Published in final edited form as:

J Comput Chem. 2018 September 15; 39(24): 2012–2021. doi:10.1002/jcc.25381.

Computational Feasibility of an Exhaustive Search of Side-Chain Conformations in Protein-Protein Docking

Taras Dauzhenka¹, Petras J. Kundrotas^{1,*}, and Ilya A. Vakser^{1,2,*}

¹Center for Computational Biology, The University of Kansas, Lawrence, Kansas 66047, USA

²Department of Molecular Biosciences, The University of Kansas, Lawrence, Kansas 66047, USA

Abstract

Protein-protein docking procedures typically perform the global scan of the proteins relative positions, followed by the local refinement of the putative matches. Because of the size of the search space, the global scan is usually implemented as rigid-body search, using computationally inexpensive intermolecular energy approximations. An adequate refinement has to take into account structural flexibility. Since the refinement performs conformational search of the interacting proteins, it is extremely computationally challenging, given the enormous amount of the internal degrees of freedom. Different approaches limit the search space by restricting the search to the side chains, rotameric states, coarse-grained structure representation, principal normal modes, etc. Still, even with the approximations, the refinement presents an extreme computational challenge due to the very large number of the remaining degrees of freedom. Given the complexity of the search space, the advantage of the exhaustive search is obvious. The obstacle to such search is computational feasibility. However, the growing computational power of modern computers, especially due to the increasing utilization of Graphics Processing Unit (GPU) with large amount of specialized computing cores, extends the ranges of applicability of the brute-force search methods. This proof-of-concept study demonstrates computational feasibility of an exhaustive search of side-chain conformations in protein docking. The procedure, implemented on the GPU architecture, was used to generate the optimal conformations in a large representative set of protein-protein complexes.

Graphical Abstract

Protein interactions are the key part of molecular mechanisms in living systems. Modeling of protein complexes requires refinement of the putative matches that takes into account structural flexibility. Given the complexity of the refinement protocols, the exhaustive search has clear advantage. The obstacle to such search is computational feasibility. In this proof-of-principle study, we address the problem of finding the optimal conformations of the side chains at protein-protein interfaces by an exhaustive search procedure.

*Corresponding authors: vakser@ku.edu or pkundro@ku.edu.

Keywords

rotamers; combinatorial geometry; conformational search; flexible docking

INTRODUCTION

Protein-protein docking procedures typically perform the global scan of the proteins relative positions, followed by the local refinement of the putative matches. Because of the size of the search space, the global scan is usually implemented as rigid-body search, using computationally inexpensive intermolecular energy approximations. In the case of free docking, the search is based on six degrees of freedom of the two rigid bodies system. In the case of comparative docking, the search is based on sequence or structure alignment to the pool of co-crystallized protein-protein templates.¹

An adequate refinement has to take into account structural flexibility.² Such refinement is supposed to be computationally feasible, due to the local nature of the search in the vicinity of the matches from the free or comparative docking scan stage. However, since the refinement performs conformational search of the interacting proteins, it is extremely computationally challenging, given the enormous amount of the internal degrees of freedom. Different approaches limit the search space by restricting the search to the side chains (most proteins have relatively small conformational change upon binding), rotameric states, coarse-grained structure representation, principal normal modes, etc. Still, even with the approximations, proper refinement presents an extreme computational challenge due to the very large number of the remaining degrees of freedom.

A number of methods for predicting side-chain conformations utilizing different search algorithms, rotamers libraries, and scoring functions have been developed in the context of protein structure prediction/computational protein design and implemented in various software applications.^{3–13} In this paper, we focus on the problem of side chain re-packing in

the context of refining protein-protein docking models, which implies fixed amino acid sequence and, consequently, no variability in the number of rotatable bonds. However, relative positions of the docking partners and conformations of the backbone loops may change. Thus, the side-chain conformations have to be predicted in the iterative manner. After each rigid-body/loop movement, a distinct interface is formed, for which the side chain conformations should be optimized/predicted and used in determining the next move. Thus, an erroneous choice of the side chain conformation by the search procedure may lead to the propagation of the errors with further iterations. Although the exact search methods such as DEE/A*,^{14–18} CFN^{11,12} and integer programming approach^{19–21} have been successfully applied to protein design and/or structure prediction, they have not been systematically tested in the refinement of the protein docking. Prediction of the side-chain conformations at the protein-protein interfaces also requires specialized rotamer libraries.²²

Given the complexity of the search space, the advantage of the exhaustive (brute force) search is obvious. The obstacle to such search is of course computational feasibility. However, the growing computational power of modern computers, especially due to increasing utilization of Graphics Processing Unit (GPU) with large amount of specialized computing cores, extends the ranges of applicability of the brute-force search methods.^{23,24}

A number of software applications to protein docking, which were initially developed for CPUs, have been transferred to the GPU architectures^{25–28} achieving significant speed-up factors. An implementation of a traditional Fast Fourier Transform (FFT) based protein docking²⁹ on supercomputers with massive numbers of CPU and GPU nodes including multiple CPU cores and Graphics Processing Unit (GPU) cards allowed conducting a million of docking jobs in < 12 hours.³⁰ The 45-fold speed-up over corresponding calculation on a single 2.3 GHz Xeon CPU core has been achieved with a 1D FFT docking scheme on a single GTX 285 GPU.³¹ GPU implementation of the FFT docking performed approximately 5 times faster than the Many Integrated Core (MIC) offload mode implementation.³² These inspiring results are contingent on successful mappings of specific docking algorithms onto GPUs, which is a major challenge.

A number of search and clustering algorithms used for structure-based computational protein design have also been successfully implemented on GPUs. A single-thread A* search algorithm implemented on CUDA-enabled NVIDIA Tesla K20c GPU yielded 45-fold acceleration compared to the traditional C implementation on Intel Xeon E5-1620 CPU.³³ A 3D protein structure similarity searching algorithm implemented on GTX 560Ti GPU has shown a 180-fold increase in speed compared to a single-thread implementation on Intel Xeon E5620.³⁴ Dang et al.³⁵ reported speed up of about one order of magnitude for CUDA-based implementation of a clustering algorithm on GeForce Titan GPU compared to a multi-threaded CPU implementation on a Core-i7 2700. TM-score-based protein structure comparison implemented on ATI 5870 GPU performed 68 times faster than the original single-threaded implementation on an AMD Phenom II 810 CPU.³⁶ However, due to the overheads associated with CPU-GPU data transfers, these impressive speedups can be severely impeded when GPU-based algorithms are integrated with pre-existent CPU-based software infrastructure. This is, for example, the case in protein structure optimization when an application takes a PDB file as an input and, after performing optimization procedures

with specified parameters, outputs results into another PDB file. When choosing between CPU vs. GPU-based implementation of the same method, one also needs to consider available hardware. A computer may have a large number of CPUs, but a single poor GPU. In such case, running in parallel the CPU-based implementations may be faster than running sequentially one GPU-based implementation.

In this proof-of-principle study, we address computational feasibility of finding the optimal conformations of the side chains at protein-protein interfaces by an exhaustive search in the rotameric state. For this purpose, we developed a procedure, implemented on the GPU architecture, and applied it to generating the optimal conformations (with the minimum number of steric clashes at the interface) of protein-protein complexes corresponding to the unbound proteins in the Dockground benchmark set.

THEORETICAL BACKGROUND

Conformational space and problem formulation

In terms of rotameric states, conformational space for a side chain of an amino acid a (where a is any of the 20 standard amino acids) is described by a set of L_a discrete rotameric states, $r_i(a)$, specific to that amino acid

$$R(a) = \{r_1(a), \dots, r_{L_a}(a)\}, \quad (1)$$

For a sequence of N side chains

$$A_N = (a_1, \dots, a_N), \quad (2)$$

the full conformational space Ω_{full} can be expressed as N -fold Cartesian product

$$\Omega_{\text{full}} = R(a_1) \times \dots \times R(a_N), \quad (3)$$

with N dimensions and

$$N(\Omega_{\text{full}}) = \prod_{j=1}^N L_{a_j} \quad (4)$$

elements (j is the position of a side chain in the sequence A_N). Each element $\omega_k \in \Omega_{\text{full}}$; in (3) corresponds to a particular conformation k of the N side chains (N -tuple)

$$\omega_k = \left(r_{i_k}(a_1), \dots, r_{i_k}(a_j), \dots, r_{i_k}(a_N) \right), \quad (5)$$

where $1 \leq i_k \leq L_{a_j}$ identifies a rotameric state of an individual side chain a_j in position j in the sequence A_N . If a permutation of amino acids in (2) yields A_N 's with distinct properties (which is the case for amino acids connected in a polypeptide chain), then ω_k 's with corresponding permutations of rotameric states can also be distinguished.

We define A_N as a sequence of interface and near-interface residues of a protein-protein complex (see above). Each conformation ω_k is characterized by a number of steric clashes, C_k

$$E(\omega_k) = c_k. \quad (6)$$

Any two heavy atoms belonging to different residues are considered to be in a steric clash if they are within $L_{\text{clash}} = 2 \text{ \AA}$ distance from each other. Closeness of any two conformations ω_k and ω_l can then be quantified as the difference in corresponding numbers of steric clashes.

Exhaustive search in conformational subspaces

The side-chain packing problem in protein design (and, consequently, in protein docking) is classified as NP-hard (with the formal theoretical proof by reduction of the SAT, Boolean satisfiability, and problem to PRODES)³⁷ and inapproximable.³⁸ Thus, there is no known polynomial-time algorithm for the solution of this problem and one has to rely on exponential-time algorithms in order to find the Global Minimum-Energy Conformation (GMEC) ω_k (Eq. 5). On the other hand, the side-chain packing in protein docking has a number of features that are distinct from the side-chain packing in protein design, and which allow development of specialized and efficient parallel algorithms.³⁹ For example, when residue interaction graphs are constructed using lower limits for the $C_\alpha - C_\alpha$ distances, inherent in the models of individual proteins,⁴⁰ an algorithm based on tree decomposition yields a globally exact side-chain assignment with computational complexity $O(Nr^{cN^{2/3}\log N})$ (where r is the average number of rotamers, and c is a constant).^{13,40} However, docking scan (the input for the refinement) often yields models with considerable steric clashes. Thus, the distances between C_α atoms in the interacting partners may not have such lower limit. Since the side-chain re-packing in such models guide other stages of the docking refinement protocol (rigid-body movement and/or backbone adjustments), a straightforward application of the tree-decomposition algorithms to the refinement of the docking models is problematic.

Lower than exponential computational complexity may also be expected when conformations of some side chains can be predicted independently (e.g., a side chain with all rotameric states not clashing with the surrounding side chains). However, identification of

such conformationally independent side chains by itself would require a number of computations that again would depend on the number of side chains in a system N (an in-depth analysis of such cases is beyond the scope of this work).

We propose to split sequence A_N into fragments (Figure 1a), each consisting of $M < N$ residues and overlapping with the neighboring fragments by M_{OV} residues (the last fragment may be shorter than M residues). The total number of such fragments in the sequence A_N is defined as

$$M_F = 1 + \left\lceil \frac{N - M}{M - M_{OV}} \right\rceil, \quad (7)$$

where brackets $\lceil \dots \rceil$ denote the ceiling function. The fragments in A_N are numbered by an index $1 < m < M_F$. Each fragment starts with amino acid a_{j_s} indexed by $j_s = 1 + (m - 1)(M - M_{OV})$, and ends with amino acid a_{j_f} indexed by $j_f = j_s + M - 1$ (Fig. 1). Thus, in accordance with Eqs. (1), (3) and (4), conformational space of fragment m is given by

$$\Omega_m^M = R(a_{j_s}) \times \dots \times R(a_{j_f}) \quad (8)$$

and has

$$N(\Omega_m^M) = \prod_{j=j_s}^{j_f} L_{a_j} \quad (9)$$

elements. Each element $\omega_m^M \in \Omega_m^M$

$$\omega_m^M = \left(r_{i_{j_s}}(a_{j_s}), \dots, r_{i_{j_f}}(a_{j_f}) \right) \quad (10)$$

is a sub-sequence of $\omega_k \in \Omega_{full}$ (Eq. 5)

$$\omega_k = \left(\dots, r_{i_s-1}(a_{j_s-1}), \omega_m^M, r_{i_f+1}(a_{j_f+1}), \dots \right). \quad (11)$$

Sequences of rotamer state indices corresponding to ω_k and ω_m^M will be referred to as *register* and *block of register*, correspondingly. Sizes of the register and blocks are determined by the numbers of interface residues N and the number of residues in a fragment M respectively.

A set of all possible combinations of indices inside each block constitutes a conformational subspace $\Omega_m^M \subset \Omega_{\text{full}}$ (Fig. 1c). Then, instead of an exhaustive search in the full conformational space Ω_{full} , one can perform a series of exhaustive searches in subspaces $\Omega_m^M \subset \Omega_{\text{full}}$ and find a set of solutions $\{\omega_{m,\text{min}}^M \in \Omega_m^M\}$, where each $\omega_{m,\text{min}}^M$ minimizes the function (6) within the corresponding subspace Ω_m^M (i.e., minimizes the number of clashes inside each interface fragment). These $\omega_{m,\text{min}}^M$ can be stapled together in order to obtain a conformation for the full interface $\omega_{\text{min}} \in \Omega_{\text{full}}$ (Fig. 1c). This allows a reduction of the number of required operations from

$$N(\Omega_{\text{full}}) \sim r^N \quad (12)$$

to

$$\sum_{m=1}^{M_F} N(\Omega_m^M) \sim r^M \times M_F, \quad (13)$$

where r is the average number of rotameric states per side chain. The results reported in this paper are obtained for the block size $M = 7$ and $M_{\text{ov}} = 2$.

Since clashes at the complex interface can occur between residues of different proteins, we constructed the fragments of the interface residues such that the neighboring positions of a fragment contain the closest (by proximity of the C^α atoms) residues from different chains (for a binary complex, the fragment construction is shown in Fig. 2a). The consistency of the stapling procedure is achieved through the requirement that each block with $m > 1$ preserves its first two indices (one residue from each interacting chain) unchanged and equal to the corresponding indices obtained during the exhaustive search in the previous block (Fig. 2b).

MATERIALS AND METHODS

Outline of the algorithm

The proposed algorithm relies on three main components: (i) splitting the conformational space of the interface side chains (ISC) into subspaces of overlapping ISC fragments (a “piecemeal” approach⁴¹ that can be traced all the way back to the *Aufbau* paradigm in conformational analysis^{42,43}), (ii) combination of geometric primitives (pre-computed representations of ISC rotameric states) in constructing rotameric states from a subspace, and (iii) GPU acceleration of the exhaustive searches in the subspaces. Conformational

splitting and the piecemeal approach have been previously successfully integrated with the dead-end elimination and Monte Carlo methods.^{41,44} Employment of geometric primitives allows reduction of the allocated memory and the CPU/GPU data transfer overheads.

The proposed method deals with rotameric states only for the side chains at the interface, while positions of all other side chains and of the backbone are fixed. We defined the interface residues as those with at least one heavy atom within 6 Å from any heavy atom of the other protein. We also distinguished a near-interface set of residues as those located within 10 Å from any of the interface residues. The conformations of the near-interface residues are kept fixed and serve as the environment in which the interface residues change their rotameric states. Our algorithm is generalized for the complexes with any number of interacting partners.

Library of rotamers

We employed our previously developed library of unbound rotameric states of the interface side chains.²² The average number of rotameric states per side chain in this library is five. The sets of rotameric states for each side chain are complemented by the unmodified unbound states of the corresponding side chains from the initial unrefined structure of the protein-protein complex. Thus, in total, one has on average six states for each side chain (5 rotamers + 1), which makes possible the exhaustive search through rotameric sub-spaces in a reasonable amount of time.

Dataset of protein-protein complexes

The procedure was tested on the Dockground set 3.0 of the X-ray unbound structures (<http://dockground.compbio.ku.edu>).⁴⁵ The set contains 99 co-crystallized complexes that have both components also crystallographically determined in the unbound state. The initial unrefined complexes for the testing were constructed by structurally aligning (by TM-align⁴⁶) the unbound monomers onto corresponding bound ones.

Realistically, docking scan does not produce models in which poses of the proteins exactly correspond to those in the native bound complex. To address the real-case scenario, we generated sets of near-native models for each of the 99 target complexes by rotating ($\pm 10^\circ$ with 5° step) and translating (± 2 Å with 1 Å step) the native ligand so that the resulting L-RMSD is within 3 Å from the native structure. The set contains 30,919 models corresponding to the 99 complexes.

Pre-computing rotameric states of interface side chains

The advantage of independent data processing by GPUs over the CPUs is based on a larger number of transistors specifically devoted to data processing rather than data caching and flow control. This permits an efficient implementation of the Single-Instruction, Multiple-Thread (SIMT) processing model.⁴⁷ In our algorithm, the single procedure comprises calculation of the number of steric clashes for a set of ISC conformations. These calculations can be performed similarly and independently for each set of conformations. Thus, GPU data processing can yield significant acceleration of the conformational search.

We implemented the proposed method for GPUs with Compute Unified Device Architecture (CUDA) from NVIDIA using C++/CUDA C programming language. Calculations were carried out on two Linux workstations (HP laptop with GPU GeForce GTX 950M, GCC 4.9.2; and Dell desktop with GPU Quadro K4000, GCC 4.4.7; both with CUDA Compilation Tools 7.5). In the CUDA heterogeneous programming model, the host (CPU) running a C/C++ program passes data and flow control to a physically separate device (GPU), which operates as a coprocessor with its own memory.⁴⁷ The time of the host/device data transfers may severely impede the algorithm speed-up. To mitigate this problem, we used pre-computed representations of rotameric states for each ISC (Fig. 3). Each Cartesian point (data structure `Point3DCart`) stores coordinates of an ISC atom in some rotameric state. The initial conformations of the ISC atoms (as in input PDB file) are stored in the data structures `ExtractedResidue` (one per interface residue) as arrays of `Point3DCart`. Positions of the ISC atoms beyond C^β (e.g., Tyr in Fig. 3c) are changed to another rotameric state and the new set of Cartesian coordinates for that state is stored in a separate `ExtractedResidue` structure (Fig. 3c,d). After generating all rotameric states for each ISC (except Ala and Gly), the corresponding `ExtractedResidue` structures are grouped into a single data structure `LibraryExtractedResidues` (Fig. 3d), which is transferred to the device memory only once. Then, each thread on a GPU uses a unique combination of `ExtractedResidue` structures, determined by a combination of indices of the ISC rotameric states.

Such data optimization storage requires only a small amount of data transferred to the GPU memory. For example, for 10 ISC with average 20 atoms and 5 rotameric states per side chain (9,765,625 distinct configurations in total) the `LibraryExtractedResidues` data structure is 12 kB (in actual calculations the size varies from 8.4 kB to 240 kB). The brute-force approach in this case would result in $9,765,625 \times 10 \times 20 \times 12 \text{ Bytes} > 23 \text{ GB}$, the amount of memory that rapidly approaches current limits for the CUDA-enabled GPUs. Splitting this conformational space, with separate scoring in each part, would require multiple host/device data transfers.

Static conformations of the near-interface residues are transferred to the GPU memory as an array of `Point3DCart` structures and participate in the scoring of conformations. The typical size of this array is 18 kB.

Details of the algorithm

The input for the kernel performing evaluation of the numbers of clashes consists of a data structure `LibraryExtractedResidues` organized in a plain array `aPointsRot` of `Point3DCart` structures; an array `aPointsEnv` of `Point3DCart` for all the near-interface residues; a number of conformations in a fragment, `numConformationsFragment` (equals to $N(\Omega_m^M)$); an array of indices of the ISC rotameric states, `conformation` (blocks of registers corresponding to conformations in the subspace Ω_m^M); a number of residues at the interface of a protein-protein complex, `numResiduesInterface`; and a value of distance for the clash criterion, `clashDistance` (Fig. 4). The output is data structure `IndexClash` (array of pairs *<index of a conformation, number of clashes in that conformation>*).

Each thread on the GPU (lines 1 through 4 in Fig. 4) calculates, independently of other threads, the number of clashes (`numClashes`) for a unique conformation and stores this number in the `indexClash`. The first part of the procedure `getNumClashesInConformation` (lines 2 – 15 in Fig. 5) computes `numClashes` between pairs of interface residues from `aPointsRot` in the rotameric states `indexRotStateI` and `indexRotStateJ`. Possible clashes between the interface residues from the array `aPointsRot` and static near-interface residues from the array `aPointsEnv` are accounted for in the second part of the procedure (lines 16 – 26 in Fig. 5).

The algorithm output `IndexClash` is searched for the index of a conformation $\omega_{m,\min}^M$ with the minimal number of clashes. Multiple conformations characterized by the same minimal number of clashes are scored by the sum of probabilities of individual rotameric states²² and conformation with the largest sum is taken as a solution. The obtained solutions $\omega_{m,\min}^M$ from each subspace Ω_m^M are concatenated using the 2-residues overlap criterion (see above), resulting in the full solution $\omega_{\min} \in \Omega_{\text{full}}$. The Cartesian coordinates of the side-chains atoms are restored using the indices of rotameric states from $\omega_{\min}$ and the full structure of a protein-protein complex is saved to a new PDB file.

RESULTS AND DISCUSSION

Computational efficiency of the algorithm

The proposed exhaustive search of the side-chain rotamers was implemented in C++/CUDA C programming language and applied to removal of steric clashes at the interfaces of protein-protein complexes constructed from the unbound interactor structures (see Methods). In the CUDA implementation, GPU threads are grouped into warps and each thread in a warp must execute the same instruction at any given time. Presence of for-loops with different sizes for different threads and code branching may lead to significant deterioration of the overall performance (warp divergence). In the algorithm (Fig. 5), the numbers of points for different conformations of a residue are the same. Thus, the for-loops in the code (lines 6, 7, 18 and 19 in Fig. 5) do not lead to a significant divergence of the threads. All if-then statements in the code contain only a single operation of increment of the integer variable `numClashes`, which also should not lead to a significant divergence of the threads.

Efficient GPU utilization is reflected in high *occupancy* of GPU MultiProcessors (MP), which is the ratio of the number of active warps per MP to the maximum number of warps that can reside on a MP. The high occupancy allows to hide the latencies of instruction pipeline and global memory access.⁴⁸ The software adjusts kernels launch parameters depending on the number of conformations in a particular fragment and the parameters of the GPU. The occupancy reported by the NVIDIA Visual Profiler for our code on GeForce GTX 950M is 47% out of the theoretical maximum 50% (30 active warps out of maximum 64; the maximum number of threads per block 1024).

The run time of the algorithm implemented on the GPU varies from 0.07 seconds for conformational space consisting of $N(\Omega_{\text{full}}) = 12$ elements (3 interface residues) to 22,351

seconds for 15 subspaces with total $\sum_m N(\Omega_m^M) = 1.5 \times 10^8$ elements (107 interface residues with $N(\Omega_{\text{full}}) \sim 5^{107}$ elements in the full conformational space). For each complex in the dataset, we have performed calculations on two GPU/CPU architectures (see Methods) and calculated the speed-up rate

$$s = \frac{t_{\text{CPU}}}{t_{\text{GPU}}}, \quad (14)$$

where t_{GPU} and t_{CPU} are wall-clock time of the execution of GPU-enabled and CPU-only implementation of the same algorithm, respectively (Fig. 6). Significant variations in the speed-up rates for different complexes are due to different numbers of the interface residues, which lead to different sizes of structures `aPointsEnv` and `LibraryExtractedResidues` and consequently to varying associated host/device data transfer overheads. In almost all the cases the GTX950M outperforms Quadro K4000 (Fig 6) due to the larger number of Streaming Multiprocessors and improved warp scheduler in GTX950M.

The CPU-only and GPU-enabled implementations utilize the same data structures and scoring procedure. This means that the observed speed-up rates come from the proper automatic adjustment of kernels launch configurations and SIMT processing model⁴⁷ inherent to the GPU architecture. We argue that if our GPU-based implementation were compared to another CPU-based implementation of the exhaustive search (with different data structures and/or without using the geometric primitives), the speed-up rates would be even higher.

Since the problem of side-chain repacking is inapproximable,³⁸ it is NP-hard to distinguish between the exact solution ω_{min} (GMEC) and an arbitrary close approximate solution ω_{sol} . This means that there is no known polynomial-time algorithm for finding ω_{sol} . The algorithm, proposed in this paper is not an exception since it requires a number of exponentially growing steps (Eqs. 12 and 13). Indeed, for the small interfaces with < 7 residues, the total number of conformations sampled by the algorithm grows with the number of residues nearly as $\sim 5^N$ (blue line in Fig. 7). This is in excellent agreement with the equation (12) with 5 rotamer states per side chain on average (which is characteristic for our rotamer library) and 7 residues fragments used in this study.

For interfaces with $N > 7$, the number of sampled conformations increases above $\sim 10^5$ (roughly corresponding to the full conformational space of one fragment 5^7) at significantly lower rate, nearly following the equation (13) (red line in Fig. 7). Without splitting to fragments, the number of conformations instead of changing the slope would follow the blue line in Figure 7. The dispersion of points in Figure 7 for the interfaces with ~ 20 residues (3 fragments) is due to statistically larger weight of the eventual significant variation of the number of conformations for fragments with different amino acid sequences (e.g., a fragment of 7 valines has 3^7 conformations, while a fragment of 7 arginines has 12^7 conformations).

Time of execution was measured by GNU/time tool on Linux workstation with CPU Intel Xeon E5-2697v2 2.7 GHz, on the set of 30,919 near-native protein-protein structures (see Methods). The average execution time for GPU-accelerated exhaustive search was 81 seconds vs. 4 seconds for SCWRL4. Both software applications were implemented as single binary executables that input and output PDB files. Thus, the reported times comprise the entire protocols and not just the conformational search. The relatively slower execution time of our procedure is expected due to the exhaustive nature of the search.

Structure refinement

The structures output by the algorithm - Fragment-Exhaustive-Search structures (FES) were evaluated in terms of differences between the number of the steric clashes (calculated by 2Å distance criterion) in the final (N_{cl}^{fin}) and the initial unrefined (N_{cl}^{init}) complexes

$$\Delta N_{cl} = N_{cl}^{fin} - N_{cl}^{init}. \quad (15)$$

The procedure reduced the numbers of clashes for 89 complexes (completely removed for 28 complexes). For 8 complexes the number of clashes did not change.

The N_{cl} was also calculated for the complexes (henceforth referred to as U_B structures) produced by changing conformations of the interface side chains in the initial unrefined complexes to the rotameric states closest (in terms of RMSD of the side-chain dihedral angles) to the conformations of corresponding side chains in the respective X-ray bound structures. Two complexes were excluded from construction of the U_B structures due to a mismatch in the interface residue composition between bound and unbound structures. We also calculated a biologically relevant docking metric - a fraction of correctly predicted native contacts,⁴⁹ for the unbound structures relative to the corresponding bound ones, based on 97 protein-protein complexes from the Dockground unbound benchmark set 3.0. The results were compared to those calculated for the same structures by SCWRL4 software¹³ with `-h` `-t` options (`-h` disables the output of hydrogen atom coordinates, and `-t` disables adding hydrogen atoms to the N-terminal nitrogen atom) and flexible sub-rotamers (enabled by default). The results of FES and SCWRL4 were close (Fig. 8), despite our use of simple backbone-independent rotamers and a trivial scoring function.

The same analysis of the number of removed clashes was performed on the set of 30,919 near-native docking models (see Methods). For this set, FES produced better results than SCWRL4 (Fig. 9a). As one can expect, SCWRL4 sub-rotamers enrich the search space and thus produced somewhat better results than just the SCWRL4 rotamers. The rotamers library in SCWRL4 contains an order of magnitude more rotamers per side-chain than our library. Thus, even the reduced by 50% SCWRL4 set of more probable rotamers is larger than our full set. Because of the small number of rotameric states in our library (maximum 12 per residue) we did not perform numerical experiment to find dependence of refinement quality on the size of the rotameric space.

Supplemental Figure S1 shows how the number of the initial clashes depends on the average number of rotatable bonds per interface residue and the number of interface residues for 30,919 docking models (with different values of L-RMSD) used in this study. Because of the nature of our set of protein complexes, which does not have sequence variability at the interface, we could not draw conclusions on the influence of the number of rotatable bonds per residue on the number of clashes in case of residue variation.

The number of structures where SCWRL4 introduced extra clashes increased with the increasing L-RMSD of the initial near-native models (Fig. 9b). This may suggest that the scoring function and/or search method implementation in SCWRL4 is less suitable for cases with a significant spatial overlap of the backbones of the docking proteins. As expected, FES reduced the number of clashes in most cases. A small number of cases, where the extra clashes were introduced, was due to imperfect stapling procedure that reconstructs the interface from the fragments.

CONCLUSIONS

The piecemeal approach and the corresponding splitting of protein-protein interfaces into fragments allowed us to consider conformational spaces Ω_m^M of the side-chain rotamers separately for each fragment. Exhaustive search in these conformational spaces produces a sequence of solutions $\{\omega_{m,\min}^{M=7} \in \Omega_m^{M=7} \mid 1 \leq m \leq M_F\}$ that are stapled together to obtain a solution for the whole interface. The method reduces the number of steric clashes that would be present after low-resolution docking. The algorithm occupies a place between the complete exhaustive search and the polynomial-time approximation schemes in a sense that it allows limiting the combinatorial explosion but does not allow polynomial-time discrimination between solution and an approximation to this solution.

This proof-of-concept study demonstrates computational feasibility of an exhaustive search of side-chain conformations in protein packing. The results of the exhaustive search using simple backbone-independent rotamers and a trivial scoring function are comparable to those obtained by state-of-the-art refinement with more accurate backbone-dependent rotamers and sophisticated scoring. The search can efficiently be employed in the refinement of protein docking where propagation of errors between different stages of an iterative algorithm (e.g., between side-chain re-packing and rigid-body rotation/translation movements) must be avoided.

The GPU-enabled exhaustive search runs on average 50 times faster than the CPU-only implementation of the same algorithm (the maximal achieved speed-up for our benchmark set was 91 times). In the future, we plan to investigate the speed-up of this implementation on multiple-GPU hardware systems. We will study applicability of the refinement to the output of low-resolution docking scan for unbound X-ray and modeled proteins.⁵⁰ The resulting protocols will be implemented in the refinement stage of our GRAMM docking software http://vakser.compbio.ku.edu/main/resources_gramm.php.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

This study was supported by NIH grant R01GM074255 and NSF grants DBI1262621, DBI1565107 and CNS133789.

References

1. Vakser IA. Biophys J. 2014; 107:1785–1793. [PubMed: 25418159]
2. Zacharias M. Curr Opin Struct Biol. 2010; 20:180–186. [PubMed: 20194014]
3. Liang S, Zheng D, Zhang C, Standley DM. Bioinformatics. 2011; 27:2913–2914. [PubMed: 21873640]
4. Liang S, Zhou Y, Grishin N, Standley DM. J Comput Chem. 2011; 32:1680–1686. [PubMed: 21374632]
5. Brown JB, Dukka KC, Tomita E, Akutsu T. Genome Inform. 2006; 17:3–12.
6. Kuhlman B, Baker D. Proc Natl Acad Sci USA. 2000; 97:10383–10388. [PubMed: 10984534]
7. Xiang Z, Steinbach PJ, Jacobson MP, Friesner RA, Honig B. Proteins. 2007; 66:814–823. [PubMed: 17206724]
8. Miao Z, Cao Y, Jiang T. Bioinformatics. 2011; 27:3117–3122. [PubMed: 21949272]
9. Lu M, Dousis AD, Ma J. Protein Sci. 2008; 17:1576–1585. [PubMed: 18556476]
10. Hartmann C, Antes I, Lengauer T. Protein Sci. 2007; 16:1294–1307. [PubMed: 17567749]
11. Traore S, Allouche D, Andre I, de Givry S, Katsirelos G, Schiex T, Barbe S. Bioinformatics. 2013; 29:2129–2136. [PubMed: 23842814]
12. Traore S, Allouche D, Andre I, Schiex T, Barbe S. Methods Mol Biol. 2017; 1529:107–123. [PubMed: 27914047]
13. Krivov GG, Shapovalov MV, Dunbrack RL Jr. Proteins. 2009; 77:778–795. [PubMed: 19603484]
14. Desmet J, De Maeyer M, Hazes B, Lasters I. Nature. 1992; 356:539–542. [PubMed: 21488406]
15. Leach AR, Lemon AP. Proteins. 1998; 33:227–239. [PubMed: 9779790]
16. Georgiev I, Lilien RH, Donald BR. J Comput Chem. 2008; 29:1527–1542. [PubMed: 18293294]
17. Chen CY, Georgiev I, Anderson AC, Donald BR. Proc Natl Acad Sci USA. 2009; 106:3764–3769. [PubMed: 19228942]
18. Gainza P, Roberts KE, Georgiev I, Lilien RH, Keedy DA, Chen CY, Reza F, Anderson AC, Richardson DC, Richardson JS, Donald BR. Methods Enzymol. 2013; 523:87–107. [PubMed: 23422427]
19. Eriksson O, Zhou Y, Elofsson A. In International Workshop on Algorithms in Bioinformatics. Springer; Berlin, Heidelberg: 2001. 128–141.
20. Allouche D, Andre I, Barbe S, Davies J, de Givry S, Katsirelos G, O'Sullivan B, Prestwich S, Schiex T, Traore S. Artif Intell. 2014; 212:59–79.
21. Kingsford CL, Chazelle B, Singh M. Bioinformatics. 2005; 21:1028–1036. [PubMed: 15546935]
22. Kirys T, Ruvinsky AM, Tuzikov AV, Vakser IA. Proteins. 2012; 80:2089–2098. [PubMed: 22544766]
23. Majumder T, Pande PP, Kalyanaraman A. IEEE Des Test. 2014; 31:8–18.
24. Stone JE, Phillips JC, Freddolino PL, Hardy DJ, Trabuco LG, Schulten K. J Comput Chem. 2007; 28:2618–2640. [PubMed: 17894371]
25. Kannan S, Ganji R. Proc IEEE Congress Evolutionary Comput; 18–23 July 2010; Barcelona. 2010. 1–8.
26. Korb O, Stutzle T, Exner TE. J Chem Inf Model. 2011; 51:865–876. [PubMed: 21434638]
27. Pechan I, Feher B. Proc 21st IEEE Int Conf Field Programmable Logic Applications; 5–7 Sept 2011; Chania. 2011. 474–477.

28. Sukhwani B, Herbordt MC. GPGPU-2: Proc 2nd Workshop on General Purpose Processing on Graphics Processing Units. ACM; New York, NY: 2009. 19–27.
29. Katchalski-Katzir E, Shariv I, Eisenstein M, Friesem AA, Aflalo C, Vakser IA. Proc Natl Acad Sci USA. 1992; 89:2195–2199. [PubMed: 1549581]
30. Ohue M, Shimoda T, Suzuki S, Matsuzaki Y, Ishida T, Akiyama Y. Bioinformatics. 2014; 30:3281–3283. [PubMed: 25100686]
31. Ritchie DW, Venkatraman V. Bioinformatics. 2010; 26:2398–2405. [PubMed: 20685958]
32. Shimoda T, Suzuki S, Ohue M, Ishida T, Akiyama Y. BMC Syst Biol. 2015; 9(Suppl 1):S6.
33. Zhou Y, Xu W, Donald BR, Zeng J. Bioinformatics. 2014; 30:i255–i263. [PubMed: 24931991]
34. Mrozek D, Brozek M, Malysiak-Mrozek B. J Mol Model. 2014; 20:2067. [PubMed: 24481593]
35. Dang HV, Schmidt B, Hildebrandt A, Hildebrandt AK. Parallel, Distributed and Network-based Processing: 22nd Euromicro International Conference; 12 – 14 Feb 2014; Torino. 2014. 1–8.
36. Hung LH, Samudrala R. Bioinformatics. 2012; 28:2191–2192. [PubMed: 22718788]
37. Pierce NA, Winfree E. Protein Eng. 2002; 15:779–782. [PubMed: 12468711]
38. Chazelle B, Kingsford C, Singh M. Inf J Comput. 2004; 16:380–392.
39. Moghadasi M, Mirzaei H, Mamonov A, Vakili P, Vajda S, Paschalidis I, Kozakov D. J Chem Inf Model. 2015; 55:872–881. [PubMed: 25714358]
40. Xu JB. Proc Res Comput Mol Biol. 2005; 3500:423–439.
41. Pitman DJ, Schenkelberg CD, Huang YM, Teets FD, Ditursi D, Bystroff C. Bioinformatics. 2014; 30:1138–1145. [PubMed: 24371152]
42. Scheraga HA. Chemica Scripta. 1989; 29A:3–13.
43. Lee J, Scheraga HA, Rackovsky S. J Comput Chem. 1997; 18:1222–1232.
44. Pierce NA, Spriet JA, Desmet J, Mayo SL. J Comput Chem. 2000; 21:999–1009.
45. Gao Y, Douguet D, Tovchigrechko A, Vakser IA. Proteins. 2007; 69:845–851. [PubMed: 17803215]
46. Zhang Y, Skolnick J. Nucleic Acids Res. 2005; 33:2302–2309. [PubMed: 15849316]
47. NVIDIA Corporation. 2014
48. Harris M. NVIDIA Corporation; 2014. devblogs.nvidia.com
49. Mendez R, Leplae R, De Maria L, Wodak SJ. Proteins. 2003; 52:51–67. [PubMed: 12784368]
50. Anishchenko I, Kundrotas PJ, Vakser IA. Proteins. 2017; 85:470–478. [PubMed: 27701777]

Figure 1. Relation between interface residues, full conformational space, fragments and corresponding subspaces

(a) Schematic illustration of splitting the sequence of the interface residues A_N into fragments. (b) Schematic representation of a register with a set of blocks (N is the number of residues in register, M is the number of residues in each block): dashed and solid boxes represent a register and a block at different values of index m of a block. (c) Schematic representation of the relation between conformational subspaces corresponding to different blocks in a register. Red, blue and green dashed boxes correspond to blocks 1, 2 and 3 of the panel (b) respectively. Results of the exhaustive search in each conformational subspace, shown by points, are concatenated in a consistent way to form a solution register.

Figure 2. Sequence of interface residues and rotameric states in different blocks

(a) Schematic illustration of construction of a sequence of the interface residues A_N from residues in different proteins within a complex. For each residue of chain H, a partner-residue from chain L is put into correspondence, provided the following conditions hold: (i) current residue in chain L has not yet been included into A_N , (ii) distance between C^α atoms of the current residue in chain L and current residue in chain H is minimal. (b) Consistency of the stapling procedure illustrated for the two blocks, each containing 6 elements and overlapping by 3 elements. Each element $r_{ij}(a_i)$ in a block is an index of a rotameric state for the residue a_i . The first two elements of the block $m+1$ (dashed rectangles) are taken from the solution $\omega_{m,\min}^7$ in the subspace Ω_m^7 of the block m and are kept fixed during the exhaustive search in the subspace Ω_{m+1}^7 of the block $m+1$.

Figure 3. Representations of TYR in different rotameric states in the algorithm data structures
(a) Interface of a model of complex Ecotin Y69F, D70P bound to D102N Trypsin (1ezu) from the Dockground unbound benchmark set 3.0; **(b)** zoom in of the interface fragment with the Tyr217 residue (chain C); **(c)** mapping of three rotameric states of the TYR217 side chain onto the data structures; **(d)** data structures associated with the atoms (Point3DCart), separate rotameric states of a side-chain (ExtractedResidue) and storage of sets of these mappings (LibraryExtractedResidues). Data structure LibraryExtractedResidues is a composition of all structures ExtractedResidue.

The total number of elements in the LibraryExtractedResidues is $N_{el} = \sum_{j=1}^{N_{int}} N_{rot}(i)$, where $N_{rot}(i)$ is the number of rotamers for the interface side chain i .

Algorithm 1: *Evaluation of scoring function by each thread on a GPU*

Input: `aPointsRot` -- LibraryExtractedResidues as a plain array of `Point3DCart`;
`aPointsEnv` -- an array of `Point3DCart` for the fixed near-interface residues;
`numConformationsFragment` -- number of conformations in a fragment Ω_m^M ;
`conformation` -- an array of indices of rotameric states of interface residues;
`numResiduesInterface` -- number of residues at the interface;
`clashDistance` -- value of distance for the criterion of a steric clash;

Output: `indexClash` -- array of pairs: index of a conformation – corresponding number of steric clashes.

```

1 for i from 1 to numConformationsFragment in parallel do
2   numClashes ← getNumClashesInConformation( conformation,
                                              aPointsRot,
                                              aPointsEnv,
                                              numResiduesInterface,
                                              clashDistance )

3   indexClash ← ( i, numClashes )
4 end for

```

Figure 4.

Evaluation of the scoring function by each thread on a GPU.

Algorithm 2: *Evaluation of the number of clashes in a conformation* (getNumClashesInConformation)

Input: conformation -- an array of indices of rotameric states of interface residues;
 aPointsRot -- an array of Point3DCart for the interface residues;
 aPointsEnv -- an array of Point3DCart for the fixed near-interface residues;
 numResiduesInterface -- number of residues at the interface;
 clashDistance -- value of distance for the criterion of a steric clash;
Output: numClashes -- number of clashes in conformation;

```

1 numClashes ← 0
2 for i from 1 to numResiduesInterface - 1 do
3   indexRotStateI ← index of rotameric state for residue i from conformation
4   for j from i + 1 to numResiduesInterface do
5     indexRotStateJ ← index of rotameric state for residue j from conformation
6     for each Point3DCart pI of residue i in rotameric state indexRotStateI do
7       for each Point3DCart pJ of residue j in rotameric state indexRotStateJ do
8         if distance between pI and pJ is less than or equal to clashDistance
9           then
10            numClashes ← numClashes + 1
11          end if
12        end for
13      end for
14    end for
15  end for
16 for i from 1 to numResiduesInterface do
17   indexRotStateI ← index of rotameric state for residue i from conformation
18   for each Point3DCart pI of residue i in rotameric state indexRotStateI do
19     for each Point3DCart pE from aEnvPoints do
20       if distance between pI and pE is less than or equal to clashDistance
21         then
22          numClashes ← numClashes + 1
23        end if
24      end for
25    end for
26  end for

```

Figure 5.

Pseudo code for evaluation of the number of clashes in a conformation.

Figure 6. Speed-up rates of the clash removal algorithm for 96 protein-protein complexes from the Dockground unbound benchmark set 3.0

The speed-up rates are from Eq. 14 (see text). The calculations were performed on a HP laptop with GPU GeForce GTX950M, CUDA Compilation Tools 7.5, gcc compiler 4.7.2, CPU Intel Core-i7-4722HQ 2.4GHz (light gray bars); and Dell desktop with GPU Quadro K4000, CUDA Compilation Tools 7.5, gcc 4.4.7, CPU Intel Xeon E5-2697v2 2.7 GHz (dark gray bars). The same codes for GPU-enabled and CPU-only versions of the algorithm were compiled on both computers. PDB codes correspond to the X-ray structure of the protein complexes, from which the unbound monomers were used for the construction of the initial unrefined complexes.

Figure 7. Number of sampled conformations $\sum_m N(\Omega_m^M)$ vs. number of residues at the interface for 99 protein-protein complexes from the Dockground benchmark set 3.0. Blue and red lines are functions $y = 5^x$ and $y = 10^4 \times x$, respectively. Inset shows the zoom in of the data for the complexes with < 7 interface residues. Blue line in the inset is the linear fit to the data points. The slope of the fitting line is 0.69 ± 0.02 (regression coefficient 0.99), which is close to $\log(5)=0.69$.

Figure 8. Refinement results

(a) Difference between the final and the initial steric clashes; and (b) fraction of correctly predicted native contacts in the protein-protein complexes from the Dockground unbound benchmark set 3.0. Results of the fragments exhaustive search (FES) are shown in comparison with SCWRL4, and the rotameric states closest to the corresponding bound X-ray structure. The initial and final numbers of clashes are determined based on the off-grid distances between atoms from different residues (a CPU side algorithm). Negative values of N_{cl} correspond to the numbers of removed clashes; positive values correspond to the numbers of introduced clashes. The $fNat$ values were calculated for the unbound structures relative to the corresponding bound ones.⁴⁹ $fNat_{final}$ is the fraction of the native contacts in a structure with repacked side-chains; and $fNat_{initial}$ is the fraction of the native contacts in the initial structure from the benchmark set.

Figure 9. Comparison of FES and SCWRL4 refinement of near-native complexes

The 30,919 near-native protein-protein structures were generated from the Dockground benchmark set 3.0. The refinement procedure is assessed in terms of (a) the absolute numbers of final vs. initial steric clashes; and (b) the difference between the final and the initial steric clashes vs. near-native/native L-RMSD. SCWRL4 was run using following configurations: “default”, “no subrotamers” (disabled subrotamers), and “MaxProbabilitySum=0.5” (all rotamers sorted by decreasing frequency, and top 50% employed). The initial and the final numbers of clashes were determined based on the off-grid atom-atom distances (a CPU side algorithm).