

A peer-reviewed version of this preprint was published in PeerJ on 18 April 2017.

[View the peer-reviewed version](https://peerj.com/articles/3139) (peerj.com/articles/3139), which is the preferred citable publication unless you specifically need to cite this preprint.

Chapman SD, Adami C, Wilke CO, B KC D. 2017. The evolution of logic circuits for the purpose of protein contact map prediction. PeerJ 5:e3139 <https://doi.org/10.7717/peerj.3139>

The evolution of logic circuits for the purpose of protein contact map prediction

Samuel D Chapman ¹, Christoph Adami ², Claus O Wilke ³, Dukka B KC ^{Corresp. 1}

¹ Department of Computational Science and Engineering, North Carolina A&T State University, Greensboro, USA

² Michigan State University, USA

³ Department of Integrative Biology, The University of Texas at Austin, Austin, USA

Corresponding Author: Dukka B KC

Email address: dbkc@ncat.edu

Predicting protein structure from sequence remains a major open problem in protein biochemistry. One component of predicting complete structures is the prediction of inter-residue contact patterns (contact maps). Here, we discuss protein contact map prediction by machine learning. We describe a novel method for contact map prediction that uses the evolution of logic circuits. These logic circuits operate on feature data and output whether or not two amino acids in a protein are in contact or not. We show that such a method is feasible, and in addition that evolution allows the logic circuits to be trained on the dataset in an unbiased manner so that it can be used in both contact map prediction and the selection of relevant features in a dataset.

The evolution of logic circuits for the purpose of protein contact map prediction

Samuel D. Chapman¹, Christoph Adami², Claus O. Wilke³, and Dukka B. KC¹

¹Department of Computational Science and Engineering, North Carolina A&T State University, Greensboro, NC, United States

²Michigan State University, East Lansing, MI, United States

³Department of Integrative Biology, The University of Texas, Austin, TX, United States

Corresponding Author: Dukka B. KC¹ E-mail: dbkc@ncat.edu

ABSTRACT

Predicting protein structure from sequence remains a major open problem in protein biochemistry. One component of predicting complete structures is the prediction of inter-residue contact patterns (contact maps). Here, we discuss protein contact map prediction by machine learning. We describe a novel method for contact map prediction that uses the evolution of logic circuits. These logic circuits operate on feature data and output whether or not two amino acids in a protein are in contact or not. We show that such a method is feasible, and in addition that evolution allows the logic circuits to be trained on the dataset in an unbiased manner so that it can be used in both contact map prediction and the selection of relevant features in a dataset.

Keywords: Protein Structure Prediction, Evolutionary Computation, Machine Learning

INTRODUCTION

Proteins are important biological molecules that perform many functions in an organism. These molecules are composed of a string of amino acids comprised of a 20-letter “alphabet” of amino acids. The sequence of amino acids is referred to as the primary structure of the protein. Beyond this primary structure, proteins are arranged in a higher-order, three-dimensional secondary structure composed of motifs such as alpha-helices and beta sheets. This secondary structure in turn is arranged into a tertiary structure that forms protein domains, which in turn can form a quaternary structure that is composed of multiple protein domains (McNaught and Wilkinson, 1997). To a large extent, the two-dimensional and three-dimensional structure of a protein is determined by its amino acid sequence. For example, two cysteine amino acids can form a disulfide bond even if they are separated by a large distance in terms of the sequence (have a large sequence separation) (Sevier and Kaiser, 2002). Protein structure in turn greatly influences the function of a protein.

However, it is still fairly time-consuming and expensive to acquire an accurate structure of a protein. Current experimental methods include crystallizing a protein and performing nuclear magnetic resonance (NMR) imaging (Wuthrich, 1986) or X-ray crystallography (Drenth, 2007). The number of structures that have been determined in these manners is still quite small—on the order of 100,000 as shown in the latest release of the Protein Data Bank (Kouranov et al., 2006) from 2016. This is in contrast to the number of proteins whose primary amino acid sequences have been determined, which is in the tens of millions according to the latest release of RefSeq (Pruitt et al., 2007) from 2016. Thus, it is desirable to find faster and cheaper methods for protein structure determination from sequence. One approach is to use computational and machine learning methods to predict protein structure based on available information. These methods are collectively referred to in this paper as protein structure prediction (PSP).

The benefits of using computational methods in PSP are numerous. In addition to providing a less costly and faster method for structural prediction and determination, such methods can also guide and confirm experimental research. Furthermore, structural prediction can help in understanding evolutionary relationships among organisms (Corbett and Berger, 2004) (Yoshikawa and Ogasawara, 1991); aiding in

46 the development of new drugs (Gaulton et al., 2012) (Koch and Waldmann, 2005); and the production of
47 synthetic proteins (Ho et al., 2001).

48 Computational methods used in protein structure prediction are quite diverse. These include clustering
49 methods (Bolten et al., 2001); neural networks (Rost and Sander, 1994); support vector machines (Cheng
50 and Baldi, 2007); and template methods (Zhang, 2007). Other examples include those using deep
51 learning (Lena et al., 2012), which has recently become popular and used in many applications such as
52 image recognition (Ciresan et al., 2012). Computational methods also predict different aspects of structure,
53 including the exact 3D coordinates of the atoms (as in NMR and X-ray crystallography mentioned above),
54 the secondary structures of the amino acids (Rost and Sander, 1994), and protein bond angles (Laskowski
55 et al., 1993). There also exist *de novo* or *ab initio* methods, such as ROSETTA (Simons et al., 1999) and
56 QUARK (Xu and Zhang, 2012), which use only the amino acid sequence to predict protein structure (Baker
57 and Sali, 2001).

58 Another type of structural determination is the amino acid “contact map”, which we discuss in this
59 paper (Cheng and Baldi, 2007; Lena et al., 2012). A contact map is the list of amino acids that are in
60 proximity to each other below a distance threshold. Computational methods used to elucidate contact
61 maps have included support vector machines (Cheng and Baldi, 2007), deep learning (Lena et al., 2012),
62 integer programming (Wang and Xu, 2013), and cascading neural networks (Ding et al., 2013).

63 In the standard approach to contact map prediction (CMP) by machine learning, the methods are
64 trained on a set of training examples, and after the training periods are tested on a set of unknown testing
65 examples. The training and testing data can encompass any kind of information and class label. For
66 example, in protein contact map prediction, a typical example would be a particular pair of amino acids
67 in a protein. This pair would have many characteristics associated with it (features such as amino acid
68 identity, protein length, and so on), with an accompanying class label (in contact or not in contact) (Cheng
69 and Baldi, 2007). The scoring need not be based on class label prediction; for example, in cases
70 where atomic coordinates are predicted, scoring can be based on the predicted distance from the true
71 coordinates (Cozzetto et al., 2009).

72 One popular type of machine learning is the use of evolutionary computation (EC), which has also
73 been applied to PSP (Pedersen and Moulton, 1996). Broadly speaking, this class of machine learning evolves
74 a population of individuals *in silico* that each represent a candidate solution to the problem at hand. The
75 best-performing individuals (*i.e.*, those with the highest score according to a given *fitness function*) tend to
76 perpetuate in the evolutionary process, improving the performance of the method (Back et al., 1997). This
77 class of methods has been used in many applications and takes many forms, in areas as diverse as medical
78 imaging (Baluja and Simon, 1998), data mining (Alcala-Fdez et al., 2009), signal processing (Fogel,
79 2000), and artificial life (Adami, 1998; Ray and Hart, 1999; Ofria and Wilke, 2004).

80 In an evolutionary computation program, the representation of an individual—that is, the digital
81 encoding of the individual—can be varied. These can include tree structures (Cramer, 1985), strings of
82 digits (Edlund et al., 2011), and even software code itself (Ofria and Wilke, 2004). The encoding of these
83 individuals must then be translated into a solution for the problem at hand; for example, in the work of
84 Ofria *et al.*, the digital organisms composed of computer code are scored based on the complexity of
85 logical operations they perform (Adami, 1998; Ofria and Wilke, 2004).

86 A particular representation of candidate solutions in evolutionary computation is known as a Markov
87 network (MN). Markov networks are a way of relating mathematical variables to one another. These
88 relations are probabilistic; when the probabilities are either 0.0 or 1.0, they are said to be deterministic.
89 The variables in Markov networks can be arranged such that they are essentially digital logic circuits
90 with deterministic probabilities. That is, they are composed of logic gates that accept binary inputs (0
91 or 1) and produce binary outputs. Markov networks have in recent years become a tool in areas such as
92 the navigational control (Edlund et al., 2011), active categorical perception (Marstaller et al., 2013) and
93 machine learning in image recognition (Chapman et al., 2013).

94 This paper examines the use of evolving, deterministic Markov networks toward the problem of
95 contact map prediction. In this work, we evolve Markov networks on a training set comprised of pairs of
96 amino acids, and at the end of the evolution test the best networks on a testing set. The class labels of the
97 examples are whether or not the pairs are in contact or not in contact, and the examples have a mixture of
98 688 binary and decimal features that describe them. The training and testing data are each comprised of
99 several hundred thousand examples containing both negative and positive contacts. The dataset used is
100 taken from SVMcon, a paper that uses support vector machines for protein contact map prediction (Cheng

Table 1. The characteristic logic table for a deterministic gate with two inputs and two outputs.

Inputs		Outputs	
<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>
1	1	0	1
0	1	0	0
1	0	1	1
0	0	0	1

and Baldi, 2007). In our work, the features are used as inputs to the MNs (logic circuits), and the outputs are the class labels. Our results show that evolving logic circuits to predict protein contact maps is a viable alternative. To our knowledge, this is the first time that Markov networks have been used in machine learning on a bioinformatics problem. Thus, it is a feasible avenue for study and has the possibility of being quite useful for this problem. Of course, as there is continual improvement in the fields of evolutionary computation and Markov networks, this study should not be seen as representing the “last word” on the capacity of this method to tackle this problem.

Although contact map prediction has improved over the years, there are still a number of challenges remaining. For example, it is unlikely that the entire feature set is useful for classification, a fact mentioned in the SVMcon paper (Cheng and Baldi, 2007). Indeed, it is extremely difficult to know beforehand whether a subset of features performs better on the task, and it may be that the only way to determine this for sure is to manually take out features and re-run the method, an approach that is impractical. In addition, if it is true that a subset of features may be better than a full set, then it follows that it is difficult to know if the addition of more features may help to solve the problem.

Evolutionary computation has a number of benefits and addresses some of the concerns outlined above. For example, once the appropriate fitness function has been selected (along with the evolutionary parameters such as mutation rate and number of individuals in the population) the program attempts to evolve a most-fit individual according to the fitness function and fitness landscape. There is no “correct” information that the EC system is told to use, but rather only the information that produces the best outcome through evolution. Thus, when combined with Markov networks, an EC system can *discover relevant features* in an unbiased manner. This is demonstrated by observing which features the Markov networks evolved to use in their structures—if a feature was used, this indicates that it helped in increasing fitness and was therefore important.

MATERIALS AND METHODS

Description of Markov Networks and their Evolution

Markov networks (MNs) are a set of probabilistically interacting state variables (Koller and Friedman, 2009). The sets of state variables in a Markov networks are often arranged into input state variables, output state variables, and “hidden state” variables that are “inside” the network and can serve as memory and can be used for processing. This arrangement can be used to represent many processes, such as robotic controllers (Edlund et al., 2011) (*e.g.* with sensory inputs and movement outputs) or in image classification (Chapman et al., 2013) (*e.g.* pixels as inputs and image classes as outputs). In this work, the tables that determine the network updates are deterministic rather than probabilistic, so that the Markov networks represent regular logic circuits.

An example of a 2-input, 2-output gate is shown in Table 1. Here, *a* and *b* represent the binary inputs and *c* and *d* represent the binary outputs.

The Markov networks (logic circuits) are created through an evolutionary process. Therefore, the networks themselves must be encoded symbolically into a “genome.” This is accomplished by representing the MNs as a string of integers (bytes), the length of which can be arbitrary but is limited to 40,000 integers in our work. The genome of an individual MN is comprised of a set of “genes,” which each specify one logic gate. Each gene that represents a logic gate begins with an integer that denotes the start of the gene, and the gene itself specifies where the specific inputs and outputs are located, as well as the logic of the gate. This information is sufficient to describe the entire Markov network. Any other

143 characteristics of the candidate solutions, such as the maximum genome size, the rates of mutation, or the
144 number of inputs and outputs to a gate, is specified in a configuration file.

145 The evolution of the Markov networks in our study follows simple evolutionary operators that act on
146 the genome of an individual. Evolutionary operators that can act upon the genes include point mutation
147 (which change a single byte) or gene duplication and deletion, which duplicate or delete an entire gene,
148 respectively. The rates of each of these are specified in the configuration file. In our work, if two or more
149 genes write into the same output, the the final output is OR'd together. More details of the general makeup
150 of Markov networks are provided in the Supplementary Information of Edlund *et al.* (Edlund *et al.*, 2011).

151 Dataset Description

152 Our work centers on protein structure prediction (PSP); specifically, we investigate how protein feature
153 data can be used to predict the amino acid contact map. In order to do this, a dataset of protein sequences
154 and protein sequence features must be prepared. We choose to use the feature dataset provided in Cheng
155 and Baldi (Cheng and Baldi, 2007). That paper described the creation of a contact map predictor based on
156 support vector machines, SVMcon. Their dataset included several hundred proteins divided into a training
157 dataset used to train the SVM (485 proteins) and a testing dataset (48 proteins). Each protein was used
158 to provide a large number of amino acid contact examples based on the many possible amino acid pairs.
159 The sequence separation threshold for pairs was six or greater, and the threshold for contact was eight
160 Angstroms or greater.

161 Each pair of amino acids in the training or testing set represented a possible contact pair, *i.e.*, each
162 pair was either in contact or not in contact. The training dataset contained 267,702 contact pairs, reduced
163 from a set of several million. This reduction was done in order to make the dataset more tractable and
164 also to increase the proportion of positive contacts; the final training dataset had 94,110 positive contacts,
165 a proportion of roughly one-third. The reason the proportion of positive contacts was increased in the
166 training dataset was to train the SVM to better recognize these examples. The testing dataset, composed
167 of 377,797 examples, used the full proportion of contact examples and contained 10,498 positive contacts.

168 Each training and testing example had 688 features associated with it. These features included many
169 types of data related to the sequences, such as mutual information, length of sequences, and secondary
170 structure, to name a few. There were 145 binary features (0 or 1) and 543 decimal features (capable
171 of being any number). Many of these features were based on a sequence alignment of the proteins,
172 which allowed all sequences to be compared according to a standardized length. The majority of features
173 were based on sliding windows centered around the amino acids in each pair and a window centered
174 halfway between the amino acids. There were nine window positions each for each amino acid in the pair
175 (including the amino acid in question) and five positions for the central segment. For each of these 23
176 positions, there were 27 features giving the entropy (one feature), secondary structure (three features),
177 solvent accessibility (two features) and the amino acid profiles for those positions (21 features). Thus,
178 the window features comprised $23 * 27 = 621$ out of 688 features. It should be noted that out of the 688
179 features, only those features associated with secondary structure and solvent accessibility could not be
180 calculated precisely from the sequence along. Secondary structure and solvent accessibility were predicted
181 using other software (Cheng *et al.*, 2005). A more-detailed description of the dataset and features used
182 can be found in the SVMcon paper (Cheng and Baldi, 2007).

183 Data Encoding

184 Markov networks take binary input. Since many of the features are continuous, it is necessary to develop
185 an input encoding that maps to the features. There are several methods that were tried.

186 The methods that were tried were based on splitting continuous features into a set number of bins. For
187 each type of encoding, a “split” value was given. Each continuous feature was divided into a split value
188 of bins based on the complete range of that particular feature (in the training set). For example, if using
189 a split of 10, and with a feature with a range from 0.0 to 2.0, the first bin would be from [0.0-0.2) the
190 second from [0.2-0.4), and so on. If the testing set had a range that was out of bounds of the training set,
191 the lowest and/or highest bin was used.

192 The methods that were tried were based on splitting continuous features into a set number of bins.
193 For each type of encoding, a split number was given (in our case, there were three splits used—4, 10,
194 or 16). Each continuous feature was divided into a split number of bins based on the complete range of
195 that particular feature (in the training set). For example, if using a split of 10, and with a feature with a
196 range from 0.0 to 2.0, the first bin would be from [0.0-0.20] the second from (0.2-0.4], and so on. If the

197 testing set had a range that was out of bounds of the training set, the lowest and/or highest bin from the
 198 training set was used. In order to make all feature representations equal, each binary feature had the same
 199 number of inputs as a continuous feature; for example, if the split was 4, then each binary input used 10
 200 bits, which were either all 0's or all 1's.

201 Two additional types of encoding were based on the fact that even-numbered splits can be expressed
 202 in terms of base 2. For example, a split of 4 bins could be compressed into 2 bits by using the binary
 203 values 11, 10, 01, and 00, and a split of 16 bins could be compressed into 4 bits along the same principle.
 204 These two binary splits used were: First, a split of 16 across four bits, and a split of four across two bits.

205 We chose these five particular splits for a number of reasons. Based on the general layout of the data,
 206 we decided that the lowest split should be 4 as anything less would be too granular on feature data that
 207 tended to have at least 4, and often more, distinct values. Second, the highest split of 16 was chosen as a
 208 reasonable upper limit, and 10 was chosen as the midpoint between the lowest and highest splits. There is
 209 no theory suggesting which split to use, but in practice Markov networks can sometimes have difficulty
 210 evolving to find the correct inputs if there are too many of them, which slows down the clock time of the
 211 simulation itself (Chapman et al., 2013). The number of inputs for the split of 4 was $4*688 = 2752$, the
 212 number for the split of 10 was $10*688 = 6880$, and the number for the split of 16 was $16*11008$. Due to
 213 time limitations, it was impossible to perform an exhaustive parameter sweep of all the splits, even from 2
 214 to 16.

215 Evolution of Markov Networks on the Dataset

216 The training (evolution) of the Markov networks is achieved by the following steps. We first create an
 217 initial population of random networks. We then evolve these networks over a number of updates on a
 218 subset of the training dataset. In our case, we ran the evolution over 100,000 updates. For every 25,000
 219 updates, we evolved the networks on a different set of 50,000 training examples. The sets of training
 220 examples were separate; thus, there used a total of 200,000 training examples. We used different sets of
 221 training examples for two reasons. First, it forced the Markov networks to evolve on different examples
 222 over time instead of focusing only on one dataset. Second, because of time limitations, it was not possible
 223 to evolve the networks on a greater number of examples.

224 During the runs, the networks “learn” (through evolution) to differentiate between positive and
 225 negative contacts based on their features. In each update, the networks are tested on the training set and
 226 a fitness is assigned to them according to a fitness function. Individuals with the highest fitness tend to
 227 survive in the population and reproduce; these “children” undergo mutation, and in the process, may do
 228 better than their parents at recognizing contacts. Mutation can affect almost any part of the networks,
 229 including the number of features used as input, the number of gates, and the gate logic. In the end, there
 230 are two binary outputs provided: an output for a negative contact decision, and one for a positive contact
 231 decision. Note that it is possible for a Markov network to give both answers, signifying both a positive
 232 and negative contact decision, but in practice, this does not happen very often.

233 In our work, we have tried a number of fitness functions, and so far accuracy has proven to be the
 234 best in terms of the final scoring method, described below. Accuracy is defined as $TP + TN / P + N$,
 235 where TP is the number of true positive (correctly-predicted) guesses and TN is the number of true
 236 negative (correctly-predicted) guesses in the dataset. Here, accuracy is calculated separately for both
 237 positive contacts and negative contacts. Also, a reward for output alone is also included for each accuracy
 238 to encourage “empty” networks to evolve to produce outputs. The vector magnitude of these results
 239 is combined as shown in Equation (1), and therefore the maximum fitness is the square root of 8.0.
 240 Calculating the square root is done in order to “smooth” the resulting fitness values. At each update,
 241 every newly-produced individual MN that has not been tested on the training set is tested according to the
 242 fitness function.

$$243 \quad f = \sqrt{(\text{acc}_{\text{pos}} + \text{out}_{\text{pos}})^2 + (\text{acc}_{\text{neg}} + \text{out}_{\text{neg}})^2}, \quad (1)$$

244 At the end of an evolutionary run, the best MN in a population is tested on the testing set, which it has
 245 not seen before, and performance is assessed according to specificity, sensitivity, and Fmax. Specificity
 246 is the number of correctly-predicted positive contacts divided by the number of total predicted positive
 247 contacts. Sensitivity is the number of correctly-predicted positive contacts over the total number of
 positive contacts. Because it is easy to get a specificity of 1.0 by simply giving one very good guess, and

Table 2. Parameters for the evolutionary algorithm.

Parameter	Value
Updates	100,000
Population size	500
Starting gates	100
Inputs per gate	4
Outputs per gate	4
Gene duplication rate per update	0.05
Gene deletion rate per update	0.05
Site mutation rate per update	0.001

248 a sensitivity of 1.0 by simply guessing all contacts to be positive, Fmax is also used, which combines the
 249 two measures and is shown in Equation (2). The maximum Fmax possible is 0.5.

$$Fmax = \frac{specificity * sensitivity}{specificity + sensitivity} \quad (2)$$

250 Note that negative contacts are not considered in the three performance measures (even though they
 251 are used in the fitness function), since positive contacts are of more interest and more difficult to determine
 252 in contact map prediction. Finally, even though Fmax is used as the scoring measure, we deemed that
 253 it was inferior to accuracy as a fitness function.

254 In order to achieve better performance, committees of MNs are assembled from the highest-fitness
 255 individuals from each of the 60 evolutionary runs. These 60 MNs are tested on the testing set, and
 256 for each testing example, the sum of all 60 negative contact answers is compared to the sum of all 60
 257 positive contact answers. The final answer is the answer with the highest number of votes (ties are broken
 258 randomly).

259 The list of parameters for the evolutionary runs is given in Table 2.

260 RESULTS

261 Performance on the Dataset

262 Markov networks were evolved using five different input data encodings (treatments), described in the
 263 Methods section. For each encoding, 60 different populations were run and the best individuals from each
 264 population (according to the fitness function) were tested on each example from the testing set and their
 265 answers combined. The evolution continued for 100k updates, and at intervals of 25k updates, results
 266 on the test set for committee sizes up to 60 were recorded. Figures 2, 3, 4, and 5 show the results for
 267 the treatments that used splits of 4 with 4 bits per feature, splits of 16 with 16 bits per feature, splits of 4
 268 with 2 bits per feature, and splits of 16 with 4 bits per feature, respectively. Even though the treatments
 269 ended at 100k updates, the best results for all treatments were from 75k updates; this is probably due
 270 to overfitting of the Markov networks to the training set in their evolution. Also, the improvement in
 271 performance of each treatment at the four update intervals tended to level out after around 20 committee
 272 members and does not tend to improve with more. This is an indication that increasing the number of
 273 committee members would not help the performance for this treatment.

274 Figure 6 shows the Fmax results at 75k updates for all treatments. It is interesting to note that the
 275 two best treatments were the two that compressed the number of bits per feature according to a base-2
 276 compression: the treatment that used a split of 16 with 4 bits/feature had a final Fmax of 0.103 and the
 277 one with a split of 4 with 2 bits/feature had a final Fmax of 0.102. The splits of 10 with 10 bits per feature,
 278 16 with 16 bits per feature, and 4 with 4 bits per feature had Fmax results of 0.098, 0.097, and 0.097,
 279 respectively.

280 All of the treatments did significantly better than random. Under random guessing that guessed 50
 281 percent of the examples to be positive contacts, the Fmax would be 0.026. Under random guessing that
 282 used the proportion of positive contacts from the training set (35.155 percent), the Fmax would also be
 283 0.026.

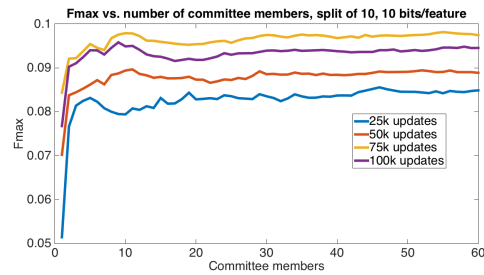


Figure 1. Results for the split of 10 with 10 bits per feature (6880 total bits). The highest Fmax at 60 committee members is at 75k, with an Fmax of 0.098.

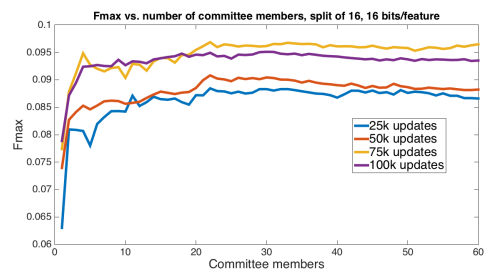


Figure 2. Results for the split of 16 with 16 bits per feature (11008 total bits). The highest Fmax at 60 committee members is at 75k, with an Fmax of 0.097.

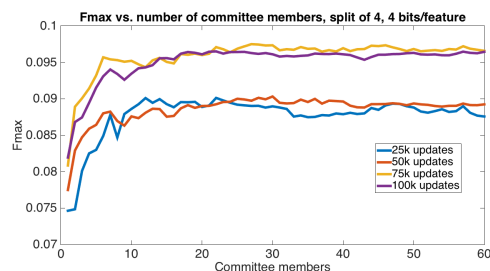


Figure 3. Results for the split of 4 with 4 bits per feature (2752 total bits). The highest Fmax at 60 committee members is at 75k, with an Fmax of 0.097.

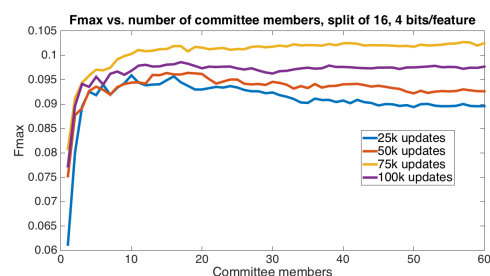


Figure 4. Results for the split of 16 with 4 bits per feature (2752 total bits). The highest Fmax at 60 committee members is at 75k, with an Fmax of 0.102.

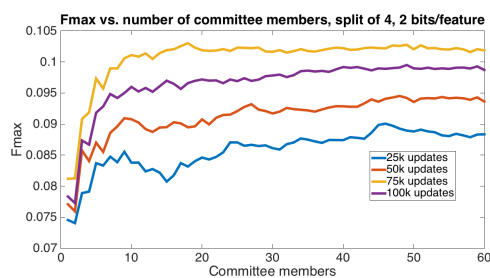


Figure 5. Results for the split of 4 with 2 bits per feature (1376 total bits). The highest Fmax at 60 committee members is at 75k, with an Fmax of 0.102.

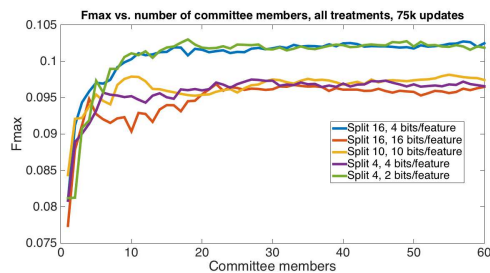


Figure 6. Results at 75k updates for all five split treatments. The highest Fmax is achieved by the split of 16, 4 bits per feature encoding, with an Fmax of 0.103.

284 We now take a detailed look at the treatment that used a split of 4 with 2 bits/feature. This treatment
 285 was chosen for two reasons. First, its Fmax performance is very close to the best (Fmax of 0.102 vs.
 286 Fmax of 0.103 obtained by the encoding using a split of 16 with 4 bits/feature); second, it is the simplest
 287 type of encoding, using only 2 bits/feature, making it simpler to analyze and describe. Figure 7 shows the
 288 specificity and sensitivity values over the committee sizes at 75k updates for the split of 4 with 2 bits per
 289 feature treatment. The specificity at 60 committee members was 0.14, and the sensitivity was 0.35. As
 290 one can see, even if sensitivity goes down, specificity can go up, leading to a higher Fmax value.

291 We also compare our results from the split 4, 2 bits/feature encoding to the results from the SVMcon
 292 paper by Cheng *et al.* (Cheng and Baldi, 2007). Table 3 gives the total specificity and sensitivity values
 293 on the testing set for Markov networks and SVMcon across several sequence separations (values under
 294 the separation of ≥ 6 are for every testing example since that is the smallest separation possible). As
 295 one can see, at all sequence separations, SVMcon has a higher specificity, but SVMcon has a higher
 296 sensitivity. Table 4 shows the Fmax results on the testing set across the six SCOP protein classes covered
 297 by the testing set and three sequence separation thresholds. Unfortunately, Markov networks do not do as
 298 well as SVMcon. It is clear that this is because SVMcon does better in terms of specificity; therefore, if
 299 Markov networks could be made to be more conservative in their guesses, they might achieve a higher
 300 overall specificity without sacrificing sensitivity.

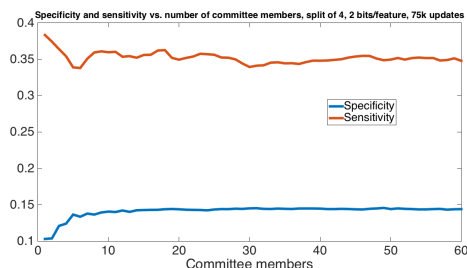


Figure 7. Specificity and sensitivity results at 75k updates for the split of 4 with 2 bits per feature treatment. Specificity at 60 committee members was 0.14, and sensitivity was 0.35.

Table 3. Average specificity and sensitivity of Markov networks and SVMcon across different sequence separations.

Method	Separation ≥ 6		Separation ≥ 12		Separation ≥ 24	
	Spec.	Sens.	Spec.	Sens.	Spec.	Sens.
Markov networks	0.144	0.347	0.132	0.280	0.109	0.209
SVMcon	0.37	0.21	0.30	0.20	0.21	0.19

Table 4. Fmax comparisons between MN—SVMcon of different SCOP protein classes and sequence separations.

SCOP class	Number	Fmax, separation ≥ 6		Fmax, separation ≥ 12		Fmax, separation ≥ 24	
		MN	SVMcon	MN	SVMcon	MN	SVMcon
alpha	11	0.076	0.12	0.012	0.087	0.009	0.050
beta	10	0.102	0.117	0.093	0.111	0.069	0.096
alpha+beta	15	0.109	0.161	0.094	0.146	0.070	0.110
alpha/beta	7	0.105	0.126	0.103	0.121	0.097	0.117
small	4	0.089	0.120	0.051	0.113	0.021	0.063
coil-coil	1	0.063	0.142	0.023	0.025	N/A	N/A
all	48	0.102	0.134	0.090	0.120	0.072	0.010

301 Network Recognition of Features

302 In an evolutionary run, the Markov networks evolve to use certain input bits (and therefore features) to
 303 make their decisions; that is, they evolve to recognize the subset of input bits that give them the best
 304 answer (“salient” bits). Figure 8 shows a typical network with its output from the treatment that used a
 305 split of 4 with 2 bits/feature, at 75,000 updates. Each green circle is an input used by the network, with
 306 the red circles representing gates and the blue circles outputs. Because this figure is illustrative, the inputs
 307 and gates are unordered. There are several things to notice in this figure. First, even though there are
 308 1,376 possible bits that the network could use, it only connects to 118 in this example. Second, in this
 309 network it is common for gates to have connections to multiple input nodes and for outputs to have many
 310 gate inputs—the networks are quite complex. Third, there are three outputs. Our system allows up to two
 311 outputs to evolve for each class label, so that one output in a pair can serve as a “veto” output to prevent
 312 too many “yes” outputs. Here, the positive contact label (on the right) evolved two outputs.

313 Because of the nature of the evolutionary process, the networks can select the best features in an
 314 unbiased manner. Figure 9 demonstrates how networks focus on some features rather than others. In
 315 this histogram, taken from the results from the encoding that uses a split of 4 with 2 bits/feature at 75k
 316 updates, we plot the frequency distribution of features used over the networks that use them, by assuming
 317 that a feature is “recognized” if there is at least one network that reads from one of the bits for that feature.

318 We see that most features are recognized by only a few networks—for example, nearly a hundred
 319 features are recognized by only 3 networks, with nearly another hundred recognized by only 4 networks.
 320 However, less than 20 features are recognized by no networks. This may be due to networks randomly
 321 evolving to recognize a feature before losing that recognition in the evolutionary process, such that it
 322 would be very difficult for no networks to recognize a feature. Also, no features are recognized by *all* 60
 323 networks, although there are a few that come close.

324 Table 5 gives information on the general statistics of the networks according to how many features
 325 they recognized. As shown in this table, the average and median tend to be quite similar when examining
 326 the number of bits recognized by the networks and the number of features. In addition, even though the
 327 maximum number of possible bits used is twice that of the number of features used, this number is not
 328 very much higher than the number of features used. In fact, overall, the average number of features used
 329 (90.0) is quite small compared to the maximum of 688 and is approximately 13 percent.

330 It is desirable to examine more closely which features the Markov networks select, because it stands to

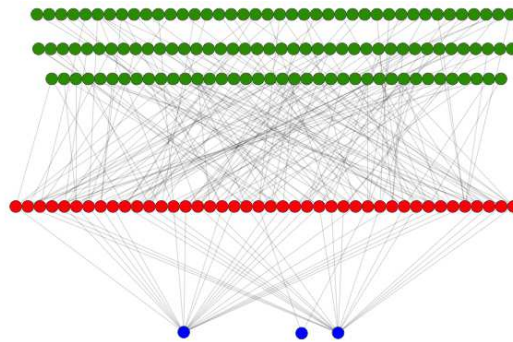


Figure 8. A sample network diagram taken from the treatment of a split of 4 with 2 bits/feature at 75k updates. Out of a possible 1376 bits, the network has evolved to recognize only 118 of these. Inputs bits are green, gates are red, and outputs are blue. The inputs and gates are unordered. Note that a pair of outputs has evolved to represent a positive contact answer (the maximum is two), but that the negative contact answer evolved only one.

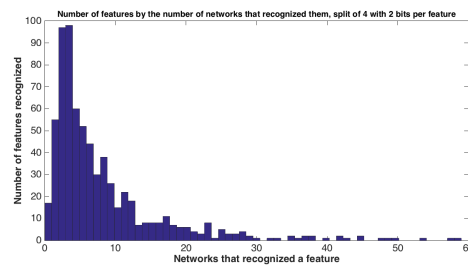


Figure 9. A histogram showing how many features are recognized by each number of networks (encoding split of 4 with 2 bits/feature). A network only has to have input from one bit of a feature to recognize it.

Table 5. Feature recognition values of the networks from the encoding of split 4 with 2 bits/feature. For a network to recognize a feature, it only has to connect to one bit for that feature.

Statistic	Networks
Average bits recognized	97.6
Median bits recognized	98.0
Average features recognized	90.0
Median features recognized	90.5

Table 6. The features most recognized by the Markov networks.

Feature	Networks
Contact pair sequence separation ≥ 50	59
C-terminus amino acid window position 5, sheet secondary structure	58
N-terminus amino acid window position 5, sheet secondary structure	54
Amino acid central segment window position 4, coil secondary structure	50
Amino acid central segment window position 4, sheet secondary structure	49
Contact pair sequence separation ≤ 49	48
Amino acid central segment window position 5, coil secondary structure	45
N-terminus amino acid window position 5, exposed solvent accessibility	45
Amino acid central segment window position 5, sheet secondary structure	43
Contact pair sequence separation of 6	42
N-terminus amino acid window position 5, buried solvent accessibility	42
C-terminus amino acid window position 5, buried solvent accessibility	40

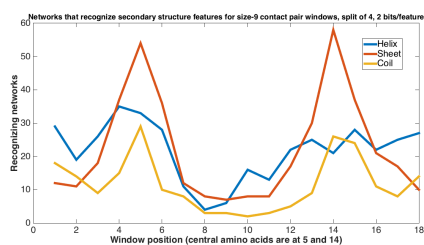
331 reason that these salient are important in general for the task of contact map determination. Table 6 gives
 332 a list of the top 12 features in the treatment according to the number of networks that recognized them. It
 333 is clear from this table that secondary structure and solvent accessibility features are very important to
 334 the Markov network decisions, and presumably to determination of contact map prediction in general.
 335 The importance of contact pair sequence separation to the networks, specifically a separation of 6 and a
 336 separation of ≥ 50 , seems to suggest that the networks use the extremes of distance as a way to help
 337 with classification.

338 Figure 10(a) demonstrates feature usage related to secondary structure features. Each pair of amino
 339 acids in a training or testing sample is situated in a sliding window of size 9, giving a total number of 18
 340 positions. There are a number of features at each of these positions, including the secondary structures. At
 341 each of these window positions, the figure shows the number of networks (out of the 60) that evolved to
 342 recognize the three kinds of secondary structure after 75,000 updates. We also see that the most common
 343 type of secondary structure recognized by the MNs is sheet. Further, the MNs focus on recognizing
 344 secondary structures that are closer to the central amino acids. Interestingly, even though the number of
 345 networks for each kind of secondary structure decreases as the window positions move away from the
 346 center, the number of networks for the “outside” positions for each window is greater for both helix and
 347 coil. It is also clear that the peaks in the centers and outsides of the windows are high in absolute terms as
 348 well; most features in the dataset are not recognized by many networks.

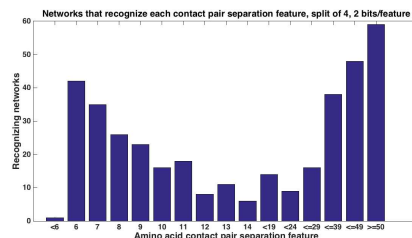
349 Similarly, Figure 10(b) shows from the same treatment how many networks out of 60 evolve to
 350 recognize each feature that describes the sequence separation between the amino acid pair. There is a
 351 clear focus on sequence separations that are either very small or very large. One can see that, at least in
 352 relation to this dataset, the Markov networks are recognizing that certain pair separations are more useful
 353 than others.

354 It is clear from these figures that a number of secondary structure and sequence separation features are
 355 important to the networks. Indeed, 9 out of the top 10 features in terms of networks that recognized them
 356 deal with either secondary structure or sequence separation (the tenth deals with solvent accessibility).

357 As mentioned before, since MNs will tend to evolve to recognize features that benefit them, they will
 358 usually only recognize a small subset of the total features. To further demonstrate that the features chosen



(a)



(b)

Figure 10. (a): Number of networks out of the 60 that evolved to recognize each kind of secondary structure along the two size-9 sliding windows. Encoding was split of 4, 2 bits/feature. (b): Number of networks out of the 60 that evolved to recognize the amino acid pair separation features. Encoding was split of 4, 2 bits/feature. Each tick shown is a different contact separation feature.

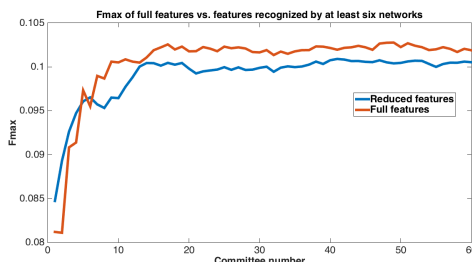


Figure 11. Fmax of the original split-4, 2 bits per feature encoding with all features, and the same kind of run with the reduced feature set that only used features recognized by at least 6 of the networks from the first run.

359 by the MNs are useful, other evolutionary runs were performed that had as input only the most-used
 360 features in the original runs. To this end, the 60 networks from the encoding that used a split of 4 with 2
 361 bits per feature were examined after a run of 75,000 updates to see which features they recognized. The
 362 training and testing datasets were changed to contain only features that were recognized by at least 6 of
 363 the networks, and a new evolutionary run with the same parameters was performed on this reduced-feature
 364 dataset. The number of features in this new run was 309, or roughly 45 percent of the original number
 365 of features. Figure 11 shows the Fmax results of this new run compared to the original run. Although
 366 the Fmax results of the new run are smaller, this difference is quite small, indicating that the Markov
 367 networks have discovered what the salient features for solving the problem. Also, it demonstrates that
 368 searching for the right features via trial and error is perhaps not necessary; simply picking the features
 369 that the most networks recognize is suitable for finding the best features.

370 This figure demonstrates two things. First, many features that were chosen for the dataset were
 371 unnecessary to obtain the same level of performance. In addition, the Markov networks successfully
 372 evolved to discover which features were useful in contact map prediction.

373 It is worth noting that many of the 688 features that were not used in the reduced feature set were
 374 features that described the amino acid percentages that were based on a sequence alignment of the
 375 proteins. Considering the two size-9 windows and the size-5 central segment window, and the fact that

376 the amino acids were the 20 canonical amino acids plus a gap, there were $23 \times 21 = 483$ total features in
377 the dataset based on amino acid percentages alone. Yet only 164 of these (34 percent) were used in the
378 reduced-feature dataset, as opposed to 145 of the remaining 205 features (71 percent of the remaining).
379 Some notable observations included the fact that all but one (22 out of 23) of the gap amino acid features
380 qualified. This could be due to the importance of gap additions in sequence alignments. In addition, most
381 of the central segment window amino acid features were salient—88 out of 104, or 85 percent—and often
382 had a relatively high number of networks that recognized them.

383 The salient features of the size-9 windows around the contact pairs were not as numerous (76 out
384 of 378 features, or 20 percent) and tended to have relatively low numbers of networks that recognized
385 them. An interesting exception was the feature for cysteine in the fifth position (central position) for the
386 C-terminus window. A total of 19 networks recognized this feature. Because cysteine is so important to
387 protein structure, this is not surprising, and the cysteine feature in the center of the N-terminus window
388 was recognized by 8 networks as well.

389 DISCUSSION

390 As demonstrated in our results, the performance of the evolution of Markov networks depends mainly on
391 two things: First, the fitness function used, and second, the encoding of the dataset. For example, with
392 respect to encoding, the two treatments that performed the best were the two that used the reduced-bit
393 binary encodings. It is hypothesized that one reason that these two treatments did better than the other
394 three is that there were fewer bits for the MNs to evolve to choose, but also that condensing bits would
395 “force” the MNs to evolve over all bits for a continuous feature due to the nature of the binary encodings.

396 Also, with respect to fitness functions, the best found so far has been accuracy. This is perhaps
397 due to the fact that accuracy is such a simple fitness function—it is simply the proportion of correct
398 guesses (true negative true positive) in the dataset and does not require a complex formula. Furthermore,
399 while other fitness functions such as Fmax measure, specificity/sensitivity, or Matthews correlation
400 coefficient (Matthews, 1975) have been tried and have not performed as well as accuracy, there is
401 the intriguing idea that a “committee of committees” could be used based on a combination of the
402 answers from runs of several fitness functions. Furthermore, it has been noted that a different fitness
403 function/encoding combination might produce better results; in addition, a different type of evolutionary
404 algorithm (such as NSGA2 (Deb et al., 2002)) or different evolutionary parameters, such as the type of
405 population replacement (in our case, we used tournament selection) (Blickle and Thiele, 1996) could
406 possibly foster a more-productive fitness landscape and therefore performance. In particular, Markov
407 networks, at least compared to the SVMcon results, were not very conservative in their answers, and thus
408 had a higher sensitivity (and lower specificity). If it could be determined *why* the networks had trouble in
409 this regard, it might be possible to improve their performance by tuning the factors mentioned above. The
410 open-ended nature of evolutionary computation is both a blessing and a curse. The usage and theory of
411 evolutionary computation is continuously being worked on and improved by many researchers, allowing
412 for the possibility of a great increase in performance of this method. However, it is always unknown
413 which specific fitness function and other parameters to use, and thus there is still an element of trial and
414 error.

415 One of the primary strengths of the evolution of Markov networks is that the evolution proceeds in
416 an unbiased manner. Thus, as demonstrated in the results section, networks evolve to recognize some
417 features more than others. This is useful, since it allows one to differentiate some features as being “better”
418 than others according to how many networks evolve to recognize them. Figure 9 shows that there is quite
419 a bit of diversity in terms of which features are recognized as salient, and Table 6 shows how certain
420 groups of related features features (*e.g.*, secondary structure and solvent accessibility) are important to
421 the decisions of the networks. Furthermore, Table 5 shows that the networks are fairly parsimonious in
422 their decisions and need only a relatively small fraction of the total bits to make their decisions. Finally,
423 Figure 11 shows that, when considering only the most-used features, they are capable of performing
424 almost as well when using all the features. This parsimonious behavior that focuses on only a few salient
425 features, and which can also identify groups of related features, shows that Markov networks do not
426 necessarily have to grow to an arbitrary and unwieldy complexity in order to achieve improved results,
427 even though they are capable of doing so.

428 CONCLUSION

429 We have shown that the evolution of Markov networks is able to produce MNs that are able to make
430 contact map predictions and recognize relevant features from the SVMcon dataset. As far as we know,
431 this is the first time that such a method has been used on a bioinformatics problem. The results show
432 that the method is promising, and may have wider applications. Naturally, because this is the first time
433 such a method has been used in this manner, there is room for improvement. In particular, the Markov
434 networks we evolved tend to overpredict, sacrificing specificity while increasing sensitivity. There are a
435 number of specific ways in which our method could be improved, including using a different evolutionary
436 algorithm, fitness function, MN structure, encoding, or feature space. We could also use our method in
437 conjunction with other methods; for example, if used with other methods, our method and others could
438 form a committee where each receives a voting weight for a contact pair example. One such additional
439 method could include PsiCOV (Jones et al., 2012), which involves finding correlated mutations between a
440 protein containing the contact pairs and a large database of proteins. Other possible additional methods
441 include the LASSO feature selection method (Tibshirani, 1996), and random forests (Liaw and Wiener,
442 2002; Rainforth and Wood, 2015), among others.

443 Furthermore, we have shown that the evolution of Markov networks can, in an unbiased manner,
444 produce networks that recognize relevant (salient), useful features from a dataset. Choosing these salient
445 features could help to remove extraneous features, a task that would otherwise be computationally
446 intensive. This could be especially relevant to a problem that uses many more features (in the thousands
447 or tens of thousands) than ours. Also, one could use this feature information as a prediction tool to guess
448 which kinds of features are useful or not. For example, if it is demonstrated that all features relating to a
449 particular class of amino acid (*e.g.* polar amino acids) are used by the Markov networks, then it might be
450 desirable to find more features of this type. Furthermore, this offers the possibility that Markov networks
451 could be used as a general-purpose feature detector in other scientific work.

452 ACKNOWLEDGMENTS

453 We wish to thank Dr. David B. Knoester for developing the Markov network codebase and for help
454 adapting it to this work. We wish to acknowledge the support of the Michigan State University High
455 Performance Computing Center and the Institute for Cyber-Enabled Research (iCER). This material is
456 based in part upon work supported by the National Science Foundation under Cooperative Agreement No.
457 DBI-0939454. Any opinions, findings, and conclusions or recommendations expressed in this material
458 are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

459 REFERENCES

- 460 Adami, C. (1998). *Introduction to Artificial Life*. Telos, Springer Verlag.
- 461 Alcalá-Fdez, J., Sánchez, L., García, S., del Jesus, M. J., Ventura, S., Garrell, J., Otero, J., Romero, C.,
462 Bacardit, J., Rivas, V. M., et al. (2009). KEEL: A software tool to assess evolutionary algorithms for
463 data mining problems. *Soft Computing*, 13(3):307–318.
- 464 Back, T., Fogel, D. B., and Michalewicz, Z. (1997). *Handbook of Evolutionary Computation*. IOP
465 Publishing Ltd.
- 466 Baker, D. and Sali, A. (2001). Protein structure prediction and structural genomics. *Science*, 294(5540):93–
467 96.
- 468 Baluja, S. and Simon, D. (1998). Evolution-based methods for selecting point data for object localization:
469 Applications to computer-assisted surgery. *Applied Intelligence*, 8(1):7–19.
- 470 Blickle, T. and Thiele, L. (1996). A comparison of selection schemes used in evolutionary algorithms.
471 *Evolutionary Computation*, 4(4):361–394.
- 472 Bolten, E., Schliep, A., Schneckener, S., Schomburg, D., and Schrader, R. (2001). Clustering protein
473 sequences—structure prediction by transitive homology. *Bioinformatics*, 17(10):935–941.
- 474 Chapman, S. D., Knoester, D. B., Hintze, A., and Adami, C. (2013). Evolution of an artificial visual cortex
475 for image recognition. In *Advances in Artificial Life (ECAL 2013)*, pages 1067–1074. MIT Press.
- 476 Cheng, J. and Baldi, P. (2007). Improved residue contact prediction using support vector machines and a
477 large feature set. *BMC bioinformatics*, 8(1):1.
- 478 Cheng, J., Randall, A. Z., Sweredoski, M. J., and Baldi, P. (2005). Scratch: A protein structure and
479 structural feature prediction server. *Nucleic acids research*, 33(suppl 2):W72–W76.

- 480 Ciresan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image
481 classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages
482 3642–3649. IEEE.
- 483 Corbett, K. D. and Berger, J. M. (2004). Structure, molecular mechanisms, and evolutionary relationships
484 in dna topoisomerases. *Annu. Rev. Biophys. Biomol. Struct.*, 33:95–118.
- 485 Cozzetto, D., Kryshchak, A., Fidelis, K., Moulton, J., Rost, B., and Tramontano, A. (2009). Evaluation
486 of template-based models in CASP8 with standard measures. *Proteins: Structure, Function, and
487 Bioinformatics*, 77(S9):18–28.
- 488 Cramer, N. L. (1985). A representation for the adaptive generation of simple sequential programs. In
489 *Proceedings of the First International Conference on Genetic Algorithms*, pages 183–187.
- 490 Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic
491 algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197.
- 492 Ding, W., Xie, J., Dai, D., Zhang, H., Xie, H., and Zhang, W. (2013). CNNcon: Improved protein contact
493 maps prediction using cascaded neural networks. *PLoS one*, 8(4):e61533.
- 494 Drenth, J. (2007). *Principles of protein X-ray crystallography*. Springer Science & Business Media.
- 495 Edlund, J. A., Chaumont, N., Hintze, A., Koch, C., Tononi, G., and Adami, C. (2011). Integrated
496 information increases with fitness in the evolution of animats. *PLoS Comput Biol*, 7(10):e1002236.
- 497 Fogel, D. B. (2000). *Evolutionary computation: Principles and practice for signal processing*, volume 43.
498 SPIE Press.
- 499 Gaulton, A., Bellis, L. J., Bento, A. P., Chambers, J., Davies, M., Hersey, A., Light, Y., McGlinchey, S.,
500 Michalovich, D., Al-Lazikani, B., et al. (2012). ChEMBL: a large-scale bioactivity database for drug
501 discovery. *Nucleic acids research*, 40(D1):D1100–D1107.
- 502 Ho, A., Schwarze, S. R., Mermelstein, S. J., Waksman, G., and Dowdy, S. F. (2001). Synthetic protein
503 transduction domains: enhanced transduction potential in vitro and in vivo. *Cancer research*, 61(2):474–
504 477.
- 505 Jones, D. T., Buchan, D. W., Cozzetto, D., and Pontil, M. (2012). PSICOV: precise structural con-
506 tact prediction using sparse inverse covariance estimation on large multiple sequence alignments.
507 *Bioinformatics*, 28(2):184–190.
- 508 Koch, M. A. and Waldmann, H. (2005). Protein structure similarity clustering and natural product
509 structure as guiding principles in drug discovery. *Drug discovery today*, 10(7):471–483.
- 510 Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. MIT
511 press.
- 512 Kouranov, A., Xie, L., de la Cruz, J., Chen, L., Westbrook, J., Bourne, P. E., and Berman, H. M.
513 (2006). The rcsb pdb information portal for structural genomics. *Nucleic acids research*, 34(suppl
514 1):D302–D305.
- 515 Laskowski, R. A., Moss, D. S., and Thornton, J. M. (1993). Main-chain bond lengths and bond angles in
516 protein structures. *Journal of molecular biology*, 231(4):1049–1067.
- 517 Lena, P. D., Nagata, K., and Baldi, P. F. (2012). Deep spatio-temporal architectures and learning for
518 protein structure prediction. In *Advances in Neural Information Processing Systems*, pages 512–520.
- 519 Liaw, A. and Wiener, M. (2002). Classification and regression by random Forest. *R news*, 2(3):18–22.
- 520 Marstaller, L., Hintze, A., and Adami, C. (2013). The evolution of representation in simple cognitive
521 networks. *Neural Computation*, 25:2079–2107.
- 522 Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of t4 phage
523 lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.
- 524 McNaught, A. D. and Wilkinson, A. (1997). *Compendium of chemical terminology*, volume 1669.
525 Blackwell Science Oxford.
- 526 Ofria, C. and Wilke, C. O. (2004). Avida: A software platform for research in computational evolutionary
527 biology. *Artificial life*, 10(2):191–229.
- 528 Pedersen, J. T. and Moulton, J. (1996). Genetic algorithms for protein structure prediction. *Current Opinion
529 in Structural Biology*, 6(2):227–231.
- 530 Pruitt, K. D., Tatusova, T., and Maglott, D. R. (2007). Ncbi reference sequences (refseq): a curated non-
531 redundant sequence database of genomes, transcripts and proteins. *Nucleic acids research*, 35(suppl
532 1):D61–D65.
- 533 Rainforth, T. and Wood, F. (2015). Canonical correlation forests. *arXiv preprint arXiv:1507.05444*.
- 534 Ray, T. S. and Hart, J. (1999). Evolution of differentiated multi-threaded digital organisms. In *Intelligent*

- 535 *Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ International Conference on,*
536 *volume 1, pages 1–10. IEEE.*
- 537 Rost, B. and Sander, C. (1994). Combining evolutionary information and neural networks to predict
538 protein secondary structure. *Proteins: Structure, Function, and Bioinformatics*, 19(1):55–72.
- 539 Sevier, C. S. and Kaiser, C. A. (2002). Formation and transfer of disulphide bonds in living cells. *Nature*
540 *reviews Molecular cell biology*, 3(11):836–847.
- 541 Simons, K. T., Bonneau, R., Ruczinski, I., and Baker, D. (1999). Ab initio protein structure prediction of
542 casp iii targets using rosetta. *Proteins: Structure, Function, and Bioinformatics*, 37(S3):171–176.
- 543 Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical*
544 *Society. Series B (Methodological)*, pages 267–288.
- 545 Wang, Z. and Xu, J. (2013). Predicting protein contact map using evolutionary and physical constraints
546 by integer programming. *Bioinformatics*, 29(13):i266–i273.
- 547 Wuthrich, K. (1986). *NMR of proteins and nucleic acids*. Wiley.
- 548 Xu, D. and Zhang, Y. (2012). Ab initio protein structure assembly using continuous structure fragments
549 and optimized knowledge-based force field. *Proteins: Structure, Function, and Bioinformatics*,
550 80(7):1715–1735.
- 551 Yoshikawa, H. and Ogasawara, N. (1991). Structure and function of dnaa and the dnaa-box in eubacteria:
552 evolutionary relationships of bacterial replication origins. *Molecular microbiology*, 5(11):2589–2597.
- 553 Zhang, Y. (2007). Template-based modeling and free modeling by I-TASSER in CASP7. *Proteins:*
554 *Structure, Function, and Bioinformatics*, 69(S8):108–117.