, M. Large-Scale Analysis of Hydrogen Bond Interaction Patterns in Protein-Ligand Interfaces. *J. Med. Chem.* **2017**, *60*, 4245−4257.
(43) Böhm, H.-J. The computer program LUDI: A new method for the de novo design of enzyme inhibitors. *J. Comput.-Aided Mol. Des.* **1992**, *6*, 61−78.
(44) Böhm, H.-J. LUDI: rule-based automatic design of new substituents for enzyme inhibitor leads. *J. Comput.-Aided Mol. Des.* **1992**, *6*, 593−606.
(45) Klebe, G. The Use of Composite Crystal-field Environments in Molecular Recognition and the de Novo Design of Protein Ligands. *J. Mol. Biol.* **1994**, *237*, 212−235.
(46) Schärfer, C.; Schulz-Gasch, T.; Hert, J.; Heinzerling, L.; Schulz, B.; Inhester, T.; Stahl, M.; Rarey, M. CONFECT: Conformations from an Expert Collection of Torsion Patterns. *ChemMedChem* **2013**, *8*, 1690−1700.
(47) Guba, W.; Meyder, A.; Rarey, M.; Hert, J. Torsion Library Reloaded: A New Version of Expert-Derived SMARTS Rules for Assessing Conformations of Small Molecules. *J. Chem. Inf. Model.* **2016**, *56*, 1−5.
(48) Daylight Chemical Information Systems, Inc., *Daylight Theory: SMARTS - A Language for Describing Molecular Patterns*. https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html.
(49) Nittinger, E.; Flachsenberg, F.; Bietz, S.; Lange, G.; Klein, R.; Rarey, M. Placement of Water Molecules in Protein Structures: From Large-Scale Evaluations to Single-Case Examples. *J. Chem. Inf. Model.* **2018**, *58*, 1625−1637.
(50) Qiu, D.; Shenkin, P. S.; Hollinger, F. P.; Still, W. C. The GB/SA Continuum Model for Solvation. A Fast Analytical Method for the Calculation of Approximate Born Radii. *J. Phys. Chem. A* **1997**, *101*, 3005−3014.
(51) Schlosser, J.; Rarey, M. Beyond the Virtual Screening Paradigm: Structure-Based Searching for New Lead Compounds. *J. Chem. Inf. Model.* **2009**, *49*, 800−809.
(52) Henzler, A. M.; Urbaczek, S.; Hilbig, M.; Rarey, M. An integrated approach to knowledge-driven structure-based virtual screening. *J. Comput.-Aided Mol. Des.* **2014**, *28*, 927−939.
(53) Grassia, F. S. Practical Parameterization of Rotations Using the Exponential Map. *J. Graph. Tools* **1998**, *3*, 29−48.
(54) Mirzaei, H.; Beglov, D.; Paschalidis, I. C.; Vajda, S.; Vakili, P.; Kozakov, D. Rigid Body Energy Minimization on Manifolds for Molecular Docking. *J. Chem. Theory Comput.* **2012**, *8*, 4374−4380.
(55) Abe, H.; Braun, W.; Noguti, T.; Gō, N. Rapid calculation of first and second derivatives of conformational energy with respect to dihedral angles for proteins general recurrent equations. *Comput. Chem.* **1984**, *8*, 239−247.
(56) Nocedal, J. Updating Quasi-Newton Matrices With Limited Storage. *Math. Comput.* **1980**, *35*, 773−782.
(57) Liu, D. C.; Nocedal, J. On the limited memory BFGS method for large scale optimization. *Mathematical Program.* **1989**, *45*, 503−528.
(58) Byrd, R. H.; Lu, P.; Nocedal, J.; Zhu, C. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM J. Sci. Comput.* **1995**, *16*, 1190−1208.

https://dx.doi.org/10.1021/acs.jcim.0c01095
J. Chem. Inf. Model. 2020, 60, 6502−6522

(59) Zhu, C.; Byrd, R. H.; Lu, P.; Nocedal, J. Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization. *ACM Trans. Math. Software* **1997**, *23*, 550−560.
(60) Nocedal, J.; Wright, S. J. *Numerical Optimization. Springer Series in Operations Research and Financial Engineering*; 2nd ed.; Springer: New York, NY, 2006; Chapter 6, pp. 136−143.
(61) Allen, M. P.; Tildesley, D. J. *Computer Simulation of Liquids*; Clarendon Press: Oxford, 1989; Chapter 5, pp. 145−146.
(62) Johnson, S. G. *The Nlopt nonlinear-optimization package, Version 2.5.0*; http://github.com/stevengj/nlopt.
(63) Lukšan, L.; Matonoha, C.; Vlček, J. Algorithm 896: LSA: Algorithms for large-scale optimization. *ACM Trans. Math. Software* **2009**, *36*, 16:1−16:29.
(64) Nocedal, J.; Wright, S. J. *Numerical Optimization. Springer Series in Operations Research and Financial Engineering*; 2nd ed.; Springer: New York, NY, 2006. Chapter 17, pp. 497−528.
(65) Niketić, S. R.; Rasmussen, K. *The Consistent Force Field: A Documentation. Lecture Notes in Chemistry, vol 3*; Springer: Berlin, Heidelberg, 1977, Chapter 5, pp. 145−148.
(66) Lindahl, E.; Abraham, M. J.; Hess, B.; van der Spoel, D. *GROMACS 2019 Manual*, 2018; pp. 334−335 DOI: 10.5281/zenodo.2424486.
(67) Fletcher, R.; Reeves, C. M. Function minimization by conjugate gradients. *Comput. J.* **1964**, *7*, 149−154.
(68) Nocedal, J.; Wright, S. J. *Numerical Optimization. Springer Series in Operations Research and Financial Engineering*, 2nd ed.; Springer: New York, 2006, Chapter 3, pp. 59−60.
(69) Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; Flannery, B. P. *Numerical Recipes: The Art of Scientific Computing*; 3rd ed.; Cambridge University Press: Cambridge, New York, 2007, Chapter 9, pp. 478−480, 523−524.
(70) Hratchian, H. P.; Schlegel, H. B. In *Theory and Applications of Computational Chemistry: The First Forty Years*; Dykstra, C. E., Frenking, G., Kim, K. S., Scuseria, G. E., Eds.; Elsevier: Amsterdam, 2005, Chapter 10, pp. 195−249.
(71) Al-Baali, M.; Grandinetti, L.; Pisacane, O. Damped Techniques for the Limited Memory BFGS Method for Large-Scale Optimization. *J. Optim. Theory Appl.* **2014**, *161*, 688−699.
(72) Urbaczek, S.; Kolodzik, A.; Fischer, J. R.; Lippert, T.; Heuser, S.; Groth, I.; Schulz-Gasch, T.; Rarey, M. NAOMI: On the Almost Trivial Task of Reading Molecules from Different File formats. *J. Chem. Inf. Model.* **2011**, *51*, 3199−3207.
(73) Urbaczek, S.; Kolodzik, A.; Groth, I.; Heuser, S.; Rarey, M. Reading PDB: Perception of Molecules from 3D Atomic Coordinates. *J. Chem. Inf. Model.* **2013**, *53*, 76−87.
(74) Liu, Z.; Su, M.; Han, L.; Liu, J.; Yang, Q.; Li, Y.; Wang, R. Forging the Basis for Developing Protein−Ligand Interaction Scoring Functions. *Acc. Chem. Res.* **2017**, *50*, 302−309.
(75) Hilbig, M.; Urbaczek, S.; Groth, I.; Heuser, S.; Rarey, M. MONA − Interactive manipulation of molecule collections. *J. Cheminformatics* **2013**, *5*, 38.
(76) Hilbig, M.; Rarey, M. MONA 2: A Light Cheminformatics Platform for Interactive Compound Library Processing. *J. Chem. Inf. Model.* **2015**, *55*, 2071−2078.
(77) Ghose, A. K.; Viswanadhan, V. N.; Wendoloski, J. J. A Knowledge-Based Approach in Designing Combinatorial or Medicinal Chemistry Libraries for Drug Discovery. 1. A Qualitative and Quantitative Characterization of Known Drug Databases. *J. Comb. Chem.* **1999**, *1*, 55−68.
(78) Geschwindner, S.; Olsson, L.-L.; Albert, J. S.; Deinum, J.; Edwards, P. D.; de Beer, T.; Folmer, R. H. A. Discovery of a Novel Warhead against β-Secretase through Fragment-Based Lead Generation. *J. Med. Chem.* **2007**, *50*, 5903−5911.
(79) Chen, Y.; Shoichet, B. K. Molecular docking and ligand specificity in fragment-based inhibitor discovery. *Nat. Chem. Biol.* **2009**, *5*, 358−364.
(80) Bietz, S.; Urbaczek, S.; Schulz, B.; Rarey, M. Protoss: A holistic approach to predict tautomers and protonation states in protein-ligand complexes. *J. Cheminformatics* **2014**, *6*, 12.
(81) Leiros, H.-K. S.; Brandsdal, B. O.; Andersen, O. A.; Os, V.; Leiros, I.; Helland, R.; Otlewski, J.; Willassen, N. P.; Smalås, A. O. Trypsin specificity as elucidated by LIE calculations, X-ray structures, and association constant measurements. *Protein Sci.* **2004**, *13*, 1056−1070.
(82) Dullweber, F.; Stubbs, M. T.; Musil, Đ.; Stürzebecher, J.; Klebe, G. Factorising Ligand Affinity: A Combined Thermodynamic and Crystallographic Study of Trypsin and Thrombin Inhibition. *J. Mol. Biol.* **2001**, *313*, 593−614.
(83) Banks, J. L.; Beard, H. S.; Cao, Y.; Cho, A. E.; Damm, W.; Farid, R.; Felts, A. K.; Halgren, T. A.; Mainz, D. T.; Maple, J. R.; Murphy, R.; Philipp, D. M.; Repasky, M. P.; Zhang, L. Y.; Berne, B. J.; Friesner, R. A.; Gallicchio, E.; Levy, R. M. Integrated Modeling Program, Applied Chemical Theory (IMPACT). *J. Comput. Chem.* **2005**, *26*, 1752−1780.
(84) Still, W. C.; Tempczyk, A.; Hawley, R. C.; Hendrickson, T. Semianalytical treatment of solvation for molecular mechanics and dynamics. *JACS* **1990**, *112*, 6127−6129.
(85) *MacroModel*; Schrödinger Release 2019-4, Schrödinger LLC: New York, N. Y. 2019.
(86) Powell, M. *The BOBYQA algorithm for bound constrained optimization without derivatives; Technical Report DAMTP 2009/NA06*; Department of Applied Mathematics and Theoretical Physics, University of Cambridge: Cambridge, 2009.
(87) Moré, J. J.; Thuente, D. J. Line Search Algorithms with Guaranteed Sufficient Decrease. *ACM Trans. Math. Software* **1994**, *20*, 286−307.
(88) Beiranvand, V.; Hare, W.; Lucet, Y. Best practices for comparing optimization algorithms. *Optim. Eng.* **2017**, *18*, 815−848.
(89) Rios, L. M.; Sahinidis, N. V. Derivative-free optimization: a review of algorithms and comparison of software implementations. *J. Glob. Optim.* **2013**, *56*, 1247−1293.
(90) Rardin, R. L.; Uzsoy, R. Experimental Evaluation of Heuristic Optimization Algorithms: A Tutorial. *J. Heuristics* **2001**, *7*, 261−304.
(91) Schönherr, H.; Cernak, T. Profound Methyl Effects in Drug Discovery and a Call for New C—H Methylation Reactions. *Angew. Chem., Int. Ed.* **2013**, *52*, 12256−12267.
(92) Malhotra, S.; Karanicolas, J. When Does Chemical Elaboration Induce a Ligand To Change Its Binding Mode? *J. Med. Chem.* **2017**, *60*, 128−145.
(93) Teotico, D. G.; Babaoglu, K.; Rocklin, G. J.; Ferreira, R. S.; Giannetti, A. M.; Shoichet, B. K. Docking for fragment inhibitors of AmpC β-lactamase. *PNAS* **2009**, *106*, 7455−7460.
(94) Li, H.; Sze, K.-H.; Lu, G.; Ballester, P. J. Machine-learning scoring functions for structure-based virtual screening. *WIREs Comput. Mol. Sci.* **2020**, *10*, e1478.
(95) Ashtawy, H. M.; Mahapatra, N. R. Machine-learning scoring functions for identifying native poses of ligands docked to known and novel proteins. *BMC Bioinf.* **2015**, *16*, S3.
(96) Khamis, M. A.; Gomaa, W. Comparative Assessment of Machine-Learning Scoring Functions on PDBbind 2013. *Eng. Appl. Artif. Intell.* **2015**, *45*, 136−151.
(97) Ragoza, M.; Hochuli, J.; Idrobo, E.; Sunseri, J.; Koes, D. R. Protein-Ligand Scoring with Convolutional Neural Networks. *J. Chem. Inf. Model.* **2017**, *57*, 942−957.
(98) Ashtawy, H. M.; Mahapatra, N. R. Task-Specific Scoring Functions for Predicting Ligand Binding Poses and Affinity and for Screening Enrichment. *J. Chem. Inf. Model.* **2018**, *58*, 119−133.
(99) Ragoza, M.; Turner, L.; Koes, D. R. Ligand Pose Optimization with Atomic Grid-Based Convolutional Neural Networks. *arXiv preprint arXiv:1710.07400* **2017**.
(100) Sieg, J.; Flachsenberg, F.; Rarey, M. In Need of Bias Control: Evaluating Chemical Data for Machine Learning in Structure-Based Virtual Screening. *J. Chem. Inf. Model.* **2019**, *59*, 947−961.