SMURFLite: combining simplified Markov random fields with simulated evolution improves remote homology detection for beta-structural proteins into the twilight zone

Noah M. Daniels1, Raghavendra Hosur2, Bonnie Berger2,∗ and Lenore J. Cowen1,∗

1Department of Computer Science, Tufts University, Medford, MA 02155 and 2Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

ABSTRACT

Motivation: One of the most successful methods to date for recognizing protein sequences that are evolutionarily related has been profile hidden Markov models (HMMs). However, these models do not capture pairwise statistical preferences of residues that are hydrogen bonded in beta sheets. These dependencies have been partially captured in the HMM setting by simulated evolution in the training phase and can be fully captured by Markov random fields (MRFs). However, the MRFs can be computationally prohibitive when beta strands are interleaved in complex topologies. We introduce SMURFLite, a method that combines both simplified MRFs and simulated evolution to substantially improve remote homology detection for beta structures. Unlike previous MRF-based methods, SMURFLite is computationally feasible on any beta-structural motif.

Results: We test SMURFLite on all propeller and barrel folds in the mainly-beta class of the SCOP hierarchy in stringent cross-validation experiments. We show a mean 26% (median 16%) improvement in area under curve (AUC) for beta-structural motif recognition as compared with HMMER (a well-known HMM method) and a mean 33% (median 19%) improvement as compared with RAPTOR (a well-known threading method), despite HHpred’s use of extensive additional training data. We demonstrate SMURFLite’s ability to scale to whole genomes by running a SMURFLite library of 207 beta-structural SCOP superfamilies against the entire genome of Thermotoga maritima, and we make over a 100 new fold predictions.

Availability and implementation: A webserver that runs SMURFLite is available at http://smurf.cs.tufts.edu/smurflite/

Received on November 15, 2011; revised on February 7, 2012; accepted on February 28, 2012

1 INTRODUCTION

Many researchers use hidden Markov models (HMMs) to annotate proteins according to homology, with popular systems such as Pfam ( Finn et al. 2008 ) and Superfamily ( Wilson et al. 2004 ) based on HMM methods integrated into UniProt. However, HMMs are limited in their power to recognize remote homologs because of their inability to model statistical dependencies between amino-acid residues that are close in space but far apart in sequence ( Cowen et al. 2009 ; Lifson and Sandkuhl 1987 ; Olmea et al. 1994 ; Steward and Thornton, 2003 ). However, as the dependency graph becomes more complex, major design difficulties emerge. First, the MRF becomes more difficult to train. Second, it quickly becomes computationally intractable to find the optimal-scoring parse of the target to the model.

We introduce SMURFLite, a method that combines both simplified MRFs and simulated evolution to substantially improve remote homology detection for beta structures. Unlike previous MRF-based methods, SMURFLite is computationally feasible on any beta-structural motif.

We first test SMURFLite on all propeller and barrel folds in the mainly-beta class of the SCOP hierarchy in stringent cross-validation experiments. We show a mean 26% (median 16%) improvement in area under curve (AUC) for beta-structural motif recognition as compared with HMMER (a popular HMM method; Fuchs et al. 1998 ) and even a mean 18% (median 10%) improvement in AUC over HHpred (a profile–profile HMM method) and even a mean 18% (median 10%) improvement in AUC over HHpred (a profile–profile HMM method), despite HHpred’s use of extensive additional training data. In order to build the first MRF models that are computationally tractable for all beta-structural proteins, even those with limited training data, we use a model that builds SMURFLite’s ability to scale to whole genomes by running a SMURFLite library of 207 beta-structural SCOP superfamilies against the entire genome of Thermotoga maritima, and we make over a 100 new fold predictions.

Availability and implementation: A webserver that runs SMURFLite is available at http://smurf.cs.tufts.edu/smurflite/

© The Author(s) 2012. Published by Oxford University Press. This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/3.0/), which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
2 METHODS

2.1 Review of SMURF MRF framework

SMURF and SMURFLite rely on training data in the form of a multiple structure alignment with beta strand annotation. This alignment is created using the Matt program [Menke et al. 2007]. Beta strand annotation is done on a structure-by-structure basis, where the beta-strand residue pairing is determined using the same algorithm implemented by the Ramachandran-Sayle and Melzer-White [2002] visualization program. A post-processing step annotates those beta-strand residues that appear in more than half the structures in the alignment as conserved. As gaps in beta strands would complicate training, this post-processing step makes beta-conserved template strands contiguous in the alignment exactly as in [Menke et al. 2007]. The result at this stage is a sequence alignment (resulting from the Matt structural alignment) with conserved beta-strand pairs annotated according to the residue positions and conformation (buried or exposed to solvent).

The pairwise probability portion of the MRF is based on the beta probability tables that were computed by collecting a set of amphipathic beta sheets from the Protein Data Bank (PDB) [Berman et al. 2002] and tabulating the frequencies of pairs of hydrogen-bonded residues in two tables, one for buried residues and one for residues exposed to solvent [Dodziewicz et al. 2008]

For each residue position, the most likely conformation (buried or exposed) is chosen based on whether that residue pairing is most probable from the buried or exposed beta-pairing tables.

Given a trained MRF, SMURF and SMURFLite align a query sequence to the MRF. The query phase of SMURF and SMURFLite computes the alignment of the query sequence to the states of the MRF that maximizes the combined score:

$$\log[HMM\ score] + \log[ pairwise\ score]$$

In this combined score, the HMM score is the conditional probability of observing the sequence given the HMM portion of the model, and the pairwise score is the conditional probability of observing the paired beta-strand components of the sequence given the beta-pair portion of the model. Let the sequence have residues $r_1...r_n$, and the MRF have match states $m_1...m_k$, deletion states $d_1...d_l$ and insertion states $i_1...i_m$. Suppose that $r_1...r_j$ and match states $m_1...m_k$ have been assigned. Then, the probability of assigning $r_k$ to the next match state $m_{k+1}$ is:

$$Pr[m_{k+1} = HMM[m_{k+1}], transition[u_{k-1}, m_k], strand[r_{k-1}, r_k, m_k]]$$

where $u_{k-1}$ can be either $d_{k-1}, i_{k-1}$ or $m_{k-1}$ depending on whether the current state is a deletion, insertion or match state. When the current state is a match state, the SMURFLite template replaces the transition $[u_{k-1}, m_k]$ term with a value of 1. The $\beta$ strand component set to be identically 1 unless the particular match state $m_k$ participates in a beta strand that is matched with a beta-strand node in the model. The pairwise component of the score can only be calculated for a given MRF node once it is determined what residue occupies the paired MRF node earlier in the sequence, each time the dynamic programming reaches a state in the MRF that corresponds to the first residue of the first beta strand in a set of paired beta strands, we need to keep track of multiple cases, depending on what residue in sequence is mapped to that state. SMURFLite uses a multidimensional array to memoize these possible subproblem solutions. A maximum gap size is set to be the longest gap seen in the training data plus 20, for computational efficiency. When paired beta strands follow each other in sequence with no interfering beta strands between them, the number of dimensions in the table for the dynamic programming is directly proportional to the maximum gap length. Let us call the last MRF state for the first of every pair of beta strands a ‘split’ state and the first MRF state for the second of that pair a ‘join’ state. Then, at every split state, the number of dimensions of the dynamic program will be multiplied by the maximum gap length, because the dynamic program must

Fig. 1. A closed beta barrel (PDB ID 1bw3, a Barwin domain) from the superfamily ‘Barwin-like endoglucanases’ to illustrate interleaving of strand pairs. Beta strands a and b, which close the barrel, have interleave 4, whereas strands c and d, which are adjacent in sequence, have interleave 1. Strands b and c have interleave 2.
keep track of scores for each possible sequence position (up to the maximum gap length) that could be mapped to that state. At the corresponding join state, the number of dimensions will be reduced by the maximum gap length, because the scoring function can calculate all the pairwise probabilities of placing that residue into the join state, and then simply take the maximum of all ways to have placed its paired residue into the split state. However, when other beta strands are interleaved, the dynamic program must open additional multidimensional tables before clearing the previous ones from memory. Thus, the number of elements in the multidimensional table is never more than the sequence length times the maximum gap length raised to the interleaving number power.

2.2 Datasets

From SCOP (Murzin et al., 1995), version 1.75, we chose the folds '5-bladed Beta-Propellers', '6-bladed Beta-Propellers', '7-bladed Beta-Propellers' and '8-bladed Beta-Propellers'. We also chose superfamilies from all of the mostly-beta folds containing the word ‘barrel’ in their description, whether open or closed, restricted to those superfamilies comprising at least four families (in order to facilitate leave-family-out cross-validation). These superfamilies were: 'Nucleic acid-binding proteins' (50249), 'Translation proteins' (50447), 'Trypsin-like serine proteases' (50494), 'Barwin-like endoglucanases' (50685), 'Cyclophilin-proteins' (50249), 'Translation proteins SH3-like domain' (50104), 'Lipocalins' (50184) and 'FMN-binding split barrel' (50475). Of these, we removed the superfamilies 'Lipocalins' and 'Trypsin-like serine proteases,' which were not structurally consistent enough to permit a multiple structure alignment for training HMMER or the SMURF variants, and which were broken into distinct superfamilies by Menke et al., 2000, with the result that 11 superfamilies containing barrels were selected. In addition, for the whole-genome search on T.maritima, out of 354 total superfamilies within the SCOP class 'All beta proteins', 288 (81%) of which contain at least two protein chains, 207 superfamilies (71%) were structurally consistent enough to be aligned using the Matt (Menke et al., 2000) structural alignment program. We built SMURFLite templates for these 207 superfamilies, and obtained from UniProt the protein sequences for T.maritima, comprising 1852 genes.

2.3 Training and testing process

For the beta-propeller folds, strict leave-family-out cross-validation was performed. The propeller folds are structurally highly consistent (Menke et al., 2010), and thus high-quality Matt (Menke et al., 2000) multiple structure alignments were possible without descending to the superfamily level. For each propeller fold, its constituent superfamilies were identified. Each superfAMILY was left out, a training set was established from the protein chains in the remaining families, with duplicate sequences removed. An HMM (in the case of HMMER and HHPredd) or MRF (in the case of SMURF and SMURFLite) were trained on the training set. Protein chains from the left-out family were used as positive test examples. Negative test examples were protein chains from all other superfamilies in SCOP classes 1, 2, 3 and 4 (including other barrel superfamilies), indicated as representatives from the nr-PDB (Berman et al., 2004) database with non-redundancy set to a BLAST E-value of $10^{-7}$. Each test example was aligned to the trained HMM (from HMMER and HHPredd) and MRF, and was also threaded, using RAPTOR, against each individual chain in the training set (RAPTOR parameters are discussed below). The score reported for HMMER and HHPredd was the output HMM score, and the score reported for SMURF and SMURFLite was the combined HMM and pairwise score from the MRF. For RAPTOR, the score reported for a test example was the highest score from all the scores resulting from threading that test example onto each chain in the training set. For each training set, the scores for each method were collected and a ROC curve (a plot of true positive rate versus false positive rate) computed. We report the area under the curve (AUC) statistics from this ROC curve (Konego and Pongor, 2004).

2.4 P-values

SMURFLite computes the P-value for an alignment similarly to HMMER, using an extreme value distribution (EVD, Eddy, 1998). An EVD is fitted to the distribution of raw scores over a random sampling of 5000 protein chains from across the SCOP hierarchy. The P-value is then simply computed as $1 - cdf(s)$ for any raw SmurfLite score $s$, where cdf is the cumulative distribution function for the EVD.

2.5 SMURFLite augmented training data

Kumar and Cowan (2009) showed that ‘simulated evolution,’ augmenting limited training data with additional sequences produced by mutating the original sequences, improved the performance of HMMER at recognizing the same-superfamily level of homology. Kumar and Cowan (2011) used two types of simulated evolution: point-wise and pairwise. Here, we add only pairwise mutations based on beta-strand pairings, as we expect long-range interactions between beta strands to be highly conserved across similar structures. We postulated that the elimination of the beta-barrel pairs SMURFLite must disregard because of computational complexity might be compensated for by augmenting the training data with artificial sequences based on likely mutations in those paired beta strands. This training data augmentation comes at insignificant runtime cost and is done before beta-strand pairs are removed from the template (but after their interleave number has been identified, where we define interleave number next below). The mutation frequencies are given by the Betawrap and SMURF (Bradley et al., 2010; Menke et al., 2010) pairwise probability tables. Using the same algorithm as Kumar and Cowan (2011), we generate 150 new artificial training sequences from each original training sequence. For each artificial sequence, we mutate at a 50% mutation rate per length of the beta strands. The parameters 150 and 50% were recommended by Kumar and Cowan (2011), we also evaluated a 10% mutation rate (a secondary peak according to their work) and performance was slightly worse (data available from the authors).

2.6 SMURFLite simplified random field

SMURFLite trains a MRF on a template built from a multiple structure alignment. Beta strands in the aligned set of structures are found by the program SmurfPrepare which is part of the SMURF package (Menke et al., 2009, Menke et al., 2009). The program not only outputs the positions of the consensus beta strands in the alignment, it also declares a position buried or exposed based on which of the two tables is the best fit to the amino acids that appear in that position in the training data. SMURFLite then assigns an interleave value to each beta-strand pair, as follows: any pairwise
interaction between beta strands whose interleave value equals or exceeds the SMURFLite threshold is removed from the training data. Consider three beta strands: A, B and C. Suppose strand A interacts with strand B and the (A,B) pair has an interleave value of 4, whereas strand B also interacts with strand C and that (B,C) pair has an interleave value of just 1. With a SMURFLite threshold of 2, the (A,B) pair would be discarded, but the (B,C) pair would remain in the training data. Thus, interleave numbers are properties of pairs of beta strands; a beta strand may be involved in multiple pairings, each of which may have a distinct interleave value. Discarding beta-strand pairs whose interleave value equals or exceeds the threshold guarantees that the MRF will have no beta-strand pairs greater than or equal to that threshold, and thus bounds the computational complexity, which is exponential in the maximum interleave value found in a training template. Note that SMURFLite with an interleave threshold of 0, which will discard all beta-strand pair information, is simply an HMM.

2.7 HMMER implementation
SMURFLite was tested against HMMER version 3.0a2 with the ‘-seqZ 1’ and ‘-seqE 10000’ options applied to hmmsearch, and the ‘-symfrac 0.2’ and ‘-ere 0.7’ options applied to hmmbuild. The -seqZ 1 option ensures that E-values are comparable regardless of the size of the sequence database, whereas the –seqE 10000 option forces HMMER to return results for all query sequences. The -symfrac 0.2 option requires that only 20% of sequences need to be in agreement to cause a match state in a given column (the default is 50%). Given the remote homology at which we were performing experiments, 50% was an unreasonably high threshold that led to few match states being found. This option was also used by Kumar and Cowd (2005). The -ere option sets the minimum relative entropy per position target to 0.7 bits (the default is 0.59). Note that HMMER versions 3.0a2 and 3.0 both SAM sequence entropy (Karplus and Hu, 2001) by default. This entropy weighting scheme has been shown to be superior for remote homology detection tasks (Bennett et al., 2004). HMMER 3.0a2 was used despite having been superseded by version 3.0, because it uniformly performs better on this task. This is because version 3.0 contains computational optimizations that cause it to reject a sequence (with no score provided) quickly if it does not appear to align well. These optimizations, however, cause nearly all query sequences outside the family level of homology to fail and return no score, with the result that HMMER version 3.0 never surpasses an AUC of 0.5.

2.8 RAPTOR implementation
SMURFLite was tested against RAPTOR, which was run with the options ‘-a nc’ indicating that the default threading algorithm described in the RAPTOR paper (Xu et al., 2005) was used. In addition, RAPTOR used the weighting parameters ‘weightMutation = 1.400976151,’ ‘weightSingleton = 1,’ ‘weightLoopGap = 16.841836238,’ ‘weightPair = 0,’ ‘weightGapPenalty = 1’ and ‘weightSStruct = 3.0137849223.’ RAPTOR uses both sequence and structural features, and these options represent the recommended balance of these features (Xu et al., 2005).

2.9 HHpred implementation
SMURFLite was tested against HHpred version 1.5.1. HHpred HMMs for each SCOP family were downloaded from the HHpred web site, and queried using hhsearch. The score of the best-scoring family HHM within each superfamily was used in comparing ROC curves.

2.10 Whole-genome search
All 1852 protein sequences from T. maritima were queried against beta-structural templates constructed from the nr-PDB (Berman et al., 2000) with non-redundancy determined by an E-value of 10^-5, organized according to those 207 beta-structural superfamilies from SCOP that were able to be aligned using the Matt structural alignment program, using SMURFLite with an interleave threshold of 2 and simulated evolution mutation rate of 50% on the residues that participate in beta strands. We computed P-values and alignments for all 1852 × 207 possible hits.

3 RESULTS
3.1 SMURFLite validation
SMURFLite’s ability to recognize beta propellers and barrels was compared with HMMER (Muñoz et al., 1998) and RAPTOR (Xu et al., 2005) and HHpred (Barton et al., 2000) in a stringent cross-validation experiment. From SCOP (Murzin et al., 1995) version 1.75, we chose the folds ‘5-bladed Beta-Propellers’, ‘6-bladed Beta-Propellers’, ‘7-bladed Beta-Propellers’ and ‘8-bladed Beta-Propellers’. We also chose superfamilies from all of the mostly-beta folders containing the word ‘barrel’ in their description, whether open or closed, restricted to those superfamilies comprising at least four families (in order to facilitate leave-family-out cross-validation). These superfamilies were: ‘Nucleic acid-binding proteins’ (50249), ‘Translation proteins’ (50447), ‘Trypsin-like serine proteases’ (50494), ‘Barvin-like endoglucanases’ (50685), ‘Cyclophilin-like’ (50891), ‘Sm-like ribonucleasepolypeptides’ (50182), ‘PDZ domain-like’ (50156), ‘Prokaryotic SH3-related domain’ (82057), ‘Tudor/PWW/PMBT’ (63748), ‘Electron Transport accessory proteins’ (50090), ‘Translation proteins SH3-like domain’ (50104), ‘Lipocalins’ (50814) and ‘FNM-binding split barrel’ (50475). Of these, we removed the superfamilies ‘Lipocalins’ and ‘Trypsin-like serine proteases’, which were not structurally consistent enough to permit a multiple structure alignment for training HMMER or the SMURF variants, and which were broken into distinct superfamilies by Daniels et al. (2013), with the result that 11 superfamilies containing barrels were selected. SMURFLite was tested on these 5 propeller folds and 11 barrel superfamilies, with interleave thresholds of 1, 2 and 3, and with and without simulated evolution on the beta-strands (Kumar and Cowd, 2014). Here, the interleave threshold is a parameter of SMURFLite that trades off the computational complexity with the ability of the MRF to capture complex long-range dependencies. The balance between accuracy and computational efficiency is determined by the interleave threshold at which SMURFLite is run. In particular, we found that SMURFLite set to an interleave threshold of 3 or less was always fast. Thus, our first question is how SMURFLite with and without simulated evolution performs on our test set when the interleave threshold is set to 3 or less. We found that SMURFLite became extremely slow at an interleave threshold of 4, and essentially intractable at an interleave threshold of 5 or above. While SMURFLite with an interleave threshold of 1 or 2 requires roughly 1 s of wall-clock time on a 12-core 2.4 GHz AMD Opteron server, an interleave threshold of 4 raises this run-time requirement to 7–10 min. Restricting the interleave threshold to 3 or less has different impacts on the different folds in our test set. In particular, the beta strands in the propeller folds never have an interleave >3, which means that full SMURF, as we know, is tractable on these folds. However, we were still interested in how simplifying the random field to an interleave of 2 or 1 would impact performance, and also whether simulated evolution would help. In contrast, the barrel superfamilies in our test set contain a maximum beta-strand interleave of between 4 and 8. Interestingly, none of these barrel families contained any beta strands with an interleave of 3 in
For two of the remaining superfamilies, HMMER performs best; for one superfamily, HHpred performs best (Table 3).

As discussed above, SMURFLite begins to test the limits of computational tractability when interleave numbers of 4 are allowed. Since many barrel structures had beta-strand pairs with interleaves of 4, we wished to test if incorporating these more long-range pairwise dependencies into our MRF would improve performance. Some barrel superfamilies on which we tested had only strand pairs of interleave 1 or 2, excepting a pair of beta strands that close the barrel and thus have an interleave equivalent to the number of strands in the barrel. Certainly, including that last strand is beyond the computational power of SMURFLite. Other barrels, whether open or closed, have more complex strand topology and interleaves of 3 or 4 are common even in the middle of the barrels. We chose to run SMURFLite with an interleave of 4 on one of the barrel superfamilies of moderately complex topology, the ‘Barwin-like endoglucanase’ superfamily, of which an example appears in Figure 1. The ‘Barwin-like endoglucanase’ superfamily contains ‘Barwin,’ a protein that may be involved in a common defense mechanism in plants (Swenson et al. 1999).

On the ‘Barwin-like endoglucanase’ superfamily, we find an enormous improvement in performance from capturing that last strand pair, with AUC improving from 0.63 for SMURFLite with an interleave threshold of 2 and simulated evolution, to 0.94 for SMURFLite with an interleave threshold of 4 and simulated evolution (Fig. 2). Note that both HMMER and RAPTOR fail entirely on this superfamily.

3.2 SMURFLite on whole genomes

We considered all 1852 genes from the bacterium T.maritima, a thermophilic organism that bears some similarity to Archaea and whose cell is wrapped in an outer membrane, or ‘toga’ (Huber et al. 1989). Out of 354 total superfamilies within the SCOP class ‘All beta proteins’, 288 (81%) of which contain at least two protein chains, 207 superfamilies (71%) were structurally consistent enough to be aligned using the Matt (Menke et al. 2008) alignment, so our restriction of running SMURFLite with an interleave threshold of 3 or less is equivalent, on the barrels, to running SMURFLite with an interleave threshold of 2.

SMURFLite with interleave threshold 2 and simulated evolution performs well on all propeller folds, with AUCs between 0.89 and 0.99. It always performs better than HMMER, and better than RAPTOR and HHpred except on the 7-bladed propellers (of which there are 39 non-redundant solved structures in 19 SCOP families), where HHpred achieves an AUC of 0.99 and RAPTOR achieves a slightly higher AUC of 0.95 versus an AUC of 0.93 for SMURFLite with interleave threshold 2 and no simulated evolution (Table 1). Interestingly, on the 5-bladed propellers (of which there are only 14 non-redundant solved structures in 7 SCOP families), adding simulated evolution seems to greatly improve performance; even SMURFLite with an interleave threshold of 2 with simulated evolution outperforms full-fledged SMURF. While these results focus on the accuracy of the MRF score for the remote homolog decision problem, as opposed to the question of alignment quality, we note that SMURFLite with an interleave threshold of 1 or 2 produces highly similar alignments to full SMURF, particularly with respect to placing the ‘blades’ of the 6-, 7- and 8-bladed propellers.

For all 11 beta-barrel superfamilies, there is a maximum interleave number that ranges from 4 (as in the ‘Sm-like ribonucleoproteins’) to 8 (as in the ‘Cyclophilin-like’ superfamily). We find that for 6 of the 11 beta-barrel superfamilies, SMURFLite with an interleave of 2 and simulated evolution outperforms HMMER, RAPTOR and HHpred. For the remaining two superfamilies, HMMER performs best; for two of the remaining superfamilies, RAPTOR performs best; and for one superfamily, HHpred performs best (Table 3).
structure, but there also is not close homology to proteins of solved structure. In particular, none have BLAST hits in UniProt with solved structure and >80% sequence identity, 18 have BLAST hits in UniProt with solved structure and between 30% and 80% sequence identity, and 4 have BLAST hits in UniProt with solved structure and <20% sequence identity. As an example, the gene Q9X087 shares only 20% sequence identity with its closest structurally-solved BLAST hit (Rhoptry protein from Plasmodium yoelii yoelii, which forms an alpha-helical structure) but we predict it to belong in the ‘beta-Galactosidase/glucuronidase domain’ SCOP superfamily with a P-value of 0.0006.

All models predicted can be found at http://smurf.cs.tufts.edu/smurflite/

### 4 DISCUSSION

We have presented SMURFLite, a method that combines long-range pairwise beta-strand interactions via a simplified MRF with simulated evolution, a method that augments training data to capture pairwise beta-strand interactions as well. SMURFLite in most cases performs considerably better than HMMER and RAPTOR; however, we examine those structures for which this is not so. We postulate that RAPTOR performs best in the case when there is significant structural conservation across families, whereas HMMER excels when there is a small but highly conserved sequence signature in members of a superfamily. In all four beta-barrel superfamilies on which RAPTOR achieves an AUC of >0.5, we see considerable structural variation in the protein backbones within each superfamily, according to the metric of Daniels et al. (2012), as compared with the other barrel superfamilies. In contrast, the barrels on which RAPTOR performed best exhibited little structural variation. The cases in which SMURFLite performs poorly exhibit an interesting property: the structural alignment of the protein chains used in the training set preserves few, or sometimes none, of the beta strands as ‘consensus’ beta strands. When a significant number of beta strands are missing in this manner from the training data, SMURFLite exhibits poor specificity, scoring some non-homologous sequences comparably to homologous ones.

The ‘Translation Proteins SH3-Like Domain,’ a superfamily in which HMMER significantly outperforms SMURFLite, is one in which the consensus alignment obtained from Matt retains zero beta strands, even though each individual structure has four strands. Thus, SMURFLite behaves like HMMER, except without HMMER’s heuristic for quickly failing bad alignments, leading SMURFLite to report more false positives. The very premise of SMURFLite rests on the conservation of beta strands, and this finding emphasizes the importance of evolutionarily faithful structural alignments. In future work, we will also consider alternative structural aligners, such as TMalign (Zhang and Skolnick, 2005), in cases where they produce alignments that better conserve secondary structure.
We also compared SMURFLite to HHpred, though in a sense this is not an apples-to-apples comparison, because HHpred uses all of protein sequence space to build profiles for training; thus it can leverage a much larger training set than HMMER, RAPTOR, or SMURF or SMURFLite. Thus it is somewhat surprising that SMURFLite outperforms HHpred in median AUC on the propellers and barrels. We expect HHpred to excel in particular on superfamilies and folds with a high HHPred NEFF (Söding propellers and barrels. We expect HHPred to excel in particular on the SMURFLite outperforms HHPred in median AUC on the can leverage a much larger training set than HMMER, RAPTOR, all of protein sequence space to build profiles for training; thus it heuristic methods (Murphy small compute cluster. We have also shown that increasing the computationally tractable enough to scale to whole genomes, MRF methodology for beta-structural motif recognition that is from 0.73 for full SMURF alone to 0.89.

We have demonstrated that SMURFLite is a powerful MRF methodology for beta-structural motif recognition that is computationally tractable enough to scale to whole genomes, requiring approximately 3h to scan the Z.maritima genome on a small compute cluster. We have also shown that increasing the interleave number for SMURFLite can have dramatic effects on performance, but at a great computational cost. Thus looking at heuristic methods Murphy et al 1995 Smith et al 1995 that approximately compute the SMURF score more efficiently may add even more power to our approach in practice.

ACKNOWLEDGEMENTS
We thank A.Kumar, M.Menke and J.Xu.

Funding: N.M.D. and L.J.C. were funded in part by NIH grant 1R01GM080330. R.H. and B.B. were funded in part by NIH grant 1R01GM08187.

Conflict of Interest: none declared.

REFERENCES

Berman,H. et al (2000) The protein data bank. Nucleic Acids Res., 28, 235-242. Bradley,P. et al (2001) Unified successful prediction of parallel \beta-helices from primary sequence reveals an association with many microbial pathogens. Proc. Natl Acad. Sci., 98, 16819–16824. Cowen,L. et al (2002) Predicting the beta-helix fold from protein sequence data. J. Comput. Biol., 9, 261-276.