Computing the Arrangement of Circles on a Sphere, with Applications in Structural Biology

Frédéric Cazals, Sebastien Loriot

Computing the Arrangement of Circles on a Sphere, with Applications in Structural Biology *

Frédéric Cazals † Sébastien Loriot †‡

October 30, 2008

Abstract

Balls and spheres are the simplest modeling primitives after affine ones, which accounts for their ubiquitousness in Computer Science and Applied Mathematics. Amongst the many applications, we may cite their prevalence when it comes to modeling our ambient 3D space, or to handle molecular shapes using Van der Waals models. If most of the applications developed so far are based upon simple geometric tests between balls, in particular the intersection test, a number of applications would obviously benefit from finer pieces of information.

Consider a sphere S_0 and a list of circles on it, each such circle stemming from the intersection between S_0 and another sphere, say S_i. Also assume that S_i has an accompanying ball B_i. This paper develops an integrated framework, based on the generalization of the Bentley-Ottmann algorithm to the spherical setting, to (i)compute the exact arrangement of circles on S_0 (ii)construct in a single pass the half-edge data structure encoding the arrangement induced by the circles (iii)report the covering list of each face of this arrangement, i.e. the list of balls containing it. As an illustration, the covering lists are used as the building block of a geometric optimization algorithm aiming at selecting diverse conformational ensembles for flexible protein-protein docking.

*This work is partially supported by the IST Programme of the 6th Framework Programme of the EU as a STREP (FET Open Scheme) Project under Contract No IST-006413 - ACS (Algorithms for Complex Shapes with certified topology and numerics) - http://acs.cs.rug.nl/

†INRIA, BP93, 06902 Sophia Antipolis cedex, France, Firstname.Lastname@sophia.inria.fr, http://www-sop.inria.fr/geometrica/team/

‡IMB - Université de Bourgogne

1 Introduction

1.1 Modeling with Balls and Spheres

Balls in Geometric Modeling. Balls and spheres are the simplest modeling primitives after affine ones. This accounts for their ubiquitousness in computer science and applied mathematics, especially when it comes to modeling our ambient 3D space. Given a known object, balls can be assembled to provide hierarchical approximations. In computer graphics, such approximations have been used to perform multi-scale visualization [24], while in robotics, they have been instrumental to perform efficient collision detection [1]. Balls are also key in inferring geometric and topological informations of an object known from sample points. For example, under mild sampling assumptions, mimicking the medial axis transform of a sampled surface allows one to reconstruct the domain enclosed within this surface [4]. From a more geometric perspective, a number of quantities can be inferred from balls centered at the sample points. As an illustration, one may cite the so-called boundary measure of a point cloud, from which singular points and sharp features of the sampled model can be estimated [9]. Additional scenarios involving balls can be found across the computer science literature in general, and we refer for example the reader to chapter 11.12 of the Visionbib bibliography project at `http://www.visionbib.com/bibliography/describe482.html` for numerous pointers.

Balls in Structural Biology. Balls are also central in structural biology, especially when it comes to manipulating Van der Waals (VdW) models. In a VdW model, each atom is represented by a ball whose radius depends on the atom type and its environment. Such models are instrumental to investigate implicit solvent models for electrostatics [19], to compute energies and statistical potentials [26], to define and model geometric shapes for docking [25], and to define molecular surfaces [10]. This latter aspect has been of special interest in computational geometry, since two of the most widely used molecular surfaces read directly from union of balls [2]. Given a collection of atomic balls, its VdW surface is the boundary of its VdW balls, while its Solvent Accessible Surface (SAS) is the boundary of the union of balls after expanding their radii by $r_w = 1.4\text{Å}$ to account for a solvation layer.

Arrangements of Circles on a Sphere. Most of the applications aforementioned are based upon simple geometric tests between balls, in particular the intersection test. A number of them, though, benefit from finer pieces of information. To see which, consider a sphere S_0 and a list of circles on it, each such circle stemming from the intersection between S_0 and a sphere S_i. The arrangement of circles on S_0 is the partition of S_0 into regions whose interior is connected. Assuming that each sphere S_i is associated a ball B_i, the *covering list* of a face of the arrangement is the list of balls that contain it, and the *multiplicity* of the face is the size of the covering list. Since our interest comes from structural biology, let us mention briefly two problems directly concerned with arrangements and covering lists.

The first one is the problem of encoding multi-body contacts between an atom and its neighbors. In representing a molecule as a collection of balls, we recalled above the importance of molecular surfaces. For a given atomic sphere within a molecule, consider the arrangement induced by the intersection circles with neighboring atoms. The contribution of this atom to the molecular surface, say the VdW or the SAS surface, consists of the cells of null multiplicity—the surface features the boundary of the union of balls i.e. the exposed spherical caps. Fig. 1 features the cumulative area (over all atoms of a complex) as a function of the multiplicity. Notice in particular that the surface area of the SAS corresponds to a mere 4.5% of the total computed surface area. Given that all previous studies overlooked faces of multiplicity ≥ 1, arrangements of circles hold great promises to refine our understanding of inter-atomic multi-body contacts.

The second problem is related to flexible docking. Recall that docking is concerned with the prediction of the geometry of a complex, say a protein-protein complex, from those of its constitutive partners. For partners undergoing significant conformational changes upon binding, a number of docking algorithms resort to pre-generated conformations of the partners called *conformers*. To maximize the chances of a docking experiment, one naturally wishes to deal with conformers providing a sampling of the conformational space as good as possible. Given a large number of conformers, we shall present in section 5 an algorithm based on arrangements of circles on a sphere to select *diverse* conformational ensembles. One such ensemble is presented on Fig. 2.

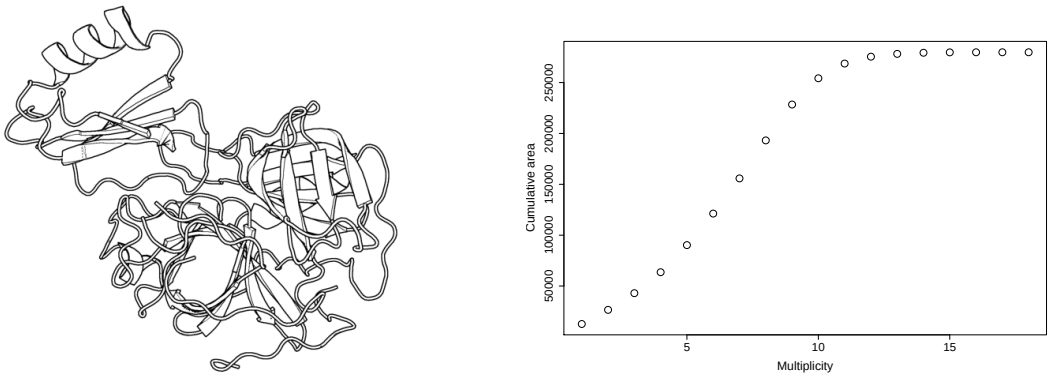


Figure 1: Modeling atomic multi-body contacts—PDB code `1ACB`. **Left**: A protein-protein complex with two partners; **Right**: Cumulative surface area, over all atoms of the complex, as a function of the multiplicity of the cells. Multiplicity 0 corresponds to the SAS surface, and accounts for a mere 4.5% of the cumulative area. See text for details.

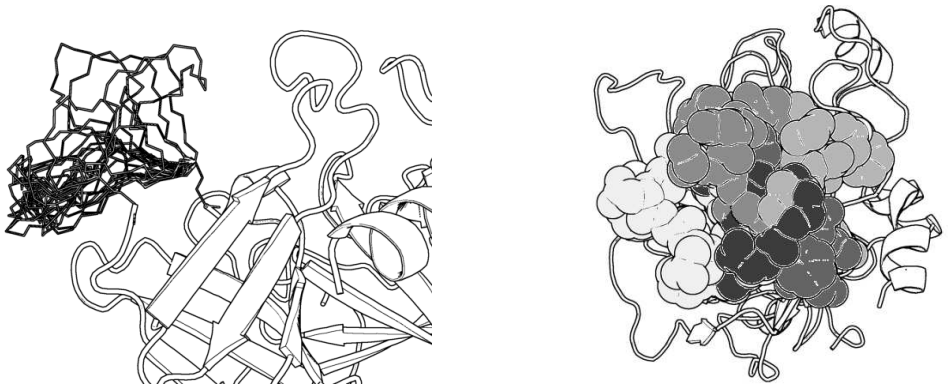


Figure 2: Conformers and flexible protein - protein docking —PDB code `1BTH`. **Left**: Side view of rigid template (cartoons), together with the backbone of a collection of 20 conformers (polylines); **Right**: Top view of five shallowly intersecting conformers (VdW representation).

1.2 Contributions and Paper Overview

Arrangement of Circles on a Sphere and Related Problems. We consider a collection of n balls $B_{i,i=1,\ldots,n}$ intersecting a given ball B_0. The sphere associated to ball B_i is denoted S_i, and the intersection circle, if any, between S_0 and S_i is denoted C_i. For a collection of n circles on sphere S_0, recall that the *arrangement* induced by the circles is the decomposition of S_0 into faces, circular arcs and vertices. In the following, we shall be concerned by the following two problems: reporting the arrangement on S_0, and reporting for each cell of this arrangement its covering list—the list of balls that contain it.

Contributions and Paper Overview. The punchline of the paper is to present an integrated framework allowing one to construct the arrangement of circles on a sphere and to compute a compact representation of the covering lists called the *covering tree*. Thus, we make three contributions.

First, we present the first effective algorithm to compute the exact arrangement of circles on a sphere—section 2. The algorithm, which generalizes the Bentley-Ottmann [5] sweep to the spherical setting, uses a segregation of events specific to circles, and gathers events into a data structure called the *event site*, so as to handle degeneracies. In passing, we mention the fact the complexity of the algorithm involves faces bounded by two arcs (the so-called lenses and lunes).

Second, we present the construction of the half-edge data structure (HDS) storing the arrangement, a construction performed on the fly during the sweep process, and using Union-Find—section 3. As an application of this construction, we present a calculation of the covering tree of the arrangement—section 4. Finally, section 5 presents an application of covering lists to flexible docking.

Position with Respect to Previous Work. For the calculation of the arrangement, the only effective alternative to our Bentley-Ottmann variation consists, to the best of our knowledge, of computing a

trapezoidal map, based upon an explicit perturbation of the spheres so as to get rid of degeneracies [16, 12]. Our strategy cumulates several advantages. First, the computation of the arrangement is exact, exactness being an obvious route to warrant robustness. (We note that while exactness is not a prerequisite in computational structural biology, it might be a must in other applications.) Second and most importantly, running Bentley-Ottmann on the sphere provides a direct route for the construction of the arrangement and of the covering lists. While the classical way to obtain the arrangement consists of merging the cells of a trapezoidal decomposition [6], we perform it on the fly using union-find—an original alternative to the best of our knowledge.

As a general comment, notice that using the stereographic projection, the arrangement of circles on a sphere could be obtained from an arrangement of circles and lines in the plane [28]. However, the spherical setting alleviates the calculations within predicates. For example, the z coordinates of the so-called critical points have rational coordinates on the sphere [7], while the x and y coordinates of their counterparts in the plane are algebraic numbers. In passing, we refer the reader to our companion paper [7] for the predicates and constructions required by the algorithms presented herein.

2 Bentley-Ottmann on a Sphere

2.1 Algorithm

Sweeping θ-monotone Circular Arcs on a Sphere. To report line-segments intersections in the plane [11], the Bentley-Ottmann (BO) algorithm [5] requires two data structures which are an event queue $\mathcal{E}$ featuring the line segment endpoints and intersection points, and another sorted data structure $\mathcal{V}$ providing the ordering along the vertical sweep line. The data structure $\mathcal{V}$ features segments intersected by the sweep-line, which are pairwise checked for intersection when they become adjacent in $\mathcal{V}$.

Consider a sphere S_0. For circles on that sphere, the extension of the BO algorithm consists of sweeping the sphere with a meridian M_θ, using cylindrical coordinates. This meridian is anchored at the poles and revolves around the sphere while θ spans the interval $(0, 2\pi]$. The sweep process handles θ-monotone circular arcs, which are stored in the vertical ordering $\mathcal{V}$. To specify these arcs from the circles, we first introduce the following classification of circles:

Definition 1 *A circle C_i is called* polar *if it goes through a single pole of S_0;* bipolar *if it goes through the two poles of S_0;* threaded *if the intersection point between the plane of C_i and the z-axis belongs to the open disk bounded by C_i;* normal *otherwise.*

The tangency point(s) between M_θ and a circle generates one or two θ-monotone circular arcs. More precisely, a normal circle yields two θ-monotone circular arcs; a polar circle defines one such arc, the second one reducing to a point (its pole). From now on, these θ-monotone circular arcs are just called arcs. During the sweep process, an arc is *active* while it is in $\mathcal{V}$. For homogeneity, we consider that a threaded circle defines an arc which is always active.

Event Sites. To accommodate degeneracies, the event queue $\mathcal{E}$ stores *event sites*, which are defined from two types of *event points* (*critical points* and *intersection points*) as follows.

The *normal critical points* (θ_c, z_c) associated to a normal circle are the two points where the meridian intersects it in a single point. Amongst these points, the *start* point (respectively *end* point) is the one where the intersection between the circle and the meridian starts (respectively ends). Consider a polar circle. Denoting z_p the z coordinate of the corresponding pole, the *polar critical points* are defined as the pairs (θ_S, z_p) and (θ_E, z_p), with θ_S and θ_E the values such that the meridian is tangent to the circle. The start point is distinguished from the end point as for normal circles. This extension carries to bipolar circles, yielding *bipolar critical points*, the θ values being those where the circle is included in the plane containing the meridian—the z coordinate is irrelevant. For such a circle, the start (respectively end) critical point is the one corresponding to the smallest (respectively largest) value of θ. An *intersection point* between two arcs is either a *tangency point*, if the arcs are tangent in their interior, or a *crossing point*, if the arcs intersect transversely in their interior.

The event points just described are encapsulated within *events*, each featuring either a circle and a start/end tag, or a pair of arcs and a first/second tag to define the corresponding event point. Notice that the qualifiers introduced above for event points are inherited by events. Equipped with these notions, events are gathered into an *event site* as follows:

Definition 2 *A* normal event site *is a collection of events with the same event point, partitioned into the following three data structures:*
– One list $\mathcal{F}$ for normal end (finish!) events, sorted by increasing circle radius.
– One list $\mathcal{S}$ for normal start events, sorted by decreasing circle radius.
– One list $\mathcal{CT}$ which contains the crossing events and tangency events. The restriction of $\mathcal{CT}$ to crossing (tangency) events is denoted $\mathcal{C}$ ($\mathcal{T}$).

Note that a *(bi)polar event site* associated to one (bi)-polar event is the event itself. A collection of event sites corresponding to distinct event points are sorted using the lexicographic order on cylindrical coordinates. Difficulties arise due to collisions between events at a given θ value and/or at the poles:

Definition 3 *Event sites of different types with same θ value are ordered as follows:*

$$e_{P_{ep}} < e_B < e_N < e_{P_{sp}}$$

where $a < b$ means a occurs before *b; e_B and e_N stand for respectively* bipolar *and* normal event site*; $e_{P_{sp}}$ and $e_{P_{ep}}$ stand for respectively* start *and* end *polar event sites. Moreover, among polar start (respectively end) event sites at the same pole, the one whose associated circle is of largest (respectively smallest) radius occurs first.*

The fact that this total order is compatible with the construction of the arrangement and the calculation of the covering lists will become apparent later.

2.2 Complexity Analysis

To state the algorithm complexity, one needs considerations on the structure of the arrangement induced on S_0 by the intersection circles. A vertex of this arrangement is either an intersection point, or a critical point. Denoting n the number of circles, and k (respectively v) the number of intersection points (respectively vertices), we have $v = O(n + k)$. An edge is an arc in-between two vertices. A face is a region of S_0 whose interior is connected. If the interior of a face is not simply connected, then the face contains hole(s). Using Euler's relationship on simply connected regions and holes of the decomposition of S_0 induced by the intersection circles, one can prove the following lemma [8]:

Lemma 1 *Denote v and e the number of vertices and edges of an arrangement of n circles on the sphere, and let l stand for the number of faces bounded by exactly two edges. One has $e \leq 3(v - 1) + l$.*

Notice this lemma involves the number of faces bounded by exactly two edges, also known as *lenses* (convex faces) and *lunes* (concave faces). For n circles of arbitrary radii in the plane, this number is known to be $O(n^{3/2+t})$, for any $t > 0$, where the constant of proportionality depends on t [3]. Using this lemma, the following theorem is proved in [8]:

Theorem 1 *Reporting the k intersection points of a family of n circles on a sphere requires $O(n)$ storage and has $O((n + k + l) \log n)$ complexity.*

2.3 Qualifying Arcs and Half-edges

Having sketched the BO algorithm, we present a classification of arcs and half-edges, to be used in sections 3 and 4.

Qualifying an Arc w.r.t. Circles: Upper or Lower. A threaded circle is called *north threaded* if its center has a z coordinate larger than or equal to that of the center of S_0, and *south* threaded otherwise. In the same way, a polar circle containing the north pole is called a north polar circle, and south polar circle otherwise. Considering the two arcs of a normal circle, we call them the *upper* and the *lower* arcs w.r.t. the z axis. A north (respectively south) polar circle has a single non trivial arc, which is *lower* (respectively *upper*). Similarly, we consider that a north (respectively south) threaded circle has a single arc which is *lower* (respectively *upper*).
Qualifying a Half-edge w.r.t. an Arc: Upper or Lower. For all but bipolar circles, to each arc A_i, we associate two active half-edges qualified w.r.t. the unique intersection between the meridian and the arc: The *upper* (respectively *lower*) half-edge associated to A_i is the half-edge leaving to its left the portion of S_0 lying above (respectively below) A_i. We now address particular cases: (i)for a threaded

circle which is not intersected by any other circle, at the end of the sweep process, the source and the target of the half-edges associated to its arcs are fixed to a common null vertex (ii)for a bipolar circle, no arc is defined. Half-edges are created on the fly at a bipolar event, so as to handle, if any, intersections with active arcs.

Qualifying a Half-edge w.r.t. a Circle: Inner or Outer. For a circle which is not a great circle, consider the spherical cap of S_0 of smallest area induced by this circle. If a half-edge on this circle induces this cap, it is called *inner*, and *outer* otherwise. For a great circle, we break the tie and call the cap of smallest area that induced by inner half-edges, that is: (i) if the great circle is bipolar, an half-edge is *inner* if it induces the spherical cap swept by M_θ between its θ_S and θ_E associated values, and *outer* otherwise; (ii) if the great circle is threaded, an half-edge is *inner* if it induces the spherical cap containing the north pole, and *outer* otherwise.

The relationship between the qualifiers upper/lower and inner/outer of half-edges is illustrated on Fig. 3, and we have:

Observation 1 *The spherical cap of smallest (respectively largest) area bounded by a circle is described by inner (respectively outer) half-edges of the circle. The lower/upper half-edge of an upper (respectively lower) arc of a all but bipolar circle is always inner/outer (respectively outer/inner).*

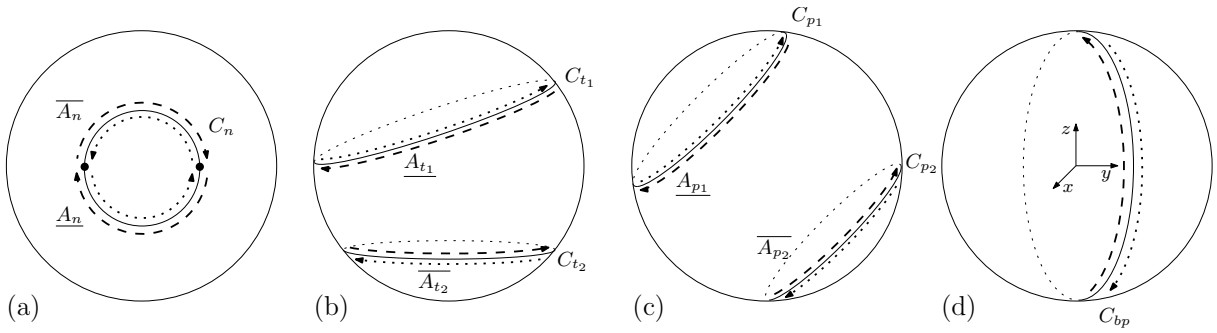


Figure 3: Qualifying arcs and half-edges: upper and lower arcs are respectively denoted as $\overline{A}$ and $\underline{A}$; inner and outer half-edges are respectively represented using dotted and dashed lines; (a) normal circle C_n and its upper and lower arcs; (b) north threaded circle C_{t_1} defining a lower arc, and south threaded circle C_{t_2} defining an upper arc; (c) north polar circle C_{p_1} defining a lower arc, and south polar circle C_{p_2} defining an upper arc; (d) bipolar circle C_{bp} inducing half-edges (no arc is defined). The θ-coordinate of the start point of C_{bp} lies in $(0; \pi]$.

3 Constructing the Arrangement During the Sweep Process

In this section, we develop the topological operations required to construct the arrangement induced by the circles and stored it into an extended half-edge data structure [18] (HDS) possibly featuring holes in faces.

3.1 Describing the Arrangement

Arcs and Half-edges. The vertices of the HDS correspond to event points, its edges are arcs delimited by such vertices, while the faces are the regions of S_0 bounded by vertices and edges. Recall that a face is a two-dimensional region whose interior is connected, and that half-edges are oriented so as to leave the interior of the face to its left. Over the course of the sweep algorithm, an half-edge is created when its *first* vertex is fixed, and remains *active* until its *second* vertex is also fixed. Following classical terminology, notice the first vertex may be the source or the target vertex of the half-edge. In the following, stitching two half-edges should be understood as fixing the next pointer of one half-edge to the second.

Faces and Holes. To describe the arrangement, we use sequences of connected half-edges (SCH), such a sequence being called *closed* if its topology is that of a circle. Two active half-edges associated to arcs in $\mathcal{V}$, with respect to the ordering along $\mathcal{V}$, are termed *adjacent* if their arcs are adjacent in $\mathcal{V}$, and if they

bound the same segment along the meridian M_θ. (Out of the four pairs of half-edges associated to two consecutive arcs along $\mathcal{V}$, a single pair corresponds to adjacent half-edges.) A *face* of the arrangement is represented by a collection of closed SCH. Each such sequence is called a *Connected Component of the Boundary* or CCB. A CCB is oriented and always induces a contractible region on the sphere S_0. The set of all CCB describing a face can be split into two categories: one CCB called *principal* defines the spherical cap containing the face; the remaining ones define *holes* in the face. See Fig. 4.

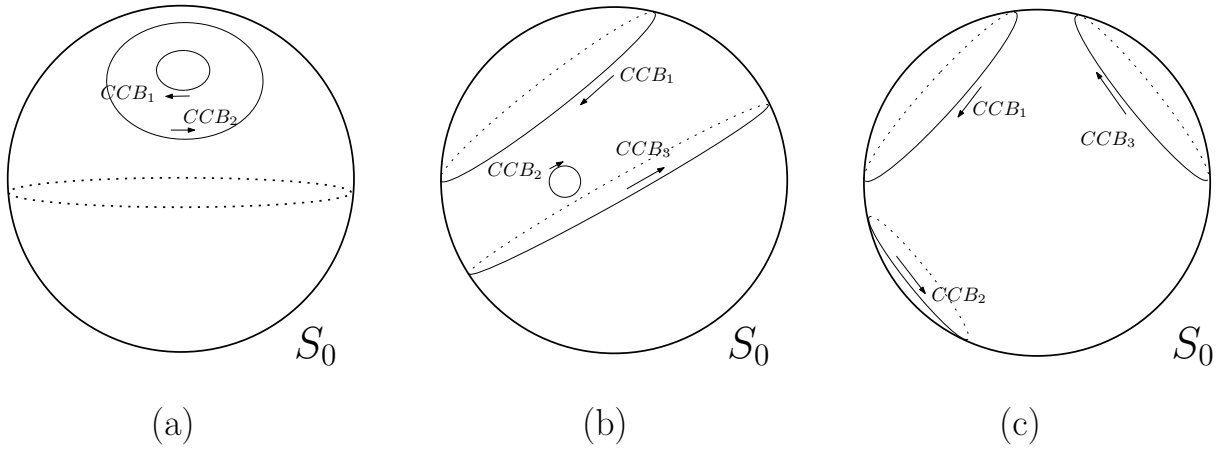


Figure 4: Defining a face with a set of CCB. Each arrow represents a CCB. In the four cases, only one face is defined by the CCB represented.

3.2 Handling Half-edges

The initialization of $\mathcal{V}$ consists of finding the ordering of arcs along M_θ for $\theta = 0^+$. To be able to define the faces cut by M_{0^+}, to each arc in $\mathcal{V}$ after initialization, we attribute a pair of opposite half-edges with one *null* vertex implicitly representing the intersection between M_{0^+} and the corresponding arc. This collection of half-edges is stored in a list H_0. At the end of the sweep process, we have a one-to-one correspondence between half-edges of arcs in $\mathcal{V}$ and the sequence of half-edges in H_0, so that a merge can be performed.

To describe the operations underwent by half-edges at a normal event site, consider the arcs involved in the three lists of a normal event site, in the neighborhood of the corresponding event point, say p. Two types of operations need to be performed. The first type, to the left (respectively to the right) of p, consists of stitching each pair of adjacent half-edges of the arcs involved, before (respectively after) updating $\mathcal{V}$. The second type consists of stitching the half-edges of the highest and lowest arcs along $\mathcal{V}$. These operations are straightforward, and the details are left to the reader.

Operations involving polar and bipolar circles are also straightforward: Following Def. 3, half-edges incident to a pole are stitched while rotating around that pole; When handling a bipolar event, from the north to the south pole, all active arcs in $\mathcal{V}$ are intersected and their half-edges are updated accordingly. If the bipolar circle goes through a point corresponding to an event site, then the event site is handled at the same time taking into account the half-edges of the bipolar circle.

3.3 Building the Faces

Building the faces of the arrangement requires two steps, namely creating CCB and creating faces by joining the CCB. To do so, we resort to two independent union-find algorithms.

Creating a CCB. A SCH becomes a CCB whenever stitching two half-edges creates a topological circle. As the intersection between a CCB and the meridian may feature several connected components (see Fig. 5), we merge SCH using union-find, which requires endowing each half-edge with a pointer to a master half-edge—called the *CCB master*. Stitching two half-edges is accompanied by a union of their CCB masters (if different), and a CCB appears when two half-edges being stitched already have the same CCB master.

Creating a Face. A face consists of a principal CCB and of CCB defining holes. We construct faces using union-find on SCH, which requires a union-find pointer called the *face master*. While closing a SCH,

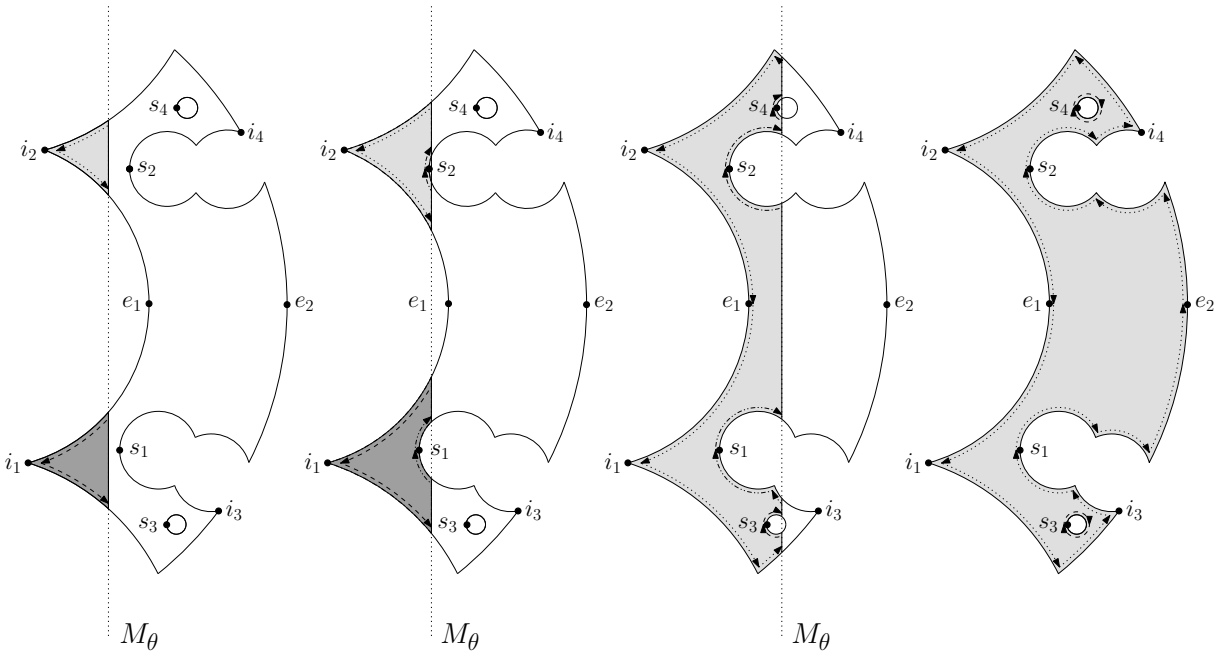


Figure 5: Illustration of Union-Find: one color per face, one line-style per CCB. Two SCH get created at events corresponding to i_1 and i_2, together with two faces. When the meridian reaches start point s_1, (and s_2) a new SCH is created and it contributes to the definition of an existing face. The handling of the event corresponding to the end point e_1 results on the one hand, and on the other hand to the union of two CCB masters and to the union of two face masters: The three remaining CCB describe a unique face. Thanks to this operation, the CCB started with points s_3 and s_4 also describe that face. Finally, the unions of CCB masters at events associated to i_3 and i_4 allows to detect the closure of the CCB at point e_2.

the resulting CCB is principal if it is its own face master; if not, the CCB becomes a hole of the face referred to by the face master. Moreover while stitching two half-edges, if the face masters are different, then a union operation is done (see Fig. 5). We complete the description by the initialization of the face master.

Whenever two arcs are adjacent in $\mathcal{V}$, a pair of active adjacent half-edges bounds the same face. When creating a new SCH, the SCH either contributes to a face in progress (i.e. an adjacent half-edge already handled can be found), or starts a new face. Difficulties arise if one of the half-edge creating the new SCH is adjacent to the north pole. This particular case is carried out by recording the *face containing the north pole*, denoted $F_N(M_\theta)$—the subscript θ indicates that this face may evolve during the sweep. If no circle is bipolar or north polar, $F_N(M_\theta)$ is set once and does not change over the course of the algorithm. Otherwise, the face $F_N(M_\theta)$ is defined as follows. If the meridian M_θ is not tangent to any circle at the pole, $F_N(M_\theta)$ is defined as the face having the north pole on its boundary and containing an infinitesimal portion of M_θ anchored at the north pole. If meridian M_θ is tangent to a (bi)polar circle, $F_N(M_\theta)$ is undefined. But we denote $F_N(M_{\theta^-})$ (respectively $F_N(M_{\theta^+})$) the face containing the north pole for an infinitesimally smaller (respectively larger) value of θ. See Table 1 for initialization and update of $F_N(M_\theta)$.

3.4 Complexity Analysis

To conclude, we give the cost of constructing the HDS storing the arrangement. The analysis consists of counting the number of find and union operations used to maintain the topological data structures i.e. SCH/CCB and faces. Denoting α the inverse of Ackermann's function, recall that the complexity of performing M union-find operations on a N elements set, with $M \geq N$, is $\Theta(M\alpha(M,N))$ [27]. $M \leq U$, then $\alpha(M,N) \leq \alpha(U,U)$.

Upon completion of the sweep, the union-find data structure features f connected components corresponding to the f faces of the arrangement. Denoting h the total number of holes, e the number of

Position of M_θ	**Operation(s) at north pole**
Initializing $\mathcal{V}$	$F_N(M_\theta)$ is set to be the face started by the upper hedge of the top arc of $\mathcal{V}$ at $\theta = 0^+$.
Bipolar or a north polar start event	$F_N(M_{\theta+})$ is the face started by the inner hedge. $F_N(M_{\theta-})$ is the face described by the outer hedge.
North polar end event	$F_N(M_{\theta+})$ is the face described by the outer hedge. $F_N(M_\theta^-)$ is the face described by the inner hedge.
Bipolar end event	$F_N(M_{\theta+})$ is the face started by the outer hedge. $F_N(M_\theta^-)$ is the face described by the inner hedge.
Normal start event or south polar start event	$F_N(M_\theta)$ is set to be the face started by the outer hedge(s) (if it is not already done).

Table 1: Initializing and updating the face containing the north pole $F_N(M_\theta)$, and its relatives $F_N(M_\theta^-)$, $F_N(M_\theta^+)$. See text for details. Notice that hedge stands for half-edge.

edges in the arrangement, n the number of circles and n_0 the number of arcs found in $\mathcal{V}$ at $\theta = 0^+$, the following theorems are proved in [8]:

Theorem 2 *Constructing CCB has complexity* $O((e + n_0)\alpha(6e + 8n_0, 6e + 8n_0))$.

Theorem 3 *Grouping CCB into faces has complexity* $O((f + h)\alpha(f + h + 20n, f + h + 20n))$.

4 Reporting Inclusion into Balls

Let B_i be the ball associated to the sphere S_i which generates the intersection circle $C_i = S_0 \cap S_i$. In this section, we describe an algorithm reporting the *covering list* of each face of the arrangement on S_0, that is the list of balls containing it.

4.1 Enclosing Balls and Unique Circles

In reporting the covering lists, two difficulties are faced.
First, a number of degeneracies must be accommodated. Notice that (i) if a sphere is tangent to S_0 the intersection reduces to a point, and if S_0 is covered by the associated ball, so are all the faces of the arrangement (ii) if two spheres intersect S_0 along the same circle, and their balls cover (respectively do not cover) the same part of S_0, a face covered by one is covered by the other (respectively is not covered by the other).
Second, the BO algorithm operates under the assumption that all circles are different. To meet this requirement, we sort the m intersecting spheres using a total ordering returning equality when two spheres intersect S_0 along the same circle. While sorting the spheres, each unique intersection circle is endowed with two special balls: the *primary* ball, which is associated to the sphere first defining such a circle; and the *opposite* ball, which is the first to intersect S_0 along the same circle, but is opposite to the primary w.r.t. the plane of the circle. Notice that the opposite ball may not exist. Each primary and opposite ball is attached a list of balls yielding the same intersection circle and covering the same part of S_0. We call these the lists of *friends* in the sequel.

Let us get back to the operator used to sort spheres. Two non-great circles are identical iff they have the same center: We first resort to a lexicographic sorting using intersection circle centers. Two great circles (their center being the center of S_0) are identical iff the centers of the two spheres defining these circles are aligned with the center of S_0—the spheres generating these circles belong to the same pencil. Thus, to distinguish great circles, we use the lexicographic order on a canonical vector describing the pencil of spheres yielding a given circle.

4.2 Inclusion into Balls

Outline. While running the sweep process, we construct an implicit encoding of the covering lists. In the following, the description focuses on the primary and opposite balls, as the remaining balls are accessible thanks to the lists of friends.

More precisely, the covering lists are represented by a tree, the *covering tree*. Each node in this tree corresponds to one face of the arrangement, and the edge connecting two nodes corresponds to an arc found in $\mathcal{V}$, stating which primary/opposite ball is added to/removed from the covering list of the father node. Since a face consists of one or several CCB and a CCB is incrementally built from SCH, the principle used to set the covering lists consists of setting one such list for each face master upon creation of the corresponding SCH. This strategy entails two things. First, upon extension at an intersection point of SCH describing a given face, the covering list does not change since the contribution (covering balls) of the corresponding circle has already been taken into account at the creation of the face using a father node. Second, consider the case of the union of two SCH having different face masters: The two face masters must be merged. The consequences are twofold: (i) one of the two covering lists can be discarded; (ii) in the covering tree, the sons of the node suppressed are attached to the node that remains.

Initialization. Let LNB_θ be the list of balls covering the face $F_N(M_\theta)$. This list is initialized at $\theta = 0^+$, as indicated in the first section of Table 4.2. The root of the covering tree is precisely LNB_θ at $\theta = 0^+$.

Updates. First, upon creation of a new face started by a new SCH, its covering list is created, using Observation 1, by updating that of its ancestor in the covering tree. This ancestor corresponds to the face pointed by the face master of the upper half-edge of the highest arc along $\mathcal{V}$ involved in the creation of the new SCH.

Second, consider the updates of LNB_θ. For a circle C, let $\texttt{CLP}(C)$ (respectively $\texttt{CSP}(C)$) be the function returning, if any, among the primary and opposite balls, the one Covering the Largest (respectively Smallest) Part of S_0 bounded by circle C. Operator + (respectively -) means that the ball is added to (respectively removed from) LNB_θ. The cases to be accommodated are listed in the second section of Table 4.2 and illustrated on Fig. 6.

Step		**Type of event**	**Action(s) on LNB$_\theta$**
Filtering spheres	1a	$S \cap S_0 =$ a point and $S_0 \subset B$	$+\ B$
	1b	B covers the largest part of S_0	$+\ B$
Classifying circles	2a	North polar circle with $\theta_E < \theta_S$	+ $\texttt{CSP}(C)$ and - $\texttt{CLP}(C)$
	2b	North threaded circle	+ $\texttt{CSP}(C)$ and - $\texttt{CLP}(C)$
Handling events	3a	North (bi)polar circle Starting	+ $\texttt{CSP}(C)$ and - $\texttt{CLP}(C)$
	3b	North (bi)polar circle Ending	+ $\texttt{CLP}(C)$ and - $\texttt{CSP}(C)$

Table 2: Updating the balls in LNB_θ: initialization and updates. The ball/sphere/circle processed is denoted $B/S/C$. See text for the definition of functions CLP and CSP, and Fig. 6 for the details.

At the end of the sweep process, the number of nodes is exactly the number of faces in the arrangement. Moreover, if n stands for the number of intersection circles, the maximum distance between the root and a node is $2n$ as $\mathcal{V}$ never contains more than $2n$ arcs.

An example of such a tree is presented on Fig. 7. Consider the case of face 6. This face got started at the start event of circle C_4 using its two inner half-edges. The highest one of the pair is the lower half-edge of the upper arc of C_4. The upper half-edge of that arc describes face 4. Therefore, the node of face 6 is a son of the node of face 4, and since the arc considered is an upper one, the edge between the two nodes indicates that the ball, among primary and opposite, covering the smallest (respectively the largest) part of S_0 according to C_4 must be added (respectively removed).

4.3 Complexity Analysis

To analyze the complexity of the implicit encoding, m denotes the number of input balls. The following is proved in [8]:

Theorem 4 *Computing the covering tree has complexity $O(f + m)$ and $O(f + m)$ space. Denote $s_T(F)$ the total number of balls covering face F. Constructing the covering lists for all faces of the arrangement requires $O(\sum_{F \in faces} s_T(F) + f)$ time.*

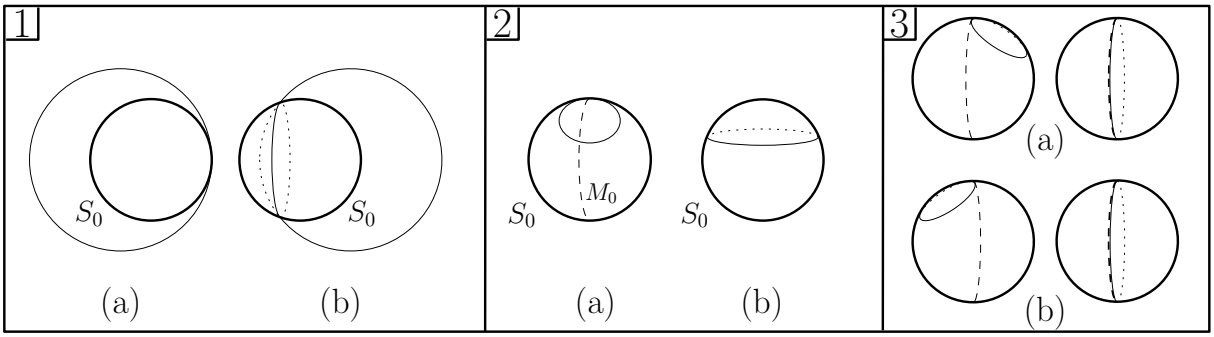


Figure 6: Updating the balls of LNB_θ: the six cases of Table 4.2. 1(a) S_0 is covered by a ball whose sphere is tangent to S_0; 1(b) A sphere intersects S_0, and the associated ball covers the largest part of S_0; 2(a) Meridian M_0 intersects a polar circle. The ball covering the smallest part of S_0 covers $F_N(M_0)$; 2(b) A north threaded circle. The north pole is always (respectively never) covered by a ball covering the smallest (respectively largest) part of S_0; 3(a) The dashed meridian M_θ stopped at a (bi)polar start event: $F_N(M_{\theta^+})/F_N(M_{\theta^-})$ is covered by a ball covering the smallest/largest part of S_0; 3(b) The dashed meridian M_θ stopped at a (bi)polar end event: $F_N(M_{\theta^+})/F_N(M_{\theta^-})$ is not covered by a ball covering the smallest/largest part of S_0.

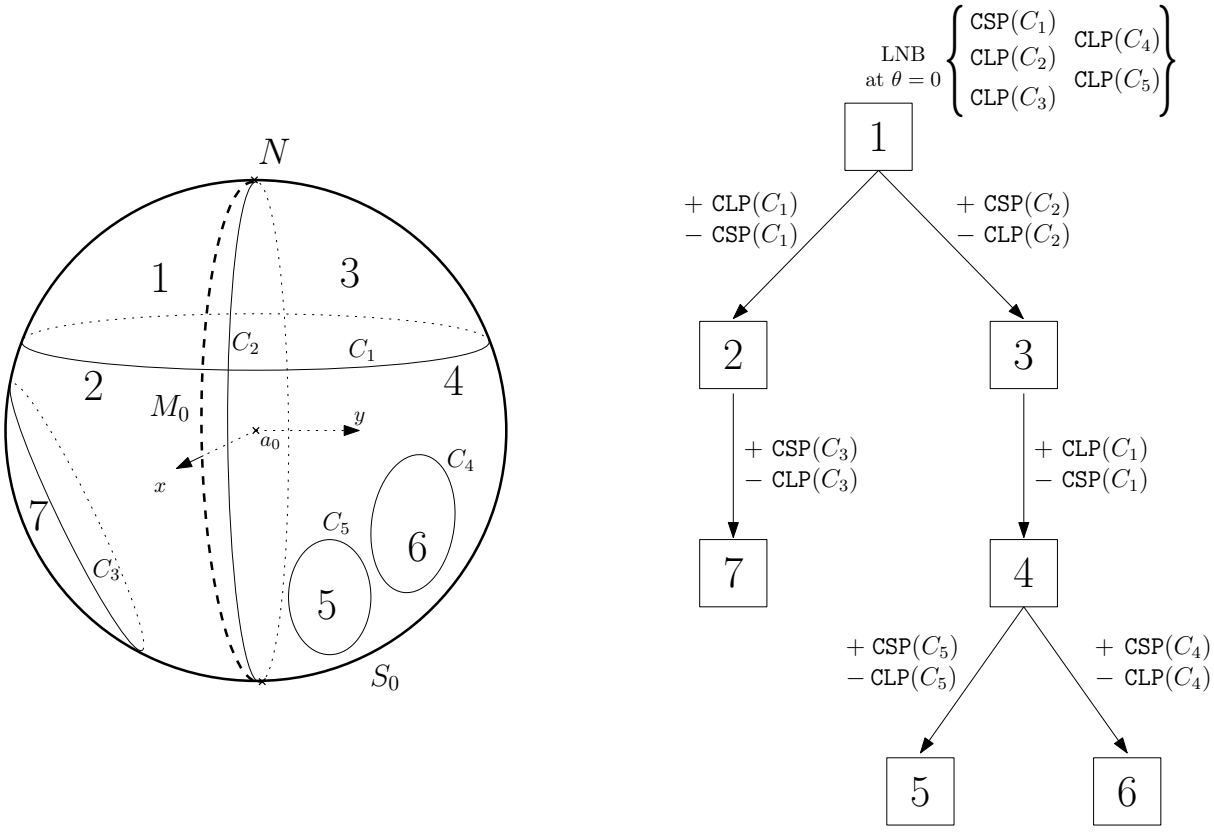


Figure 7: Implicit encoding of covering lists. Circle C_1 is north threaded, C_2 is bipolar, while C_3, C_4 and C_5 are normal circles. Faces of the arrangement depicted on the left are identified using numbers from 1 to 7. Notice that node associated to face 2 reflects the modification of LNB_θ induced by bipolar circle C_2.

5 Application to Flexible Docking

In this section, we discuss one structural biology problem involving spherical arrangements, namely that of selecting diverse conformational ensembles. The molecular models used are Van der Waals (VdW) models.

5.1 Selecting Diverse Conformational Ensembles

Flexible Protein Docking with Conformer Ensembles. Protein-protein complexes are paramount to all biological processes, and predicting the conformation of a complex from the unbound partners is known as the docking problem. Unfortunately, docking is especially challenging, as evidenced in the CAPRI experiment, by the low number of *medium* and *high* predictions—as opposed to *incorrect* and *acceptable* ones [17]. (The CAPRI experiment is a community wide experiment during which participants are asked to predict the conformation of an unpublished co-crystallized complex from the geometries of the isolated partners. Since the structure of the complex has been solved experimentally, the predictions can be compared to the crystallographic reality.) For such challenging cases, which feature large amplitude motions beyond the time ranges accessible to all-atom molecular dynamics (MD) simulations, discrete ensembles of conformations known as *conformer ensembles* can be pre-generated and considered simultaneously. (A MD simulation consists of simulating the atomic motions by integrating Newton's equations of motion. Since the integration time-step is of the order of the femtosecond, the simulation times are of the order of tens of nanoseconds for classical systems.) Such ensembles are particularly appropriate when dealing with macromolecular docking, since one wishes to explore the relative position and orientation of the partners, but also their conformational space. In the Monod-Wyman-Changeux interpretation [23], the unbound proteins are considered as two collections of conformers in thermodynamic equilibrium. When the partners bind, the equilibrium is shifted towards the structure observed in the complex. Implementing this strategy poses two problems, namely generating conformational ensembles, and handling them for docking. We examine this latter problem in the sequel.

Diversity from Geometric Optimization. Given a collection of conformers for each partner, flexible docking algorithms manipulating conformational ensembles aim at identifying the two conformations most likely to form a complex. (For a co-crystallized complex, ideally, these conformations should contain at least one conformer as close as possible to the one observed in the crystal structure.) But since the total number of conformations of a given molecule is in general infinite, the question of selecting an ensemble of reasonable size arises. To further specify the problem, assume we are given a large pool $\mathcal{C}$ of *stable* candidate conformations. By stable, we mean that the conformations are energetically favorable, as trying to dock conformations with serious flaws (e.g. steric clashes) would be meaningless. Given such a pre-computed collection $\mathcal{C}$ of conformers, we introduce the following two selection problems:

▷ Problem #1: Find out a subset $\mathcal{S}$ of $s \ll n$ conformers, called the *selection*, such that the volume occupied by the union of the conformers in the selection is maximized.

▷ Problem #2: Find out a subset $\mathcal{S}$ of $s \ll n$ conformers, called the *selection*, such that the area of the VdW surface of the union of the conformers in the selection is maximized.

Notice that the rationale underlying these optimization problems is rather simple: maximizing the volume or the surface exposed by the selection are two ways to ascertain that the conformers are non redundant. For a fixed budget of conformers, this non redundancy warrants the conformational diversity called for by the docking algorithms.

The first problem is intimately related to *max-k cover* [14, 13], a well-known **NP**-Complete combinatorial optimization problem. In fact, the problem generalizes max-k cover, since the weight of each cell in the $3D$ arrangement decomposing the union of balls is not a unit weight but the Euclidean three-dimensional volume. The second problem is more tricky, since the surface area of the selection is not, as opposed to the volume, a monotonic quantity: add one conformer, and the surface area may decrease. Because of these difficulties, algorithms solving exactly these problems in time polynomial in both n and s cannot be expected, so that approximation strategies must be sought. An obvious such strategy is the greedy one, which consists of incrementally selecting the s conformers: The conformer selected at the i-th stage is that yielding the best increment. These strategies are examined in [22], where the following theorems are proved:

Theorem 5 *For Problem #1, the greedy strategy has an approximation guarantee of* $1-(1-1/s)^s > 1-1/e$*, which is optimal. (e is the base of the natural logarithm.)*

Theorem 6 *For Problem #2, the greedy approach may have an approximation guarantee as bad as* $1/s^2$.

Implementing the Greedy Strategy using Arrangements of Circles on a Sphere. In the following, we focus on Problem #2 for a practical reason: Problem #2 can be solved using arrangements of circles on a sphere, which can be done efficiently as sketched in section 2, using the spherical kernel developed in [7]; on the other hand, Problem #1 requires decomposing the 3D volume occupied by a collection of balls, and we are not aware of any effective algorithm to perform this task.

To solve Problem #2, assume we have computed for each atom (i) the decomposition of its surface induced by the intersection circles with all atom from all conformers in $\mathcal{C}$, as explained in section 3; (ii) the covering lists for all faces of this arrangement, as explained in section 4.

Implementing the greedy strategy from these pieces of information is elementary: one maintains a priority queue based on the increments each conformer would bring to the selection; upon selecting the top conformer C_t, using the covering lists involving balls of C_t, one updates the weight of conformers remaining in the queue which feature spherical caps covered by balls of C_t. To state the algorithm complexity, define

$$\tau = \sum_{C_i \in \mathcal{C}} \sum_{S_j \in C_i} \Big(\sum_{F_k \in S_j} s_T(F_k) + 1 \Big), \tag{1}$$

where $\mathcal{C}$ is the set of all conformers, S_j a sphere of a conformer C_i, F_k a face on the arrangement on sphere S_j and $s_T(F_k)$ the total number of balls covering F_k, as introduced in Thm. 4.

Implementing the priority queue with Fibonacci yields the following theorem, as proved in [22]:

Theorem 7 *The greedy selection based on the spherical arrangements of balls of the conformers in* $\mathcal{C}$ *has amortized complexity* $O(\tau + s \log n)$.

Alternatively, one may skip the calculation of the covering lists, in which case the greedy selection can be carried out by recomputing for each candidate conformer C_k the surface area of $S_{j-1} \cup C_k$, with S_{j-1} the selection obtained so far. In that case, one has [22]:

Theorem 8 *The naive greedy selection of conformers has complexity* $O(ns^3)$.

From a practical standpoint, and since one has $s \ll n$, the best strategy depends on the value of τ given by Eq. (1): for sparse arrangements, the algorithm coming along with Thm. 7 should be used; for dense and cluttered arrangements, as τ takes over the cost of running the priority based selection, the algorithm associated to Thm. 8 is the alternative of choice.

5.2 Validation

Docking Protocols. To validate the conformer selection strategy based upon surface area maximization, we ran docking simulations between one rigid protein called the ligand (L), and one flexible called the receptor. The receptor itself decomposes into a rigid template (R) and a flexible loop (F). While performing flexible protein docking with conformer ensembles, the strategy consists of using a conformer ensemble for the flexible loop F, this ensemble being selected from a larger pool. Thus, specifying a docking protocol requires specifying the triple R/L/F.

To see how, recall that a binary complex used for docking validation features two molecules which have been crystallized under two forms: on their own, i.e. the unbound forms, and in complex i.e. the bound forms. Thus, to specify the rigid parts (R and L), we provide a tag indicating the origin of the partner, namely U for Unbound and B for Bound.
To specify the ensemble associated to F, we provide three pieces of informations: (i) the bound/unbound tag which indicates the loop geometry used to generate the pool of conformers (ii) the selection size, and (iii) the algorithm used to select the conformers from this pool. Two selection algorithms were used, namely a standard one based on a hierarchical clustering of the conformers (called HClust) [15], and the greedy strategy presented above (called Greedy). For example, F=B-Greedy-10 refers to 10 conformers selected by algorithm Greedy, out of a pool of conformers generated from the Bound structure of the receptor.

To summarize, we report on the following four docking protocols: two using the Bound form of the receptor, namely B/B/B-HClust-10, B/B/B-Greedy-10; and two using the Unbound form of the receptor, namely U/B/B-HClust-10, U/B/B-Greedy-10. In passing, notice that the incentive for using the Bound conformation of the flexible region to generate the conformers is the following: for very flexible systems,

the reconstruction of the unbound conformation of the flexible loop from the crystallographic data may not be possible. (If the conformations of the loop changes across the crystallographic units, the signal is not strong enough for the reconstruction to be carried out.)

Initial Conditions for a Protocol. For a given protocol, we ran N_t docking tests using algorithm `ATTRACT` [30]. Each such test corresponds to a specific position and orientation of the ligand with respect to the receptor. Given these initial conditions, ATTRACT performs a sequence of minimizations so as to explore the six degrees of freedom of the ligand. At each stage, the energy of each conformation of the complex is computed, from which a fitness score (between 0 and 1) is attributed to each loop. Upon termination, the loop selected is that having the highest fitness score. An assessment of the quality of the proposed complex is then based upon two figures: (i) the interaction potential energy E of the complex (ii) the *i-rmsd* of the atoms of the ligand. (The i-rmsd is the standard deviation on the atomic positions of the alpha carbons of the *extended interface* of two conformations: that output by the docking program, and that found in the co-crystallized complex. The extended interface is defined as the set of residues of the ligand having an atom of the receptor within 5Å in the X-ray structure.) For the N_t tests associated to a given protocol, the plot of the pairs (E,i-rmsd) defines the energy landscape of the docking experiment. Thus, a conformer ensemble is satisfactory if the landscape features at least one conformer yielding a large number of points (E,i-rmsd) next to the bottom left corner of the energy landscape. In particular, following the CAPRI assessment rules, a *medium* (respectively *acceptable*) prediction corresponds to a i-rmsd smaller than 5Å (respectively 10Å) [20]. Practically, we represent an energy landscape using buckets. For a given bucket B_i and conformer C_j, let $s_{i,j}$ be the number of times conformer C_j yields a complex whose energy and i-rmsd fall in bucket B_i. (Notice that $\sum_{i,j} s_{i,j} = N_t$.) Finally, for a given bucket B_i, denote l_i the index of the conformer that yields the largest value of $s_{i,j}$, and let $r_i = \sum_{j=1,\ldots,n;j\neq l_i} s_{i,j}$. In bucket B_i we display the score s_{i,l_i}, together with r_i when $r_i \neq 0$. The color used to do so is that associated to conformer l_i—one color per conformer.

System Tested and Assessment. As an illustration, we report docking energy landscapes for complex `1BTH`, whose receptor is a thrombin mutant, the ligand being the pancreatic trypsin inhibitor. The flexible region is a loop of 10 amino acids on the receptor. A pool of $n = 500$ conformers was generated using Loopy [29], from which $s = 10$ were selected using either the HClust or Greedy algorithms aforementioned. For each selection method, a total of $N_t = 30,000$ docking tests were run using the same 10 selected conformers. Out of the N_t tests, Fig. 8 and Fig. 9 present plots of the results corresponding to a i-rmsd ≤ 15Å and an energy in the range $[-30, 0]$ units.

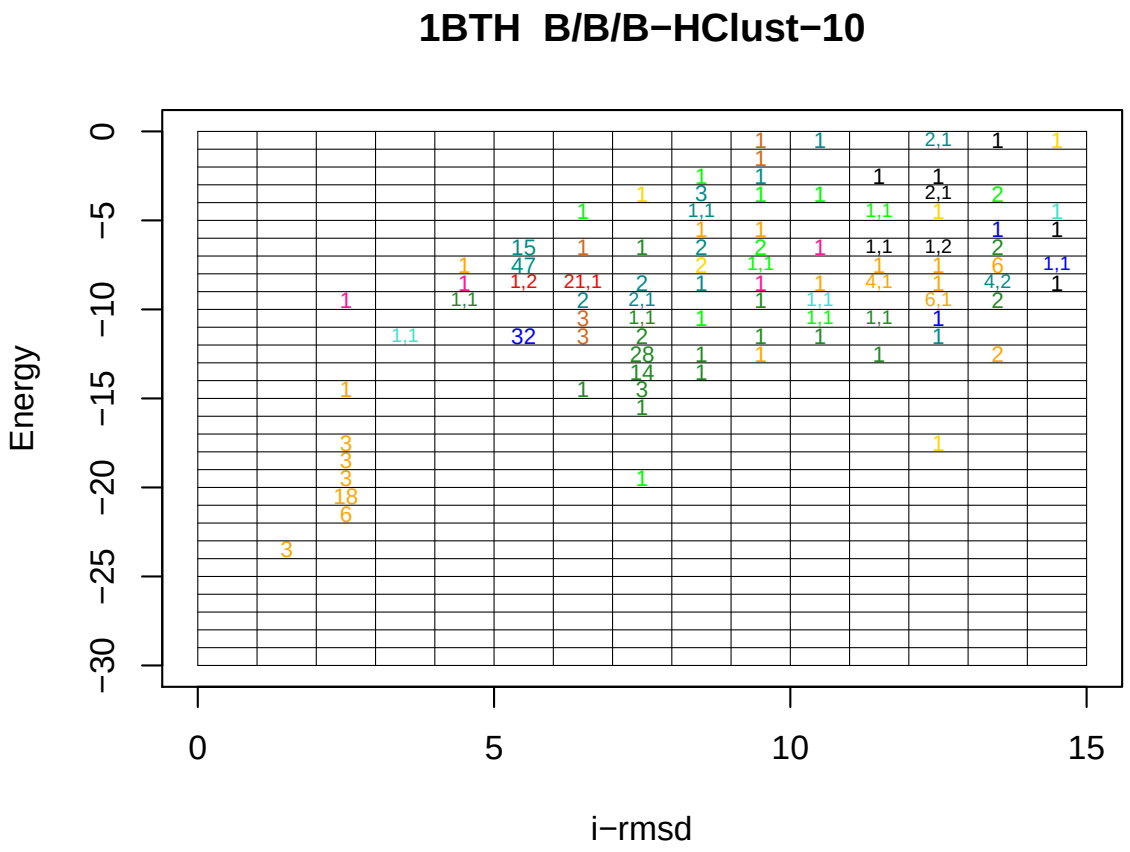
Fig. 8 shows that the conformer selection provided by Greedy is of higher quality than that provided by HClust. To see why, consider conformations of the complex with i-rmsd less than 3Å and energy below -20 units: the number of hits for the top scoring conformer are respectively of about 40 against 160 for HClust and Greedy respectively. On Fig. 9, which corresponds to the unbound protocol, the results are more mitigated. Consider conformers of energy below -15 units, and i-rmsd in the range $[6, 9]$. Two of the ten conformers selected by HClust are well represented, while one conformer selected by Greedy features a high concentration in-between -17 and -18 energy units. In such cases, a finer inspection of the structures is in order to select the best complex(es) [21].

As a conclusion, on this system, using diverse conformational ensembles increases the chance of ending up with a putative complex with low i-rmsd and low energy with respect to the co-crystallized complex. This trend is general, and a more detailed discussion comparing protein-protein complex will appear elsewhere.

6 Conclusion

This paper presents an integrated framework based upon the generalization of the Bentley-Ottmann algorithm to the spherical setting. This framework accommodates the calculation of the exact arrangement of circles on a sphere, as well as the construction of the corresponding half-edge data structure on the fly. Moreover, assuming that each circle comes from the intersection between the central sphere and neighboring spheres, each coming with an accompanying ball, we explain how to efficiently compute the covering lists of faces of the arrangement, i.e. the lists of balls that contain it. In passing, we notice that while exactness of the arrangement is not a goal per se, it is one way to achieve robustness, at a modest computational overhead as testified in a companion paper.

From an application perspective, we use the framework in structural biology, namely for the problem of flexible docking. Application-wise, a number of modeling situations should benefit from the pieces of



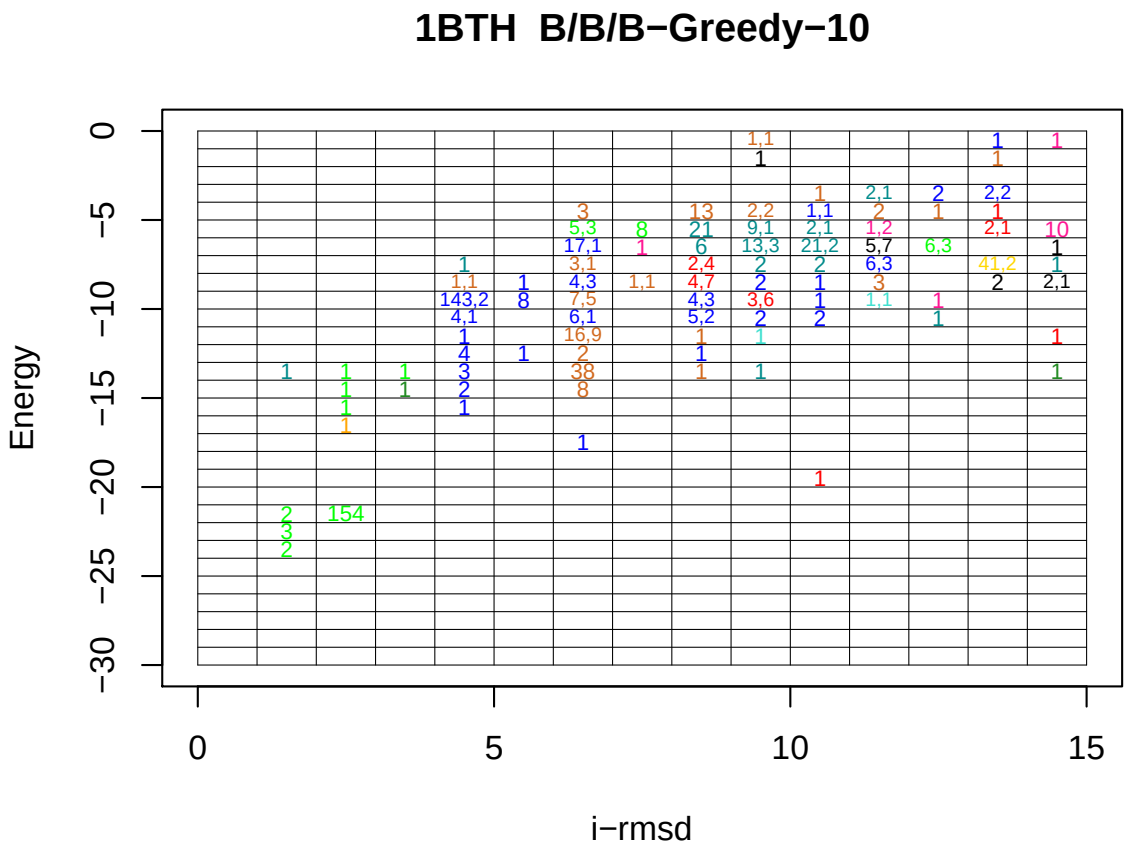
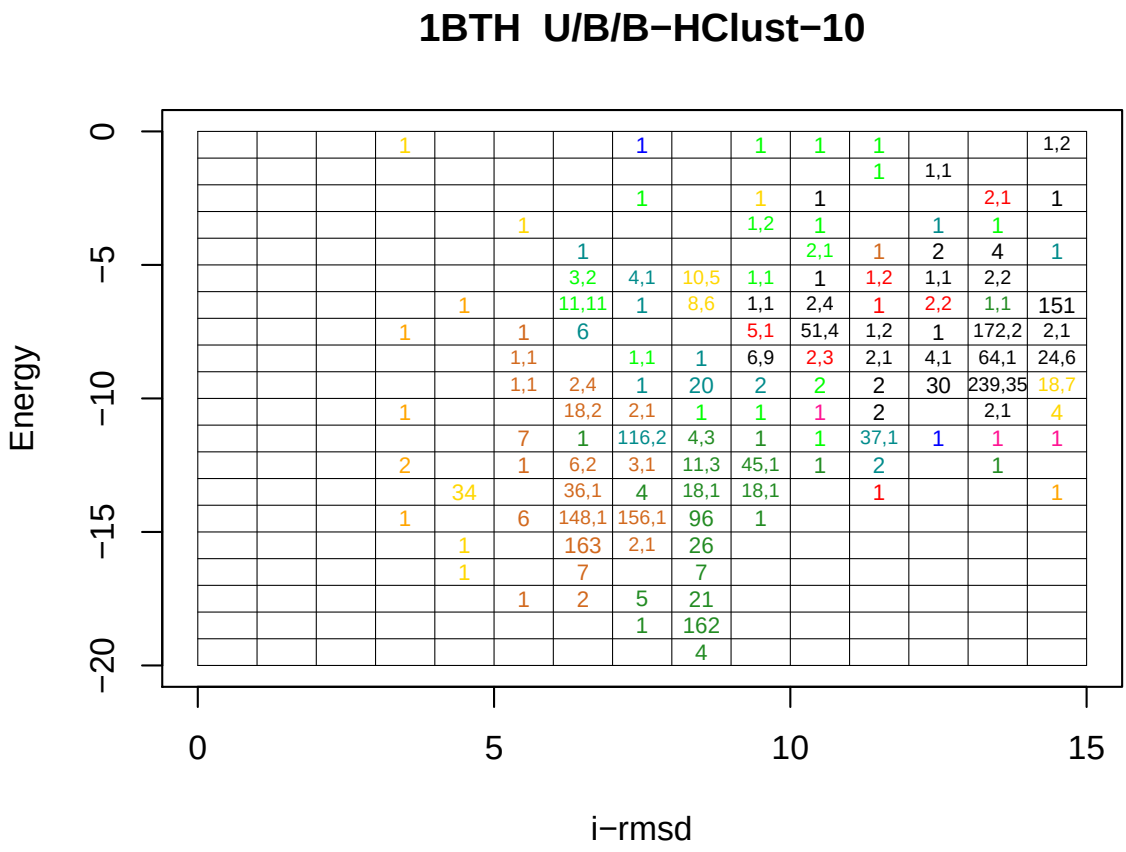


Figure 8: `1BTH`: binning the docking tests using the Bound form of the receptor. See text for details.



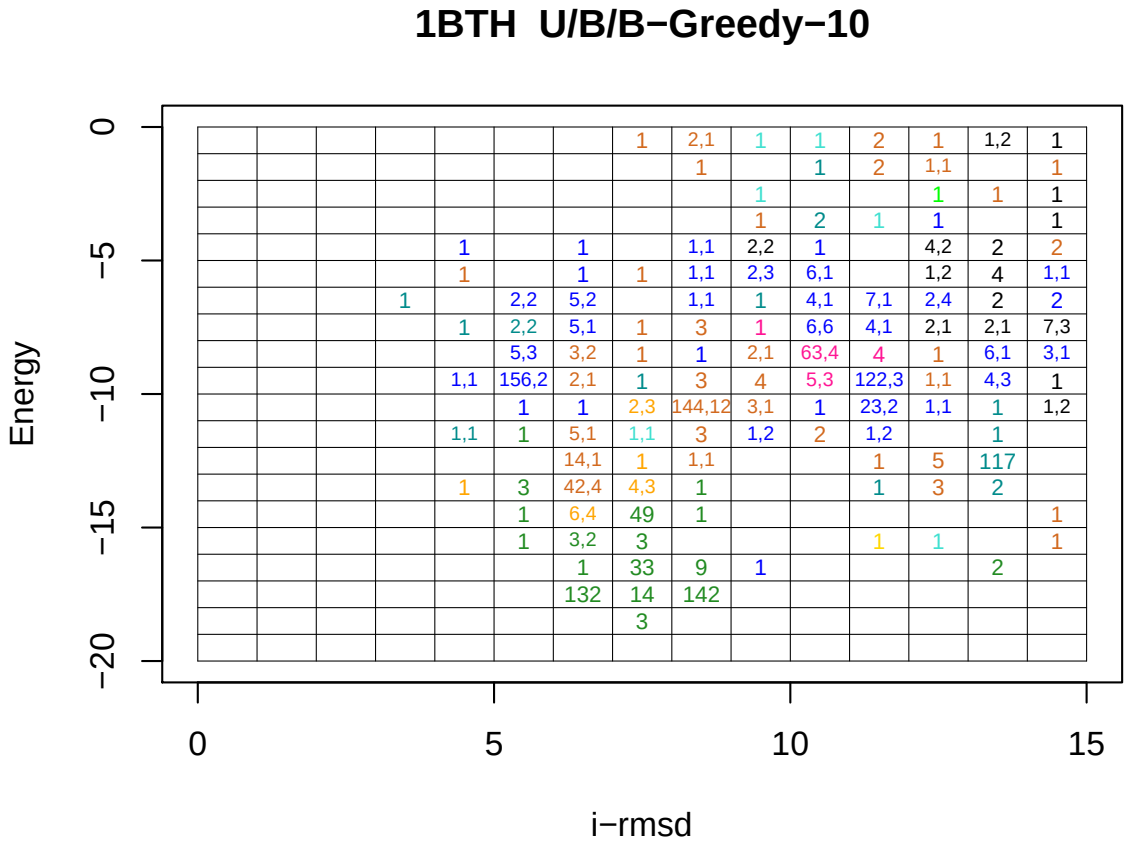


Figure 9: `1BTH`: binning the docking tests using the Unbound form of the receptor. See text for details.

information encoded in the covering lists. From an algorithmic perspective, as our algorithm is dedicated to circles on a sphere, the question of coming up with more general algorithms featuring the same performances deserves investigations.

References

[1] P. Agarwal, L. Guibas, A. Nguyen, D. Russel, and L. Zhang. Collision detection for deforming necklaces. *Computational Geometry: Theory and Applications*, 28:137–163, 2004.

[2] N. Akkiraju and H. Edelsbrunner. Triangulating the surface of a molecule. *Discrete Appl. Math.*, 71:5–22, 1996.

[3] N. Alon, H. Last, R. Pinchasi, and M. Sharir. On the Complexity of Arrangements of Circles in the Plane. *Discrete and Computational Geometry*, 26(4):465–492, 2001.

[4] N. Amenta, S. Choi, and R. K. Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry : Theory and Applications*, 19(2):127–153, 2001.

[5] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, 28(9):643–647, 1979.

[6] J.-D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, UK, 1998. Translated by Hervé Brönnimann.

[7] P. M. M. de Castro, F. Cazals, S. Loriot, and M. Teillaud. Design of the CGAL spherical kernel and application to arrangements of circles on a sphere. *Computational Geometry : Theory and Applications*. Submitted.

[8] F. Cazals and S. Loriot. Computing the exact arrangement of circles on a sphere, with applications in structural biology. Research Report 6049, INRIA, 2006. `https://hal.inria.fr/inria-00118781`.

[9] F. Chazal, D. Cohen-Steiner, and Q. Mérigot. Stability of boundary measure. Research Report 6219, INRIA, 2007. `http://hal.inria.fr/inria-00154798`.

[10] M. L. Connolly. Molecular surfaces: a review. *Network Science*, 14, 1996. `http://www.netsci.org/Science/Compchem/feature14.html`.

[11] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[12] E. Eyal and D. Halperin. Dynamic maintenance of molecular surfaces under conformational changes. In *Proc. 21st Annual Symposium on Computational Geometry*, pages 45–54, 2005.

[13] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.

[15] A. Gordon. *Classification*. Chapman & Hall/CRC, 1999.

[16] D. Halperin and C. R. Shelton. A perturbation scheme for spherical arrangements with application to molecular modeling. *Computational Geometry: Theory and Applications*, 10(4):273–288, 1998.

[17] J. Janin and S. Wodak. The Third CAPRI Assessment Meeting Toronto, Canada, April 20–21, 2007. *Structure*, 15(7):755–759, 2007.

[18] L. Kettner. Halfedge data structures. In C. E. Board, editor, *CGAL User and Reference Manual*. 3.3 edition, 2007.

[19] P. Koehl. Electrostatics calculations: latest methodological advances. *Current Opinion in Structural Biology*, 16(2):142–151, 2006.

[20] M. Lensink, R. Mendez, and S. Wodak. Docking and scoring protein complexes: CAPRI 3rd Edition. *Proteins*, 69(4):704–18, 2007.

[21] N. London and O. Schueler-Furman. Funnel Hunting in a Rough Terrain: Learning and Discriminating Native Energy Funnels. *Structure*, 16(2):269–279, 2008.

[22] S. Loriot, S. Sachdeva, K. Bastard, C. Prevost, and F. Cazals. On the characterization and selection of diverse conformational ensembles. Research Report 6503, INRIA, 04 2008. `https://hal.inria.fr/inria-00252046`.

[23] J. Monod, J. Wyman, and J.-P. Changeux. On the nature of allosteric transitions: a plausible model. *Journal of Molecular Biology*, 12:88–118, 1965.

[24] S. Rusinkiewicz and M. Levoy. QSplat: a multiresolution point rendering system for large meshes. *Proc. 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 343–352, 2000.

[25] B. K. Shoichet, D. L. Bodian, and I. D. Kuntz. Molecular docking using shape descriptors. *Journal of Computational Chemistry*, 13(3):380–397, 1992.

[26] C. M. Summa, M. Levitt, and W. F. DeGrado. An Atomic Environment Potential for use in Protein Structure Prediction. *Journal of Molecular Biology*, 352(4):986–1001, 2005.

[27] R. E. Tarjan. *Data Structures and Network Algorithms*, volume 44 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1983.

[28] R. Wein, E. Fogel, B. Zukerman, and D. Halperin. Advanced programming techniques applied to CGAL's arrangement package. *Computational Geometry: Theory and Applications*, 38(1-2):37–63, 2007.

[29] Z. Xiang, C. S. Soto, and B. Honig. Evaluating conformational free energies: The colony energy and its application to the problem of loop prediction. *Proceedings of the National Academy of Sciences*, 99(11):7432–7437, 2002.

[30] M. Zacharias. Protein-protein docking with a reduced protein model accounting for side-chain flexibility. *Protein Science*, 12(6):1271–1282, 2003.