Poleksic *Algorithms Mol Biol (2015) 10:27*
DOI 10.1186/s13015-015-0058-0

AMB ALGORITHMS FOR MOLECULAR BIOLOGY

RESEARCH

Open Access

A polynomial time algorithm for computing the area under a GDT curve

Aleksandar Poleksic*

Abstract

Background: Progress in the field of protein three-dimensional structure prediction depends on the development of new and improved algorithms for measuring the quality of protein models. Perhaps the best descriptor of the quality of a protein model is the *GDT* function that maps each distance cutoff θ to the number of atoms in the protein model that can be fit under the distance θ from the corresponding atoms in the experimentally determined structure. It has long been known that the area under the graph of this function (*GDT_A*) can serve as a reliable, single numerical measure of the model quality. Unfortunately, while the well-known *GDT_TS* metric provides a crude approximation of *GDT_A*, no algorithm currently exists that is capable of computing accurate estimates of *GDT_A*.

Methods: We prove that *GDT_A* is well defined and that it can be approximated by the Riemann sums, using available methods for computing accurate (near-optimal) *GDT* function values.

Results: In contrast to the *GDT_TS* metric, *GDT_A* is neither insensitive to large nor oversensitive to small changes in model's coordinates. Moreover, the problem of computing *GDT_A* is tractable. More specifically, *GDT_A* can be computed in cubic asymptotic time in the size of the protein model.

Conclusions: This paper presents the first algorithm capable of computing the near-optimal estimates of the area under the *GDT* function for a protein model. We believe that the techniques implemented in our algorithm will pave ways for the development of more practical and reliable procedures for estimating 3D model quality.

Keywords: Protein structure, Structure modeling, Structure prediction, Model quality

Background

Advances in the area of protein three-dimensional structure prediction depend on the ability to accurately measure the quality of a protein model. One of the most popular and most reliable measure of the protein model quality is *GDT_TS*. It is defined as the average value of GDT_P_θ computed for four distance cutoffs $\theta = 2^i$, $i = \overline{0,3}$, where GDT_P_θ is the percentage of model residues (represented by their C_α atoms) that can be placed under θ ångströms from the corresponding residues in the experimental structure [1, 2]. In a "high-accuracy" version of *GDT_TS*, denoted by *GDT_HA*, the distance cutoffs are cut in half ($\theta = 2^i$, $i = \overline{-1,2}$) [3]. In both approaches, the underlying assumption is that the experimental (crystallographic or NMR) structure is close to the real (native) structure (which is sometimes not true due to experimental errors).

Several methods exist for computing *GDT_TS*. The LGA algorithm [4] can estimate *GDT_TS* quickly, but those estimates deviate from the true *GDT_TS* values in about 10 % of the cases [5]. Rigorous algorithms for computing *GDT_TS* have also been developed [6–9], but they are computationally much more expensive.

The *GDT_TS* is commonly interpreted as an approximation of the area under the *GDT* curve, denoted by *GDT_A* [10–12]. Unfortunately, since the measure is approximated using the *GDT* function values at only several distance cutoffs, the errors in the area approximation are large. As we demonstrate later, *GDT_TS* is not only overly sensitive to small but also insensitive to large changes in the protein model's coordinates.

In this paper, we present a polynomial time algorithm for computing *GDT_A*. Our method runs on the order

*Correspondence: poleksic@cs.uni.edu
Department of Computer Science, University of Northern Iowa, 305 ITTC, Cedar Falls, Iowa 50613, USA

$\widetilde{O}(n^3)$, where n represents the length of the protein model (and $\widetilde{O}$ hides the log factor). The algorithm returns "near-optimal" *GDT_A* scores, meaning that the errors in our estimates can be made arbitrary small i.e., smaller than any upfront specified vale. Although our method is theoretical, we believe that its parallel implementations, coupled with carefully designed speed up techniques, can result in a practical and widely used software tool.

The rest of this paper is structured as follows. First, we present three examples that illustrate drawbacks of *GDT_TS* and advantages of *GDT_A*. Then, we place our theory on a firm mathematical ground, which enables us to formally define the *GDT_A* computation problem. Finally, we describe the actual algorithm for *GDT_A* and provide its running time analysis.

Methods

Definition of the GDT function

The *GDT* function is a mapping that relates each distance cutoff θ to the percentage of model residues that can be placed at distance $\leq\theta$ from the corresponding residues in the experimentally determined structure. The graph of a *GDT* function provides a valuable insight into the quality of a protein model (Fig. 1). More specifically, the closer the graph runs to the horizontal axis (in other words, the smaller the area under the graph), the better the model.

As a single numerical measure of the model quality, *GDT_TS* is extensively used at CASP to rank different models for the same target [13, 14]. Since it represents the average of GDT_P_θ at several distance cutoffs, *GDT_TS* is often viewed as an approximation of the area under the *GDT* curve (*GDT_A*) [10–12]:

$$GDT\_TS = \sum_{i=0}^{3} GDT\_P_{2^i}. \quad (1)$$

However, as we demonstrate below, such a sparse sampling of the values of *GDT* function compromises the reliability of *GDT_TS*.

In our first example, we analyze the protein model for the target T0482, submitted by the group TS208 at CASP8 (Fig. 1). The *GDT_TS* score of this particular model was not even among the best dozen at CASP8, despite the fact that it fits the largest number of residues at distance $\leq\sim 4$ from the corresponding residues in the experimental structure. In fact, the blue model (Fig. 1a) can be superimposed onto the experimental structure so that all of its residues are at distance ≤ 8 from the residues in the experimental structure (Fig. 1b), while no such superposition exists for any other model, even for the distance cutoff of 10Å. Interestingly, according to the MAMMOTH algorithm [15], the blue model is the best model for this particular target, while the DALI [16] algorithm ranks it as the second best.

Although it is impossible to tell whether #13 *GDT_TS* rank is more or less fair than #1 and #2 rank assigned by MAMMOTH and DALI, respectively, it is also not difficult to see that the ranking by the area under the *GDT* plot (*GDT_A*) would serve as a good compromise between these extremes.

The next two examples illustrate further disadvantages of *GDT_TS*. As seen in Fig. 2, better *GDT_TS* scores can be assigned to obviously worse models. Moreover, as demonstrated in Fig. 3, very similar models can have significantly different *GDT_TS* scores.

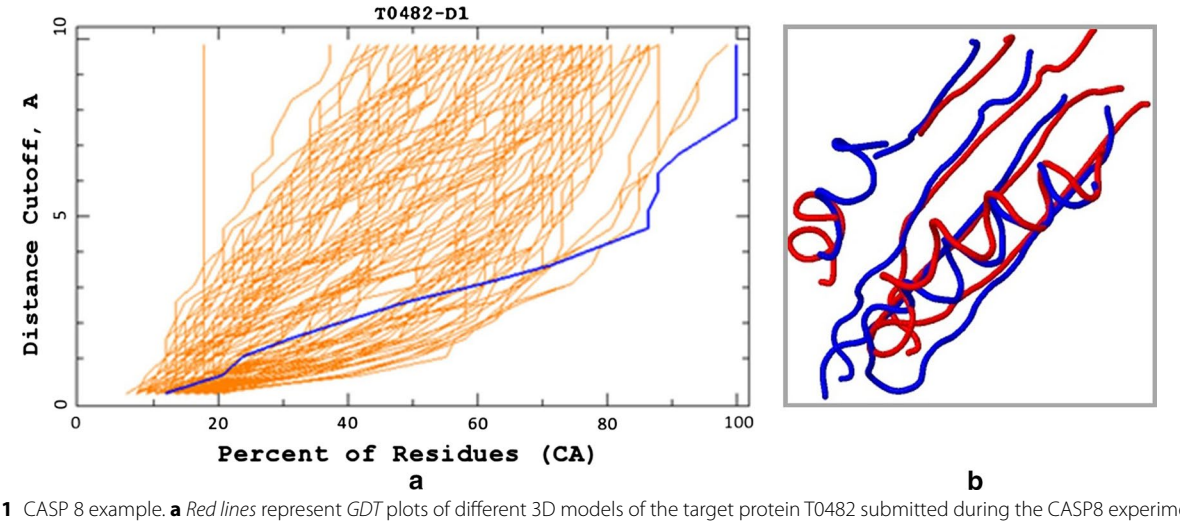

Fig. 1 CASP 8 example. **a** *Red lines* represent *GDT* plots of different 3D models of the target protein T0482 submitted during the CASP8 experiment. The model submitted by the group TS208 is represented in *blue*. **b** Structural alignment of the model submitted by the group TS208 (*blue*) and the experimental structure (*red*) over 67 residues evaluated by the CASP assessors

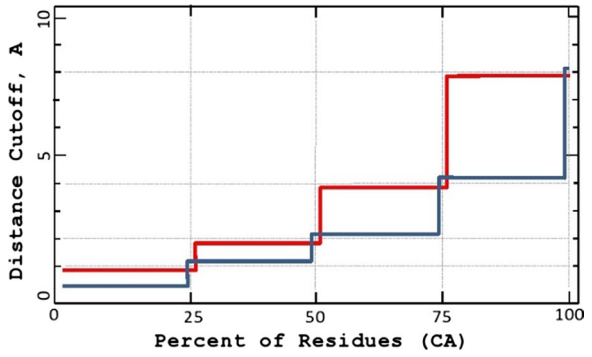

Fig. 2 Insensitivity of GDT_TS. This theoretical example shows no sensitivity of *GDT_TS* to large variations in model quality. Surprisingly, the *red model* has a better *GDT_TS* score than the better *blue model*, even though it is worse by all standards. Notice that, unlike *GDT_TS*, the *GDT_A* measure is not skewed by the values at the cutoff points 1, 2, 4 and 8 Å. In fact, the *GDT_A* score of the blue model is twice as good as that of the *red model*

Mathematical formalism

Strictly speaking, the *GDT* function is not well-defined. Zooming into the plot of the model highlighted in Fig. 1a, we see a set of many small vertical segments, meaning that each point on the horizontal axis is mapped to zero or more points on the vertical axis (Fig. 4). On the other hand, the inverse function (mapping each distance cutoffs θ to the percentage of residues in the model structure that can be fit under the distance θ from the corresponding residues in the experimental structure) is obviously well defined. This allows us to define the area under the *GDT* plot as the complement of the area under the inverse function:

$$GDT\_A = Total\_Area - \overline{GDT\_A} \tag{2}$$

where *Total_Area* represents the area of the rectangular region under consideration (100×10). We start our mathematical formalism by first defining a protein structure.

Definition 1 A *protein structure* a is a sequence of points in the three dimensional Euclidean space $\mathbb{R}^3$

$$a = (a_1, \ldots, a_n). \tag{3}$$

The sequence elements a_i can represent individual atoms, but it is more typical (in particular in protein structure prediction experiments) to assume that each point a_i corresponds to the alpha-carbon atom of the protein's *ith* amino acid.

In what follows, we formally define the *GDT* function [17]. For simplicity of presentation, we will modify the codomain of *GDT* to represent the "fraction of residues" (ranging from 0 to 1) instead of "percentages of residues" (ranging from 0 to 100). We note that this simple rescaling of the ordinate values will have no effects on the results obtained in our study.

Definition 2 Let $a = (a_1, \ldots, a_n)$ be a protein structure consisting of n amino acids, let $b = (b_1, \ldots, b_n)$ be a 3D model of a, and let $\overline{\theta}$ be a positive constant. The *Hubbard function (or GDT function)* is the function $H_b : [0, \overline{\theta}] \to (0, 1]$, defined by $H_b(\theta) = \max_\tau |\{i \mid \|a_i - \tau(b_i)\| \leq \theta\}|/n$, where $\| \ \|$ denotes the Euclidean norm on $\mathbb{R}^3$ and τ is a rigid transformation (a composition of a rotation and a translation).

Theorem 1 *H_b is a stepwise function with finitely many steps $\theta_1, \ldots, \theta_k$, $1 \leq k \leq n-1$.*

Proof Since H_b is monotony non-decreasing and since the range of H_b is a finite subset of (0,1], it follows that H_b must be a stepwise function. To complete the proof, we note that the number of steps in H_b matches the size of its range, which does not exceed $n-1$, where n is the length of b.

For simplicity of presentation, from now on (and whenever the model b is implied), we will omit the subscript in H_b and denote the Hubbard function only by H.

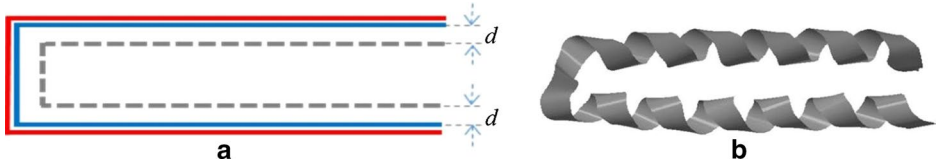

Fig. 3 Oversensitivity of GDT_TS. **a** A four helix bundle-like (toy) protein (*dashed grey line*) along with two of its, almost identical, models (*red* and *blue*). A realistic example of such a target protein (PDB ID: 1JM0A) is shown on the *right* (**b**). In this example, we assume that the protein and its models are extended to the right to include 100 or more residues. Note that, if $d \in \{1$ Å, 2 Å, 4 Å, 8 Å$\}$ then the *GDT* score of the *blue model* is significantly higher than that of the *red model*. For instance, if $d = 2$ Å, then the blue model has the *GDT_TS* score of about 87.5 since ~50 % all of its residues can be fit at distance ≤ 1 Å and 100 % under each distance 2, 4 and 8 Å from the corresponding residues in the experimental structure (*dashed grey*). On the other hand, the *GDT_TS* score of the *red model* is only about 75, since only ~50 % of the *red model's* residues can be placed under 1 and 2 Å and 100 % under 4 and 8 Å. In fact, no matter how close the *red model* gets to the *blue model*, its *GDT_TS* score will never improve. Note also that the *blue* and *red models* have almost identical *GDT_A* scores, since *GDT_A* is not sensitive to small coordinate changes

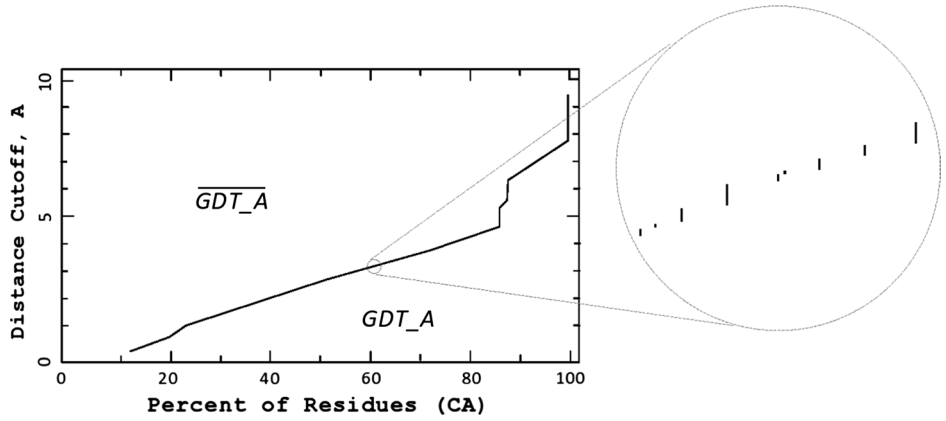

Fig. 4 A closer look at the GDT_TS function. Zooming into the *GDT* plot of the model highlighted in Fig. 1. What appears to be the graph of a continuous function is, in fact, a set consisting of many separated *vertical line* segments

Algorithm for GDT_A

The area under H is the sum of the areas of the rectangular regions $(\theta_i)(\theta_i - \theta_{i-1})$:

$$Area = \sum_{i=1}^{k+1} H(\theta_i)(\theta_i - \theta_{i-1}), \tag{4}$$

where $\theta_0 = 0$ and $\theta_{k+1} = \bar{\theta}$ (Fig. 5). It would be trivial to compute *Area* had we known all θ_i and all function values $H(\theta_i)$. Unfortunately, even if we knew the step points θ_i, it would be computationally very difficult to compute the function values at them, since the best to date algorithm for computing $H(\theta_i)$ runs on the order of $O(n^7)$ [7]. Hence, we resort to using the Riemann sums to approximate (instead of to compute exactly) the area under the graph of H.

The following definition and an accompanying theorem can be found in virtually any mathematical analysis textbook.

Definition 3 If $f : [a, b] \to \mathbb{R}$ is a function then $R = \sum_{i=1}^{n} v_i(x_i - x_{i-1})$, where $a = x_0 < x_1 < \cdots < x_n = b$ is the partition of the interval $[a, b]$ and v_i denotes the supremum of f over $[x_{i-1}, x_i]$, is called the *upper Riemann sum* of f on $[a, b]$.

Theorem 2 *Let f be a real, non-decreasing, Riemann integrable function on an interval $[a, b]$. Then*

$$\left| \int_a^b f(x)dx - R \right| < \Delta x\big(f(b) - f(a)\big), \tag{5}$$

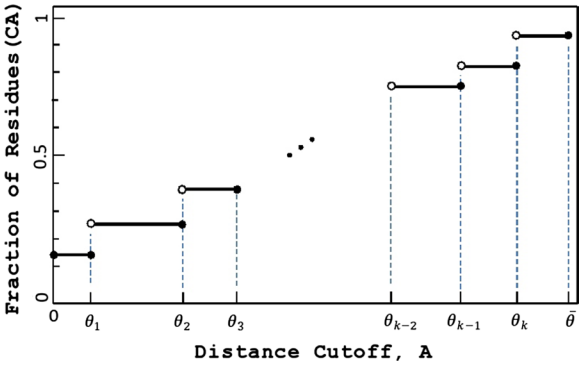

Fig. 5 The general shape of the Hubbard function. Notice that the values θ_i along with the function values $H_b(\theta_i)$, $i = \overline{1,k}$, uniquely determine the area under the graph of H_b. At the biannual CASP experiment, $\bar{\theta} = 10$

where

$$R = \sum_{i=1}^{n} v_i(x_i - x_{i-1}) \tag{6}$$

is the upper Riemann sum of f the and $\Delta x = \max_i(x_i - x_{i-1})$.

Observe that, since H_b is piecewise continuous, it must be integrable on $[0, \bar{\theta}]$. Thus, the area under the graph of H is

$$Area = \int_0^{\bar{\theta}} H(\theta)d\theta. \tag{7}$$

To approximate *Area* with a Riemann sum, one can define the partition points $\epsilon, 2\epsilon, \ldots, m\epsilon$, where $m = \lceil \bar{\theta}/\epsilon \rceil$ (Fig. 6) and then compute an estimate $Area(\epsilon)$ of *Area* as

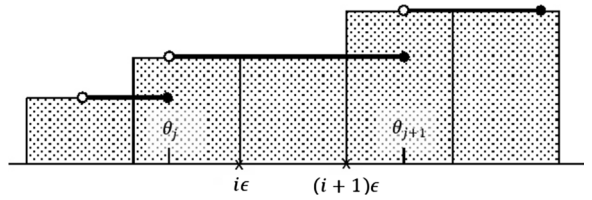

Fig. 6 Approximation of the Hubbard function by the Riemann sum. *Area*(ϵ) is the sum of the areas of all rectangular regions

$$Area(\epsilon) = \sum_{i=1}^{m} \epsilon H(i\epsilon) \tag{8}$$

The error $|Area - Area\ (\epsilon)|$ in the estimate (8) is below 2ϵ. Up to a half of this error is due to the error in the Riemann sum with the remaining error being due to the possible placement of the last partition point $m\epsilon$ outside the interval $[0, \bar{\theta}]$.

Unfortunately, computing the area estimates according to (8) is still a challenging problem, because (as we mentioned above), there is no computationally effective procedure for finding the function values $H(i\epsilon)$. To circumvent the problem, we utilize an efficient algorithm capable of computing the lower bound estimates H_i of $H(i\epsilon)$, satisfying $H((i-1)\epsilon) \le H_i \le H(i\epsilon)$, $i = \overline{1,m}$. We then compute an estimate $\widetilde{Area}(\epsilon)$ of *Area* as

$$\widetilde{Area}(\epsilon) = \sum_{i=1}^{m} \epsilon H_i. \tag{9}$$

Since $\left|\widetilde{Area}(\epsilon) - Area(\epsilon)\right| < 2\epsilon$, it follows that $\widetilde{Area}(\epsilon)$ is a 4ϵ-approximation of *Area*. Below we show how to compute all H_i's, and, in turn, $\widetilde{Area}(\epsilon)$ in time $O\left(n^3 logn/\epsilon^6\right)$, where n is the length of b. Our algorithm takes advantage of an efficient procedure for computing near optimal *GDT_TS* values [5].

Let $T(b)$ denotes the image of the model structure b under the transformation T. Denote by $MAX(T,\theta)$ the largest fraction of residues from $T(b)$ that are at distance $\le\theta$ from the corresponding residues in the experimental structure a. To find each H_i, it is enough to compute a rigid body transformation T_i satisfying $H((i-1)\epsilon) \le MAX(T_i, i\epsilon) \le H(i\epsilon)$.

Denote by T_θ a transformation that places a largest subset b_θ of residues from b at distance $\le\theta$ from the corresponding residues in the experimental structure. Given T_θ, one can easily compute b_θ by calculating all n distances between the residues a_i and $T_\theta(b_i)$. Note that $P(T_\theta, \theta) = H(\theta)$. We approximate the transformation T_θ by a so-called "near-optimal" transformation i.e., a transformation that places at least as many residues from the model structure under distance $\theta + \epsilon$ as the optimal transformation T_θ places under the distance θ. From now on, we will use T_θ^ϵ to denote a "near-optimal" transformation and the corresponding set of residues will be denoted by b_θ^ϵ. Observe that $P\left(T_\theta^\epsilon, \theta+\epsilon\right) \ge P(T_\theta, \theta) = H(\theta)$.

Building upon any procedure for computing T_θ^ϵ, one can develop an algorithm for $\widetilde{Area}(\epsilon)$ by substituting $P\left(T_{\theta_i}^\epsilon, \theta_i+\epsilon\right)$ for H_i in (10), where $\theta_i = (i-1)\epsilon$. Several existing methods can be modified and made suitable for finding T_θ^ϵ. The most efficient such method relies on the concept of "radial pair" [5].

Definition 4 Let $S = \{s_1, \ldots, s_n\}$ be a set of points in the three-dimensional Euclidean space. An ordered pair of points (s_i, s_j) is called a *radial pair* of S if s_j is the furthest point from s_i among all points in S.

Theorem 3 *Let T_1 and T_2 be two transformations and let (s_k, s_l) be a radial pair of S. If $\|T_1(s_k) - T_2(s_k)\| < \epsilon/3$ and $\|T_1(s_l) - T_2(s_l)\| < \epsilon/3$ then there exists a rotation R around the line through $T_1(s_k)$ and $T_1(s_l)$ such that $\|R(T_1(s_p)) - T_2(s_p)\| < \epsilon$, for any s_p in S. The rotation R can be found in time O ($nlogn$), where n is the size of S.*

A proof of the above theorem can be found in [5]. The algorithm for finding R is fairly straightforward and it relies on the so-called *plane-sweep* approach [18].

The Theorem 3 implies that one choice for the near-optimal transformation T_θ^ϵ is the transformation $R \circ T$, where T is any transformation that maps the points b_k and b_l from the radial pair (b_k, b_l) of b_θ to the $\epsilon/3$ neighborhoods of $T_\theta(b_k)$ and $T_\theta(b_l)$, respectively, and R is the rotation around the radial axis $\overline{T(b_k)T(b_l)}$ that maps the remaining points from $T(b_\theta)$ to the ϵ-neighborhoods of the corresponding points from $T_\theta(b_\theta)$.

In search for a radial pair of b_θ, the algorithm in [5] explores all n^2 possible pairs of residues in b. For each candidate radial pair (b_k, b_l), the algorithm generates a finite, representative set of transformations that map b_k and b_l into $\theta + \epsilon/3$ neighborhoods of a_k and a_l, respectively (see the paragraph below for more details). For every such transformation T, a plane-sweep algorithm [18] is used to find a rotation R around the axis $\overline{T(b_k)T(b_l)}$ that maximizes the number of residues from $R(T(b))$ that can be placed at distance $< \theta + \epsilon$ from the corresponding residues in a.

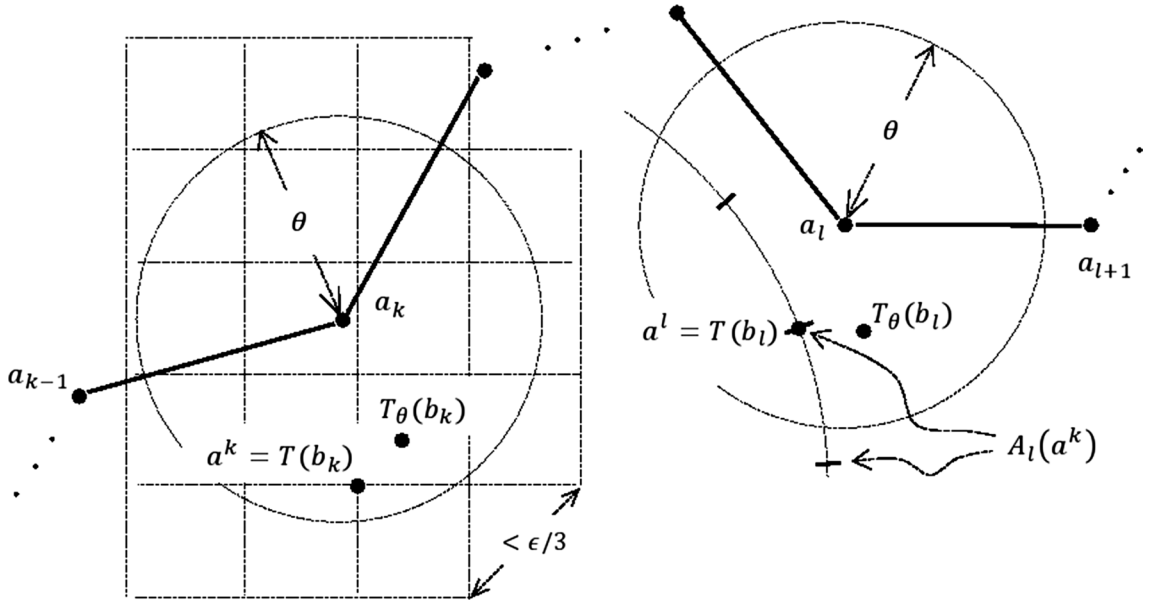

Fig. 7 Discretizing the space of rigid body transformations. 2D illustration of A_k (the set of the vertices of the *squares* shown on the *left*) and the set $A_l(a^k)$ generated for one $a_k \in A_k$

A finite set of transformations that map the residues b_k and b_l into the $\theta + \epsilon/3$ neighborhoods of a_k and a_l, respectively, is constructed in such a way to ensure that for at least one of those transformation T, $\|T(b_k) - T_\theta(b_k)\| < \epsilon/3$ and $\|T(b_l) - T_\theta(b_l)\| < \epsilon/3$. This can be achieved by partitioning $\mathbb{R}^3$ into small cubes of side length slightly smaller than $\sqrt{3}\epsilon/9$ and then collecting the vertices of the cubes that are inside the open ball of radius $\theta + \epsilon/6$ around a_k (Fig. 7). The elements of this set, denoted by A_k, are the candidate points $T(b_k)$. The number of points in A_k is $O(1/\epsilon^3)$ and at least one of them must be at distance $< \epsilon/6$ from $T_\theta(b_k)$ (Fig. 7). For each point $a^k \in A_k$, the set $A_l(a^k)$ of possible images of b_l under T is computed by discretizing the spherical cap $S(a^k, \|b_k - b_l\|) \cap B(a_l, \theta + \epsilon/3)$, where $S(a, r)$ and $B(a, r)$ denote the sphere and the open ball in $\mathbb{R}^3$ with center a and radius r, respectively, in such a way that at least one point from $A_l(a^k)$ is found at distance $< \epsilon/3$ from $T_\theta(b_l)$ (Fig. 7). We note that size of $A_l(a^k)$ is $O(1/\epsilon^2)$. Hence, the total number of candidate pairs of points $(T(b_k), T(b_l))$ is $O(1/\epsilon^5)$.

An obvious to compute $T^\epsilon_{\theta_1}, \ldots, T^\epsilon_{\theta_m}$ is to run the just described algorithm m times in succession, for $\theta = \theta_1, \ldots, \theta = \theta_m$. However, such an approach results in many unnecessary repeated calculations as the area around a_k and the corresponding spherical cap in the neighborhoods of a_l are discretized over and over again. Moreover, all transformations T and R, generated and inspected during the procedure for finding $T^\epsilon_{\theta_i}$, are inspected again during the procedure for finding $T^\epsilon_{\theta_j}$, for each $j > i$.

NEAR_OPTIMAL_HUBBARD_VALUES $(a, b, \bar{\theta}, \epsilon)$

Let K be the set of the small cubes obtained by dividing $\mathbb{R}^3$ by the planes $x = ri, y = ri, z = ri; i \in \mathbb{Z}$, where r is a constant (slightly) smaller than $\sqrt{3}\epsilon/9$.

Let $m = \left\lceil \bar{\theta}/\epsilon \right\rceil$ and let $\theta_i = (i-1)\epsilon$ and $H_i = 0$ for $i \in \{1, \ldots, m\}$

for each $k, l \in \{1, \ldots, n\},\ k < l$

Let A_k be the set of vertices of the cubes $k \in K$ that are inside the open ball of radius $\theta_m + \epsilon/6$ around a_k. For each $a^k \in A_k$, let $A_l(a^k)$ be a set of uniformly spaced points from the spherical cap $C = S(a^k, \|b_k - b_l\|) \cap B(a_l, \theta_m + \epsilon/3)$ such that, for any $c \in C$, $A_l(a^k) \cap B(c, \epsilon/6) \neq \emptyset$

for each pair $(a^k, a^l) \in A_k \times A_l(a^k)$

Let T be a transformation such that $T(b_k) = a^k$ and $T(b_l) = a^l$

for each i such that $\|a_k - a^k\| < \theta_i + \epsilon/6$ and $\|a_l - a^l\| < \theta_i + \epsilon/3$

Find a rotation R around the axis through a^k and a^l maximizing $P(R \circ T, \theta_i + \epsilon)$

if $P(R \circ T, \theta_i + \epsilon) > H_i$

$T^{\epsilon}_{\theta_i} = R \circ T$

$H_i = P(R \circ T, \theta_i + \epsilon)$

end for

end for

end for

We show that all transformations $T^{\epsilon}_{\theta_1}, \ldots, T^{\epsilon}_{\theta_m}$ and the corresponding values H_1, ..., H_m can be computed, at once, during the procedure of finding the last transformation, namely $T^{\epsilon}_{\theta_m}$. As demonstrated in the pseudocode above, the transformation T is generated only once for each pair of points $(a^k, a^l) \in A_k \times A_l(a^k)$ and a sweep-plane algorithm for finding R is called only once for each i satisfying $\|a_k - a^k\| < \theta_i + \epsilon/6$ and $\|a_l - a^l\| < \theta_i + \epsilon/3$. The values of H_i are updated on the fly.

Running time analysis

To analyze the algorithm's running time, we note that the number of iterations of the first *for* loop is equal to the number of candidate radial pairs (b_k, b_l), which is $O(n^2)$. The number of iterations of the second *for* loop matches the number of pairs of grid points around a^k and a^l, which is $O(1/\epsilon^3) \times O(1/\epsilon^2) = O(1/\epsilon^5)$. Each one of $O(m) = O(\lceil \bar{\theta}/\epsilon \rceil) = O(1/\epsilon)$ iterations of the third *for* loop calls a $O(nlogn)$ plane-sweep procedure to compute an optimal rotation and (if needed) to update the value H_i. Hence, the asymptotic time complexity of the three nested *for* loops is $O(n^3 logn/\epsilon^6)$.

Conclusions

Estimating the quality of a protein 3D model is a challenging task. Automatically generated *GDT_TS* score is helpful as the first raw approximation but this measure is neither sensitive nor selective enough to be exclusively relied upon in ranking different models for the same target. In this paper, we show that using a more accurate approximation of the area under the *GDT* curve as the criterion of model quality addresses many of the drawbacks of *GDT_TS*. We also present a rigorous $\widetilde{O}(n^3)$ algorithm for computing the area under the *GDT* curve for a given model, where n is the model's length. The area estimate returned by our method is "near-optimal", meaning that the error in the estimate can be made smaller than any upfront specified value.

Despite the cubic asymptotic running time with a relatively large hidden constant, we believe that the techniques presented in this paper can guide a future development of a computationally efficient computer program, in particular since our methodology is amenable to parallel implementations. A heuristic version of the algorithm for estimating the area under the *GDT* plot can be found at http://bioinfo.cs.uni.edu/GDT_A.html.

Acknowledgements
This project was supported by the University of Northern Iowa Professional Development Award. The structure alignment figures were prepared in Jmol (http://www.jmol.org).

Competing interests
The author declares that there is no competing interests regarding the publication of this article.

Received: 7 March 2015 Accepted: 9 October 2015
Published online: 26 October 2015

References

1. Zemla A, Venclovas C, Moult J, Fidelis K. Processing and analysis of CASP3 protein structure predictions. Proteins. 1999;37(S3):22–9.
2. Zemla A, Venclovas C, Moult J, Fidelis K. Processing and evaluation of predictions in CASP4. Proteins. 2001;45(S5):13–21.
3. Read RJ, Chavali G. Assessment of CASP7 predictions in the high accuracy template-based modeling category. Proteins. 2007;69(S8):27–37.
4. Zemla A. LGA—a method for finding 3D similarities in protein structures. Nucleic Acids Res. 2003;31:3370–4.
5. Li SC, Bu D, Xu J, Li M. Finding nearly optimal GDT scores. J Comput Biol. 2011;18(5):693–704.
6. Li SC, Ng YK. On protein structure alignment under distance constraint. Theor Comput Sci. 2011;412:4187–99.
7. Choi V, Goyal N. A combinatorial shape matching algorithm for rigid protein docking. CPM Lecture Notes Comput Sci. 2004;3109:285–96.
8. Akutsu T. Protein structure alignment using dynamic programming and iterative improvement. IEICE Trans Ins Syst. 1995; E79-D(12):1629–36.
9. Poleksic A. Improved algorithms for matching r-separated sets with applications to protein structure alignment. IEEE/ACM Trans Comput Biol Bioinform. 2013;10(1):226–9.
10. Kryshtafovych A, Fidelis K, Moult J. CASP8 results in context of previous experiments. Proteins. 2009;77(9):217–28.
11. Tramontano A, Cozzetto D, Giorgetti A, Raimondo D. The assessment of methods for protein structure prediction. Methods Mol Biol. 2008;413:43–57.
12. Kryshtafovych A, Milostan M, Szajkowski L, Daniluk P, Fidelis K. CASP6 data processing and automatic evaluation at the protein structure prediction center. Proteins. 2005;61(S7):19–23.
13. Huang YJ, Mao B, Aramini JM, Montelione GT. Assessment of template-based protein structure predictions in CASP10. Proteins. 2014;82(S2):43–56.
14. Tai CH, Bai H, Taylor TJ, Lee B. Assessment of template-free modeling in CASP10 and ROLL. Proteins. 2014;82(S2):57–83.
15. Ortiz AR, Strauss CE, Olmea O. MAMMOTH (matching molecular models obtained from theory): an automated method for model comparison. Protein Sci. 2002;11:2606–21.
16. Holm L, Sander C. Protein structure comparison by alignment of distance matrices. J Mol Biol. 1993;233:123–38.
17. Pevsner J. Bioinformatics and functional genomics. 2nd edn. Wiley-Blackwell; 2009.
18. Alt H, Mehlhorn K, Wagener H, Welzl E. Congruence, similarity, and symmetries of geometric objects. Dicrete Comput Geom. 1988;3:237–56.