Rapid Approximation to Molecular Surface Area via the Use of Boolean Logic and Look-Up Tables

Scott M. Le Grand and Kenneth M. Merz, Jr.*

Departments of Molecular and Cell Biology and Chemistry, The Pennsylvania State University, University Park, Pennsylvania 16802

Received 11 June 1992; accepted 14 September 1992

We report the development of a new approximate method of calculating molecular surface areas. Our technique is based upon the method of Shrake and Rupley but incorporates several major advances. First, we represent the state of surface points as bits in a bit string so we can utilize Boolean operations to simultaneously turn off multiple test points in one Boolean AND operation. Second, we use a series of Boolean mask look-up tables to reduce the time complexity of the calculation of molecular surface area down to the same magnitude as doing a potential energy evaluation. When we use a 256 surface point sphere for all of the atoms in BPTI, a 454 nonhydrogen atom protein, and a 1.4-Å solvent probe, we in general underestimate the total solvent-accessible surface area (SASA) by approximately 1.25% with a correlation coefficient of 0.9990 over a wide range of conformations. The average CPU time required to calculate the SASA of a BPTI conformer is 0.58 s on an SGI 4D/220 workstation. We also describe a method by which we can calculate an approximate finite difference SASA gradient for BPTI in 0.79 of CPU time. © 1993 by John Wiley & Sons, Inc.

INTRODUCTION

The solvent-accessible surface area (SASA) of a molecule can be used to approximate its free energy and enthalpy of hydration,[1-3] measure its degree of compaction, and generate a 3-D profile to assist sequence data base searches.[4] There are now many algorithms to calculate this.[5-20,22] These algorithms calculate the net surface area of a set of overlapping spheres of various radii. Each sphere represents an atom. Unfortunately, all but two of these algorithms are prohibitively slow for incorporation into molecular dynamics and minimization calculations at each step on anything short of a supercomputer. Wodak and Janin[18] derived a rapid approximation to SASA utilizing only pairwise interactions between atoms. Their algorithm could approximate the SASA of proteins if each amino acid residue was treated as a single united sphere rather than a collection of atomic spheres. Häsel et al.[17] created a parameterized version of this algorithm for use on small molecules. However, it does not give good results for proteins when using an all- or united-atom representation.[21] Perrot et al. recently described MSEED, a rapid analytic method based upon the method of Connolly.[20] MSEED can be used to rapidly calculate both the SASA and its derivatives but will be inaccurate if the molecular surface is not one continuous piece (i.e., if there is a cavity inside the molecule) or if the atomic radii used are too small. In this article, we present an approximation to the method of Shrake and Rupley[7] based upon the use of look-up tables and Boolean logic. Using our algorithm, we can calculate the SASA or BPTI (454 nonhydrogen atoms) to within 1.2% accuracy in less than 1 s of CPU time on an IRIS 4D/220, a four MFLOP CPU workstation. We can also calculate the finite difference SASA gradient of BPTI in 0.79 s on the same machine. This is roughly six times faster than the MSEED algorithm and appropriate for use in situations where MSEED may fail. This makes our algorithm a useful procedure for energy minimization calculations and molecular dynamics simulations.

*Author to whom all correspondence should be addressed.

METHODS

Similar to the method of Shrake and Rupley,[7] we represent the surface of each spherical atom a_i in an n atom molecule by an array of l points (hereafter known as surface points) distributed on the surface of the sphere of radius r_i, which is the van der Waals radius of a_i (Fig. 1). The configuration of the l surface points is determined by minimizing a potential (1) where β_{xy} is the arc length in degrees on the unit sphere between surface points x and y. U is minimal when the l surface

$$U = \sum_{x<y} (180 - \beta_{xy})^2 \qquad (1)$$

points are maximally dispersed on the surface of the unit sphere. Unfortunately, the optimal distribution

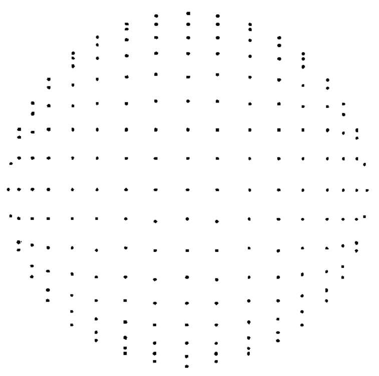

Figure 1. Atomic surface approximated by an array of test points.

of l surface points on a sphere is one of the unsolved problems of pure mathematics.[23] However, U can be approximately minimized by integrating the equations of motion of the surface points on the unit sphere in response to U. Therefore, we minimized U from many different starting configurations derived from both tesselated polyhedra and random placement on the sphere surface and used the configuration that produced the lowest U for all calculations presented here. The molecular surface area is now approximated by determining the status of every surface point. A surface point p_{ij} (where i is the atom index and j is the index of the point on a_i) is considered exposed if it is outside the sphere of every other atom in the molecule and buried otherwise. Finally, we determine the exposed surface area of each atom a_i using (2), where l_{exposed} is the number of exposed surface points on a_i and l_{total} is the total number of surface points on a_i.

$$A_i = \frac{4\pi r_i^2 l_{\text{exposed}}}{l_{\text{total}}}. \tag{2}$$

At this point, if we were to calculate the surface area of a molecule with n atoms using the l surface points on each atom it would require nkl computations, where k is the average number of neighboring atoms that intersect with each atom. It has been estimated to be approximately 40.[19] The accuracy of the calculation depends upon l: The more surface points used, the better the accuracy. For BPTI, a 58 amino acid protein that has 454 nonhydrogen atoms, this would require 4,648,960 distance computations for l equal to 256. This would require approximately 1 min of CPU time on an SGI IRIS 4D/220 and be accurate to within about 1%.

Because the surface points of each atom are either exposed or buried, they have only two possible states. This means that we can represent the state of a_i's surface points at any time by an $l/8$-byte bit string b_i, where a 1 is b_{ij} indicates a surface point p_{ij} is exposed and a zero in b_{ij} indicates it is unexposed (Fig. 2). Once we do this, calculating the area removed by the intersection of a_i with its k neighbors

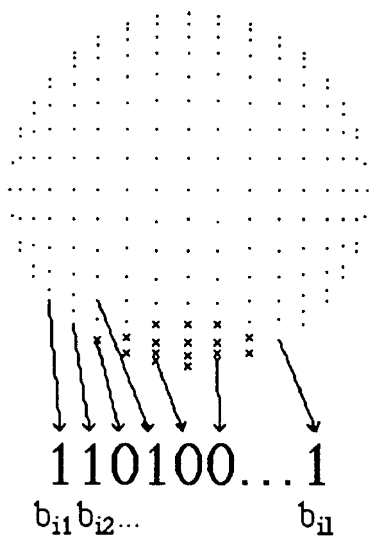

Figure 2. Mapping of the state of l test points on the surface of a sphere into an $l/8$ = byte bit string. Dots represent exposed points and xs buried points.

is isomorphic to k Boolean AND operations on b_i. For each intersection of a_i with an atom a_q, a mask m_{iq} that has zeroes wherever surface points of a_i are within a_q, and ones elsewhere must be generated. ANDing m_{iq} with b_i will mark all surface points that are within a_q as buried and have no effect on those that are not buried by it (3). Once a surface point has been marked as buried, no further AND operations will affect it. Because AND operations are both

$$b_i = (11 \ldots 1) \,\&\, m_{i1} \,\&\, m_{i2} \,\&\, \ldots \,\&\, m_{ik} \tag{3}$$

commutative and associative, the actual order in which these operations are performed is irrelevant. The end result will be identical. This makes this algorithm easily vectorized. The final SASA of a_i is calculated by setting l_{exposed} for a_i to $bc(b_i)$ where $bc(x)$ is the number of on bits in bit string x, and plugging this value into (2).

Calculating m_{iq} requires two applications of the law of cosines. First, θ_{iq} is calculated by (4) (Fig. 3).

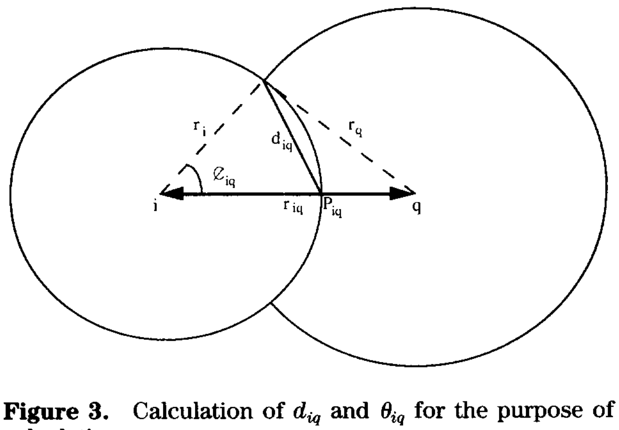

Figure 3. Calculation of d_{iq} and θ_{iq} for the purpose of calculating m_{iq}.

Next, d_{iq} is calculated from θ_{iq} by (5). All surface points

$$\cos\theta_{iq} = \frac{r_i^2 + r_{iq}^2 - r_q^2}{2r_i r_q} \tag{4}$$

$$d_{iq} = r_i\sqrt{2(1 - \cos\theta_{iq})} \tag{5}$$

on a_i within d_{iq} distance units from P_{iq} (which is the point on the surface of a_i on the line connecting the centers of a_i and a_q) are buried by a_q. As described above, we now set all bits in m_{iq} representing surface points buried by a_q to zero and the rest to one. Next, we AND m_{iq} with b_i. At this point, m_{qi} can be similarly calculated and applied to b_q. If these masks had to be generated each time we calculated the intersection of two spheres, then this method would still require *nkl* operations. However, the masks required for the operation can be calculated ahead of time and stored in a look-up table for instant retrieval when needed. First, note that by (5) d_{iq} can be made to range from 0–2 independent of atomic radius by removing the factor of r_i (6). This will allow us to handle all intersections with spheres of arbitrary radius using one set of masks generated on the

$$d_{iq} = \sqrt{2(1 - \cos\theta_{iq})} \tag{6}$$

sphere. Next, we generate masks for d_{iq} values ranging from 0–2 with an increment of 0.01 (Fig. 4). This generates 200 masks for each of the l surface points for a total of $200l$ masks. This will cover all possible overlaps that will be encountered in a surface area calculation with one caveat, which we discuss next.

After we calculate all the possible masks for intersections centered exactly on a surface point of a sphere, we use them to approximate intersections that are not exactly centered on surface points. To do this, we determine the closest surface point to a large number of spherical polar coordinate positions on the sphere and store this information in a second look-up table. We use increments of 1.8° for both θ and ϕ. To calculate the intersection of a_i with a_q, we first determine the spherical polar coordinates of the point of atomic intersection and then map this over

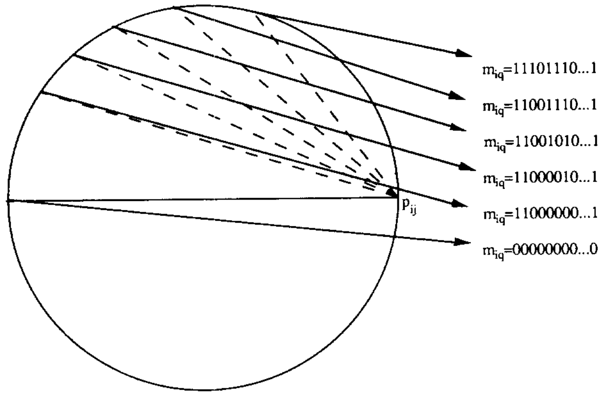

Figure 4. Generation of the look-up table of Boolean masks of p_{ij}.

to the closest surface point p_{ij} via this second look-up table and then calculate d_{iq} by (4) and (6). Using d_{iq}, we retrieve the appropriate mask m_{iq} from the p_{ij} look-up table and AND it with b_i. Next, we set θ to 180° − θ and ϕ to ϕ + 180° to similarly determine m_{qi}. This introduces a small amount of error into our calculation. However, we have never seen it exceed 3% while using 256 surface points.

Overall, this reduces the magnitude of SASA calculation from *nkl* to *nk*. This means that it can be incorporated easily into molecular dynamics, conformational search, and energy minimization routines.

SASA gradients are useful for incorporating atomic solvation parameters into molecular dynamics calculations. These are usually calculated through the use of various differential geometry theorems but are in general CPU intensive.[22] However, on a 30 MFLOP machine MSEED calculates the SASA and SASA gradient of BPTI in 0.71 s. Although we do not use any concepts from differential geometry, our algorithm can be modified to calculate finite difference approximations to the SASA gradient. In our algorithm, the partial derivative of A_i with respect to any of the three coordinate axes is calculated by determining the change in A_i, $\mathrm{d}A_i$, upon moving a_i a distance $\mathrm{d}r$ along a coordinate axis. The ratio of these two differentials is an approximation to the true partial derivative (7). This will be referred to as the *finite difference SASA gradient*. If one

$$\frac{\partial A_i}{\partial r} \approx \frac{\mathrm{d}A_i}{\mathrm{d}r} \tag{7}$$

further assumes that relative to a_i the only change in orientation of a_q upon moving a_i a distance d_r along a coordinates axis is in r_{iq}, then one can speed this calculation at the cost of some accuracy. This will be referred to as the *approximate finite difference SASA gradient*. Finally, we assume that the gradient terms of all fully buried atoms are zero.

Overall memory requirements are proportional to l^2. For all calculations reported here, we use 256 surface points. The look-up tables for this require 1.6 Mb of memory and $32n$ bytes are required to store the status of each bit sting b_i. If memory is a concern, one can reduce the look-up table storage requirement by a factor of four by using 128 test points. The look-up table for using 128 surface point requires ~400 K and $16n$ bytes are required to store the status of each bit string b_i.

RESULTS

Figure 5 illustrates the success of our approximation at calculating the SASA of 200 randomized conformations of BPTI, a 454 nonhdyrogen atom protein. As compared to the exact SASA (as calculated by the exact method of Dodd and Theodorou[16]), we

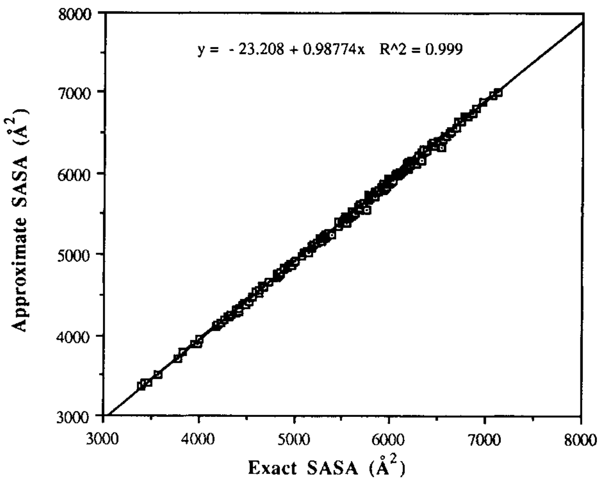

Figure 5. Plot of our approximate SASA vs. the exact SASA of 200 random conformations of BPTI using 256 test points.

underestimate it by 1.25% with a correlation coefficient of 0.9990. However, on average our algorithm required only 0.58 s of CPU time to calculate the SASA of each BPTI conformer as compared to the method of Dodd and Theodorou, which required approximately 44 s. Clearly, this algorithm is fast enough for use in conformational search, molecular dynamics, and energy minimization.

The finite difference SASA gradient can be calculated in 2.66 s of CPU time. The calculation of the approximate finite difference SASA gradient is faster at 0.79 s of CPU time but introduces some error. For BPTI, 80% of the approximate finite difference gradient terms are exactly the same as finite difference gradient terms, 7% are off by one surface point, and 13% are off by greater than one surface point. In a molecular dynamics simulation, this will introduce some error, but the body of gradient terms will be correct and the small magnitude of atomic solvation parameters should dampen the effect of this error. We are currently testing this hypothesis, as well as using temperature coupling and Brownian dynamics in our simulations to further reduce the effect of the error. To calculate the BPTI SASA and approximate finite difference SASA gradient, 1.37 s of CPU time on a four MFLOP processor were required. MSEED required 0.71 s to calculate the same quantities on a 30 MFLOP processor. This means that our approach is approximately six times faster than MSEED when the SASA gradient is desired. Because no figures were provided for SASA calculations alone by MSEED, there is no way to compare the two algorithms for the raw SASA calculation.

CONCLUSIONS

We described a rapid method of calculating molecular surface areas and their gradient. Although the gradient terms have some error associated with them, we believe they are suitable for use in energy minimization, molecular dynamics, and conformational search. Source code in C is available upon request to the authors.

The authors thank Lawrence Dodd and Dr. Doros Theodorou for providing them with their surface area calculation software, Drs. Randy Zauhar and David Heisterberg for mathematical assistance, Dr. Todd Yeates for a valuable suggestion about the SASA gradient calculation, and the Pittsburgh Supercomputing Center for CRAY YMP time. Finally, they thank the Office of Naval Research for supporting this research (N00014-90-3-4002).

References

1. D. Eisenberg and A.D. McLachlan, *Nature,* **319,** 199 (1986).
2. T. Ooi, M. Oobatake, G. Nemethy, and H.A. Scheraga, *Proc. Natl. Acad. Sci. USA,* **84,** 3086 (1987).
3. J. Vila, M. Vasquez, and H.A. Scheraga, *Proteins,* **10,** 218 (1991).
4. J. Bowie, R. Luthy, and D. Eisenberg, *Science,* **253,** 164 (1991).
5. B. Lee and F.M. Richards, *J. Mol. Biol.,* **55,** 379 (1971).
6. T.J. Richmond and F.M. Richards, *J. Mol. Biol.,* **119,** 537 (1978).
7. A. Shrake and J.A. Rupley, *J. Mol. Biol.,* **79,** 351 (1973).
8. M. Connolly, *J. Appl. Cryst.,* **16,** 548 (1983).
9. M. Connolly, *J. Am. Chem. Soc.,* **107,** 1118 (1985).
10. M.Y. Pavlov and B.A. Fedorov, *Biopolymers,* **22,** 1507 (1983).
11. C.J. Alden and S.H. Kim, *J. Mol. Biol.,* **132,** 411 (1979).
12. M.H. Zehfus, J.P. Seltzer, and G.D. Rose, *Biopolymers,* **24,** 2511 (1985).
13. A.Y. Meyer, *J. Comp. Chem.,* **9,** 18 (1988).
14. H.R. Karfunkel and V. Eyraud, *J. Comp. Chem.,* **10,** 628 (1989).
15. H. Wang and C. Levinthal, *J. Comp. Chem.,* **12,** 868 (1991).
16. L.R. Dodd and D.N. Theodorou, *Mol. Phys.,* **72,** 1313 (1991).
17. W. Häsel, T.F. Hendrickson, and W.C. Still, *Tet. Comp. Meth.,* **1,** 103 (1988).
18. S.J. Wodak and J. Janin, *Proc. Natl. Acad. Sci. USA,* **77,** 1736 (1980).
19. G. Perrot and B. Maigret, *J. Mol. Graphics,* **8,** 141 (1990).
20. G. Perot, B. Cheng, K.D. Gibson, J. Vila, K.A. Palmer, A. Nayeem, B. Maigret, and H.A. Scheraga, *J. Comp. Chem.,* **13,** 1 (1992).
21. S.M. Le Grand and K.M. Merz Jr., unpublished results.
22. T.J. Richmond, *J. Mol. Biol.,* **178,** 63 (1984).
23. H.S.M. Coxeter, *Introduction to Geometry,* John Wiley & Sons, New York, 1961.