Algorithms for Modeling and Measuring Proteins

Donna K. Slonim*
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139
slonim@theory.lcs.mit.edu

June 9, 1995

Abstract

In this paper we investigate efficient algorithms for computing the volume and surface area of protein molecules and for graphically displaying the molecules in real time. Protein molecules are modeled by sets of overlapping spheres in $\mathcal{R}^3$. We summarize and critique three papers in the field, and we add several new contributions of our own. First, we present and discuss a new randomized algorithm for computing volumes of proteins. This algorithm is faster than any previously-known algorithm. We also suggest several extensions to this research, including ideas for detecting errors in the x-ray crystallography data used as input. Finally, we propose applying a recent machine-learning result [BCGS95] to determine the tolerance of errors in the data.

Keywords: computational biology, protein modeling, efficient algorithms, protein volumes, measuring unions of spheres, computational geometry.

*Supported by a grant from the Siemens Corporation.

1 Introduction

In this paper we investigate efficient algorithms for computing the volume and surface area of protein molecules and for graphically displaying the molecules in real time. We first summarize and critique three papers in the field: "Spheres, Molecules and Hidden Surface Removal," by Halperin and Overmars [HO94], "The Union of Balls and its Dual Shape," by Edelsbrunner [Edl93], and "Measuring Proteins and Voids in Proteins," by Edelsbrunner, Facello, Fu, and Liang [EFFL95]. We then present several new contributions of our own. First, we give a randomized linear algorithm that computes volumes and surface areas of proteins. This is the best bound we know of for the problem; our algorithm is faster and simpler than that of Edelsbrunner. We also suggest several directions for future work, including ways to detect errors in the x-ray crystallography data used as input for these algorithms. Finally, we propose using a new learning model [BCGS95] to determine the tolerance of errors in the data and to ensure that the data actually contains the important structural information about the protein.

Our paper is organized as follows. The rest of this section contains some relevant background about proteins and the definitions used in the rest of the paper. Sections 2 and 3 contain summaries and discussions of previous work. In Section 4, we present our new, faster algorithm for computing volumes of proteins. Section 5 suggests some areas for future research, and in Section 6 we elaborate on one such suggestion: applying a recent result from learning theory to the problem of noisy protein data.

1.1 Background Information About Proteins

Proteins are molecules that control much of the function of living cells. They play diverse roles, from regulating RNA synthesis to facilitating chemical reactions to governing the cell cycle. Many diseases are caused by malfunctioning or missing proteins. Thus, one challenge of modern medicine is to understand these defective proteins well enough to design new molecules to assist or replace them.

Structurally, proteins are long chains of amino acids folded into a specific three-dimensional structure. A protein's behavior depends at least in part on this structure. Protein molecules range in size from fewer than one hundred atoms to several thousand. Thus, even though proteins generally contain atoms of only five different elements, there are a vast number of possible sequences and structures to work with. The problem of predicting structure from sequence is currently the focus of major research efforts, but it is not relevant to this paper. Instead, we treat the problems of modeling proteins to display them quickly on a computer screen, and of computing properties of proteins intended to help us better understand their nature and function.

Proteins inside cells are surrounded by solvent molecules, primarily water. Different amino acids have different affinities for water, and this affinity governs protein structure to some degree. There are also spaces inside folded proteins, sometimes called "cavities" or "voids," which may contain solvent molecules. Thus the total surface area of a protein molecule is an important property to investigate. Determining the surface of the molecule is also essential for displaying the molecule on the screen. The volume of a protein provides information about its packing density and thermodynamic properties. Efforts to compute volumes of proteins based on Voronoi diagrams are at least 20 years old [Ric77]. This paper discusses recent improvements to such approaches.

Proteins are frequently represented by a *hard-sphere* model, in which each atom is considered to be a ball in $\mathcal{R}^3$. The center of the ball corresponds to the nucleus of the atom, while the sphere represents the space occupied by the atom's electron shells. The radius chosen for each atom is the *van der Waals radius* of the element: the distance from the nucleus at which the repellent forces of two such atoms exactly balance the attracting forces between the two. The exact values for this distance are the subject of some debate, and slightly different estimates of the van der Waals radii for protein atoms are common.

Protein structure is currently determined primarily by x-ray crystallography. In this procedure, x-rays are diffracted through a protein crystal and the protein's structure is derived from the resulting diffraction pattern. Since the atoms move a little even in crystallized proteins, and since reconstructing the structure from the diffraction pattern is a difficult process, the data may contain errors.

1.2 Definitions

First we formally define the hard-sphere representation of proteins. Let $B = \{b_1, b_2, \ldots, b_n\}$ be a set of n possibly intersecting balls in $\mathcal{R}^3$. Each ball b_i has center c_i and radius r_i. Let $C = \{c_i \mid b_i \in B\}$ be the set of center points. We write $\mathcal{U}(B)$ to denote the union of balls in B.

Several of the results discussed in this paper rely on a class of proximity diagrams commonly used in computational geometry. A *Voronoi diagram* of a set S of points in $\mathcal{R}^d$ divides the d-dimensional space into $|S|$ regions. The *Voronoi region* of a point $s \in S$ is the set of all points closer to s than to any other point in S under the Euclidean distance metric. Such diagrams have many practical applications. For example, suppose there are a number of fire stations spread throughout a city. When a fire alarm is raised at a particular point p, the dispatcher first notifies the station closest to p. If the Voronoi diagram of the set of fire stations is already known, the dispatcher merely needs to determine which Voronoi region contains p; this region corresponds to the closest fire station.

In this paper we use a variation of the Voronoi diagram called the *power diagram*. The power diagram of a set of spheres is a proximity diagram using a distance metric that depends on the radii of the spheres as well as their centerpoints. Thus the power diagram of B is exactly the Voronoi diagram when all balls in B have radii equal to zero. The power diagram partitions

$\mathcal{R}^3$ by the planes of intersection of spheres in $\mathcal{U}(B)$.

We now formally define the power diagram of a set of balls. The *power distance* from a ball b_i to a point x, denoted pow(x, b_i), is $|xc_i|^2 - r_{b_i}^2$, where $|xc_i|$ is the Euclidean distance between x and c_i. The *power cell* p_i of a ball b_i is the set of all points in $\mathcal{R}^3$ closer to c_i than to any other point in C, using power distance as the distance metric. Formally, $x \in p_i \iff \forall b_j \in B, \text{pow}(x, b_i) \leq \text{pow}(x, b_j)$.

The *power diagram* P of B is a representation of the power cells of all balls in B. These cells cover all of $\mathcal{R}^3$, and they intersect only along their borders. Since the balls are in general position, no point can be in more than four power cells. (See Figure 1 for illustrations of the power diagram and the related diagrams defined below.)

If we superimpose the power diagram on $\mathcal{U}(B)$ and consider only the portion of the power diagram contained in $\mathcal{U}(B)$, we get the diagram $Q = \bigcup_{b \in B} q_b$, where each $q_b = p_b \cap b$. Note that the q_b's may intersect only along their borders and that $\bigcup_{b \in B} q_b = \mathcal{U}(B)$. Like the p_b's, no point may be in more than four q_b's.

We can now define duals to the diagrams P and Q. D, the dual of P, is a triangulation of the convex hull of C. For $T \subseteq B$, let σ_T be the convex hull of the centers of the balls in T. Formally, D is a simplicial complex in $\mathcal{R}^d$ such that $\sigma_T \in D$ if and only if $\cap_{b \in T} p_b$ is nonempty and is contained in P. In other words, a simplex is in D if the power cells of its vertices intersect (touch) in P.

Finally, K is the subcomplex of D restricted to $\mathcal{U}(B)$. $K = \{\sigma_T \mid \cap_{b \in T} q_b \in Q\}$; that is, a simplex is in K if the power cells of its vertices intersect *inside* the union of balls.

2 Displaying Molecular Surfaces

In this section we discuss the paper "Spheres, Molecules and Hidden Surface Removal," by Halperin and Overmars [HO94], which describes algorithms for displaying and manipulating hard-sphere models of proteins on the screen. One application of such algorithms is in drug design, where computer-drawn models are used to design molecules that bind to specific proteins or replace defective ones. The algorithms in the paper are geared to such applications.

2.1 Summary

The essence of the Halperin and Overmars paper is a straightforward but powerful theorem with a simple proof. The authors define two restrictions that they claim hold true for any hard-sphere representation of protein molecules. They then prove that given an arrangement of balls satisfying these two assumptions, there is a constant upper bound on the number of balls that intersect any given ball in the arrangement.

The two restrictions are the following. Let r_{max} be the maximum radius of any atom in the molecule, and let r_{min} be the minimum radius. The first assumption is that there is a constant

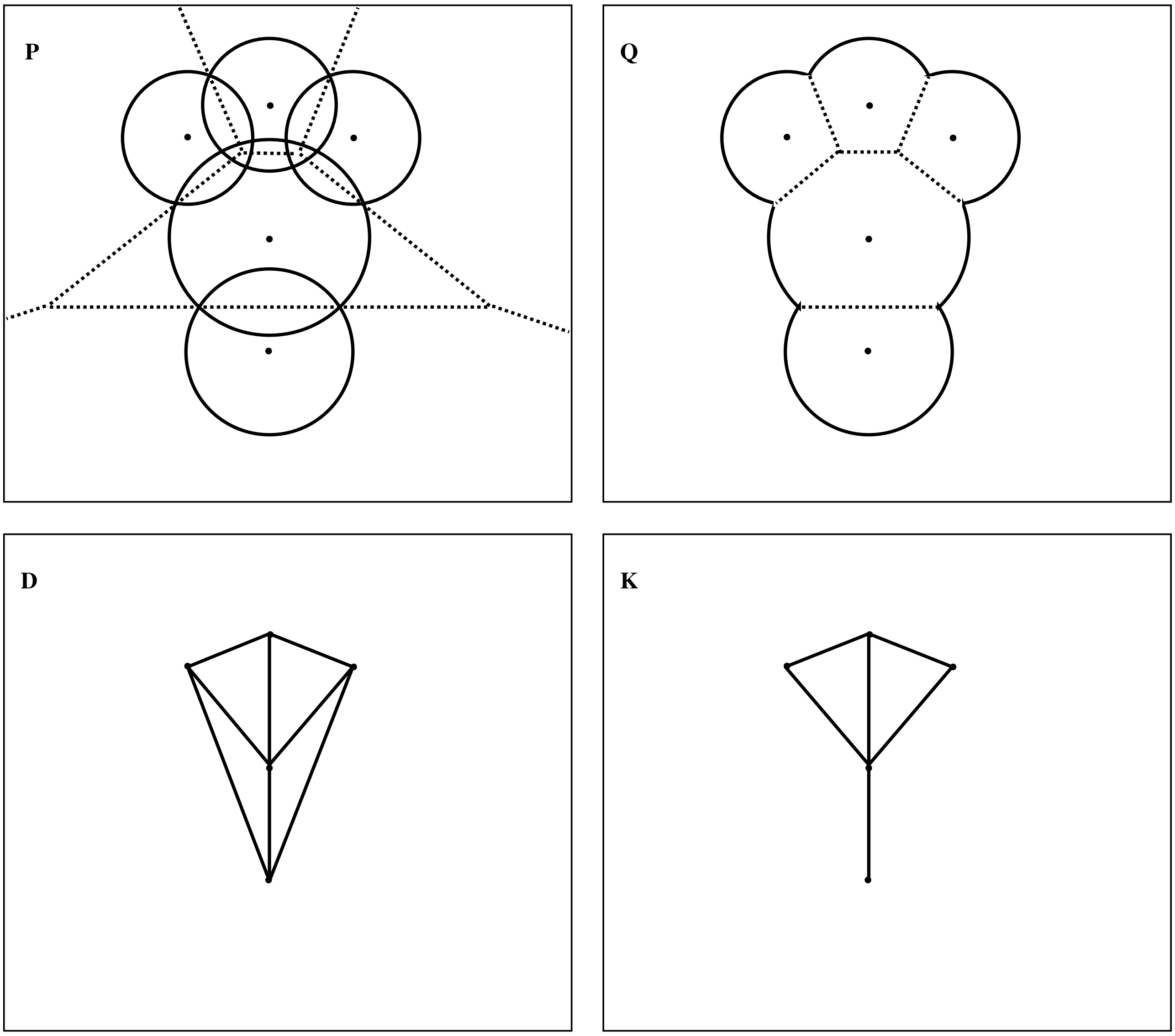

Figure 1: The power diagrams and duals for a set B of five balls in the plane. The dashed lines define power cells. The power cell of a ball b contains all points in the plane closest to c_b under the power distance metric. Note that the power diagram P covers all of $\mathcal{R}^2$. Diagram Q is just the restriction of P to the union of the balls, $\mathcal{U}(B)$. D, the dual of P, contains an edge between any two centerpoints whose power cells share an edge in P. A triangle in D, therefore, corresponds to three power cells meeting at a common vertex in P. Similarly, in K, the dual of Q, there is an edge between two centerpoints b and b' only if cells q_b and $q_{b'}$ in Q share an edge.

upper bound $k \geq r_{max}/r_{min}$. This is certainly true for any fixed data set; k can be computed directly from the van der Waals radii. The second assumption formalizes the notion that no two atoms can overlap too much. This seems a reasonable assumption since the repellent forces between the atoms become stronger as their nuclei move closer together. The paper formalizes this constraint by requiring a positive constant ρ such that for each b_i, the smaller concentric ball with center c_i but radius ρr_i does not contain the center of any other ball in B.

The key to the proof is the observation that any ball intersecting b_i must be contained in the larger concentric ball with center c_i and radius $r_i + 2r_{max}$. It is then easy to show that at most $216(k/\rho)^3$ balls satisfying the restrictions can fill a ball of radius $r_i + 2r_{max}$. Let $\mathcal{N}_b$, the *neighbors of ball* b, be the set of other balls which are contained within a ball with center c_b and radius $r_b + 2r_{max}$. Then any ball which intersects b must be in $\mathcal{N}_b$.

The authors derive a great deal of power from this theorem. Using a similar proof, they show that there is a decomposition of the arrangement of spheres in $\mathcal{R}^3$ (*e.g.*, the subdivision of space induced by $\mathcal{U}(B)$) with total complexity $O(n)$. The paper describes efficient algorithms based on these results to answer intersection queries (to determine if a single query-ball intersects the molecule), to compute the surface of $\mathcal{U}(B)$, to compute a visibility map, and to display the molecule graphically, performing hidden surface removal. We describe some of these algorithms briefly to indicate how they rely on the main theorem.

An intersection query is a test to see whether a single query ball intersects the molecule represented by B. Intersection queries can be used to analyze how a new molecule fits together with the protein or to determine that two molecules do not overlap. One can certainly test whether the query ball intersects each of the other balls in the molecule in $O(n)$ time. However, with some preprocessing it may be possible to optimize the time for each intersection query. The best bound previously known for intersection queries is due to Aurenhammer [Aur88], who proves that by computing the power diagram of B (using $O(n^2)$ time and space for an arbitrary arrangement of balls), the question of whether a given *point* intersects $\mathcal{U}(B)$ can be answered in $O(\log^2 n)$ time.

Halperin and Overmars use the restrictions on proteins to find an algorithm with $O(\log n)$ query time, given $O(n \log n)$ preprocessing time and $O(n)$ space. The idea behind the algorithm is to divide the space up into cubes of size $2r_{max}$ on each side. Using a proof similar to that of the main theorem, they show that each cube intersects only a constant number of balls in B. Thus they can compute all the cubes and the constant-length list of the balls they intersect and insert them into a binary search tree in $O(n \log n)$ time. To answer an intersection query, their algorithm calculates the (at most 8) grid cubes the query ball intersects, looks up the balls of B in each of these cubes ($O(\log n)$ time), and tests whether each ball intersects the query ball. This method produces a list of all balls of B that the query ball intersects. Furthermore, if the binary search tree is replaced by a randomized hash table, the bound on the algorithm is improved to $O(1)$ expected query time and $O(n)$ preprocessing time.

Halperin and Overmars also give an efficient algorithm for computing the surfaces on the boundary of $\mathcal{U}(B)$. For each ball, they find the $O(1)$ balls intersecting it and determine the ball's contribution to the outer boundary of $\mathcal{U}(B)$: that portion of the surface of b not contained in the intersection with any other ball. They then combine the boundary patches from each ball to give a global description of the surface of the union. The main theorem implies that this algorithm can be implemented to run in $O(n)$ randomized or $O(n \log n)$ deterministic time.

Finally, Halperin and Overmars show how to implement hidden surface removal to display the molecule in $O(n \log n)$ time. Given a depth order on the objects to be displayed, one would normally use a painter's algorithm for hidden surface removal: display the farthest objects first, then draw the closer ones later so they appear to be in front. However, for an arrangement of intersecting balls there may be no valid depth order on the balls, so it is not clear how to display them. Halperin and Overmars prove that by determining the faces on the surface of $\mathcal{U}(B)$ and by restricting attention to the parts of these faces on the hemispheres facing the viewer, it is possible to construct a depth order on the faces simply by sorting the balls in B. Again, the efficiency of the algorithm is due to the main theorem; for a general arrangement of balls, this algorithm might require $\Theta(n^2)$ steps.

2.2 Discussion

Halperin and Overmars have used their main theorem to improve the analysis of a number of algorithms commonly used in displaying and manipulating proteins. Some of the ideas behind their main theorem and its applications have appeared elsewhere. Yip and Elber [YE89] present an $O(n)$ algorithm for determining neighboring atoms by dividing the space into cubes. Independently, Varshney and Brooks [VB93] note that each protein atom has only a constant number of neighbors. They use this information to compute the accessible surface of the molecule in linear time, or in $O(n/p)$ time on p parallel processors. Perrot *et al.* [PCG+92] describe an $O(nk)$ algorithm for approximating the surface area, where k is the average number (over the molecule) of neighbors per atom. However, Halperin and Overmars are the first to put all these ideas together and to apply them to real-time computer-aided molecule design.

One essential issue to address is the validity of the assumptions behind the model. Even the fixed-sphere model of atoms is subject to criticism. Atoms in proteins are not stable but move with respect to one another. The standard protein data banks list for each atom both a position, which is essentially the average position over time, and a quantification of how much the atom moves. However, it is hard to model the motions of all the individual atoms in a protein molecule, and it is not clear that the added complexity needed to do so is worth the improvement in accuracy [LTK95]. Representation by fixed balls is thus fairly common, and appears to be accurate to a first approximation [Mez93, Ric77].

The main theorem's restrictions that the atoms' radii are bounded in size and that no two nuclei can be too close seem reasonable for the reasons outlined in Section 2.1. However, there might be comparable formalizations of these ideas which are better suited to our purposes. For example, superoxidise dismutase, a typical molecule from the paper, has constants $k = 1.95$ and $\rho = .76$. Thus the main theorem states that at most 3648 atoms intersect any single atom in the molecule. On the other hand, simulation of the molecule has shown that on average, each atom intersects 5.5 others; the maximum number of others that a single atom intersects is 16. These bounds are not a fluke; for the nine atoms simulated in both the conference and technical report versions of the Halperin and Overmars paper, no atom intersects more than 16 others, and the average over each molecule ranges from 4.5 to 7.5. In all cases the constant bound derived by the theorem is at least two orders of magnitude larger. Surely something can be done to close the gap.

There are several possible causes of this gap. One is that the proof of the theorem uses a worst-case analysis; k is an upper bound on the ratio r_{max}/r_{min}. In fact, this bound is probably seldom realized; an average-case factor corresponding to k might give a more accurate prediction. Also, ρ is generally computed from the x-ray crystallography data directly. This means it is particularly susceptible to errors in the x-ray crystallography data, which are common. We address this issue in Section 5.2.

Another issue is that the theorem does not account for repulsive forces between atoms or for the fact that atoms in proteins generally have some space between them. Richards [Ric77] analyzes the packing of atoms in proteins and concludes that there is little empty space. However, he derives this conclusion by comparing the number of protein atoms with the number of *non-overlapping* balls that can theoretically be packed in the same space. Thus, there are two opposing tendencies to account for when evaluating his conclusions: the fact that the balls overlap somewhat, and the fact that they repel each other. It is therefore unclear how Richards's results would change by accounting for both factors. It might be interesting to obtain similar statistics comparing the number of atoms in a protein with the number of overlapping balls that could fill the same space, subject to the main theorem's constraint on ρ.

Varshney, Wright and Brooks [VWB94] study the general problem of packing unit spheres inside a larger sphere. They motivate their work by the problem of calculating the number of neighbors of protein atoms under some simplifying assumptions about the atoms' radii and the average bond length. Their best bound for the average number of neighbors of an atom satisfying their assumptions is 139. This bound is substantially tighter than the theoretical bounds derived from Halperin and Overmars's paper, but it is still quite large compared to the experimentally observed values.

Another question one might ask is whether the algorithms from this paper are easily parallelized. Since one goal of the paper is real-time computer-aided molecule design, it is essential to represent the molecule on the screen quickly, and parallel implementation might provide the necessary speed. The problem of computing each ball's contribution to the surface of $\mathcal{U}(B)$ can easily run in $O(n/p)$ time on p processors. However, displaying the molecule relies on sorting the balls; the time for sorting in parallel depends on the structure of the parallel network and is in any case no better than $O(\log n)$ [Lei92]. Thus, even with n processors it is not possible to implement Halperin and Overmars' method to display molecules in constant time.

Finally, the algorithms presented in this paper assume that the initial data on the structure of the algorithms, generally derived from x-ray crystallography, is accurate. However, x-ray crystallography is a difficult science and errors in the determined protein structures are common. The algorithms given in this and the following papers do not account for these errors and do nothing to detect them. In sections 5.2 and 6 we suggest some ways to use these algorithms to detect and possibly correct errors in the x-ray crystallography data.

3 Measuring Proteins

The papers "The Union of Balls and its Dual Shape," by Edelsbrunner [Edl93], and "Measuring Proteins and Voids in Proteins," by Edelsbrunner, Facello, Fu, and Liang [EFFL95], discuss the same algorithms and are therefore treated together. The latter paper describes the implementation of algorithms for determining surface area and volume of various protein models, while the former treats the theoretical justification for the algorithms.

3.1 Summary

The primary breakthrough in these papers is a polynomial-time simplification of the inclusion-exclusion formula for computing volumes. The basic inclusion-exclusion formula states that the volume of the union of a family B of potentially intersecting sets is

$$\sum_{T \subseteq B} (-1)^{|T|-1} \text{volume}(\cap_{b \in T} b).$$

This formula is accurate but unfortunately requires computation time exponential in the size of B. In general, approximations to the formula consisting of fewer than $\binom{n}{\sqrt{n}}$ terms are inaccurate [LN90]. However, Edelsbrunner proves that the volume of $\mathcal{U}(B)$ in d dimensions can be computed exactly using an inclusion-exclusion formula where all subsets have size at most d. The formula does not include *all* subsets of size d; only those sets of balls whose power cells intersect in Q are considered.

In "The Union of Balls and its Dual Shape," Edelsbrunner presents a topological proof of the following formula:

$$\text{volume}(\mathcal{U}(B)) = \sum_{T \subseteq B \mid \sigma_T \in K} (-1)^{|T|-1} \text{volume}(\cap_{b \in T} b).$$

To evaluate this formula one need only consider $O(|K|)$ sets of size at most $d+1$. The paper further explains that it is possible to speed up the computation by calculating the volume of all simplices in K directly and then adding the volume of the *fringe*, the part of $\mathcal{U}(B)$ *not* contained in K. The volume of the fringe can be computed by summing only subsets of up to d balls.

The implementation paper applies the algorithms to arrangements of balls in $\mathcal{R}^3$ and analyzes the running time of the algorithms. To compute the volume of a protein, the algorithm first computes the regular triangulation D and then uses D to derive K. An unsupported claim argues that for proteins, $|D| = O(n)$. The algorithm then computes D and K in $O(n \log n)$ expected time and evaluates the inclusion-exclusion formulas in time proportional to $|K| \leq |D|$. Thus if $|D|$ is $O(n)$, then the total running time of their algorithm is $O(n \log n)$. However, in section 4.1 we show that, while $|D|$ might be larger than $O(n)$, it is possible to compute the volume of a protein molecule obeying the restrictions from Halperin and Overmars in $O(n)$ steps.

Edelsbrunner's algorithm also computes the number and volume of the cavities in the molecule. This information can help determine whether solvent molecules such as water might be trapped in cavities inside the protein molecule. The algorithm can also compute the number of vertices and arcs on the surface of the molecule, although it is unclear what purpose such statistics serve.

More exciting, therefore, are some of the topological properties of molecules that Edelsbrunner proves in the "Union of Balls...." He first shows homotopy equivalence between $\mathcal{U}(B)$ and the space occupied by K. (Homotopy equivalence is similar to homeomorphism except that it need not preserve dimension.) This result yields an efficient algorithm for computing the homology groups of $\mathcal{U}(B)$. Donald and Chang [DC91] explain that homology groups can be used to determine if two structures are topologically equivalent. This might be useful when designing a new molecule to fulfill the function of a missing or malfunctioning protein. Homology groups can also tell if a structure can be embedded in $\mathcal{R}^3$, another essential test for molecule design.

The proof of the inclusion-exclusion formula relies on a variation of the Euler relation for convex polyhedra also derived in the paper. Using an inductive argument, Edelsbrunner shows it is possible to determine if a point x is contained in a convex polyhedron P in d dimensions by considering only sets of up to d of the defining half-spaces. He generalizes this result to any compact convex set, and to some non-convex compact sets as well. He then shows how to measure the volume of the intersection of a convex set A and a polyhedron P by integrating over all points in the intersection. Using the Euler relation, he derives a formula for the intersection which again considers only sets of d half-spaces.

Finally, the general inclusion-exclusion formula for $\mathcal{U}(B)$ is proved by embedding $\mathcal{R}^d$ as a hyperplane in $\mathcal{R}^{d+1}$. The hyperplane is mapped to its dual sphere in $\mathcal{R}^{d+1}$; the balls in B are mapped from their containing hyperplane to dual half-spaces intersecting the sphere. Thus, the intersection of the half-spaces defines a polyhedron P and the sphere is a convex set A, so the volume of $P \cap A$, determined using the shallow inclusion-exclusion formulas above, maps to $\mathcal{U}(B)$.

3.2 Discussion

The fact that it is possible to compute the volume of $\mathcal{U}(B)$ using an inclusion-exclusion formula whose terms contain intersections of at most $d+1$ balls is quite a surprising and interesting result. It may very well have applications to other areas where inclusion-exclusion formulas are used, such as cryptography and approximation algorithms. The result is particularly surprising in light of the hardness bounds for general approximation of such formulas (with fewer than $\binom{n}{\sqrt{n}}$ terms) given by Linial and Nisan [LN90]. However, in order for the inclusion-exclusion result to be accessible to the computer science communities that might use it, it needs a more intuitive proof. The proof would be most appropriate in the "Measuring Proteins..." paper, which currently describes the algorithms at an intuitive level but refers to "The Union of Balls..." for all proofs. The proofs in the latter paper, however, are inaccessible to anyone unfamiliar with topology and lack the basic definitions needed to be self-contained. It would thus be extremely helpful to present a short proof or at least some intuition in future applications papers.

The analysis of the implementation in "Measuring Proteins..." relies on the statement that "for dense distributions common to proteins," the number of simplices in D is generally $O(n)$. (The worst-case theoretical bound for $|D|$ in $\mathcal{R}^3$ is $O(n^2)$ [Aur88]. The paper states that such arrangements are unlikely in practice but does not support this claim.) Since the paper states that the software "makes no use of the fact that the union of balls is defined by a molecule," it implies that the time bounds it derives hold for general unions of balls in $\mathcal{R}^3$, which is untrue. In the general case, the new algorithm for finding the volume of $\mathcal{U}(B)$ is no more efficient than the best previously known, Aurenhammer's simpler $O(n^2)$ algorithm for computing volumes of unions and intersections of balls in $\mathcal{R}^3$. Some implementation results might help to support the claim that $|D| = O(n)$. For example, the running times needed to compute volumes of various proteins could illustrate that the algorithm effectively scales as $n \log n$.

It is once again worth asking whether the algorithms in these papers could be implemented efficiently in parallel. The difficult part is the parallel computation of the power diagram. Using algorithms due to Amato, Goodrich, and Ramos [AGR94], one can construct a power diagram for n 3-dimensional balls in parallel in $O(\log n)$ time in the EREW PRAM model. Once K has been derived, evaluating the inclusion-exclusion formula takes constant time for each simplex in K and can be done in parallel.

4 An O(n) Algorithm for Volume and Surface Area

In this section we apply the assumptions about proteins from Halperin and Overmars's main theorem to obtain an faster algorithm for computing the volume of $\mathcal{U}(B)$. Our algorithm is a simple extension of Halperin and Overmars's algorithm for computing the boundary of $\mathcal{U}(B)$. Recall that their main theorem assumes that all proteins satisfy the following constraint:

Protein Constraint:

Let r_{max} be the maximum radius of any atom in the molecule, and let r_{min} be the minimum radius. Then there exist positive constants k, ρ such that $r_{max}/r_{min} < k$ and for each $b_i \in B$, the concentric ball with center c_i and radius $\rho\, r_i$ does not contain any other centerpoint $c_j \in C \mid j \neq i$.

Our algorithm runs in randomized linear time for proteins obeying this constraint, whereas Edelsbrunner's algorithm requires $O(n \log n)$ time. Note that the $O(n \log n)$ bound for Edelsbrunner's algorithm holds even with the constraint, since the algorithm requires the computation of D. We show here that it is enough to determine the neighbors of each ball and that this can be done without computing D. A corollary of our result is that $|K| = O(n)$ for proteins satisfying the constraint.

Recall that $\mathcal{N}_b$ is the set of balls contained within the sphere with center c_b and radius $r_b + 2r_{max}$. Any ball which intersects b must be in $\mathcal{N}_b$. Also recall that Halperin and Overmars's main theorem proves that for any set B of balls satisfying the protein constraint, the number of neighbors of each atom (and therefore, the number of other atoms intersecting each atom) is a constant.

4.1 The Algorithm and Proof

We determine the volume of $\mathcal{U}(B)$ by computing the volume of q_b for each ball b. Recall that $q_b = p_b \cap b$. Since for any two balls b and b', the interiors of q_b and $q_{b'}$ are disjoint, and since $\mathcal{U}(B) = \bigcup_{b \in B} q_b$, the volume of $\mathcal{U}(B)$ is exactly the sum over all $b \in B$ of the volumes of q_b.

If we were to compute q_b by finding p_b and taking its intersection with ball b, the time needed to compute each q_b could be as large as $\Theta(n)$. Even in sets of balls satisfying the protein constraint there may be a ball whose power cell has $\Theta(n)$ bounding halfplanes. However, as we show below, we can compute q_b from $\mathcal{N}_b$ *without* constructing all of p_b. Thus, in the special case of protein molecules we can compute each q_b in constant time, so we can compute q_b for all the balls in $O(n)$ time overall. To show this we need the following definitions.

Let H_{ij} be the halfspace such that $\mathrm{pow}(x, b_i) \leq \mathrm{pow}(x, b_j)$. Note that H_{ij} is bounded by the plane that would separate the power cells of b_i and b_j if there were no other balls in B.

Define the *feasible cell* of b, F_b, as $\bigcap_{b' \in \mathcal{N}_b} H_{bb'}$. Note that the power cell $p_b = \bigcap_{b' \in B} H_{bb'}$, so $p_b \subseteq F_b$.

For our argument we also need the following lemma, which is stated in Edelsbrunner [Edl93] without proof.

Lemma 1 $\forall b' \in B,\ p_b \cap b' \subseteq p_b \cap b$.

Proof: If $p_b \cap b' = \emptyset$ then the lemma is trivially true. So suppose $p_b \cap b'$ is not empty. Let x be any point in $p_b \cap b'$. By definition, if $x \in p_b$ then $\mathrm{pow}(x, b) \leq \mathrm{pow}(x, b')$. Thus,

$|xc_b|^2 - r_b^2 \leq |xc_{b'}|^2 - r_{b'}^2$. Since x is also in ball b', $|xc_{b'}| \leq r_{b'}$, so $|xc_{b'}|^2 - r_{b'}^2 \leq 0$. Thus, $|xc_b|^2 - r_b^2 \leq 0$, so $|xc_b| \leq r_b$, showing that x is also in b. □

Lemma 1 immediately yields this corollary, which we will use to prove Theorem 3.

Corollary 2 *If $p_b \cap b' \neq \emptyset$, then $b \cap b' \neq \emptyset$.*

We can now show that it is possible to compute the volume of $\mathcal{U}(B)$ in linear time. The following theorem states that q_b, the part of the power cell of b restricted to within ball b, can be computed just from the feasible cell; that is, by considering only the neighbors of ball b.

Theorem 3 $F_b \cap b = p_b \cap b = q_b$.

Proof:

Note that the second equality is true by definition; we need only prove the first one. Recall that $F_b \supseteq p_b$, so $F_b \cap b \supseteq p_b \cap b$. For the theorem to be false, there would have to be some points in $F_b \cap b$ but not in $p_b \cap b$. So suppose that there were some ball $\beta \notin \mathcal{N}_b$ such that halfspace $H_{b\beta}$ borders $p_b \cap b$ but not $F_b \cap b$. Since $H_{b\beta}$ borders $p_b \cap b$, there exists some point x on the boundary of $H_{b\beta}$ such that $x \in H_{b\beta} \cap p_b \cap b$. Since x is on the boundary of $H_{b\beta}$, $\mathrm{pow}(x, b) = \mathrm{pow}(x, \beta)$. And since $x \in b$, $\mathrm{pow}(x, b) \leq 0$. Thus, $\mathrm{pow}(x, \beta) \leq 0$, so $x \in \beta$ as well. Thus $x \in p_b \cap \beta$, showing that $p_b \cap \beta \neq \emptyset$. Then by Corollary 2, $b \cap \beta \neq \emptyset$. Since $\mathcal{N}_b$ contains all balls that intersect b, this contradicts the assumption that $\beta \notin \mathcal{N}_b$. □

Thus, we can find q_b from the feasible cell without determining the entire power cell.

Let $X = \{H_{b,b'} \mid b' \in \mathcal{N}_b\}$ so that the feasible cell $F_b = \bigcap_{H_{b,b'} \in X} H_{b,b'}$, and let constant $k = |X|$. We now show how to compute the surface area and volume of $\mathcal{U}(B)$, given X, by determining the surface area and volume of each $q_b = F_b \cap b$.

To compute F_b from X, we use the technique of Varshney and Brooks [VB93, PS85, pp. 316 ff.]. They compute the dual point for each halfspace in X, take the convex hull of the dual, and then compute the dual of the hull to find the vertices of the convex (but potentially unbounded) polyhedron F_b. More specifically, they express each halfspace H_{bi} as an equation of form $a_i x + b_i y + c_i z + d_i \leq 0$, where all the $d_i < 0$. (It may be necessary to translate the origin to make all the $d_i < 0$, but this will have no effect on the area or volume of the intersection.) Then the dual of each halfspace is the point $(a_i/d_i, b_i/d_i, c_i/d_i)$. There will be $O(k)$ such points, so finding their convex hull in $\mathcal{R}^3$ takes $O(k \log k)$ time [PS85]. There can be only $O(k)$ points on the hull, each corresponding to the intersection of three halfspaces (since the balls are in general position), so computing the duals in both directions takes only $O(k)$ time.

The above procedure produces a list of adjacent vertices of F_b, along with the halfspaces whose intersection defines each vertex. The next job is to test whether each vertex is contained in the ball b. If the vertex is *not* contained in b, the intersection of the bounding halfplanes with the surface of ball b defines a spherical patch of b on the surface of $\mathcal{U}(B)$. The dimensions of this patch can be computed from the radius of the ball and the intersection of the sphere and halfspaces. Thus, we can compute the surface area of q_b.

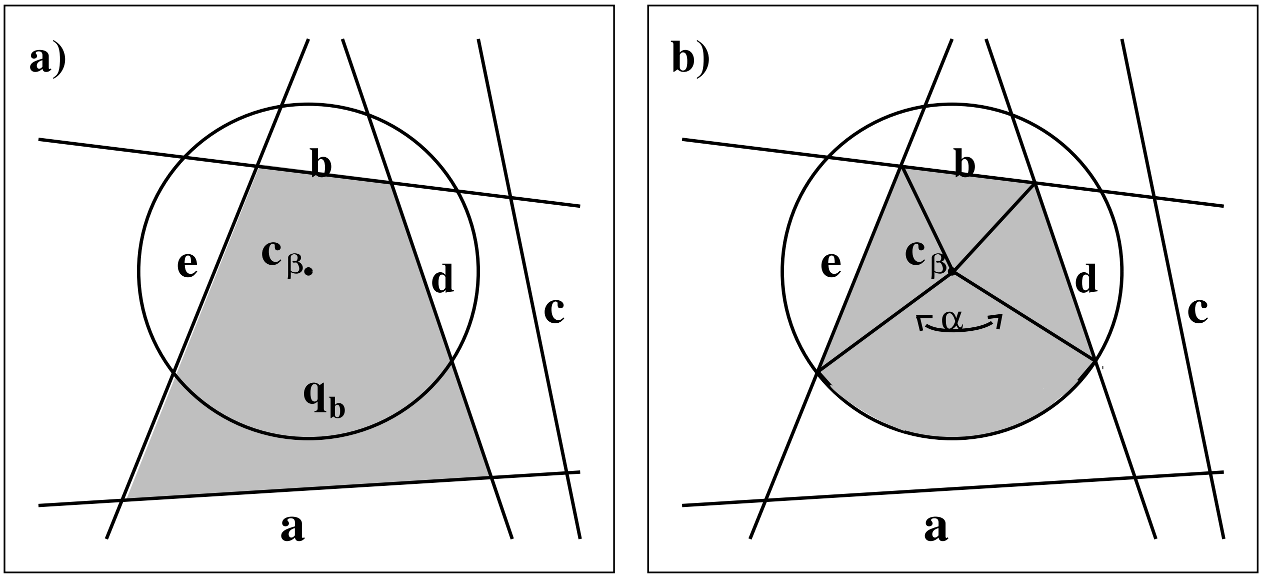

Figure 2: a) The ball β and five half-planes, a through e, defined by balls in $\mathcal{N}_\beta$. The shaded area represents the region satisfying all the inequalities. We want to determine the area of q_b, the intersection of the ball β and all the half-spaces. Notice that only e, b, and d are on the border of q_b. b) The triangulation divides q_b (the shaded region) into three triangles adjoining the centerpoint, and a sector with internal angle α. We can easily compute the area of all these components of q_b.

To compute the volume of q_b, we divide $b \cap (\bigcap_{x \in X} x)$ into tetrahedrons, each of which is incident to the center of ball b, and into sectors where the surface of b is on the surface of q_b. (Aurenhammer [Aur88] shows how to do this in $\mathcal{R}^3$ in $O(k^2)$ time.) Thus, we can compute the volume of q_b by summing the volumes of all such tetrahedrons and adding the volumes of all the sectors.

We illustrate the two-dimensional analog of this algorithm in Figure 2. The lines in the figure are the borders of halfplanes in X; the shaded area represents the region satisfying all the inequalities. The convex-hull procedure determines that the vertices on the hull of $\bigcap_{x \in X} x$ are the points of intersection of halfspaces $e \cap b, b \cap d, d \cap a$ and $a \cap e$. Since the first two of these are contained in the ball β, we compute only the intersection of the border of β with halfplanes e and d. This gives us the triangulation of q_b shown in Figure 2b, along with the sector with internal angle α. (We can determine α from the coordinates of $e \cap \beta, d \cap \beta, c_\beta$, and radius r_β.) Thus, we can compute the length of the circular portion of q_β, and we find the area of q_β by summing the area of the three triangles and the sector.

The results of Halperin and Overmars imply that, given $O(n)$ preprocessing time and randomized perfect hashing, we can determine the constant number k of halfspaces in $\mathcal{N}_b$ in $O(k)$ steps. Thus, the computation of q_b's contribution to the surface area and volume of $\mathcal{U}(B)$

should take $O(k^2)$ time for each ball $b \in B$. (The dominant term is the time needed to divide the three-dimensional polytope into tetrahedrons.) Thus, if k_{max} is an upper bound on the number of elements in any $\mathcal{N}_b$, we can compute the surface area and volume of $\mathcal{U}(B)$ in $O(nk_{max}^2)$ steps. For proteins k_{max} is a constant, so this algorithm runs in randomized linear time.

Finally, we observe that given a list of the adjacent vertices and faces of $F_b \cap b = q_b$, it is easy to compute K. Each q_b corresponds to a vertex of K; any face on the border of q_b defined by the intersection of balls b and b' corresponds to an edge $\overline{bb'} \in K$; and so on. Therefore, our algorithm yields the following interesting corollary:

Corollary 4 *For any arrangement of balls satisfying the protein constraint, $|K| = O(n)$ and K can be computed in $O(n)$ expected time.*

4.2 Discussion and Extensions

We first consider the problem of implementing our algorithm in parallel. In the CREW PRAM model, the core of the algorithm can be implemented in constant time on n processors (or $O(n/p)$ time on a p-processor machine). The preprocessing needed to do intersection queries can be done independently for each ball in constant time, and only constant time is needed to compute each N_b in parallel in any concurrent-read model. Therefore the volume of all the individual q_b can be computed in constant time, given enough processors. The limiting step of the computation is thus summing the volumes of each of the n components to find the total volume.

Our new algorithm has both advantages and disadvantages when compared to that of Edelsbrunner. Our result certainly does not diminish the importance of the inclusion-exclusion result in "The Union of Balls and its Dual Shape." However, our algorithm can calculate the volume and surface area of proteins obeying some simple assumptions in linear time, and can easily be implemented in parallel. The improved speed of our algorithm is clearly an advantage. Furthermore, the simplicity of the approach of summing the volumes of each q_b makes our algorithm more accessible than that of Edelsbrunner, even for sets B that do not satisfy the protein constraint.

One important advantage of the Edelsbrunner algorithm, however, is that it provides additional information about the cavities in the molecule which our algorithm currently does not. Edelsbrunner's algorithm computes not only the volume of $\mathcal{U}(B)$, but the volume of each individual cavity in the molecule. It also determines which surface patches border each cavity. This information, along with the coordinates of the vertices in K, can help determine which cavities of the molecule are large enough to contain solvent molecules such as water. Our algorithm provides only information about the space occupied by the molecule itself; it says nothing about interior and exterior surfaces or volume.

It would therefore be interesting to know if we can determine in linear time the faces of $\mathcal{U}(B)$ that are inside cavities of the molecule and the volumes of each of these cavities. We discuss some approaches to this problem here.

4.2.1 Measuring Cavities in Protein Molecules

To derive information about the cavities of the molecule, it is useful to discuss the subdivision of $\mathcal{R}^3$ by the spheres in B, called the *arrangement* $\mathcal{A}(B)$. A corollary of Halperin and Overmars's main theorem is that for sets of balls B satisfying the protein constraint, the arrangement $\mathcal{A}(B)$ has linear complexity. However, the vertical decomposition of this arrangement into simple pieces may have quadratic complexity. Halperin and Overmars show that by dividing the space into cubes with unit sides $2r_{max}$ and then finding the vertical decomposition of each cube, one can obtain a linear-sized decomposition of $\mathcal{A}(B)$ into simple pieces. Since $\mathcal{A}(B)$ describes the complement of $\mathcal{U}(B)$ as well as $\mathcal{U}(B)$ itself, there is hope that this decomposition will enable us to compute the volume of the cavities in a molecule efficiently.

In order to determine which surface patches are on the interior of the molecule, we need only determine the connected components of the surface. Since each face of the surface is determined by the balls bordering it, our algorithm can easily compute the connected components of the surface by starting at one face and moving to adjacent ones. This can be done in $O(n)$ steps (although it is worthwhile to note that such an algorithm is inherently sequential). Then the only question is which of the components is the exterior one. The exterior component can be found in linear time by choosing some vantage point outside of $\mathcal{U}(B)$ and finding some face of the surface visible from that vantage point. The connected component of the surface containing this face must be on the exterior; all other components are interior surfaces.

The question of computing the *volume* of the cavities is somewhat more difficult. One approach uses the fact that by Corollary 4, we know we can compute K for proteins in linear time. (Note that the corollary says nothing about the size of D, which may still be $\Theta(n \log n)$). So perhaps if we determine which balls border each cavity, we could find a way to fill the cavity with additional balls satisfying the protein constraint without adding too many balls. Then the volume of the original molecule subtracted from the volume of the augmented molecule would be the volume of the cavity.

A more promising approach uses the linear decomposition of $\mathcal{A}(B)$ as described by Halperin and Overmars. We first break up the space into unit cubes. By the protein constraint, each cube intersects only a constant number of balls. The vertical decomposition of the cube has a constant number of pieces, where each piece is essentially a box with some well-defined spherical patches removed. Thus, we should be able to determine the volume of each of these patches in constant time (given a fast method for determining which balls border which piece of the decomposition). If so, we need only determine which patches of the decomposition are in the interior of the molecule and which are exterior. We can do this by checking if any defining spherical patch of the decomposition is on an external face of the molecule (using the linear-time surface area algorithm described above). So we could sum the volumes of all pieces of cavities to get the total description of the cavities of $\mathcal{U}(B)$. Some additional work would be needed to describe individual cavities by re-assembling cavities that span multiple cubes.

5 Ideas for Future Work

The previous sections have contained several suggestions for expanding and improving the individual papers and for improving the linear volume algorithm to provide information about cavities. The following are some additional directions that might be interesting starting points for future research.

5.1 Using Secondary Structure to Bound $|\mathcal{N}_b|$

As we suggested in section 2.2, the constant bound on the number of neighbors per atom from Halperin and Overmars's main theorem appears to be far from tight. We have already suggested some possible methods for improving the bound. Here we suggest two different approaches.

One option is to use whatever information is known about the secondary structure of the molecule to predict the amount of space between atoms. If, for example, we know that the protein consists primarily of α-helices, then we can use information about the space between atoms in α-helices to derive a more accurate bound for at least the main chain atoms. If we know that hairpin-β-motifs are common, we can analyze the spacing in those.

It might even be possible to use general information about how α-helices and β-strands fit together, along with some estimate of what fraction of protein atoms are in one of these structures, to derive general bounds on ρ for all proteins or for certain classes of proteins. These bounds might be useful in error-detection, as discussed below.

Another method might be to analyze the bonding forces acting on the atoms. There are only five elements found in proteins: hydrogen, carbon, nitrogen, oxygen, and sulfur. Perhaps with more chemical information about minimum covalent bond lengths, hydrogen bonds and van der Waals forces for each type of atom, one could derive a better estimate of the degree of overlap expected between each pair of elements. This information might lead to a tighter bound on the number of neighbors.

5.2 Error Detection for X-Ray Crystallography Data

If our estimates of ρ and k are derived from some source other than our x-ray crystallography data for a specific molecule, (for example, if derived as in section 5.1 above), we could use these values to detect and flag some errors in the x-ray crystallography data. One common error [Hal95] occurs when a single atom is described with two slightly different centers, so that there appear to be two different atoms. Certainly, this sort of error might be detected by applying secondary structure information, although it may be difficult to map that information onto the x-ray crystallography data. But if ρ is derived from other sources it would be easy to check for pairs of atoms of the same element whose centerpoints violate the ρ constraint. If the centers are extremely close, or if the atoms are known to move a lot, then it is likely that these pairs represent a single atom. An algorithm that flagged cases in which individual atoms violated the pre-determined constraints might detect other types of errors in the data as well.

6 Learning Proteins with Unreliable Boundary Queries

A recent paper by Blum, Chalasani, Goldman and Slonim [BCGS95] describes a model of learning with a limited type of noise: noise near the boundary of the target concept. Because of the unreliability of x-ray crystallography data, which derives estimates of stationary positions for moving atoms, the unreliable boundary query (UBQ) noise model seems particularly appropriate for representing proteins.

Dyer, Frieze and Kannan [DFK91], in their paper on estimating volumes of convex bodies, suggest using a model in which information about the body is derived using *membership queries*. A membership query for this problem just asks whether a point in $\mathcal{R}^3$ is contained in $\mathcal{U}(B)$ or not. We considered the Dyer, Frieze and Kannan approach to computing volumes as a possible method for improving Edelsbrunner's algorithms but decided that the amount of sampling needed to estimate volumes accurately is inherently superlinear. (The best known bound using these methods is an $O(n^5)$ algorithm for general convex bodies [Kan94]. While this might be improved for unions of "fat" bodies such as spheres, we believe it unlikely that we can approximate the volume of $\mathcal{U}(B)$ in $o(n \log n)$ time using this method.)

However, this work sets a precedent for a membership-query representation of a molecule. Furthermore, we argue that it is a reasonable representation for some design applications. Halperin and Overmars have already shown that intersection queries can be answered in $O(1)$ expected time. (Note that membership queries are simply intersection queries for a query ball with radius zero.) For an application focused on the protein's interaction with a new molecule, defining the protein by a set of membership queries may be reasonable. Finally, we now suggest how a modified membership query model would be a good representation of the type of noise we expect in x-ray crystallography data.

An "unreliable-boundary query" is one in which an oracle answers if the query point is an element of the target concept, but is allowed to answer incorrectly if the point is within a constant distance of the target concept's boundary. The distance measure here depends on the individual task; the Blum, Chalasani, Goldman and Slonim paper describes the application of these queries in both geometric and boolean domains. For proteins the Euclidean distance measure is appropriate. The border size could be determined as some function of the van der Waals radii of adjacent atoms. Then a query would be required to be correct in the interior of the molecule, but could be a bit "fuzzy" near the border. Learning in such a model means deriving an approximation to the target concept which is accurate everywhere except perhaps on the border. In other words, we do not require that the learner have a better idea of the target concept than its teacher, but it must be as good, even though the learner is only allowed a small polynomial number of queries to learn in an exponentially large or even infinite domain.

The errors likely to be derived from x-ray crystallography are precisely those in which the border of the molecule is undetermined. Choosing potentially inaccurate van der Waals radii for the atoms makes it even more likely that such errors will occur. Thus the UBQ model is a good model for representing proteins.

It would therefore be exciting to show that one can accurately learn proteins (or unions of balls satisfying the protein constraint) in this model. Such a result would show that even though the x-ray crystallography data may be inaccurate near the boundary, we derive a good idea of the structure of most of the protein, proving that the model is accurate everywhere except near the boundary. It is possible that in fact we do *not* get an accurate model of protein structure from the noisy data. If so, the learning results can quantify the needed improvements in accuracy. We could therefore use the UBQ model to determine the optimal temperature at which to perform x-ray crystallography so that the noise does not dominate the resulting structure.

7 Acknowledgements

A preliminary version of this paper was submitted as part of the MIT Area Exam in EECS Area II. I would like to thank Albert Meyer, Bonnie Berger, and Tomas Lozano-Perez for serving on my exam committee and for encouraging publication of this paper. I am especially grateful to Danny Halperin for helpful discussions of his paper, for his comments on an earlier draft of this paper, and for the suggestion that his work on decompositions of arrangements might provide a linear algorithm for computing the volumes of cavities, as described in section 4.2. Thanks are due also to Esther Jesurum for valuable comments on an earlier draft of the paper, to Andy Oakland for explaining the graphics algorithms mentioned in the Halperin and Overmars paper, and to Lisa Tucker-Kellogg for helpful discussions of X-ray crystallography and protein structure.

References

[AGR94] Nancy Amato, Michael Goodrich and Edgar Ramos. Parallel algorithms for higher-dimensional convex hulls. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 683–693, 1994.

[Aur88] F. Aurenhammer. Improved algorithms for discs and balls using power diagrams. *Journal of Algorithms*, 9:151–161, 1988.

[BCGS95] Avrim Blum, Prasad Chalasani, Sally Goldman and Donna Slonim. Learning with unreliable boundary queries. To appear in the *Proceedings of the Eighth Annual ACM Conference on Computational Learning Theory*, July, 1995.

[BT91] Carl Branden and John Tooze. *Introduction to protein structure.* Garland Publishing, Inc., New York, 1991.

[CEG+90] K. L. Clarkson, H. Edelsbrunner, L. J. Guibas, M. Sharir and E. Welzl. Combinatorial complexity bounds for arrangements of curves and spheres. *Discrete and Computational Geometry*, 5:99–160, 1990.

[Con83] M. L. Connolly. Solvent-accessible surfaces of proteins and nucleic acids. *Science*, 221:709–713, 1983.

[DE92] C. J. A. Delfinado and H. Edelsbrunner. An incremental algorithm for betti numbers of simplicial complexes. Technical Report *UIUCDCS-R-93-1787*, Computer Science Department, University of Illinois at Urbana-Champaign, 1992.

[DC91] B. R. Donald and D. R. Chang. On the complexity of computing the homology type of a triangulation. In *32nd Annual Symposium on Foundations of Computer Science*, pages 650-661, 1991.

[DFK91] M.E. Dyer, A. Frieze, and R. Kannan. A random polynomial time algorithm for approximating the volume of convex bodies. *Journal of the ACM*, 38:1–17, 1991.

[Dug70] James Dugundji. *Topology.* Allyn & Bacon, Inc., Boston, 1970.

[EFFL95] H. Edelsbrunner, M. Facello, P. Fu and J. Liang. Measuring proteins and voids in proteins. In *Proceedings of the 28th Annual Hawaii International Conference on System Sciences*, pages 256–264, 1995.

[Edl93] H. Edelsbrunner. The union of balls and its dual shape. In *Proceedings of the 9th Annual Symposium on Computational Geometry*, pages 218–231, 1993.

[FV82] James D. Foley and Andries van Dam. *Fundamentals of Interactive Computer Graphics.* Addison-Wesley, Reading, MA, 1982.

[HO94] Dan Halperin and Mark H. Overmars. Spheres, molecules, and hidden surface removal. In *Proceedings of the 10th Annual Symposium on Computational Geometry*, pages 113–122, 1994.

[Hal95] Dan Halperin. Personal communication.

[Kan94] Markov chains and polynomial time algorithms. In *35th Annual Symposium on Foundations of Computer Science*, pages 656–671, 1994.

[LR71] B. Lee and F. M. Richards. The interpretation of protein structure: Estimation of static accessibility. *Journal of Molecular Biology*, 55:379–400, 1971.

[Lei92] F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures.* Morgan Kaufmann, San Mateo, CA, 1992.

[LN90] Nathan Linial and Noam Nisan. Approximate Inclusion–Exclusion. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 260–270, 1990.

[Mez93] P. G. Mezey. *Shape in chemistry: An introduction to molecular shape and topology.* VCH Publishers, Inc., New York, 1993.

[PCG+92] G. Perrot, B. Cheng, K. D. Gibson, J. Vila, A. Palmer, A. Nayeem, B. Maigret and H. A. Scheraga. MSEED: A program for rapid determination of accessible surface areas and their derivatives. *Journal of Computational Chemistry*, 13:1–11, 1992.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry — An Introduction.* Springer Verlag, New York, 1985.

[Ric77] F. M. Richards. Areas, volumes, packing, and protein structures. In *Ann. Rev. Biophys. Bioeng.*, 6:151–176, 1977.

[Sei91] R. Seidel, *et al.* Class Notes for *Computational Geometry*, University of California at Berkeley, Spring 1991.

[LTK95] Lisa Tucker-Kellogg. Personal communication.

[VB93] A. Varshney and F. P. Brooks, Jr. Fast analytical computation of Richards's smooth molecular surface. In *Proc. Visualization*, pages 300–307, 1993.

[VWB94] A. Varshney, W. Wright, and F. P. Brooks, Jr. Estimating the number of unit spheres inside a larger sphere. Technical Report *TR93-039*, Department of Computer Science, University of North Carolina at Chapel Hill, May 1994.

[WHR+87] J. Watson, N. Hopinks, J. Roberts, J. Steitz, and A. Weiner. *Molecular Biology of the Gene.* Benjamin/Cummings, Menlo Park, CA, 1987.

[YE89] V. Yip and R. Elber. Calculations of a list of neighbors in molecular dynamics simulations. *Journal of Computational Chemistry*, 10: 921–927, 1989.