Pokefind: a novel topological filter for use with protein structure prediction

Firas Khatib[*], Carol A. Rohl[**] and Kevin Karplus
Department of Biomolecular Engineering, University of California, Santa Cruz, CA 95064

ABSTRACT

Motivation: Our focus has been on detecting topological properties that are rare in real proteins, but occur more frequently in models generated by protein structure prediction methods such as Rosetta. We previously created the Knotfind algorithm, successfully decreasing the frequency of knotted Rosetta models during CASP6. We observed an additional class of knot-like loops that appeared to be equally un-protein-like and yet do not contain a mathematical knot. These topological features are commonly referred to as slip-knots and are caused by the same mechanisms that result in knotted models. Slip-knots are undetectable by the original Knotfind algorithm. We have generalized our algorithm to detect them, and analyzed CASP6 models built using the Rosetta loop modeling method.

Results: After analyzing known protein structures in the PDB, we found that slip-knots do occur in certain proteins, but are rare and fall into a small number of specific classes. Our group used this new Pokefind algorithm to distinguish between these rare real slip-knots and the numerous classes of slip-knots that we discovered in Rosetta models and models submitted by the various CASP7 servers. The goal of this work is to improve future models created by protein structure prediction methods. Both algorithms are able to detect un-protein-like features that current metrics such as GDT are unable to identify, so these topological filters can also be used as additional assessment tools.

Contact: firas@u.washington.edu

1 INTRODUCTION

During the fifth Critical Assessment of Techniques for Protein Structure Prediction (CASP) experiment (Moult *et al.*, 1995), a high frequency of occurrence of knots were observed for certain targets among the protein structure prediction models built using the Rosetta homology-based structure prediction method (Bradley *et al*., 2003; Rohl *et al.*, 2004a). This required a significant effort in manual inspection to discard those models containing knots (Rohl *et al.*, 2004a), but this step was necessary as the assessors in CASP4 had deemed knotted models "impossible structures" (Tramontano *et al.*, 2001). This was the motivation behind Knotfind, a rapid algorithm for knot detection which our group implemented during CASP6 in the context of the Rosetta homology-based method (Khatib *et al.*, 2006). We were interested in finding topological properties that were common in Rosetta models, but rare in real proteins.

In the CASP6 experiment, the assessors reported that knotted models were still being submitted and that such knotted models submitted for comparative modeling targets were rejected out of hand without additional assessment (Tress *et al.*, 2005). Knots in polypeptide chains are often difficult to detect simply by visual inspection, as evidenced by the fact that the assessors still accepted several knotted CASP6 comparative modeling models, most likely because it was not visually apparent that these models contained knots (Khatib *et al.*, 2006). We noticed the same phenomenon for a similar protein topology that occurs in protein structure prediction models.

After the CASP6 experiment, while analyzing models generated by the automated Robetta server (Chivian *et al.*, 2003; Kim *et al.*, 2004), which utilizes the Rosetta method, our group noticed an interesting topology for Target T0199. By visual inspection, it would seem as though Robetta's model 1 is knotted (Figure 1). Following the orange region of the chain towards the red terminus in the backbone ribbon diagram, it seems as if the chain wraps itself around the cyan region, behind the blue region and next to the yellow region. If one were to increase the tension in this protein chain, as described in the Knotfind algorithm (Khatib *et al.*, 2006), it would correctly simplify the chain to a straight line, denoting an un-knotted chain. This Robetta model does not contain a knot; it has what is more commonly known as a slip-knot.

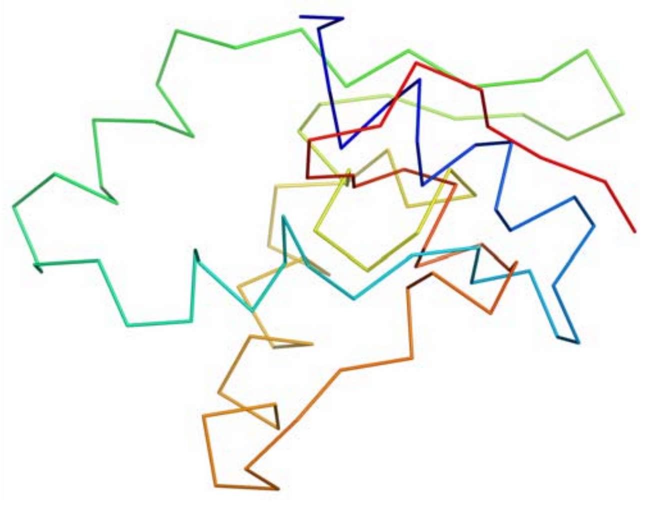

Fig. 1. A backbone ribbon diagram of Robetta's model 1 for CASP6 target T0199. Following the orange region to the red region, the chain seems to wrap around the cyan region, behind the blue and next to the yellow region. It seems as if it would become tangled into a knot if one were to pull both ends of this chain. That is not the case, however, because if the red and blue ends are pulled apart then the chain simplifies to a straight line. This is commonly referred to as a slip-knot.

Like untying shoelaces, which are commonly considered to be knotted, a slip-knot will simplify to a straight line if one pulls both ends of the chain. It is very difficult to detect a slip-knot, and the Knotfind algorithm will simply report the polypeptide chain to be knot-free. After noticing this particular case in CASP6, we set out to create a new algorithm that would detect this complex un-protein-like topology. Although the Robetta model does not contain a mathematical knot, by visual inspection one can tell that its

[*]To whom correspondence should be addressed. Current address: Department of Biochemistry, University of Washington, Seattle, WA 98195
[**]Current address: Merck Research Laboratories, 33 Avenue Louis Pasteur, Boston, MA 02115

fold is not protein-like. As seen in Figure 1, it seems as though the cyan region pokes through a small loop in the orange region. Our goal was to be able to detect this computationally, since servers such as Robetta have no human intervention.

The model in Figure 2 is also from T0199, but is one predicted by the Rohl group (Group 079) at UCSC, also using the Rosetta homology-based method. Although the chain does not contain a knot, it does seem that the red loop and cyan loop thread through one another. Just looking at these two interconnecting loops led us to believe that it would be topologically unfavorable to pass a segment of the chain through such a small red loop and likewise wrapping such a red loop around another segment of the chain would also be unfavorable and thus un-protein-like. In the Rosetta modeling process, however, such loops have no problem wrapping around one another or poking through small loops in the chain. We refer to these slip-knots as *pokes* and set out to create a Pokefind algorithm that would be able to detect such topologically unfavorable conformations in our Rosetta models, referred to as *decoys*.

The goal of this work is to improve future models created by the protein structure prediction community. Pokefind is an attempt to capture another topological property that distinguishes decoys from real protein structures.

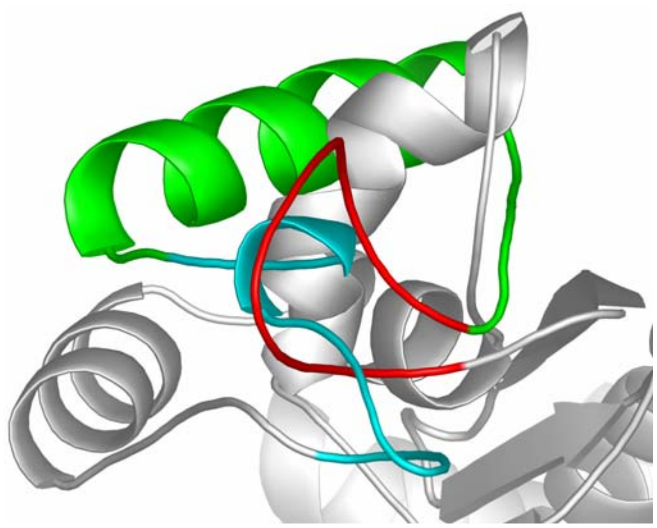

Fig. 2. One of the CASP6 decoys for target T0199 generated by the Rohl group using Rosetta. This protein chain does not contain a knot even though the red and cyan regions thread through one another. This chain is unknotted because the red and cyan regions are connected by the green region. Increasing the tension between both terminal ends of the protein chain causes the white helix (which lies between the red and green regions) to simplify away. This results in the red/green/cyan region easily simplifying to a straight line.

2 METHODS

2.1 Closed Loops

In 2003, Trifonov *et al.* reported that "analysis of the closed loops in crystallized protein structures reveals that the contour length of 20-50 residues is dominant, with the majority of 25-30 residues." Trifonov *et al.* cited a paper by Berezovsky *et al.* from 2000 where for closed loops with ends within 7 Å from one another they reported how often they had observed different loop lengths. The Pokefind algorithm defines these closed loops as any segment of the protein backbone where the ends are within 7.0 Å from one another and the ends span between 3 and 33 Cα atoms. Any part of the remaining Cα trace of the chain that pokes through this closed loop is deemed to be a poke. The maximum closed loop length of 33 residues was chosen so as not to detect entire domains that might poke through a large closed loop. After examining our CASP6 Rosetta dataset (see Section 2.3), 32 residues was the longest closed loop containing what we believed to be an un-protein-like poke (Figure 3).

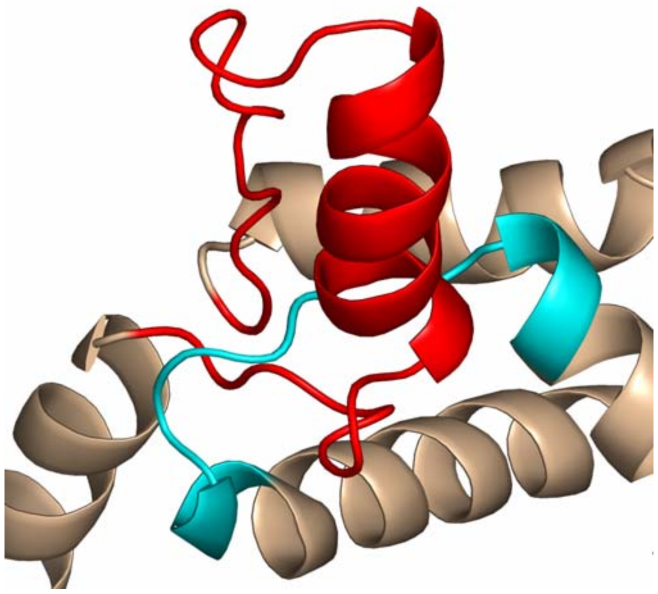

Fig. 3. The longest closed loop found in CASP6 Rosetta decoys containing a poke we want Pokefind to report. The closed loop of length 32 (shown in red) is being poked by the cyan segment of the chain. This is the exact kind of poke that the Pokefind algorithm was designed for, because this chain does not contain a knot.

For the Pokefind algorithm, we wanted a method that would generate a surface out of the closed loop and detect anything that poked through this surface. If a closed loop is non-planar, forming a U-shape for example, it would only make a surface of the actual U-shaped closed loop and not a surface that includes the entire space that the bent closed loop occupies. Our solution to this problem was to break up this surface into as many small triangles as possible, and then inspect those triangles for pokes.

2.2 Poke-detection algorithm

Pokefind begins by searching for the two closest Cα atoms in a given closed loop and connecting them. Next it splits up the closed loop into two different areas based on this connection and runs again the exact same way on both sections. It does this recursively until a section is left with only three points. Once that occurs, it checks that triangle for any pokes by detecting if any line segments that lie outside the closed loop intersect the triangle. Any line segment that pokes through the triangle is reported and Pokefind continues to solve the remaining sections (Figure 4).

Pokefind is able to divide any closed loop into many smaller regions and analyze each region separately. This is useful when the closed loop has many different topologies since Pokefind will not be looking at line segments that may poke the global fold of the closed loop, but rather will detect more local pokes within a single closed loop that may not be planar. By triangulating the surface of the closed loop, this method will not report segments that poke through the concavity of a closed loop.

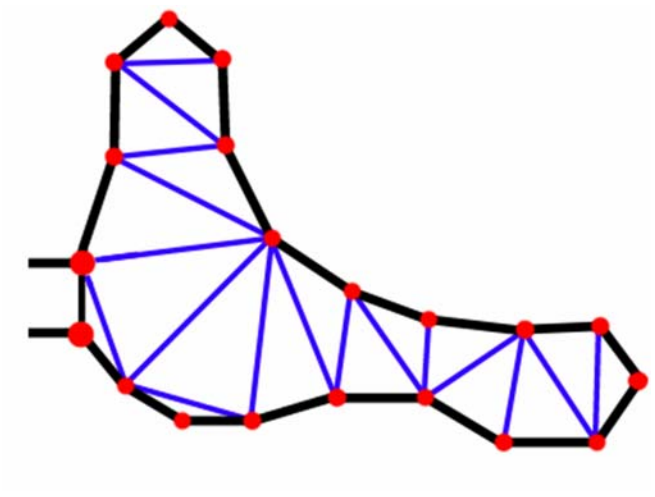

Fig. 4. Example of the Pokefind algorithm. A closed loop is shown in black, with Cα residues as red dots. The two large red Cα residues are within 7 Å from one another, defining the closed loop. Pokefind breaks up this closed loop into smaller sections, shown by the blue lines. It then checks all these triangles to see if any line segments that lie outside the closed loop are poking through them. Even if the turn at the top and the turn at the lower right are not in the same plane as the rest of the closed loop, Pokefind will still be able to correctly distinguish between pokes going through these turns and non-pokes that go through the concavity of a turn.

2.3 Pokefind Training Set

The Rosetta decoy sets built during CASP6, including models from the Robetta server, use the Rosetta homology-based structure prediction method (Bradley *et al*., 2003; Chivian *et al.*, 2005). Predictions begin from an alignment to a parent protein of known structure and coordinates for the aligned regions are taken directly from the parent structure and serve as a fixed template. Coordinates for structurally variable regions (SVRs), corresponding to both gaps in the alignment as well as regions of uncertain alignment, are constructed by assembling short fragments of known structure. The selected fragments are combined using a Monte Carlo simulated annealing search by means of a knowledge-based potential function derived from the observed distributions of residues in known protein structure along with a gap penalty to ensure chain continuity in the final model (Rohl *et al.*, 2004a,b).

The PISCES server was used to identify 9,553 protein chains in the RCSB PDB (Berman *et al.*, 2000) with less than 90% sequence identity, with x-ray structures of resolution better than 3.0Å and no R-factor filtering ($R \leq 1.0$) (Wang *et al.*, 2003). Running the Pokefind algorithm on both this PDB set and on our CASP6 Rosetta decoy set resulted in many pokes being reported in real proteins: 5,543. The most common types of pokes found in real proteins occur near the ends of the closed loop and barely poke through it (Figure 5). Similar to a knotted protein chain that becomes unknotted if a few residues are trimmed from each end; if a poke occurs within a few residues of either end of the closed loop, we are less interested in it than a poke which is further down the protein chain.

We also noticed that shorter closed loops often had pokes in the decoys, but rarely in real proteins. In the rare cases where real proteins with short closed loops contained pokes, most of these all had a similar topology. The poke in the short closed loop would be the result of a beta sheet forming with a middle strand poking though the closed loop formed by the two outer strands (Figure 6).

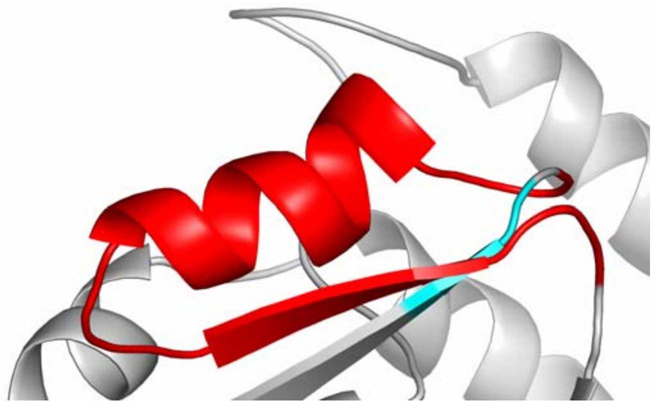

Fig. 5. Example of the most commonly seen type of poke, taken from 1a8s. The closed loop (shown in red) is being poked by a segment of the chain (in cyan) adjacent to one of the ends of the closed loop. This poke barely punctures the plane created by the closed loop. These are not the types of pokes that we are interested in reporting since they are very common in real proteins.

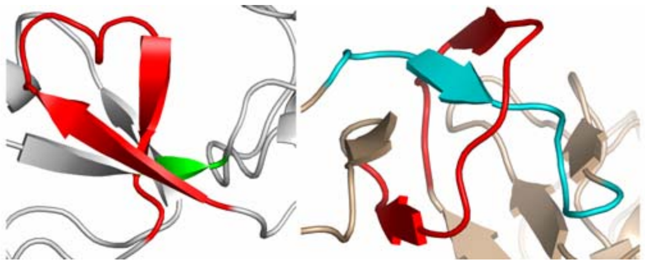

Fig. 6. Examples of real proteins with short closed loops containing pokes. Two closed loops of length 14 are shown in red. The poke on the left occurs in the real protein 1g8lA, where the middle strand (shown in green) pokes through the closed loop, between the two red strands. The poke on the right is from the real protein 2viu, where the cyan strand pokes through the red closed loop. In both of the real proteins, these short closed loops are being poked by strands that form a sheet with the closed loop.

While examining these rare pokes in real proteins, it became apparent that these occurrences are even more infrequent in Rosetta decoys. Further inspection of pokes in real proteins revealed that these beta sheet topologies not only occur with short closed loops, but also with longer ones. Figure 7 shows two different real proteins, with different closed loop lengths, that have strands poking through them. The protein on the left, 1cex, has one green strand poking through the red closed loop. 2aqj, on the right, has two strands that form a poke; the green strand pierces through the red closed loop in one direction and pokes through in the opposite direction a few residues later with the blue strand. These strands all form a beta sheet with the closed loops that they are poking, whereas this does not occur as often with pokes found in Rosetta decoys (see Section 2.5).

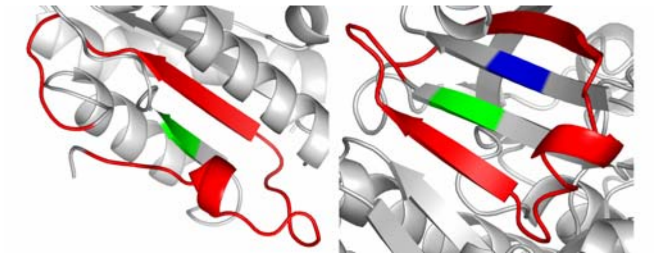

Fig. 7. Examples of real proteins with closed loops being poked by strands. Two real proteins, 1cex (on the left) and 2aqj (on the right) have closed loops that are poked by strands. The closed loops are shown in red with the strands poking in green and blue. These strands form a beta sheet with the closed loop that they poke through.

2.4 Sheet Filter

We implemented a sheet filter to detect and ignore most of the pokes that were found in real proteins. This sheet filter uses the transitive property of strands to group them all together. If two strands form a sheet with one very long strand which forms a sheet with three strands on the other side, then all six of these strands are considered to be in the same group. The sheet filter then checks if a poke is in the same sheet group as any residue in the closed loop and filters the poke out if it is, so as to not report it as an un-protein-like poke. The Undertaker program was used to establish whether a hydrogen bond exists between atoms to determine if a sheet is present (Karplus *et al*., 2005). The detailed methods of how Undertaker models hydrogen bond geometry are explained in Archie and Karplus (2008).

The sheet filter was able to throw away 3,977 of the 5,543 pokes in real proteins. One promising aspect is that it conveniently filters out all pokes that occur in knotted proteins. None of the remaining 1,566 pokes in real proteins are from proteins that contain knots. This makes the sheet filter even more effective, because all knots that are found in real proteins will never be reported by Pokefind. Pokefind will still report severely knotted models, but the types of knots that have been found in nature so far will not be reported. Therefore, if a model is using a knotted region of a real protein as a template, Pokefind will not incorrectly classify that region as having a poke or as being un-protein-like; it will simply be ignored.

Of the 137,057 pokes found in 58,498 CASP6 Rosetta decoys, 27,548 pokes were filtered out by the sheet filter. Although it may seem as though 20% of all pokes in Rosetta decoys are being filtered out incorrectly—that these 27,548 cases are all false negatives—this is not the case. All Rosetta decoys created at UCSC during CASP6 were built using templates of known protein structures, so many of these sheet pokes in Rosetta decoys are actually template regions that are copied directly from real proteins. 27,327 of the 27,548 decoy pokes that were filtered out by the sheet filter occurred in template regions and of the remaining 221 sheet pokes that occurred solely in SVRs, none of those decoys had pokes that were 15 or more residues away from the closest end of the closed loop.

2.5 Co-pokes

While examining Rosetta decoys having two conflicting SVRs, such as the decoy in Figure 3, we noticed another problematic topology that could easily be identified. If two closed loops become intertwined and poke one another, then they can be classified as bad pokes. This means that if a closed loop has a poke and that poke is part of another closed loop which is being poked by the initial closed loop, then both closed loops are poking one another resulting in what we call a *co-poke* (Figure 8).

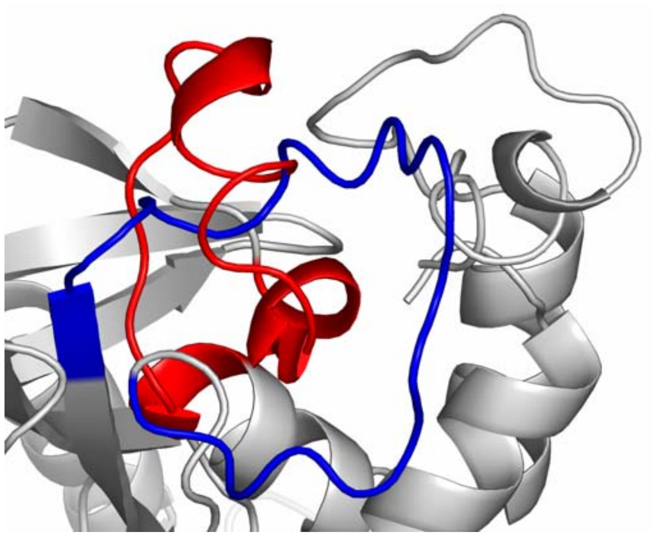

Fig. 8. Example of a co-poke. This CASP6 Rosetta decoy contains two closed loops (shown in red and blue) that thread through one another. The red closed loop is being poked by the blue loop and the blue closed loop is being poked by the red loop. We have defined this topological feature of two closed loops poking one another as co-pokes.

After running the sheet filter, to ignore all the pokes that form beta sheets with the closed loops they are poking, we ran a co-poke identifier to classify two closed loops that poke one another as bad pokes. Of the 137,057 pokes found in 58,498 CASP6 Rosetta decoys, 24,551 of these pokes had co-pokes. With this simple identifier we are able to classify 18% of all decoys pokes as definitively having an incorrect topology, without any additional assessment.

Just as there are rare cases of deeply knotted proteins, our co-poke identifier discovered 37 co-pokes in real proteins. That translates to only 0.67% of all real pokes being co-pokes. This very low co-poke rate in real proteins was very exciting, since the corresponding rate for co-pokes in Rosetta decoys was 18%. Using these results, we looked at the ratio between the pokes per model in the decoys compared to the pokes per model in the reals as a measure of how un-protein-like a poke is.

In Figure 9 this ratio of pokes per model is shown at each step in the filtering process. Initially, after running the Pokefind algorithm, the ratio of decoy pokes per model to real pokes per model is 4.04. After implementing the sheet filter, that ratio increases almost three fold to 11.42. The ratio for pokes thrown out by the sheet filter is very low at 1.13, whereas the ratio for co-pokes is 108.36, demonstrating how un-protein-like co-pokes are since they are observed at a much higher rate in decoys than in real proteins.

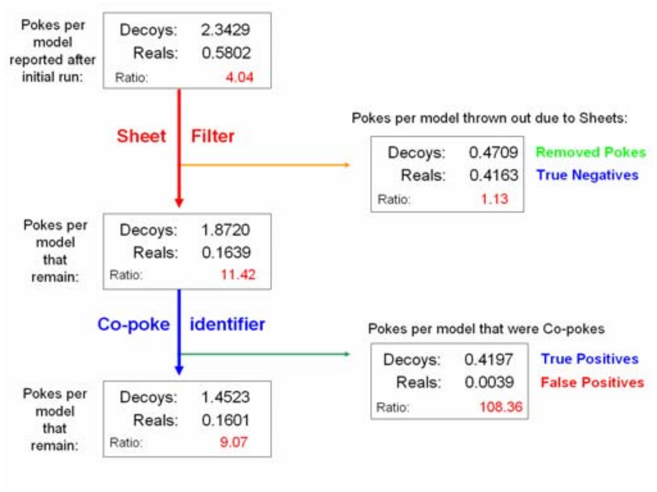

Fig. 9. Chart showing how many pokes per model are separated out by the sheet filter (first arrow in red) and the co-poke identifier (bottom arrow in blue). Numbers in black indicate the total number of pokes reported at each step, divided by the total number of models examined by Pokefind. The numbers in red represent the ratio between the pokes per model in the decoys compared to the pokes per model in the reals. Not all the pokes removed from decoys by the sheet filter are false negatives—only 0.0038 pokes per model are. The remaining pokes were present in the templates used by the decoys (see Section 2.4).

2.6 Assigning costs to different pokes

We plotted the exact separations for pokes that were 29 residues or less away from their closed loops. For example, if line segment 45-46 pokes through a closed loop spanning residues 20-40, the separation between the poke and the closest closed loop end would be five. If line segment 19-20 poked the same closed loop, it would have a separation of zero. Figure 10 shows that due to very few data points in the reals, the ratios of decoy pokes per model to real pokes per model range from 3.55 to 192 when looking at the exact separation between a poke and the closed loop it is poking. In order to smooth out the values in Figure 10 as much as possible, we manually grouped individual separations into bins. Figure 11 shows a histogram of the smoothest manual binning. For example, exact separations of zero, one and two are assigned an average ratio of 4.88, and separations of 29 or more are all given a ratio of 97.66 (including higher separation values not shown on the graph). Using the ratios in Figure 11, we can assign how much worse a poke is the further it is from the closest end of the closed loop.

Energy is usually presented as a negative log probability, so to get an energy-like cost function we take the log of the sum of the ratios to assign each poke a cost. For example, a decoy with a poke that is adjacent to the closed loop it is poking will have a cost of 0.69 (the log of 4.88) whereas a decoy with a poke that is 153 residues away from the closest end of its closed loop will have a cost of 2.78 (the log of 97.66). A decoy containing both these pokes will have a cost of 2.01 (the log of 97.66+4.88). Co-pokes had a ratio of 108.36, therefore any co-poke will be given a cost of 2.03. Since pokes that were thrown out by the sheet filter had a ratio of 1.13, we assign these pokes a cost of 0.053, to differentiate them from completely poke-free proteins.

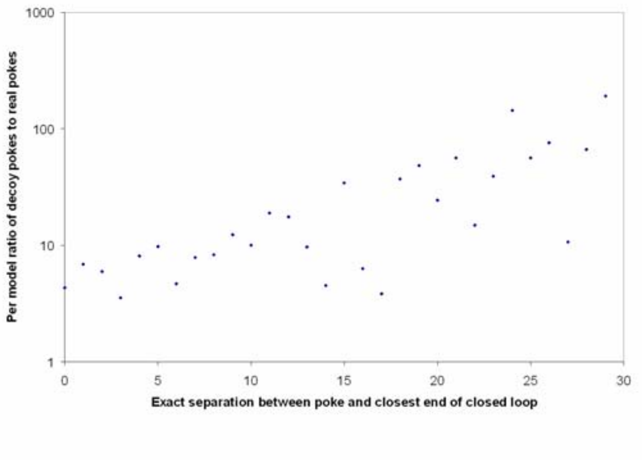

Fig. 10. Graph plotting the ratio of decoy pokes per model to real pokes per model at exact poke to closed loop separation cutoffs. The x-axis shows the exact separation between a poke and the closed loop it is poking for all pokes within 29 residues of a closed loop. A separation of 10, for example, indicates that the poke is exactly 10 residues from the closest end of the closed loop it is poking. The y-axis shows the ratio between the pokes per model in the decoys compared to the pokes per model in the reals. Due to the small data set of pokes in real proteins, these ratios vary highly from one separation value to the next. This led to a manually smoothed version of the graph, shown in Figure 11.

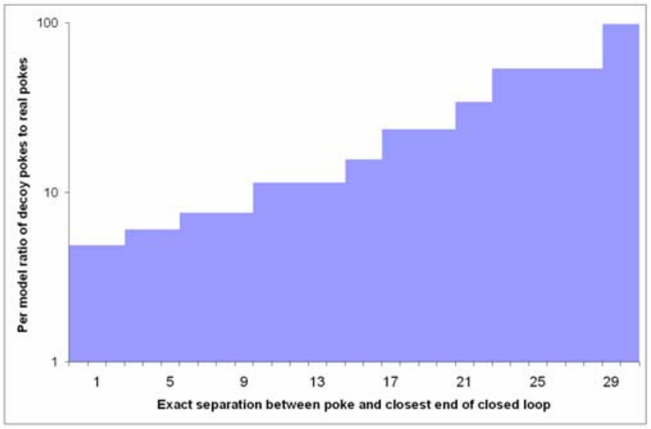

Fig. 11. Histogram showing the ratio of decoy pokes per model to real pokes per model at various binned poke to closed loop separation cutoffs. Using the data points from Figure 10, we manually grouped individual separations into bins and calculated the average ratios for those values. This histogram represents the smoothest binning we were able to produce. All separations of 29 and higher, including those not shown on this graph, are all given the same ratio value of 97.66.

We use the log of the sum of the ratios rather than the sum of the logs because a decoy with one very bad poke, such as a co-poke, is much worse than many pokes that are adjacent to the end of a closed loop. For example, a decoy with five pokes that are adjacent to the ends of the closed loops that they are poking is given a poke cost of 1.39, which is less than 2.03 (the poke cost for a decoy containing a co-poke), whereas the decoy with five pokes would have a cost of 3.45 had we summed the log scores. By assigning these various pokes different costs, we are reporting how un-protein-like each poke is. Based on our analysis of CASP6 Rosetta decoys and our PDB set of real proteins, co-pokes are extremely rare in real proteins making up only 0.67% of all the real pokes, whereas 18% of all decoy pokes were co-pokes. This difference indicates how un-protein-like co-pokes are, which is why we assign them the highest cost, compared to pokes that occur near the ends of a closed loop.

It is important to note that all our observations of pokes in decoy sets have been solely based on models built by our Rosetta group at UCSC during CASP6. The decoy training set for all the Pokefind work was only the CASP6 Rosetta models built at UCSC, so we needed a completely independent decoy set to use as our test set. It would not be sufficient to simply simulate another CASP6 experiment using the Pokefind algorithm, because the template regions would be the same and all our poke analysis was done on those exact same templates.

3 RESULTS

3.1 Pokefind Test Set

We carefully investigated pokes in our CASP6 decoy training set and in real proteins to come up with a metric for how un-protein-like a given poke is. To be certain that the poke costs we had assigned were adequate, we required a decoy test set that was completely unrelated from our decoy training set. We selected two different datasets from CASP7 to use as a decoy test set. We examined the models submitted by the various servers at CASP7—containing predictions from 93 different structure prediction methods—as an independent decoy test set. In addition to these 11,071 server models, we also looked at the Rosetta decoys built at the University of Washington, in the Baker Lab, during CASP7. This Rosetta test set contained all 16,392 low energy decoys built for CASP7 using Rosetta's loop modeling protocol and one round of full-atom relax.

Prior to running Pokefind on this CASP7 decoy test set, we ran it on the CASP7 solutions. Two of the CASP7 targets contained knots in the solved structures, so we needed to determine whether any of the solutions contained bad pokes. Our CASP7 Rosetta decoy test set consisted of models built using Rosetta*'s* loop modeling protocol, therefore these were all decoys from the Template Based Modeling (TBM) category at CASP7, and we only included CASP7 server models for the exact same targets. When running Pokefind on the solutions, we only looked at the solved structures from the same TBM category, ignoring the four targets that had not been solved: T0320, T0333, T0355, and T0386.

Most of the solved structures, 29 out of 39, had no pokes whatsoever. Four of the 10 remaining proteins had pokes that were filtered out by the sheet filter; this included the two knotted CASP7 targets: T0332 and T0378. Five of the remaining six proteins containing pokes that were within three residues of the end of the closed loop, and one solved structure had a poke 15 residues from the end of its closed loop. None of the 39 real proteins contained co-pokes, but the same was not true for our CASP7 decoys test set. An example of a CASP7 Rosetta decoy containing a co-poke is shown in Figure 12. This prediction for target T0316 had a low Rosetta energy score of -28.63 despite the fact that this co-poke is very un-protein-like; the two closed loops that thread through one another (shown in red and blue in Figure 12) are 169 residues apart.

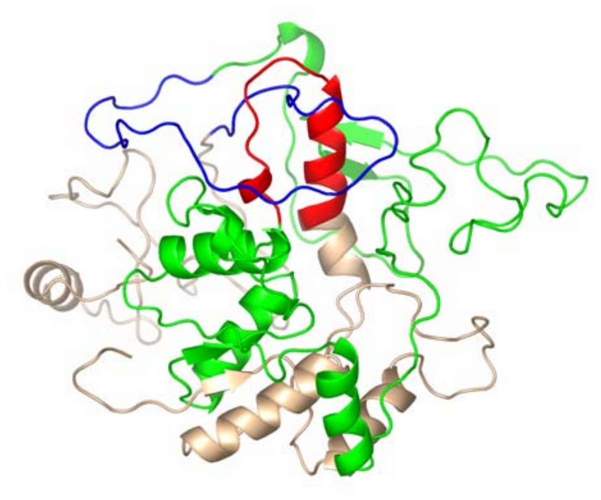

Fig. 12. Example of a CASP7 Rosetta decoy containing a co-poke. This decoy for CASP7 target T0316 was built using Rosetta*'s* loop modeling protocol. The red closed loop pokes the blue closed loop, while the blue closed loop pokes the red closed loop, resulting in a co-poke. These closed loops are 169 residues apart from one another, denoted by the green region.

3.2 CASP7 Results

The histogram in Figure 13 shows the results of running the Pokefind algorithm on our CASP7 Rosetta test set, as well as the corresponding CASP7 server models test set and solved structures. The red line represents all the CASP7 decoys built using the Rosetta loop modeling protocol, for which the corresponding target was successfully solved, with the associated total poke cost per decoy. We wanted to look beyond Rosetta and ran Pokefind on the other methods that were used at CASP7. The green line denotes all the CASP7 server predictions for the same solved targets, after filtering out models that contained missing density. If server predictions with missing density are included, the histogram values for the green line are even higher. The majority of decoys in our test set either did not contain any pokes (poke cost of zero) or contained pokes that were filtered out by the sheet filter (poke cost of 0.053), but there were still many decoys in the test set having higher poke costs than the actual solved structures.

None of the solved structures in the TBM CASP7 category had co-pokes, yet our test set of CASP7 Rosetta decoys contained 6,335 co-pokes. Clearly, these un-protein-like features were still being created by Rosetta's loop modeling method, since there are many CASP7 Rosetta decoys with poke costs higher than those of the corresponding solved structures. The blue crosses in Figure 13 show the actual poke costs for the solved structures in the TBM category at CASP7. These results imply that any decoy with a poke cost greater than that of the rightmost blue cross in the figure contains an un-protein-like topological feature, since such pokes do not exist in the corresponding solved structures.

The histogram in Figure 13 shows that there are many files in our Rosetta test set that have a poke cost higher than the solved CASP7 targets, but this problem is not exclusive to Rosetta. The results are similar for the CASP7 server predictions as well, shown in green in Figure 13. There were 11,071 CASP7 server predictions, but only 5,231 of them did not have any missing density and are shown in the histogram. The green and red lines are similar, despite the fact that the Rosetta decoy set contained 16,392 files, 11,161 more than the server models. Even with only 5,231 predictions, there were more CASP7 server models with poke scores between 2.1 and 5.6 than Rosetta decoys.

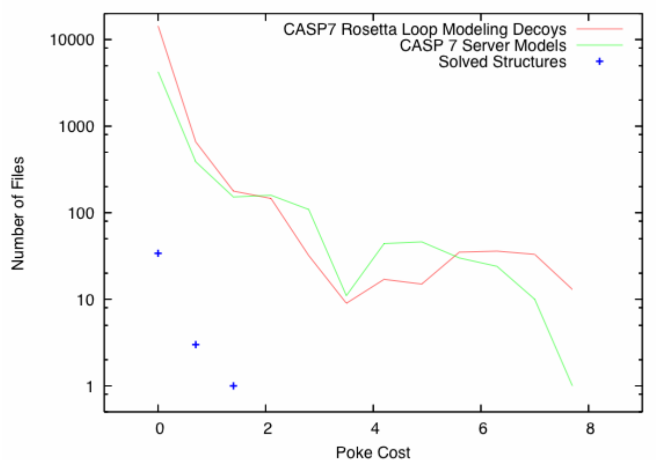

Fig. 13. Histogram showing the results of running Pokefind on CASP7 decoys. The red line represents all CASP7 decoys built using Rosetta*'s* loop modeling protocol, for which there was a solved structure, with the associated total poke cost for each decoy. The green line denotes all the template-based models with no missing atoms from CASP7 servers. The blue crosses show the actual poke costs for the solved structures in the "template based modeling" category at CASP7, implying that any decoy with a higher poke cost contains an un-protein-like topological feature.

An example of a server prediction with un-protein-like features, but no missing density, is shown in Figure 14. This model was submitted as model 2 for target T0364 by the 3Dpro server and has a GDT_TS score of 76.701, which is the highest GDT_TS score for all submitted models for this target, including human predictions. The solved structure for target T0364 does not contain any pokes, not even a poke that would be discarded by the sheet filter, but Figure 14 shows that this highest-ranked prediction contains a co-poke. The blue closed loop is being threaded by the red segment of the chain while the red closed loop is being poked by the blue segment of the chain. By visual inspection it looks as if this chain contains a knot, but that is not the case and Knotfind correctly reports this model to be unknotted.

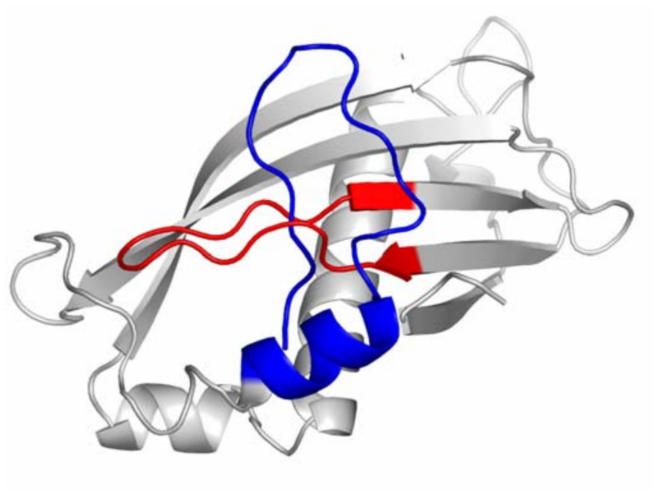

Fig. 14. Example of a high-ranking CASP7 server prediction containing un-protein-like pokes. This prediction for CASP7 target T0364 by the 3Dpro server, submitted as model 2, has a GDT_TS score of 76.701, the highest GDT_TS score for all human and server submissions. This unknotted prediction contains a co-poke, the blue and red closed loops thread through one another, showing that these un-protein-like features are not exclusive to the Rosetta method. The solved structure for this target does not contain any pokes.

3.3 Topological Filters can detect what other metrics have not

The particular example in Figure 14 highlights the need for topological filters, especially in the case of servers where there is no human intervention. Just as Robetta was submitting knotted models in CASP6, many servers submitted models containing bad pokes in CASP7. Even though certain models might obviously be un-protein-like by visual inspection, such as the prediction in Figure 14, servers have no human interference to detect such a feature. Algorithms like Knotfind and Pokefind can be useful to discriminate between various models generated by servers.

Topological filters such as Knotfind and Pokefind are also useful additions to the standard metrics that are currently used to evaluate how protein-like a prediction is. There is the example from CASP4, where a decent scoring model, with "reasonable" GDT and AL0 values, was deemed "an impossible structure" by the CASP4 assessors because it contained a trefoil knot (Tramontano *et al.*, 2001). This particular knot was identified by visual inspection, but many other submitted knotted models went unnoticed in CASP6 (Khatib *et al.*, 2006). Knotfind and Pokefind could be used by assessors in addition to the other metrics that are currently used. When using Rosetta to evaluate the 9,553 PDB chains taken from the PISCES server, incorporating the Pokefind algorithm added 571 seconds to the overall run time of 45 minutes on an Intel(R) Xeon(TM) CPU 2.80GHz, compared to evaluating the chains using Rosetta without Pokefind. The Knotfind algorithm is even faster, adding only 90 seconds to the same 45 minute overall run time.

In CASP7, the assessors added a metric for hydrogen bond conservation, HBscore, in order to assess local atomic interactions (Kopp *et al.*, 2007). For the TBM CASP7 category, they combined GDT, AL0 and HBscore to determine which groups had submitted the best predictions. The assessors showed examples of models with good GDT scores but low HB scores, compared to models with lower GDT scores and higher HB scores. A model that has a decent GDT score, yet does not resemble a protein, is not a useful prediction. This shows the importance of using different metrics that are not correlated with one another. GDT and AL0 are highly correlated already, so combining useful uncorrelated metrics, such as HBscore, will help assessors in the future.

The average correlations with GDT, using Kendall's Tau, was 0.070 with Pokefind and 0.010 with Knotfind across all CASP7 server targets with no missing density. Since both the Knotfind and Pokefind algorithms are uncorrelated with GDT, they could be useful additional metrics, just as HBscore was in CASP7, to detect un-protein-like features that GDT cannot identify. Neither Rosetta's energy function nor Undertaker's cost function are correlated with Pokefind (unpublished data), which may explain why so many poked models are submitted to CASP. If current protein structure prediction algorithms have no penalty for pokes, and the current metrics used by assessors do not penalize these un-protein-like features, then poked predictions will continue to be submitted. This is especially true for server predictions, where there is no human intervention.

Protein structure prediction is regarded as one of the hardest problems in biology today. We have introduced two novel algorithms that can be applied to structure prediction methods and can be used for assessment of predictions. Both Knotfind and Pokefind are able to detect topological features that current metrics are unable to discover. Implementing both these algorithms as metrics in future CASP assessments would force prediction methods to avoid creating these un-protein-like features in their models. Removing these un-protein-like features will hopefully result in better models produced by the protein structure prediction community.

ACKNOWLEDGEMENTS

We thank Josue Samayoa, David Bernick, and Craig Lowe for the decoy structures predicted for CASP6, and Bin Qian and David Baker for CASP7 decoy sets.

REFERENCES

Archie, J. and Karplus, K. (2008) Applying Undertaker Cost Functions to Model Quality Assessment. *Proteins*, PMID 19004017.

Berezovsky, IN. *et al.* (2000) Closed loops of nearly standard size: common basic element of protein structure. *FEBS*, **466**, 283-6.

Berman, HM. *et al.* (2000) The Protein Data Bank. *Nucleic Acids Research*, **28**, 235-242.

Bradley, P. *et al*. (2003) Rosetta predictions in CASP5: successes, failures, and prospects for complete automation. *Proteins*, **53** (Suppl 6), 457-468.

Chivian, D. *et al.* (2003) Automated prediction of CASP-5 structures using the Robetta server. *Proteins*, **53**, 524-533.

Chivian, D. *et al.* (2005) Prediction of CASP6 structures using automated Robetta protocols. *Proteins*, **61** (Suppl 7), 183-92.

Karplus, K. *et al*. (2005) SAM-T04: what is new in protein-structure prediction for CASP6. *Proteins*, **61** (Suppl 7), 135-142.

Khatib, F. *et al*. (2006) Rapid knot detection and application to protein structure prediction. *Bioinformatics*, **22**(14), e252-9.

Kim, DE. *et al.* (2004) Protein structure prediction and analysis using the Robetta server. *Nucleic Acids Res.*, **55**, 656-677.

Kopp, J. *et al*. (2007) Assessment of CASP7 predictions for template-based modeling targets. *Proteins*, **69** (Suppl 8), 38-56.

Moult, J. *et al.* (1995) A large-scale experiment to assess protein structure prediction methods. *Proteins*, **23**, ii-v.

Rohl, CA. *et al.* (2004a) Modeling structurally variable regions in homologous proteins with Rosetta. *Proteins*, **55**, 656-677.

Rohl, CA. *et al*. (2004b) Protein structure prediction using Rosetta. *Methods Enzymol.*, **383**, 66-93.

Tramontano, A. *et al.* (2001) Analysis and assessment of comparative modeling predictions in CASP4. *Proteins*, **45** (Suppl 5), 22-38.

Tress, M. *et al.* (2005) Assessment of predictions submitted for the CASP6 comparative modeling category. *Proteins*, **61** (Suppl 7), 27-45.

Trifonov, EN. and Berezovsky, IN. (2003) Evolutionary aspects of protein structure and folding. *Curr. Opin. Struct. Biol*. **13**, 110-4.

Wang, G. *et al*. (2003) Jr. PISCES: a protein sequence culling server. *Bioinformatics*, **19**, 1589-1591.