Local Search Heuristic for Rigid Protein Docking [⋆]

V. Choi[⋆⋆], P. K. Agarwal[⋆⋆⋆], H. Edelsbrunner[†] and J. Rudolph[‡]

Duke University, Durham, North Carolina

Abstract. We give an algorithm that locally improves the fit between two proteins modeled as space-filling diagrams. The algorithm defines the fit in purely geometric terms and improves by applying a rigid motion to one of the two proteins. Our implementation of the algorithm takes between three and ten seconds and converges with high likelihood to the correct docked configuration, provided it starts at a position away from the correct one by at most 18 degrees of rotation and at most 3.0Å of translation. The speed and convergence radius make this an attractive algorithm to use in combination with a coarse sampling of the six-dimensional space of rigid motions.

1 Introduction

Protein interactions are the molecular basis for many essential components of life. In this paper we contribute to the growing body of work on *protein docking,* which is the computational approach to predicting protein-protein interactions.

Field of protein docking. The reliable prediction of protein interactions from three-dimensional structures alone is one of the grand challenges in computational biology. There is ample experimental evidence from X-ray crystallography and other structure determination methods that interactions require proteins to exhibit extensive local shape complementarity. Nevertheless, the precise mechanism that brings about interactions is poorly understood. The observed shape complementarity of docked proteins suggests we start with the geometric structures of individual proteins and search for a good local fit. This raises intriguing but hard questions about the relative importance of physical forces (e.g. van der Waals interactions, hydrogen bonds, ion pairs, etc.) and shape, particularly as to the precise meaning of shape when objects are not rigid.

The known structures of protein complexes form a benchmark for computational tools and the attempt to reassemble proteins to their observed, native configurations is referred to as *bound protein docking*. Even if we ignore physical forces and focus exclusively on shape, the high dimension of the search space makes this a difficult problem.

[⋆] All authors are supported by NSF under grant CCR-00-86013. VC, JR, and HE are also supported by a BGT Postdoc Program from Duke University. JR and HE are also supported by NIH under grant R01 GM61822-01. PA is also supported by NSF under grants EIA-01-31905 and CCR-02-04118 and by the U.S.-Israel Binational Science Foundation.

[⋆⋆] Departments of Computer Science and Biochemistry.

[⋆⋆⋆] Departments of Computer Science and Mathematics.

[†] Departments of Computer Science and Mathematics, and Raindrop Geomagic.

[‡] Departments of Biochemistry and Chemistry.

Following the policy of small steps, it thus makes sense to simplify the problem by assuming rigidity. The task in *rigid protein docking* is to find a motion that positions one rigid protein relative to the other into the correct docked configuration. The dimension of the search space is still high, three for translations plus three for rotations. Implementation of a fast and accurate rigid docking algorithm would allow for future methods that add the higher dimensionality of conformational changes seen in real docking problems. Possible approaches to flexibility include tolerance to collisions [8, 14] and pre-calculation of multiple residue conformations [16].

Prior work. Many different approaches have been taken to solving the rigid docking problem and we refer to several survey articles in the general area [7, 10, 13]. All of these methods consist of essentially two parts. First, one creates a scoring function that discriminates correctly docked conformations from incorrect ones. The scoring is based primarily on shape recognition but often includes electrostatics or hydrogen bonds. Because of the size of the search space and the number of atoms for each protein, simplifications or data reduction methods are often employed. Second, one creates a search algorithm that finds the correct solution using the scoring function. Many rigid protein docking algorithms based primarily on shape have been implemented using diverse approaches to search the space of rigid motions, including cube coverings, fast Fourier transforms, spherical harmonics, and geometric hashing. One major limitations in these methods is that they can yield anywhere from a few to thousands of false positives, incorrect configurations that have a higher score than the native configuration. Thus, a re-ranking of the docked configurations is usually implemented based on a wide variety of methods including solvation potentials, empirical atom-atom or residue-residue contact energies, optimal positioning of hydrogen bonds, etc. Bespamyatnikh et al. developed a shape-based docking algorithm and demonstrated that it correctly docks a diverse set of 25 protein complexes without any false positives [3]. This result was achieved using a scoring function that approximates the van der Waals interactions by counting pairs of atoms and by high-resolution sampling of the space of rigid motions. The major limitation of this method is the amount of computation time needed, with even a modest size problem taking a day on a cluster of 100 processors. This does not allow for facile experimentation or implementation of flexibility.

Local search. We envision a more efficient algorithm that first uses a coarse sampling in the space of rotations to generate a set of possible solutions with at least one not too far from the correct docking configuration. The second step is a rapid search for the solution using a local improvement method. Based on our previous results [3], the correct docking configuration will yield the highest score following this second step. A multistage local search method for rigid protein docking has been recently reported that mimics the physical events of protein binding [5]. Starting from complexes as much as 10Å root-mean-square-distance from the native configuration, the method guides protein docking, first with desolvation and electrostatics, then adding partial van der Waals potentials as the proteins come closer together. This energy funnel method was shown to work well with a set of eight different complexes, but appears computationally expensive.

Our results. As in [5], we do not explicitly address the generation of initial configurations in this paper. Instead the main question we pursue is the convergence radius of our local search heuristic. In other words, how far away from the native configuration can we start and still have a good chance to recover (a good approximation of) the native configuration? For the time being, we do not have any theoretical results and our approach to finding answers is purely experimental and restricted to the case of known structures of docked proteins.

We performed computational experiments using the barnase-barstar complex whose structure can be found in the protein databank [2]. Our findings show that the chances of recovering the correct, native configuration are about 80% provided we start with a configuration generated by a local perturbation with rotation angle at most 18° and translation distance at most 3.0Å. We extended the experiments to nineteen additional protein complexes and found that the bounds on the local perturbation are about the same and perhaps universal for protein complexes.

Outline. Section 2 introduces the geometric and algorithmic background used in this paper. Section 3 presents the local search heuristic. Section 4 describes the results of the computational experiments that test the performance of the heuristic. Section 5 summarizes our findings and points toward future directions.

2 Background and Definitions

In this section, we introduce the notation and the main geometric and algorithmic concepts used in the design of our local search heuristic.

Notation and assumptions. We use solid spheres to represent atoms and space-filling diagrams to model proteins as unions of such spheres. Writing a_i for the center and r_i for the radius of the i-th sphere, we let $\mathcal{A} = \{A_i = (a_i, r_i) \mid 1 \leq i \leq m\}$ be the set of spheres defining the first protein. Similarly, we let $\mathcal{B} = \{B_j = (b_j, s_j) \mid 1 \leq j \leq n\}$ be the set of spheres defining the second protein. Following the work of Bespamyatnikh et al., we say A_i and B_j *collide* if the two spheres overlap, and they *score* if they are within a pre-specified distance but do not overlap. Formally,

$$\text{score}(i,j), \text{collision}(i,j) = \begin{cases} 0, 1 & \text{if} \qquad\qquad\quad \|a_i - b_j\| < r_i + s_j, \\ 1, 0 & \text{if } r_i + s_j \qquad \leq \|a_i - b_j\| \leq r_i + s_j + \lambda, \\ 0, 0 & \text{if } r_i + s_j + \lambda < \|a_i - b_j\|, \end{cases}$$

where the constant is experimentally set to $\lambda = 1.5\text{Å}$ [3]. The *total score* and the *total collision number* are $\text{Score}(\mathcal{A}, \mathcal{B}) = \sum_{i,j} \text{score}(i,j)$ and $\text{Collision}(\mathcal{A}, \mathcal{B}) = \sum_{i,j} \text{collision}(i,j)$. Using a second constant, χ, we can now formally define the rigid docking problem as finding a rigid motion μ that maximizes the total score between $\mathcal{A}$ and $\mu(\mathcal{B})$ while keeping the number of collisions at or below χ. The second threshold is experimentally set to $\chi = 5$ [3]. Using van der Waals radii for the spheres, physics dictates that there are no collisions at all, but in order to compensate for measurement errors and other modeling inaccuracies, we allow for a small number of violations of that dictum. We make two assumptions on the geometric input data motivated by the application to organic molecules. To state them, let δ be the minimum distance between centers of any two spheres in $\mathcal{A}$ or in $\mathcal{B}$, and let $R_{\min}$ and $R_{\max}$ be the minimum and maximum radii of the spheres in these sets.

I. There are constants $c \leq C$ such that $R_{\max}/C \leq \delta \leq R_{\min}/c$.
II. The difference between the extreme radii satisfies $R_{\max} - R_{\min} < \lambda$.

We note that Assumption II is implied by Assumption I and $C - c < \lambda/\delta$, requiring that the two constants in I are not too different. Our algorithm crucially depends on Assumption I, and it makes use of Assumption II, but that dependence could be avoided. In the data retrieved from the protein databank [2], we observe $\delta = 1.18\,\text{Å}$ and get $c = 1.14$ and $C = 1.60$. In our experiments, we use only five different radii, between 1.348Å and 1.880Å, which clearly satisfy Assumption II.

Preprocessing. Consider a sphere B with radius $s = (R_{\min} + R_{\max})/2$. It scores with a sphere A_i iff its center b lies in the shell centered at a_i whose inner and outer radii are $r_i + s$ and $r_i + s + \lambda$, respectively. Consistent with the terminology introduced above, we define the *score* of B equal to the number of shells that contain b. Each sphere in $\mathcal{A}$ defines a shell, giving an arrangement of $2m$ (non-solid) spheres that decompose $\mathbb{R}^3$ into cells of constant score, as considered in [6]. The arrangement is useful for distinguishing desirable from undesirable positions for the sphere B but it has two drawbacks, namely it suggests regions and not specific positions, and not all spheres in $\mathcal{B}$ have radius s. We remedy both by replacing the shell around a_i by its mid-sphere,

$$S_i = \{x \in \mathbb{R}^3 \mid \|x - a_i\| = r_i + s + \frac{\lambda}{2}\}.$$

By Assumption II, S_i lies within the shell around a_i defined for each radius r in $[R_{\min}, R_{\max}]$. The mid-spheres intersect pairwise in circles and triplewise in points, the latter being the vertices of the arrangement. We use some of these vertices as target positions for the spheres B_j in $\mathcal{B}$, as illustrated in Figure 1. We compute and evaluate these vertices in a preprocessing step, which we now describe.

`Step 1.` Compute the set of vertices of the arrangement of mid-spheres.
`Step 2.` For each vertex u, compute the score of $B_u = (u, s)$ and the number of collisions between $B'_u = (u, s + \tau)$ and spheres in $\mathcal{A}$.
`Step 3.` Let $\mathcal{U}$ be the set of vertices u for which B'_u has zero collisions and there is no vertex $v \in \mathcal{U}$ nearby that dominates u in terms of scoring.

The constant τ used in `Step 2` will be discussed shortly. `Step 1` is greatly helped by Assumption I, which implies that each mid-sphere intersects only a constant number of other mid-spheres. It follows that the number of vertices is only O(m), and using the grid data structure of Halperin and Overmars [9] we find them in time O($m \log m$). Using the same data structure, we compute the scores and collision numbers of the spheres B_u and B'_u in time O($m \log m$).

The purpose of the vertices is to act as target locations for the spheres in $\mathcal{B}$. It thus makes sense to eliminate vertices u for which the enlarged sphere B'_u has non-zero collisions with spheres in $\mathcal{A}$. We use the experimentally determined constant $\tau = 0.2\,\text{Å}$ for the enlargement. Of the remaining vertices, we keep only the ones with locally maximum score. More specifically, we remove a vertex u for which there is a vertex v at distance at most $\lambda/2$ such that the set of spheres A_i scoring with B_u is a proper subset of the set scoring with B_v. Although the vertices do not observe a constant separation

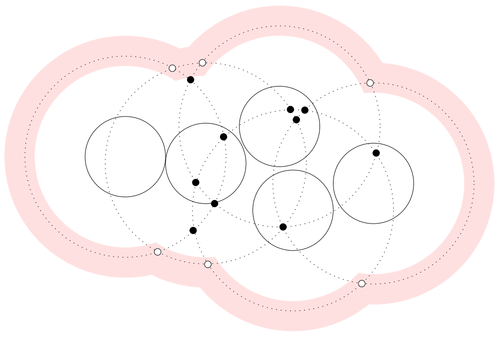

Fig. 1: The dotted circles represent mid-spheres. The vertices formed by the mid-spheres are black or white depending on whether a sphere centered at that vertex forms a near collision or not. The region of scoring, non-colliding positions is shaded.

bound, it is easy to prove from Assumption I that there are only a constant number of vertices within a constant distance from any point in space. We can therefore use the same grid data structure to implement `Step 3` within the same time bound as the first two steps. It follows that all three preprocessing steps together take time $\mathrm{O}(m \log m)$. We note that instead of one we may use several arrangements, each catering to a small range of radii of spheres in $\mathcal{B}$. In our implementation, we use five arrangements, one each for the five different radii in our data sets. As long as the number of arrangements is a constant, the running time is not affected by more than a constant factor.

Least square rigid motion. In the local search heuristic, we will repeatedly compute local rigid motions by solving a least-square optimization problem. An instance is given by a subset $G = \{g_1, g_2, \ldots, g_\ell\}$ of the vertices in $\mathcal{U}$, a subset $Z = \{z_1, z_2, \ldots, z_\ell\}$ of the centers of spheres in $\mathcal{B}$, and a bijection between G and Z specified by shared indices. The objective is to find a rigid motion μ that minimizes the sum of square distances, $\sum_{k=1}^{\ell} \|g_k - \mu(z_k)\|^2$. The problem of computing μ is known as the absolute orientation problem in computer vision. Every rigid motion can be written as a translation followed by a rotation about the origin. Assuming the centroid of the vertices is the origin, $\sum_{k=1}^{\ell} g_k = 0$, the translational component of the optimal motion necessarily moves the centroid $\bar{z} = \frac{1}{\ell}\sum_{k=1}^{\ell} z_k$ to the origin. It remains to compute the optimal rotation for the points $z_k - \bar{z}$, which reduces to solving a small eigenvalue problem. The matrix for this problem can be computed in time $\mathrm{O}(\ell)$ from the sets G and Z, using either the formalism of rotation matrices [15] or that of quaternions [11]. We remark that the reduction to an eigenvalue problem allows for more general correspondences between G and Z than bijections, and it can be modified to use a set of weights $W = \{w_1, w_2, \ldots, w_\ell\}$. In other words, we can compute in time $\mathrm{O}(\ell)$ the rigid motion $\mu = \mathrm{OptRM}(G, Z, W)$ that minimizes $\sum_{k=1}^{\ell} w_k \|g_k - \mu(z_k)\|^2$.

3 Local Search Heuristic

In this section, we describe the algorithm that locally improves the fit between the space-filling representations of two proteins. We begin by explaining the overall structure of the algorithm and follow up by detailing its loops.

High-level structure. Given sets of spheres $\mathcal{A}$ and $\mathcal{B}$, we aim at finding a local rigid motion that we can apply to $\mathcal{B}$ to improve the fit. By a local rigid motion we mean a rigid motion that is small, and we will be specific in Section 4 about how small. Here, we focus on the structure of the algorithm, which repeatedly solves one of two types of weighted least-square problems. The types are distinguished by the intended effect on the fit:

– *score-improving* instances are prepared and solved in the outer loop, and
– *collision-reducing* instances are prepared and solved in the inner loop.

The success of the algorithm crucially depends on how we define these instances. We follow two intuitions:

1. worthwhile target positions for spheres in $\mathcal{B}$ are collision-free and locally maximize the score;
2. an effective collection of target positions is approximately congruent to the configuration of corresponding sphere centers.

We satisfy the first intuition by using the vertices in $\mathcal{U}$ and the second intuition by limiting our attention to spheres and vertices that are near each other. Letting Z be the set of centers of spheres in $\mathcal{B}$ that have a vertex in their neighborhood, we define a bijection between Z and a subset G of the vertices; we refer to G as the set of *tentative goals*. Assuming a set of weights, W, and a bijection between Z and G, we can now describe the algorithm.

```
for $\#_{\text{outer}}$ times do
    prepare a score-improving instance of the least-square problem,
    compute $\mu = \text{OptRM}(Z, G, W)$, and let $\mathcal{B} = \mu(\mathcal{B})$
    while $\text{Collision}(\mathcal{A}, \mathcal{B}) > \chi$ and $\mu$ is not negligible do
        prepare a collision-reducing instance of the least-square
        problem, compute $\mu = \text{OptRM}(Z, G, W)$, and let $\mathcal{B} = \mu(\mathcal{B})$
    endwhile
endfor; return best fit encountered during the iteration.
```

We cannot prove, and indeed do not expect, that the algorithm always successfully finds a fit with sufficiently high score and small collision number. We therefore implement the algorithm with a constant limit, $\#_{\text{inner}}$, on how often the inner loop can be repeated. the outer loop $\#_{\text{outer}}$ times and return the best fit encountered during any of the iterations.

Score-improving outer loop. We describe how to prepare an instance of the weighted least-square problem that aims at improving the score of the fit. We define a distance threshold, D_j, for each sphere B_j in $\mathcal{B}$, and we let the corresponding tentative goal of B_j be a vertex u_j in $\mathcal{U}$ within distance D_j from b_j that maximizes the score as computed in the preprocessing step. If there is no vertex within distance D_j from b_j, the tentative goal of B_j is undefined. We let Z be the set of centers of spheres with tentative goals, and G the corresponding set of tentative goals. Finally, we set all weights to one.

It remains to describe how we choose the distance threshold $D_j = \min\{D, d_j\}$, where D is a general and d_j is an individual threshold. The general threshold depends on the current fit between $\mathcal{A}$ and $\mathcal{B}$ and gets smaller as the fit gets better. The individual threshold depends on whether or not B_j collides with a sphere in $\mathcal{A}$. Let $A_i \in \mathcal{A}$ be the sphere that minimizes $\|b_j - a_i\| - s_j - r_i$. If A_i and B_j are disjoint then we choose d_j small enough so that moving b_j within this limit avoids a collision. Otherwise, we choose d_j large enough to give B_j a chance to undo the collision:

$$d_j = \begin{cases} \|b_j - a_i\| - s_j - r_i & \text{if } \|b_j - a_i\| \geq s_j + r_i, \\ \|b_j - a_i\| + s_j + r_i + \lambda & \text{if } \|b_j - a_i\| < s_j + r_i. \end{cases}$$

When we compute the sphere A_i of B_j, we do not have to look farther than distance $D + s_j + R_{\max}$ from b_j. Since this is a constant, we can again use the grid data structure and explore the neighborhood of b_j in constant time. After collecting Z and G, we compute the optimal rigid motion, μ, and apply it to $\mathcal{B}$. We expect that the total score increases but there is no guarantee that this really happens. Simultaneously, the total collision number may also increase.

Collision-reducing inner loop. This brings us to the preparation of instances of the weighted least-square problem that aim at reducing the number of collisions. We distinguish spheres B_j with and without collisions. If B_j collides with at least one sphere in $\mathcal{A}$ then we find the closest collision-free position in $\mathcal{U}$ within distance $D + s_j$ from b_j and let it be the tentative goal of b_j. If B_j scores and has no collision then we encourage it to stay put by setting its tentative goal equal to its own center, b_j. Let Z contain the centers of all spheres B_j that receive tentative goals, and let G be the corresponding set of tentative goals. Finally, we use weights to counterbalance the usual relative abundance of collision-free spheres. Letting $F \subseteq Z$ be the subset of centers of collision-free spheres, we set the weight of a sphere center $z_k \in Z$ equal to

$$w_k = \begin{cases} 1 & \text{if } z_k \in F, \\ \omega / \|z_k - g_k\|^2 & \text{if } z_k \in Z - F, \end{cases}$$

with *weight factor* ω. Initially, we set $\omega = |F|/|(Z - F)|$, but if this leads to an increase in the number of collisions we adjust the weight factor as explained shortly. After computing Z, G, and W, we determine the optimal rigid motion, μ, and apply it to $\mathcal{B}$. We expect that the total number of collisions decreases but there is no guarantee that this really happens. Simultaneously, the total score may decrease although we counteract that tendency by including collision-free scoring spheres in the weighted least-square problem.

Adjusting the weight factor. If the rigid motion $\mu = \mathrm{OptRM}(Z, G, W)$ leads to an increase in the number of collisions, then we adjust the weight factor, ω, and redo the step in the inner loop. Recall that the rigid motion $\mu = \mathrm{OptRM}(Z, G, W)$ minimizes

$$\mathrm{Error}(\mu') = \sum_{z_k \in F} \|\mu'(z_k) - z_k\|^2 + \sum_{z_k \in Z-F} w_k \|\mu'(z_k) - g_k\|^2,$$

where the minimum is taken over all rigid motions μ'. We set

$$\omega_{\mathrm{new}} = \sum_{z_k \in F} \|\mu(z_k) - z_k\|^2 \Big/ \sum_{z_k \in Z-F} \left(1 - \frac{\|\mu(z_k) - g_k\|^2}{\|z_k - g_k\|^2}\right).$$

We then set $\omega = \omega_{\mathrm{new}}$ and repeat the computations unless $\mathrm{Collision}(\mathcal{A}, \mu(\mathcal{B})) \leq \mathrm{Collision}(\mathcal{A}, \mathcal{B})$. Experimentally, the adjustment is not always needed, and if necessary it seems to take at most four iterations. In our implementation, we limit the number of iterations to at most a constant $\#_{\mathrm{wf}}$. If the iteration ends with higher collision number for $\mu(\mathcal{B})$ than for $\mathcal{B}$ then we set $\mu = \mathrm{id}$.

Implementation and running time. We complete the description of the algorithm by defining the constants we used but have left unspecified. Most importantly, we use a distance threshold D in the preparation of various least-square problems. In the inner loop, we use $D = 2\text{Å}$ for the collision reducing instances. The situation is more complicated in the outer loop, where D depends on the current fit. Assuming $\mathrm{Score}(\mathcal{A}, \mathcal{B}) > 180$, we set

$$D = \begin{cases} 2\text{Å} & \text{if } \quad\quad \mathrm{Collision}(\mathcal{A}, \mathcal{B}) \leq 5, \\ 3\text{Å} & \text{if } \ 5 < \mathrm{Collision}(\mathcal{A}, \mathcal{B}) \leq 15, \\ 4\text{Å} & \text{if } 15 < \mathrm{Collision}(\mathcal{A}, \mathcal{B}) \leq 30. \end{cases}$$

We set $D = 4.5\text{Å}$ if $\mathrm{Collision}(\mathcal{A}, \mathcal{B}) > 30$ or $\mathrm{Score}(\mathcal{A}, \mathcal{B}) \leq 180$. The other constants we use are the upper bounds on the number of iterations, $\#_{\mathrm{inner}} = 20$, $\#_{\mathrm{outer}} = 13$, and $\#_{\mathrm{wf}} = 5$. In each case, the search for the tentative goal of a sphere B_j is limited to a constant size neighborhood. Assumption I implies that there are only a constant number of spheres and vertices to consider, which takes only constant time. The total amount of time needed for an iteration of either loop is therefore bounded by O(n). However, the constant number of spheres and vertices searched for each B_j can be rather large, warranting the implementation of a hierarchical search structure, e.g., the kd-tree, assisting the grid data structure. The total number of iterations is again at most a constant. It follows that the running time of the entire algorithm is O(n), not including the time for preprocessing, which has been accounted for in Section 2.

We have implemented the algorithm in C++ and refer to the software as LILAC. Depending on the protein complex, LILAC takes between half a minute and five minutes for the preprocessing step, and between three and ten seconds for a local search, on a PC with a Pentium III processor with clock speed of 929 MHz and memory of 600 MB. Our main focus is on accelerating the search since we are interested in applications in which the preprocessing time can be amortized over a potentially large number of local searches.

4 Experimental Results

We applied the software to a number of known protein-protein complexes. In this section, we describe these experiments and analyze the results we obtain.

Statement of questions. We test the effectiveness of the local search heuristic by running LILAC on a number of protein-protein complexes with known structure. For each such complex, we generate two sets of spheres, $\mathcal{A}_{\mathrm{nat}}$ and $\mathcal{B}_{\mathrm{nat}}$, one for each protein. We then perturb the native configuration by applying a local rigid motion π to the second set and use the local search heuristic on $\mathcal{A} = \mathcal{A}_{\mathrm{nat}}$ and $\mathcal{B} = \pi(\mathcal{B}_{\mathrm{nat}})$, obtaining another rigid motion, μ. Define $\mathcal{C} = \mu(\mathcal{B}) = \mu(\pi(\mathcal{B}_{\mathrm{nat}}))$. In the ideal case, μ is the inverse of π

and $\mathcal{C} = \mathcal{B}_{\text{nat}}$. In general, we measure how different $\mathcal{C}$ is from $\mathcal{B}_{\text{nat}}$ and use this information to distinguish successful from unsuccessful applications of the search heuristic. Letting $\mathcal{B}^*_{\text{nat}} \subseteq \mathcal{B}_{\text{nat}}$ be the set of scoring spheres in the native configuration, we define $\mathcal{C}^* = \mu(\pi(\mathcal{B}^*_{\text{nat}}))$ and compute the root-mean-square-distance between corresponding centers,

$$\text{RMSD}^* \;=\; \text{RMSD}(\mathcal{B}^*_{\text{nat}}, \mathcal{C}^*) \;=\; \sqrt{\frac{1}{\ell} \sum_{b_j \in \mathcal{B}^*_{\text{nat}}} \|b_j - \mu(\pi(b_j))\|^2},$$

where $\ell = |\mathcal{B}^*_{\text{nat}}|$. Recall the main question formulated in Section 1, which we can now rephrase to asking how much the native configuration can be perturbed so we still have a good chance to recover the complex using our local search heuristic. To approach this question, we let $\bar{b}^*$ be the centroid of the centers of the spheres in $\mathcal{B}^*_{\text{nat}}$ and we write each rigid motion, π, as a rotation about $\bar{b}^*$ followed by a translation. We measure the rotation and the translation separately, letting $\theta(\pi)$ be the angle of the rotation and $t(\pi)$ be the distance of the translation. Using oriented rotation axes, we may assume that both the angle and the distance are non-negative. A *local rigid motion* is one for which θ and t are small. We can now rephrase our main question again.

QUESTION A. What are the largest thresholds Θ and T such that for a perturbation with $\theta(\pi) \leq \Theta$ and $t(\pi) \leq T$ we have a good chance the local search heuristic recovers a good approximation of the native configuration?

We also address several related, more detailed questions, such as whether or not the success rate of the search heuristic is influenced by the angle between the rotation axis and the translation vector, or by the number of collisions in the perturbed starting configuration. The second main question addresses the variation over different complexes.

QUESTION B. Does the behavior or the search heuristic depend significantly on the protein complexes or are there universal thresholds T and Θ that apply to all or most complexes?

Statistical results of the experiments aimed at answering the two questions are now presented.

Convergence thresholds. Our first experiment explores the dependence of the performance of the local search heuristic on the size of the initial perturbation. We use the experimentally well-studied barnase-barstar complex (1BRS) as a test case. In this complex, the ribonuclease (barnase) exhibits extensive geometric surface complementarity docked to its natural protein inhibitor (barstar). The interaction surface is large, measuring about 800Å^2, and contains many of the features typically seen at protein interfaces, including a few hot-spot residues that contribute the majority of the interaction energy, electrostatic interactions, buried water molecules and the lack of a deep binding groove that characterizes small molecule binding. To define a perturbation, we select two directions $\mathbf{u}, \mathbf{v} \in \mathbb{S}^2$, an angle θ and a distance t. The thus specified perturbation first rotates by θ around the oriented axis passing through $\bar{b}^*$ in the direction $\mathbf{u}$ and second translates by adding $t\mathbf{v}$. We note that a uniform sampling of directions, angles, and distances favors small over large rigid motions. While generally undesirable, this bias is acceptable in our application, which focuses on local rigid motions. We sample the space of perturbations using 32 directions (defined by the vertices and face normals of the regular icosahedron), 10 angles ($0^\circ, 3^\circ, \ldots, 27^\circ$), and 10 lengths ($0\text{Å}, 0.5\text{Å}, \ldots, 4.5\text{Å}$). This gives a total of about one hundred thousand perturbations or trials. For each trial, we compute the score, collision number, RMSD^* of the native, perturbed, and computed configurations. We declare a trial successful if the computed configuration has score at least some fraction of the native score and small RMSD^*. To be specific, we consider the computation of $\mathcal{C}$ from $\pi(\mathcal{B})$ a *success* if

$$\begin{aligned} \mathrm{Score}(\mathcal{A}, \mathcal{C}) &\geq p \cdot \mathrm{Score}(\mathcal{A}, \mathcal{B}_{\mathrm{nat}}) \text{ and} \\ \mathrm{RMSD}(\mathcal{B}^*_{\mathrm{nat}}, \mathcal{C}^*) &\leq \varepsilon, \end{aligned}$$

where we use $p = 0.9$ and $\varepsilon = 1.5\text{Å}$. We note that $\mathrm{Collision}(\mathcal{A}, \mathcal{C}) \leq \chi$ is understood since this is the threshold used in the search heuristic. The results of this experiment are displayed in Figure 2. Using these results, we set $\Theta = 18^\circ$ and $T = 3.0\text{Å}$.

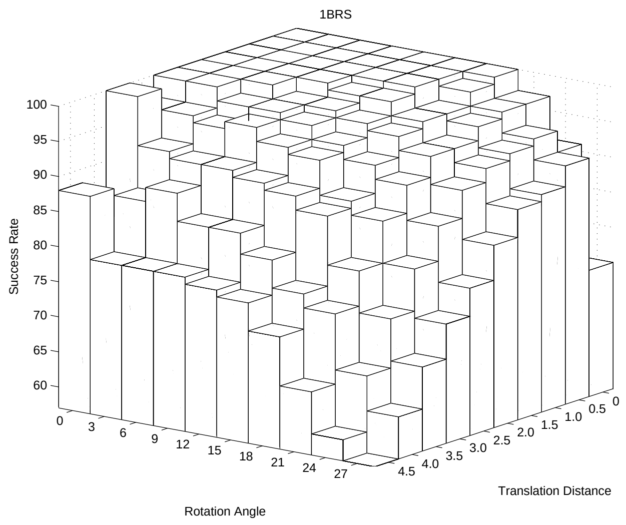

Fig. 2: The success rate of the local search heuristic as a function of the rotation angle and translation distance. Each prism corresponds to a pair θ, t and represents 1,024 trials. Of course for $\theta = t = 0.0$ all rigid motions are the same. The height of a prism gives the percentage of successes within the corresponding collection of trials.

For convenience, we introduce a norm to measure the size of a perturbation, $\|\pi\| = [(\theta(\pi)/\Theta)^2 + (t/T)^2]^{1/2}$. We use it to display the experimental results with graphs of one-dimensional functions, such as in Figure 3. We looked into the question of whether or not the search heuristic performs better for some perturbations than others. Comparing the RMSD^* caused by rotations versus by translations it seems that the algorithm is about twice as sensitive to the rotational part of the perturbation. We did not find any correlation between the success rate and the angle formed by the directions defining the rotation and the translation. We also did not find any correlation between the success rate and the number of collisions in the initial, perturbed configuration.

Universality of convergence thresholds. We repeated the same computational experiment for nineteen additional protein complexes to test the applicability of our algorithm to a diverse set of protein-protein complexes. The types of interactions tested include some that fall into commonly observed classes such as protease-inhibitor complexes or antibody-antigen complexes, and some that are one-of-a-kind complexes, often with broad featureless interfaces that have historically been harder to dock by computational methods. We used the same thresholds, $\Theta = 18^\circ$ and $T = 3.0\text{Å}$ to define the norm of a perturbation and $p = 0.9$ and $\varepsilon = 1.5\text{Å}$ to distinguish successful from unsuccessful trials. As shown in Figure 3, the results exhibit the same trend as those for 1BRS. The search heuristic was more successful for twelve and less successful for seven of the additional complexes. The answer to Question B is therefore that the thresholds Θ and T appear to be universal for protein complexes. Of course, this applies only to bound, rigid structures.

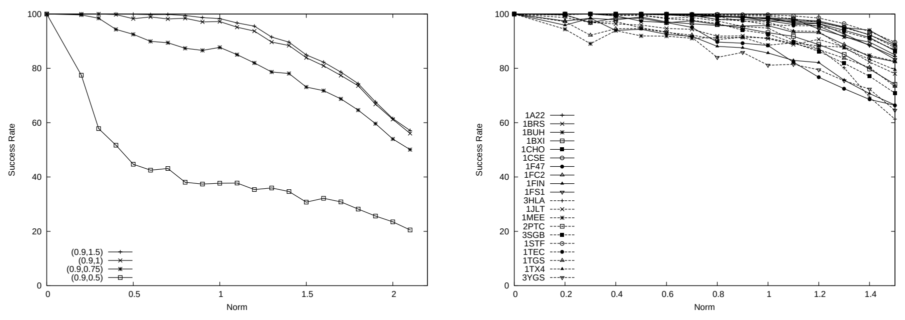

Fig. 3: Left: comparison of success rates obtained for different root-mean-square thresholds $\varepsilon = 0.5, 0.75, 1.0, 1.5\text{Å}$. Right: comparison of success rates for different complexes, while setting $p = 0.9$ and $\varepsilon = 1.5\text{Å}$.

5 Discussion

The main contribution of this paper is a local improvement algorithm for rigid protein docking and an experimental study of its performance. Comparison of our algorithm to other docking methods is beyond the scope of this paper as we have presented here only a local improvement step, not an overall docking method. The results are encouraging but we hope to improve them through additional tests and fine-tuning of the various steps in the algorithm. We plan to use this algorithm to accelerate the exhaustive search method of Bespamyatnikh et al. [3] using a coarse sampling of rigid motions that relates to the observed convergence radius. We also plan to use the algorithm in combination with more sophisticated strategies for coarse sampling that are based on protein shape measurements [1].

The most important new research direction is the inclusion of flexibility in protein docking. This can be approached in a variety of ways, including the use of rigidity analysis of protein structures [12], normal mode analysis [4], and ensembles of alternative configurations [16]. The experimental results reported in [3] suggest that a successful prediction of protein interaction is possible based on geometric criteria only but requires a fine sampling of the space of rigid motions and careful accounting of details. Any method incorporating flexibility will sacrifice some of the specificity we find in rigid docking and will need to balance speed and specificity.

References

1. P. K. AGARWAL, H. EDELSBRUNNER, J. HARER AND Y. WANG. Extreme elevation on a 2-manifold. *In* “Proc. 20th Ann. Sympos. Comput. Geom., 2004”, 357–365.
2. H. M. BERMAN, J. WESTBROOK, Z. FENG, G. GILLILAND, T. N. BHAT, H. WEISSIG, I. N. SHINDYALOV AND P. E. BOURNE. The protein data bank. *Nucleic Acid Res.* **28** (2000), 235–242.
3. S. BESPAMYATNIKH, V. CHOI, H. EDELSBRUNNER AND J. RUDOLPH. Accurate bound protein docking by shape complementarity. Manuscript, Dept. Comput. Sci., Duke Univ., Durham, North Carolina, 2003.
4. B. R. BROOKS AND M. KARPLUS. Harmonic dynamics of proteins: normal modes and fluctuations in bovine pancreatic trypsin inhibitor. *Proc. Natl. Acad. Sci.* **80** (1983), 3696–3700.
5. C. J. CAMACHO AND S. VAJDA. Protein docking along smooth association pathways. *Proc. Natl. Acad. Sci.* **98** (2001), 10636–10641.
6. V. CHOI AND N. GOYAL. A combinatorial shape matching algorithm for rigid protein docking. *To appear in* “Proc. 15th Ann. Sympos. Combin. Pattern Matching, 2004”.
7. A. H. ELCOCK, D. SEPT AND J. A. MCCAMMON. Computer simulation of protein-protein interactions. *J. Phys. Chem.* **105** (2001), 1504–1518.
8. J. FERNANDEZ-RECIO, M. TOTROV AND R. ABAGYAN. Soft protein-protein docking in internal coordinates. *Protein Sci.* **11** (2002), 280–291.
9. D. HALPERIN AND M. H. OVERMARS. Spheres, molecules, and hidden surface removal. *Comput. Geom. Theory Appl.* **11** (1998), 83–102.
10. I. HALPERIN, B. MA, H. WOLFSON AND R. NUSSINOV. Principles of docking: an overview of search algorithms and a guide to scoring functions. *Proteins* **47** (2002), 409–443.
11. B. K. P. HORN. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Amer. A* **4** (1987), 629–642.
12. D. J. JACOBS, A. J. RADER, L. A. KUHN AND M. F. THORPE. Protein flexibility predictions using graph theory. *Proteins* **44** (2001), 150–165.
13. J. JANIN AND S. J. WODAK. The structural basis of macromolecular recognition. *Adv. Protein Chem.* **61** (2002), 9–73.
14. F. JIANG AND S.-H. KIM. “Soft-docking”: matching of molecular surface cubes. *J. Mol. Biol.* **219** (1991), 79–102.
15. W. KABSCH. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallogr. Sect. A* **34** (1978), 827–828.
16. D. M. LORBER, M. K. UDO AND B. K. SHOICHET. Protein-protein docking with multiple residue conformations and residue substitutions. *Protein Sci.* **11** (2002), 1393–1408.