An important advantage of Korea in bioinformatics is its mature IT industry. The bioinformatics onslaught in Korea is led by the chaebols such as LG, SK, and Samsung. These companies have established new bioinformatics teams. The bioinformatics industry also finds support from pharmaceutical companies, such as Chongkeundang and Green Cross. Some of the start-up bioinformatics companies in Korea are Bioinfomatix, Inc., Pax GENETICA, Inc, Macrogen, IDRTech, Badasoft Co., SmallSoft Co., BITEK CHEMS, Inc., lStech Co., IDGENE, Inc, RNA, Inc., Genoprot Co., and Proteogen.

Some of the public institutions involved in bioinformatics research include Korean Society of Bioinformatics (KSBI), Biological Research Information Center (BRIC), National Institutes of Health (NIH), Korean Institute of Science and Technology Information (KISTI), Korea Research Institute of Bioscience and Biotechnology (KRIBB) and the 21c Frontier the Center for Functional Analysis of Human Genome.

4 Conclusion

Asia Pacific definitely enjoys a distinct advantage in bioinformatics. The strength of its IT sector along with the presence of a large base of man power skilled in IT would enable a number of countries in the region to grab opportunities in bioinformatics. This large pool of technically-skilled personnel can be used to meet the demand for bioinformatics in other regions as well. Growth in the volume of genomic data along with technological advances and increased automation of the R&D process present bright growth prospects for bioinformatics worldwide and Asia Pacific is well placed to harness the same.

If there are a couple of constraints, it would be Asia Pacific’s ability to hard and soft sell its skill sets to get a chunk of the global pie as well as the low expenditure on biopharma R&D locally. Only time can reveal whether countries in the region have the resourcefulness needed to play a winning game in bioinformatics.

Abstract

This paper considers the problem of inferring an original sequence from a number of erroneous copies. The problem arises in DNA sequencing, particularly in the context of emerging technologies that provide high throughput or other advantages at the cost of an increased number of errors. We describe and compare two approaches that have recently been developed by the authors. The first approach searches for a sequence known as a Steiner string; the second searches for the most probable original sequence with respect to a simple Bayesian model of sequencing errors. We present the results of extensive tests in which erroneous copies of real DNA sequences were simulated and the algorithms were used to infer the original sequences. The results are used to compare the two approaches to each other and to a third, more conventional, approach based on multiple sequence alignment. We find that the Bayesian approach is superior to the Steiner approach, which in turn is superior to the alignment approach.
The two new algorithms can also be used to construct multiple sequence alignments. We show that the two methods produce alignments of approximately equal quality, and conclude that the Steiner approach is better for this purpose because it is faster. Both methods produce better alignments than a well-known multiple sequence alignment package, for the cases tested.

Keywords: Consensus sequences; multiple sequence alignment; sequencing errors.

1 Introduction

The need to infer an original sequence from a number of erroneous copies is common in DNA sequencing. All DNA sequencing technologies are imperfect and therefore one cannot have high confidence in a sequence obtained from a single read. By generating multiple overlapping reads and comparing them, errors can be detected and a closer approximation to the actual sequence can be obtained. This issue is particularly important for certain emerging technologies that promise rapid sequencing at the cost of an increased number of errors. Kececioglu et al. (1997) cite single-molecule DNA sequencing as an example of such a technology. In another example, the authors are currently developing a technology in which as many as 20–30 percent of the bases in any given read may be erroneous. If such technologies gain currency, the ability to accurately and efficiently infer a sequence from erroneous copies may become increasingly important. Indeed, that ability may be crucial in rendering such technologies feasible.

By far the most common approach to this problem, used in all major sequencing projects, is to form a multiple sequence alignment of the erroneous reads and to determine a (possibly weighted) consensus character for each column of the alignment. This approach is attributable to no single author or group, but Gusfield (1997) provides an interesting discussion of it. Kececioglu et al. (1997) describe an elegant approach based on aligning sequences in groups of three. In this present paper, two recent approaches developed by the authors are described and compared. The first approach (Keith et al. 2002) uses a simulated annealing algorithm to search for a sequence known as a Steiner string. The second approach (Keith et al. 2003) also uses a simulated annealing search algorithm, but differs in that it searches for the most probable sequence under a Bayesian probabilistic model of the processes by which errors are introduced.

The paper is structured as follows. The Steiner approach is described in Section 2 and the Bayesian approach in Section 3. In Section 4, we present and discuss results of extensive tests in which erroneous copies of an original sequence were simulated and the original sequence was inferred. We also present results of similar tests in which consensus sequences were obtained from multiple sequence alignments produced by the multiple sequence alignment package, CLUSTAL W. Finally, in Section 5 we present the results of simulations used to evaluate multiple sequence alignments generated by the two methods and compare them to alignments obtained using CLUSTAL W.

2 The Steiner Approach

A consensus sequence for a family of related biological sequences is, loosely speaking, a sequence that embodies the characteristics shared by all or most members of the family. It is thus a different concept from an inferred original sequence. However, if the errors are randomly distributed without strong location-specific biases then the consensus sequence for a number of erroneous copies may be considered to be an approximation to the original sequence.

The first algorithm we discuss in this paper searches for an object known as a Steiner string. To formalise this concept, we require the following notation and definitions. Let W be the set of all finite sequences that can be formed from characters of the alphabet S = {A,C,G,T}. Given a set S = {Y₁, Y₂, ..., Yₗ} of sequences in W, an arbitrary sequence X, and a metric D defined on W, the consensus error of X relative to S is:

\[ E(X) = \frac{1}{q} \sum_{k=1}^{q} D(Y_k, X) \]

A sequence s* is a Steiner string for S if E(s*) ∈ E(s) for all s ∈ W. These definitions resemble those given by Gusfield (1997). In this paper, the metric D is the edit distance, defined as the minimum number of insertions, deletions and substitutions required to transform the first sequence into the second, or vice versa. A Steiner string is, in some sense, the centre of the set S, in that it minimises the sum of distances to the elements of S. It should not, however, be confused with the center string of S, which is the element of S that minimizes the sum of distances to the other elements of S.

We use a simulated annealing algorithm to search for a Steiner string for the input sequences. A detailed description of the search technique is given by Keith et al. (2002). The time required for a single iteration of the algorithm is \( O(L_{\text{max}} \sum_{k=0}^{q-1} n_k) \), where \( L_{\text{max}} \) is the length of the longest sequence generated in that iteration and \( n_k \) is the length of \( Y_k \). The memory requirement for a single iteration is \( O(L_{\text{init}} \sum_{k=1}^{q} n_k) \), where \( L_{\text{init}} \) is the length of the initial sequence for that iteration.
The Bayesian Approach

The Bayesian approach to this problem involves searching for the most probable original sequence with respect to a simple Bayesian model of sequencing errors. In this paper, it is not our intention to accurately represent the actual mechanisms by which sequencing errors are introduced. Rather, we use an idealized model, which should nevertheless provide a good approximation to these mechanisms. The result is a powerful, practical method for inferring an original sequence.

Most DNA sequencing techniques read molecules sequentially. Consequently, once a part of the molecule has been read, no further errors can be introduced to that part of the sequence. This observation suggests the following model, in which insertions, deletions and substitutions are introduced into the sequence in order from left to right. Let the original sequence be \( X \) and suppose that it is formed from characters of a finite alphabet \( S \). Let the left-most character of \( X \) be temporarily designated the current character. Now, consider inserting a character immediately to the left of the current character. Let the probability of inserting character \( x \in S \) be \( r(-,x) \) and the probability of not inserting any character be \( r(-,-) \). Note that \( r(-,-) + \sum_{x \in S} r(-,x) = 1 \). If a character was inserted, consider a second insertion immediately to the right of the inserted character, that is, immediately to the left of the current character. The probability of inserting a character \( x \), or of not inserting any character, is here assumed to be the same for the second (and subsequent) insertions as for the first insertion, namely \( r(-,x) \) and \( r(-,-) \). Characters are inserted in this manner until a decision is made not to insert a character.

Next, choose whether to substitute the current character with a different character, delete the current character, or leave the current character unaltered. Let the value of the current character be \( x \in S \) and let the probability of substituting this character with character \( y \in S \) be \( r(x,y) \). Let the probability of deleting character \( x \) be \( r(x,-) \) and let the probability of leaving the character unaltered be \( r(x,x) \). Note that \( r(x,-) + \sum_{y \in S} r(x,y) = 1 \). Having made this choice, let the next character in the sequence be designated the current character and consider making insertions, deletions and substitutions as before. Continue this process until the last character is reached. Finally, consider insertions at the end of the sequence until a decision is made not to insert a character.

In summary, the process consists of the following steps. For ease of description, we suppose that a termination character has been appended at the right end of \( X \).

1. Set \( i := 1 \).
2. Consider an insertion immediately to the left of character \( i \) of \( X \).
3. If an insertion was made at Step 2,
   a) Set \( i := i+1 \).
   b) Go to Step 2.
4. If character \( i \) is not the termination character,
   a) Consider deleting or substituting character \( i \) of \( X \).
   b) If a deletion was not made at Step 4(a), set \( i := i+1 \).
   c) Go to Step 2.

Now, we want to compute the probability that the sequence generated by applying this process to \( X \) is a given sequence \( Y \). To do this, we must consider all possible alignments of \( Y \) to \( X \). Each alignment may be interpreted as showing a possible way in which \( Y \) could have been obtained from \( X \). In a given alignment, each character in \( Y \) that is aligned to a character in \( X \) is interpreted as having been substituted for that character if the characters are different, or as having been preserved from \( X \) if the characters are the same. Each character of \( X \) that is aligned to a space in \( Y \) is interpreted as having been deleted, and each character of \( Y \) that is aligned to a space in \( X \) is interpreted as having been inserted. A probability can thus be assigned to each alignment, specifically, the probability that the errors implied by the alignment would occur under the model. The probability \( p(Y \mid X) \) that a sequence \( Y \) is obtained by modifying a sequence \( X \) is thus the sum of these probabilities over all alignments. Letting \( M(X,Y) \) denote the set of all such alignments, this probability may be written as follows:

\[
p(Y \mid X) = \sum_{M \in M(X,Y)} r(M) \prod_{k=1}^{\text{len}(M)} r(s_k, t_k).
\]

Here \( m \) is the length of sequence \( X \), \( \text{len}(M) \) is the length of the alignment \( M \) (number of columns in the alignment), \( s_k \) is the \( k \)th token in the row of the alignment corresponding to \( X \) (a token being a character or a space), and \( t_k \) is the \( k \)th token in the row of the alignment corresponding to \( Y \). Note that the term \( r(-,-)^{m+1} \) results from the fact that the decision
not to insert another character must be made for each of the \( m+1 \) positions between adjacent characters of \( X \) and at the ends of \( X \). Summing over all possible alignments of two sequences sounds like a formidable computational task, but in fact it can be done efficiently using dynamic programming (see Keith et al. 2003 for details).

Suppose that we are now given \( q \) independently generated modified copies \( Y_1, Y_2, \ldots, Y_q \) of an unknown original sequence. Using Bayes’ rule, the posterior probability that the original sequence was \( X \) is given by:

\[
p(X | Y_1, Y_2, \ldots, Y_q) = \frac{p(X) \cdot p(Y_1, Y_2, \ldots, Y_q | X)}{p(Y_1, Y_2, \ldots, Y_q)}
\]

where \( p(X) \) is the prior probability that the original sequence was \( X \). The prior probability encapsulates information about \( X \) that was available prior to observing the erroneous sequence reads, such as information about the length and composition of \( X \). Here we assume that all sequences of equal length are equally probable prior to observing the reads. Since there are \( |S|^L \) reads of length \( L \), where \( |S| \) is the number of characters in the alphabet, the prior probability can be written as \( p(X) = f(L(X))/|S|^L \). Here \( L(X) \) is the length of \( X \) and \( f \) is a prior probability distribution with regard to the length of the original sequence. We further assume that all sequence lengths are equally likely prior to observing the reads, and hence that \( f \) is uniform.

We use a simulated annealing algorithm to search for a sequence \( X^* \) that is maximal with respect to the posterior probability \( p(X | Y_1, Y_2, \ldots, Y_q) \). One advantage of our search algorithm is that it does not require the above proportionality to be normalized. However, we do not regard this search technique as a defining property of the method, and there may be many alternative search techniques that could be used with equal success.

Although this approach seems entirely different to the Steiner approach, it can be implemented using a similar algorithm. In practice, there are only three practical differences between the algorithms: the inclusion of a prior probability term in the scoring function, the use of different scores for different types of error and the practice of summing over all alignments.

The advantage of a Bayesian approach is that it enables diverse kinds of information to be taken into account when drawing an inference. For example, prior information about the length and composition of the original sequence, and more importantly, information about the kinds and frequencies of errors that can occur, can be used to inform the inference. The result is a reconstructed sequence that is potentially more accurate, and also more meaningful in the sense that one may make probabilistic statements about what it represents. For example, one may state that the reconstructed sequence is, with some probability, the original sequence from which the erroneous copies were derived. One may also determine probabilities for other candidate sequences.

Using various computational techniques described elsewhere (Keith et al. 2003), the Bayesian approach can be implemented in time \( O(L_{\text{max}} \cdot q \cdot n_k) \), where \( L_{\text{max}} \) is the length of the longest sequence generated in that iteration. This is of the same order as the Steiner approach, although in practice the Bayesian approach is somewhat slower. The memory requirement for a single iteration is \( O(n_{\text{init}} \cdot q \cdot k \cdot n_k) \), where \( n_{\text{init}} \) is the length of the initial sequence for that iteration. It should be possible to reduce memory requirements to \( O(q \cdot k \cdot n_k) \) by taking advantage of the reversibility of certain calculations used in this approach. This technique is not available in the Steiner approach.

All DNA sequencing technologies are imperfect and therefore one cannot have high confidence in a sequence obtained from a single read.

4 Results: Inferred Original Sequences

To test the two approaches, the following simulations were performed. Fragments of known sequence were selected at random from a database of human DNA. A number of erroneous copies were then simulated for each original sequence, in accordance with the error model described in Section 3. This was done for various choices of model parameters, although all substitutions were assumed to be equally likely in all tests. That is, \( r(x,y) \) was assumed to be independent of \( x \) and \( y \) for all \( x, y \in S \) with \( x \neq y \). We attempted to reconstruct each original sequence by searching for a Steiner sequence as described in Section 2. The edit distance between each reconstructed sequence and the corresponding original sequence was then calculated, as a measure of the correctness of the reconstruction. Similar simulations were performed using the Bayesian approach described in section 3. We also attempted to reconstruct the original sequence by forming a multiple sequence alignment of the erroneous sequences using the well-known alignment program CLUSTAL W (Thompson et al. 1994) with the default parameters,
and then taking a consensus character for each column of the alignment.

Fig. 1 to 4 show how the average number of errors (that is, the average edit distance between the inferred and original sequences) varied with the number of erroneous sequences used, for various probabilities of insertion, deletion and substitution. The substitution probabilities given in the captions are the values of \( r(x, y) \) for any \( x, y \in S \). The length of the original sequence was 400 bases in all of these tests. Each data point in the figures represents an average over approximately 1000 simulations. Results are shown for the multiple sequence alignment approach (triangular marker), Steiner approach (diamond marker) and Bayesian approach (square marker).

Fig. 1. Average error when probabilities of insertion, deletion and substitution are 0.01, 0.01, 0.03 respectively.

Fig. 2. Average error when probabilities of insertion, deletion and substitution are 0.01, 0.01, 0.2 respectively.

Fig. 3. Average error when probabilities of insertion, deletion and substitution are 0.05, 0.05, 0.2 respectively.

Fig. 4. Average error when probabilities of insertion, deletion and substitution are 0.1, 0.1, 0.1 respectively.

Fig. 5 shows how the average number of errors varies with sequence length for the three methods, using five reads and with probabilities of insertion, deletion and substitution being 0.01, 0.01, and 0.2 respectively. The three approaches correspond to the same markers as in Fig. 1 to 4. Note that the curves shown in all of these figures serve mainly to link data points obtained using the same method; they provide only a rough approximation to the results that would be obtained between data points.

This paper considers the problem of inferring an original sequence from a number of erroneous copies. We describe and compare two approaches that have recently been developed by the authors. The first approach known as a Steiner string; the second a simple Bayesian model of sequencing errors.
In Fig. 1 to 4, observe that the number of errors decreases exponentially with the number of sequences when sequences are inferred using either of the two new algorithms. This is a desirable behaviour, as it means that a highly accurate sequence can be inferred from a surprisingly small number of highly inaccurate sequences. In most cases the sequences inferred using CLUSTAL W resulted in a curve of similar shape, but the limiting number of errors is non-zero. In Fig. 1, the average number of errors actually begins to increase as the number of reads increases. These are highly undesirable behaviours, and we do not know how to account for them. We suspect, however, that the problem is with the sequential alignment approach on which CLUSTAL W is based.

In Fig. 5, observe that the number of errors increases approximately linearly with the length of the original sequence, for all approaches. Consequently, the proportion of errors in a reconstructed sequence is independent of length. This property should facilitate estimating the number of reads required to achieve the desired accuracy.

The Bayesian approach produces substantially better results than the Steiner approach, which in turn produces better results than the alignment approach. The difference in performance between the two new approaches must be due to either the use of prior information, the use of known probabilities for various types of error, or the practice of summing over alignments, as these are the only practical differences between the algorithms. This analysis does not reveal the relative contribution of these differences to the improvement. However, we suspect that the appropriate use of known error probabilities accounts for most of the improvement. A compromise between the two methods is possible, taking the error probabilities into account when determining the ‘distances’ between the consensus and the erroneous copies, but not using prior probabilities or summing over alignments. Similarly, the results obtained using CLUSTAL W could be improved by taking the error probabilities into account when defining the scoring matrix. We suspect, however, that summing over alignments makes a significant difference for sequences where many alternative alignments are possible, such as sequences containing micro-satellite repeats.

When the proportion of errors in the uncorrected sequences is already quite low, as it is for the data presented in Fig. 1, the number of errors obtained using any of the approaches is small. Based on these results, it would not be appropriate to claim that there is an urgent need for major sequencing projects to adopt a Bayesian approach to inferring an original sequence. We do, however, make the following points. Firstly, there seems to be a limit to the accuracy that can be achieved by inferring the original sequence from a multiple sequence alignment. At least, this appears to be the case when the alignment is performed using CLUSTAL W. That this should be so for such a widely used alignment package is concerning. It would be interesting to determine whether the same is true of the consensus sequences produced in major sequencing projects. Secondly, the Bayesian approach has the flexibility to incorporate application-specific information about the kinds and types of errors that may occur, such as specific sequence patterns that are known to cause errors, or more generally how error probabilities are affected by local sequence characteristics. Detailed models of this kind could enable automated sequence editing of a quality that is currently only possible for an informed human editor.

The advantage conferred by the Bayesian approach is much more significant when the proportion of errors is high, as it is for the data displayed in Figs. 2 to 4. The ability to infer high-quality sequence from a small number of inaccurate reads could make the difference between a competitive and a non-competitive sequencing technology. An important conclusion that may be drawn from this study is that error-prone sequencing technologies may in fact be feasible if they possess compensating advantages such as high throughput or low cost.

5 Results: Multiple Sequence Alignments

An inferred original sequence can induce a multiple sequence alignment of the input sequences, using a version of the well-known algorithm of Feng and Doolittle (1987). The idea is to perform pair-wise
alignments between the inferred sequence and each input sequence, and use these to construct a multiple sequence alignment. This can be done even if the input sequences are not erroneous copies of an original sequence, in which case the reconstruction can be interpreted as a common ancestor or merely as a convenient reference sequence. This technique is called consensus alignment (Altshul and Lipman 1989, Gusfield 1997).

We used the two new approaches to construct multiple sequence alignments of the simulated data described in Section 4, and compared these to alignments obtained using CLUSTAL W. We also compared them to alignments induced by the original sequences used in each simulation. We refer to this latter approach as the ideal approach because it implies knowledge of the original sequence and because it should, in theory, provide the best possible consensus alignment.

Pair-wise alignments to a sequence inferred using the Bayesian approach were constructed to minimise the probability of the implied errors, and so too were pair-wise alignments to an original sequence. However, pair-wise alignments to a sequence inferred using the Steiner approach were constructed to minimise the number of differences between rows of the alignment. The latter approach can thus be used when error probabilities are not available. To score a multiple sequence alignment, we calculated the average number of differences between any two rows of the alignment. This is a rough, but objective, measure of alignment quality.

The results for various error probabilities are shown in Figs. 6 to 9. The top curve in each figure corresponds to results obtained using CLUSTAL W. Where the other three curves can be distinguished from each other they correspond to the ideal, Bayesian and Steiner approaches, in order from top to bottom.

The Steiner Approach analyzes consensus sequence for a family of related biological sequences is, loosely speaking, a sequence that embodies the characteristics shared by all or most members of the family.

![Fig. 6. Alignment scores with probabilities of insertion, deletion and substitution 0.01, 0.01, 0.03 respectively.](image)

![Fig. 7. Alignment scores with probabilities of insertion, deletion and substitution 0.01, 0.01, 0.2 respectively.](image)

![Fig. 8. Alignment scores with probabilities of insertion, deletion and substitution 0.05, 0.05, 0.2 respectively.](image)
Fig. 9. Alignment scores with probabilities of insertion, deletion and substitution 0.1, 0.1, 0.1 respectively.

The most obvious feature of these graphs is the qualitatively different behaviour of CLUSTAL W in comparison to the other algorithms. The quality of the CLUSTAL W alignments deteriorates as additional sequences are added, whereas the quality of the induced alignments rapidly converges to a constant value. In fairness, it should be noted that the new algorithms were designed for data of the kind used here, whereas CLUSTAL W was not. Nevertheless, the observed behaviour of CLUSTAL W is disconcerting.

The results obtained using the two new approaches are almost indistinguishable from the results obtained using the ideal approach. This suggests that the consensus approach to alignment is not sensitive to small variations in the reference sequence. Moreover, since the probability of different types of error is not used in the Steiner approach, it seems that such information has little impact on the quality of the induced alignment. The Steiner approach is significantly faster than the Bayesian approach, so it may be the better approach for alignment purposes unless extremely high accuracy is called for. Incidentally, these results suggest that the map taking a reference sequence to the alignment it induces may be many-to-one. This is an additional argument against inferring the original sequence from a multiple sequence alignment.

Where the lower three curves can be distinguished, the Steiner approach gives the lowest alignment scores, followed by the Bayesian approach and then the ideal approach. This may seem odd if the ideal approach truly is ideal, and given that the Bayesian approach uses more information than the Steiner approach. A likely explanation is that the new approaches are responding to accidental patterns in the erroneous copies, and are thus inferring a sequence that implies fewer errors than actually occurred. This effect becomes less apparent as more sequences are used or when the error probabilities are used, as in the Bayesian approach. The extra information helps to distinguish between true and accidental patterns.

The difference in performance between the two new approaches must be due to either the use of prior information, the use of known probabilities for various types of error, or the practice of summing over alignments, as these are the only practical differences between the algorithms.

References


Managing Editors
Limsoon Wong
Laboratories for Information Technology
21 Heng Mui Keng Terrace
Singapore 119613
Tel.: 65-874-8406
Fax: 65-776-8109
E-mail: limsoon@lit.a-star.edu.sg
Ming Li
Department of Computer Science
University of California
Santa Barbara, CA 93106
USA
Tel.: 1-805-893-8759
Fax: 1-805-893-8553
E-mail: mli@cs.ucsb.edu
Dr. John Wooley
Mail Code #0043
University of California San Diego
University Center Building 402
9500 Gilman Drive
La Jolla CA 92093-0043
USA
Tel.: 858 822 3630
Fax: 858 822 4767
Cell phone: 858 353 1971
E-mail: jwooley@ucsd.edu

Editorial Board
Vladimir Bajic (Laboratories for Information Technology, Singapore)
David Galas (Keck Graduate Inst, California, US)
Yike Guo (Imperial College of Science Technology & Medicine, UK)
Tao Jiang (University of California, Riverside, US)
Charles Lawrence (Wadsworth Center, New York, US)
Sang Yup Lee (Korea Advanced Institute of Science & Technology, Korea)
Satoru Miyano (University of Tokyo, Japan)
Gene Myers (Celera Genomics, US)
Pavel Pevzner (University of California, San Diego, US)
Manfred Sippl (University of Salzburg, Austria)
Masaru Tomita (Keio University, Japan)
Ying Xu (Oak Ridge National Labs, US)

Aims & Scope
The Journal of Bioinformatics and Computational Biology aims to publish high quality, original research articles, expository tutorial papers and review papers as well as short, critical comments on technical issues associated with the analysis of cellular information and the use of such information in biomedicine.

The research papers will be technical presentations of new assertions, discoveries and tools, intended for a narrower specialist community. The tutorials, reviews and critical commentary will be targeted at a broader readership of biologists who are interested in using computers but are not knowledgeable about scientific computing, and equally, computer scientists who have an interest in biology but are not familiar with current thrusts nor the language of biology. Such carefully chosen tutorials and articles should greatly accelerate the rate of entry of these new creative scientists into the field.

For information on Guidelines for Contributors, please visit
http://www.worldscinet.com/jbcb/jbcb.shtml

Published by Imperial College Press and distributed by World Scientific Publishing Co.
The Protein Surface Properties Calculator

Dan V. Nicolau Jr.*, Florin Fulga † and Dan V. Nicolau Sr. ‡

*Department of Mathematics and Statistics, Faculty of Science University of Melbourne, Parkville 3010, Victoria, Australia.

sarmisegetusa@yahoo.com

‡Industrial Research Institute
Swinburne University of Technology, Hawthorn 3122, Victoria, Australia.

dnicolau@swin.edu.au, ifulga@swin.edu.au

Abstract

The interactions of large molecules with surfaces and with each other are strongly dependent upon their surface, rather than their bulk properties. In addition, the local properties of biomolecular surfaces are very important in their own right in biomedicine and other areas, for example for locating binding sites.

Following to previous work, we have developed a program to compute to compute amino acid and atom-based surface descriptors, and used it to generate a small database of charge and hydrophobicity-related surface properties for a set of proteins. The program requires the user to input two text files: one assigning a real number to each atom of each amino acid, and one assigning a real number to each amino acid. Although we have so far only computed surface charge (atom-based) and surface hydrophobicity (amino acid-based), we note that this program could be used to compute any surface parameter whatsoever, since the user can assign arbitrary atom-by-atom and amino acid properties. We discuss possible applications of this program and describe one current application, the Biomolecular Adsorption Database.

Keywords: Molecular surface.

1 Introduction

The geometric properties of molecular surfaces have received constant attention during the last 20 years due to their importance to inter-molecular, and in particular, biochemical interactions. These properties include solvent-accessible surface area and surface roughness. In addition, a few other parameters, e.g. electrostatic surface potential can also currently be computed, for example using the program GRASP.

Information about the solvent-accessible surface can be used, for example, to make inferences about the interactions of molecules with each other and with surfaces, or to help in locating binding sites (Pettit & Bowie 1999) and so on. However, despite their importance, non-geometric biomolecular surface properties have received very little attention; and the studies that do exist have remained for the most part qualitative.

On the other hand, breakthroughs in this area could shed light on other important problems. As a prime example, the process of protein adsorption has resisted all attempts at predictive modeling despite considerable attention over the last two decades or so. One important reason for this may be that molecular detail is usually ignored or over-simplified in phenomenological models of this phenomenon. Since most proteins are at the smaller end of the colloid spectrum, the bulk of the interaction between a protein and a surface can be attributed to the interaction of the protein surface with the interface (and hence with the solid’s molecular surface). For example, both electrostatic and hydrophobic interactions occur largely between the two surfaces, and hence the systematic and quantitative investigation of the magnitude and distribution of such surface parameters should prove fruitful; and this should be the case wherever the interaction of biomolecules with each other and with other substances is important.
To this end, we have developed a set of algorithms to compute arbitrary local properties of molecular surfaces. The corresponding global parameters can be obtained by approximating the integral of the local parameter over the molecular surface. We divide the computational foundations of these algorithms into two categories. The first is “atom-based”, where we assume that we can assign constant and homogenously distributed properties to individual atoms of the same element; the second assumes that properties can be assigned to individual amino acids rather than atoms in the same way.

Fig. 1. Illustration of Connolly’s algorithm (a) the probe sphere is “rolled”, i.e. placed tangentially to all pairs of atoms (b) the solvent-accessible surface is the resulting locus of the probe sphere’s contact point with the atomic spheres.

2 Methods

We have already briefly described some of the ideas behind these algorithms with respect to the computation of atom-by-atom charge and amino acid-based hydrophobicity in recent work (Nicolau and Nicolau 2002). In this section, we present general algorithms for the computation of arbitrary surface properties of molecules and biomolecules. These form the computational basis for our program.

2.1 Overview of Connolly’s algorithm

Connolly’s algorithm has been described in great detail elsewhere, for example in Connolly (1983) and we will only briefly describe it here. The rationale behind the computation of the “accessible” surface of a molecule is that of the part of the molecule that is exposed to the solvent, only those features that are at least the same size as a solvent molecule actually have interactions with the solution: smaller features are not “accessible” to the solvent.

Based on this idea, Connoly’s algorithm rolls an imaginary sphere of a radius close to the effective size of a solvent molecule (in the case of water, this is 1.4 Å) over the 3D structure of the molecule, which consists of a cloud of spheres (the constituent atoms) of appropriate radii (usually the van der Waals radius of each atom is used). The “pivot points” of the ball are recorded as it is rolled over the molecule, and together, these constitute an approximation to the solvent-accessible surface.

From this set of points, one can compute the surface area of the molecule, surface curvature and other features. The “ball-rolling” procedure is illustrated in Fig. 1.

Clearly, the surface obtained in this way depends strongly on the size of the probe sphere. Therefore, all the properties of this surface also depend on the dimensions of the probe. This is illustrated in Fig. 2, where using different probe sphere radii results in quite different electrostatic potential surfaces for the same protein, crambin. This dependence is natural and reflects the philosophy of the algorithm: the interaction of the molecule with the solvent depends on the solvent itself, which is here modelled as a sphere.

2.2 Computing atom-based surface properties

Here we concern ourselves with computing an arbitrary surface property $P_S$ (such as surface charge or surface hydrophobicity), constant values of which can be assigned, under reasonable assumptions, to each atom. Moreover, we assume that the property is concentrated entirely at the surface of the atomic sphere and distributed evenly over that surface.

Then let the solvent-accessible surface of a molecule be $S(r)$ (where $r$ is the radius of the probe sphere used). Clearly $S$ will be a three-dimensional

Fig. 2. The solvent-accessible surface of a molecule depends on the radius of the probe sphere (a) crambin probed with a sphere of radius 1 Å, (b) crambin probed with a sphere of radius 5 Å.
vector, intuitively, an “empty shell”. Consider a function \( p: S \rightarrow \mathbb{R} \), which assigns to each point \((x_S, y_S, z_S)\) of the surface \(S\) a real number \( p(x_S, y_S, z_S) \). Then we will compute the global surface property \( P_S \) as
\[
P_S = \sum_{S_S p(x_S, y_S, z_S) \, dS}
\tag{1}
\]
where the integral will be taken over the surface \(S\).

In practice, we will approximate \( P_S \) by assigning to each point generated by our algorithm a small area element, and summing the contributions of all the elements. In order to do this, we assign to each atom \( i \) a real number \( p_i \), and scale it by the surface area of the atom in question. We have done this by computing the van der Waals area of the atom using \( A_i = 4\pi r_{vW}^2 \), where \( r_{vW} \) is the van der Waals radius of each atom. We obtained these radii from the paper by Gavezotti (1983).

The resulting approximation can be stated as
\[
p(x_S, y_S, z_S) \approx p_i / A_i \tag{2}
\]
where the relation holds only if the point \((x_S, y_S, z_S)\) “belongs” to the surface of atom \( i \). More precisely, a point “belongs” to a certain atom’s surface if the distance between the point and that atom’s centre is the smallest such distance.

This leads us to the approximation
\[
P_S \approx \frac{P_i}{A_i} \int_{S} A(x_S, y_S, z_S) \, dS
\tag{3}
\]
where \( dA(x_S, y_S, z_S) \) is the area of each surface element and the dependence of the coordinates indicates that there is a small variation in these areas due to the local surface curvature.

Therefore, our algorithm for determining \( P_S \) can be stated as follows:

for each surface area element
determine to which atom the element belongs
look up the value of \( p_i / A_i \) for this atom in a table
record the value of \( p_i / A_i \) at this point in a file
increment \( P_S \) by this value of \( p_i / A_i \)

The property that we have computed in this way is surface charge. First, we calculated formal charges for each atom of each isolated amino acid using the program HyperChem from Hypercube\textsuperscript{TM}, disregarding the effect of connecting amino acids together. Then, we calculated the surface area of all atom types using their van der Waals radii. Finally, we divided the two values and stored them in a table.

Our program was then used to compute several surface charge descriptors such as surface charge at each point, total positive charge at the surface, total negative charge at the surface, average charge per square Angstrom at the surface, and standard deviation of surface charge at the surface. This work is described in Nicolau and Nicolau (2002). We have also developed a scheme to assign hydrophobicities to each atom in this way (to be published).

We note that any property which can be assigned atom-by-atom values under reasonable circumstances could be computed in this way, as long as the assumptions that it is concentrated at the surface of the atom and is homogenously distributed over that surface are also reasonable. In fact, it would even be possible to assign to each atom a complex (instead of real) or vectorial value, in which case the summation in (2) would have to be carried out two or more times, once for each independent component of the vector.

### 2.3 Computing amino acid-based surface properties

Some properties of proteins cannot be assigned to individual atoms, but can be assigned to the amino acids. For example, hydrophobicity scales usually attribute a hydrophobicity or hydrophilicity value to each amino acid (e.g. Kyte-Doolittle hydrophobicities). Therefore, we have added a variation of the above algorithm to our program, capable of computing an arbitrary amino acid-bases surface property of a molecule.

The algorithm is very similar to the atom-based one, but the approximation to \( p \) this time takes the form
\[
p(x_S, y_S, z_S) \approx p_{aa} / A_{aa} \tag{4}
\]
and holds true for all points \((x_S, y_S, z_S)\) that belong to the surface of the amino acid denoted by the subscript “aa”. In this case, a point “belongs” to the surface of an amino acid if it belongs (in the previous sense) to the surface of an atom that is part of that amino acid. \( A_{aa} \), the area of the amino acid in question, was computed by us as the solvent-accessible surface area of that amino acid using a probe radius of 1.4 Å. This was done for all the amino acids, and must be used to scale the assigned value \( p_{aa} \) for the same reasons as above. Therefore, (3) now becomes
\[
P_S \approx \frac{p_{aa}}{A_{aa}} \int_{S} A(x_S, y_S, z_S) \, dS
\tag{5}
\]

The modified algorithm is then

for each surface area element
determine to what amino acid it belongs
look up the value of \( p_{aa} / A_{aa} \) for this amino acid in a user-supplied table
record the value of \( p_{aa} / A_{aa} \) at this point in a file
increment \( P_S \) by this value
So far, the amino acid-based property feature has only been used by us to compute hydrophobicities, following the Kyte-Doolittle scale (a default file is supplied with the program). However, any property could again be assigned to the amino acids (for example, adsorption on a chromatographic column).

3 Description of program

We have given our program the acronym PSPC (Protein Surface Properties Calculator), although of course its use is not restricted to proteins or even to biological molecules. It was written in Visual Fortran and runs under Windows™ environments.

We briefly describe the main features of the program in terms of input and output characteristics and a few remarks on running time and space requirements. A screenshot of the program window is shown in Fig. 3.

3.1 Description of input

3.1.1 Probe Radius

This is the radius, in Angstroms, of the sphere used to probe the molecular surface. This parameter must be entered in the Probe Radius input text box at the top of the window. Note that a sphere representing a water molecule has a radius of about 1.4 Å, and this value is usually used to compute the solvent-accessible surface of molecules in aqueous solution.

3.1.2 Selecting Mode/Input files

Here the user can enter a file, or set of files to perform the calculations on. If Batch Mode is not checked, then one must select the file of interest. This file must be in either PDB/ENT (.pdb/.ent) format or in Alchemy (.hin) format. If Batch Mode is checked, then one must select a text file containing the names of the PDB files of interest (one name per line). For example, to perform calculations on the three files 1IGT.ENT, 1LYZ.ENT and 1MYO.ENT, the batch file would contain

1IGT
1LYZ
1MYO

3.1.3 Selecting a Property

Here the user can specify the files containing the atom-based and amino acid-based properties to be used in calculations. For atom-based properties, this can either be an atomic charge file (*.chg), an atomic hydrophobicities file (.hph), or in principle any file in which a real number is assigned to each atom of each amino acid, in the format:

Amino-Acid 3-letter code, Atom Code, Property/van der Waals area of atom

A file named aminochg.chg, included with the program, is the default atomic charge file and is an example of this format. It assigns to each atom of each amino acid, a value equal to the charge on that atom divided by the atom’s van der Waals surface area. Atom-based hydrophobicities follow an identical format, with each atom being assigned a value equal to its predicted hydrophobicity/hydrophilicity contribution divided its van der Waals area. A default file is supplied with the program in this case also. In general, the property assigned to the atom must always be divided by the atom’s surface area, since the program computes surface properties by summing the contributions of small area elements on the molecular surface, as described above.

The geometric properties of molecular surfaces have received constant attention during the last 20 years.

A file named aminochg.chg, included with the program, is the default atomic charge file and is an example of this format. It assigns to each atom of each amino acid, a value equal to the charge on that atom divided by the atom’s van der Waals surface area. Atom-based hydrophobicities follow an identical format, with each atom being assigned a value equal to its predicted hydrophobicity/hydrophilicity contribution divided its van der Waals area. A default file is supplied with the program in this case also. In general, the property assigned to the atom must always be divided by the atom’s surface area, since the program computes surface properties by summing the contributions of small area elements on the molecular surface, as described above.
In similar fashion, the user can select the amino acid-based property file, in which a real number is assigned to each amino acid instead of each atom. The format is:

- Amino-Acid 3-letter code, Property/Surface
- Accessible Area of Amino Acid
- Amino-Acid 3-letter code, Property/Surface
- Accessible Area of Amino Acid

... For the same reasons as above, the property must again be divided by the area of the amino acid (we have used the solvent-accessible areas of the amino acids, computed using a probe radius of 1.4 Å — this file will be supplied on request).

3.2 Description of Output

3.2.1 Output Data Files

The primary output is a text file, containing the numerical results of the run, including both atom-based and amino acid-based property results, together with some information about the input file. If the program is running in Batch Mode, then more than one primary output file will be generated.

3.2.2 Generating Graphics

Three sets of files will be generated if this feature is used. The first is a point-by-point table of the atom-based property; the second is a point-by-point table of the amino acid-based property. Finally, two MATLAB™ files will be generated, allowing the user to view the graphics generated. Running these in MATLAB™ will produce a 3D, two-color, rotatable surface showing red for a positive property and blue for a negative property. This can be used to qualitatively answer questions such as “is this molecular surface like a zebra or like a leopard, with respect to this property?”

Finally, a progress bar at the bottom of the window provides a rough indication of the current status of the computation. Running times vary greatly with input file size, molecular shape (globular molecules take less time than elongated ones) and probe radius. To a first approximation, the time and space complexities are quadratic in probe radius and linear in the number of atoms, as expected.

An average run time for a small molecule, such as crambin, using a small probe radius, such as 1.4 Å, is no more than 30 seconds on an average desktop machine running Windows NT, for example. On the same machine, running a computation on a large molecule such as IgG, using a large probe radius, say 20 Å, can take as long as 30 minutes, if graphics are also generated. In general, the program is quite demanding in terms of both CPU and memory usage, and takes a large share of the system resources, even on a fast machine.

Although initial space requirements are very small (the program itself, together with a few auxiliary files occupies less than 200 KB of hard disk space), the temporary file generated during the run (and deleted afterwards) can reach a few MB in size, and if graphics are generated, each of the six graphics files can also be as large as a few MB. Consequently, we recommend that it be run on a machine with at least 10–20 MB of free space, and if Batch Mode is used, then this should be scaled accordingly.

4 Discussion

We have purposefully built a large degree of flexibility into this software: the user can and is encouraged to not only supply custom atomic and amino acid property data files, but he or she can actually determine what property is to be computed in each case. The limits of this flexibility are only determined by the availability of per-atom or per-amino acid data related to the property of interest, and of course the validity of assigning numerical properties in the manner described here. In keeping with this spirit, the ventual applications of this software are also left to the technical interests and area of expertise (and the imagination) of the user.

This is one of several programs that can be used to investigate molecular surfaces; GRASP and STING are among the better-known of these — the former in particular is used extensively for this task. Several important differences exist between our program and these. GRASP, STING and similar programs are entire molecular visualisation and analysis packages, with functionality split more or less evenly between these very general tasks. The focus on visualisation and/or the broad scope of these packages mean that the molecular surface analysis functionality is fixed and limited. Even with GRASP, only a small number of properties can be calculated, and there is very little flexibility in computing these.

The program we present here is dedicated entirely to the analysis of molecular surfaces, and while it is
possible to produce images, the intended purpose of these is to aid the user in interpreting the data produced, rather than on visualization itself. On the other hand, every effort has been made to give the user as much latitude as possible in surface analysis. Not only is it possible to vary the probe radius and the input property files, but both “atom-based” and “amino acid-based” can be computed simultaneously. Another difference from general-purpose software is that SPPC returns a large number of parameters for each property computed. Possibly the greatest degree of flexibility comes from the ability to compute any surface property whatsoever (for which the user can supply the needed input files). Although the immediately obvious use of the program is the calculation of surface charge and hydrophobicity and input files are supplied for these tasks, the true power of the software would come from the computation of arbitrary surface properties. Finally, computations on several or many molecules can be run “in batch mode”.

To exemplify the above comparisons, we will briefly describe three possible applications of the program. The first is the calculation of the part of an exposed molecular surface that is due to a certain amino acid or group of amino acids. All the user needs to do to achieve this is to assign a 1 to the amino acid(s) of interest and a 0 to all others. An example input file might be:

```
ALA, 1
LYS, 0
MET, 0
... (all others 0)
```

The primary output file would contain the exposed area, in Å², which can be attributed to Alanine. The same can be done to discover what part of a molecular surface can be attributed to a particular element or set of elements; in this case, it is the atomic property file that would contain the ones and zeroes. In fact, it would even be possible to find out if and to what extent one particular atom or some other part of one particular amino acid, e.g. the nitrogen or COO⁻ group in Alanine is exposed, in the same way.

A second possible application is the determination of the fractal dimension or surface roughness of molecular surfaces. The fractal dimension of a 2D molecular surface can be defined by

\[ D = 2 - \frac{\log A}{\log r} \]  

where \( D \) is the fractal dimension, \( A \) is the surface area of the molecule and \( r \) is the radius of the probe sphere.

Since the derivative is generally smaller than 0, the fractal dimension is larger than 2 — an indication of the fractal nature of protein surfaces.

Although it was previously possible to compute the fractal dimension (which is an indication of the surface roughness of a molecule) of the molecular surface geometry, using our program one can study the variation (and possibly fractal nature) of any surface property with probe radius. For example, preliminary investigations suggest that the properties of protein surfaces exposed to water-sized probe spheres are vastly different quantitatively, but also qualitatively, to those exposed to larger probe spheres, representing perhaps other molecules or other solvents. Investigations of this nature should prove of some service in molecular biology and bioinformatics.

Finally, we will mention a more pragmatic and bioinformatics-related application. We have used these algorithms to compute the following surface properties for a set of some tens of proteins of interest in protein adsorption to solid surfaces (Nicolau and Nicolau 2002): surface area, positive/negative surface areas, total positive/negative surface charges, average surface charge, standard deviation of surface charge, total hydrophobic/hydrophilic surface areas, total surface hydrophobicity/hydrophilicity and average surface hydrophobicity.

The program we present here is dedicated entirely to the analysis of molecular surfaces, to aid the user in interpreting the data.

This data was compiled into a small database (the Biomolecular Descriptors Database). We used this, together with our Biomolecular Adsorption Database (B.A.D.) — which contains experimental data describing protein adsorption to solid surfaces from solution, to build semi-empirical models of protein adsorption from solution (Nicolau and Nicolau 2002). This database is available at www.bionanoeng.com/bad.

We note in closing that although the intended application areas for this software are bioinformatics and molecular biology, in principle these algorithms can be applied to any structures which can be represented in the same way as molecules are in the basic PDB format — as a set of spheres in three dimensions, the coordinates of whose centers and whose radii are given. Discussion of this possibility is beyond the scope of the present work.
5 Conclusion

We have developed general algorithms for the computation of arbitrary atomic and amino acid-based properties of molecular and especially biomolecular surfaces. These have been implemented in a Windows™ program, which we call the Protein Surface Properties Calculator (PSPC). The algorithms in question are described, as are the functionalities of the program. We also give some examples of possible applications. The program can be downloaded from the website www.bionanoeng.com.

References


(NMR) techniques or by molecular dynamic simulations. However, the experimental approaches are limited to small proteins and marred by long experimental time.

Protein 3D structure prediction directly from amino acid sequences still remains as an open and important problem in life sciences. The bioinformatics approach first predicts the protein secondary structure which represents an 1D projection of the very complicated 3D structure of a protein (Mount 2001). The goal of secondary structure prediction is to classify a pattern of residues in amino acid sequences to a corresponding secondary structure element: an $a$-helix (H), $b$-strand (E), or the remaining type, coil (C). Let us denote the given amino acid sequence by $r = (r_1, r_2, ..., r_n)$ where $S_R$ and $S_g$ is the set of 20 amino acids, and $t = (t_1, t_2, ..., t_n)$ denote the corresponding secondary structure sequence where $S_T$ and $S_g = \{H, E, C\}; n$ is the length of the sequence. The prediction of the secondary structure sequence, $t$, from an amino acid sequence, $r$, is the problem of finding the required mapping from the space of $S_R$ to the space of $S_T$.

Many computational techniques have been proposed in the literature to solve the protein secondary structure prediction problem, which can be broadly fallen into three categories: (1) statistical methods, (2) neural network approaches, and (3) nearest neighbor methods. The statistical methods are mostly based on likelihood techniques (Garnier, Osguthorpe, & Robson 1978; Gibrat, Garnier, & Robson 1987; Garnier, Gibrat, & Robson 1996). Neural network approaches use residues in a local neighborhood or window to predict the secondary structure at a particular location of an amino acid sequence (Rost & Sander 1993, Jones 1999). The nearest neighbor method often uses the k-nearest neighbor techniques (Salamon & Solovyev 1995; Salamon & Solovyev 1997).

SVMs have been earlier applied to protein secondary structure prediction (Hua & Sun 2001). One of the drawbacks in this approach is that the method does not capture the global information of the amino acid sequence due to the limited size of the local neighborhood. Bayesian approach provides a framework to take into account non-local interactions among amino acid residues (Schmidler, Liu, & Brutlag 2000). GOR techniques maximize the mutual information between amino acid sequences and the corresponding secondary structure sequences (Garnier et al. 1978, Gibrat et al. 1987, Garnier et al. 1996). Most existing secondary structure methods, such as SVM, GOR, and Bayesian techniques are single-stage approaches, except the PHD method (Rost et al. 1993) which combined two multi-layer perceptron (MLP) networks. The second-stage neural network captures the contextual relations among secondary structures and provides an prediction accuracy higher than that attained only with the first stage. However, two cascaded MLPs can be seen as a single MLP network having twice the number of layers of the single MLP. The improvement of the accuracy obtained with the two MLPs could also be achieved by using one MLP with larger input window.

Despite the existence of many approaches, the current success rates of existing approaches are insufficient; further improvement of the accuracy is necessary. Single-stage approaches are unable to find complex relations (correlations) among different elements in the sequence. This could be improved by incorporating the interactions or contextual information among the elements of the output sequence of secondary structures. We argue that it is feasible to enhance present single-stage approaches by augmenting with another prediction scheme at their outputs and propose to use SVMs as the second-stage.

In this paper, we present SVM and PHD techniques to protein secondary structure prediction. And then, a general framework for two-stage secondary structure prediction scheme is proposed by using SVMs as second classifiers for predicting the sequence of secondary structure elements from the output of the first stage. By using the SVMs to predict the output of the GOR and Bayesian predictors, the new prediction scheme achieves four percent of improvement in the accuracy on a database of 126 nonhomologous globular proteins. And by using SVMs to enhance the prediction of the SVM secondary structure scheme, an improvement of five percent was possible on the same dataset.

2 Support Vector Machine (SVM) Approach

![Architecture of the Support Vector Machine classifier for protein secondary structure prediction](Image)

**Fig. 1.** Architecture of the Support Vector Machine classifier for protein secondary structure prediction.
Support Vector Machine (SVM) learns the decision surface through a process of learning that achieves good generalization characteristics. SVMs have been proven to be the most efficient method in many real-world applications (Cristianini 2000, Vapnik 1995, Vapnik 1998). A typical SVM model used for protein secondary structure prediction is illustrated in Fig. 1. Each input amino acid unit is made up of 21 input positions, one for each amino acid and one for a padding space when the window overlaps the end of a sequence. Only one of the 21 components has a value of 1 for given input residue while the rest of the components are 0. When padding for the end of the sequence is required, the padding input component is set to 1.

Let \( v_i \) be the orthogonal binary vector representing 21-dimensional coding of the residue \( r_i \). Let the input pattern to the SVM to predict the residue at site \( i \) be \( s_i = (v_{i-1}, v_{i+1}, \ldots, v_{i+n}) \) where \( 2w + 1 \) is a window size around the element \( i \). Three binary SVM classifiers, \( C_1^k, k = 1, \ldots, n \), are constructed, each predicting that the secondary structure at the local site \( i \) belongs to the secondary structure type \( k \) or not. The input vectors, usually derived from a window of 7–17 amino acid residues, are transformed to a hidden-space and compared to the support vectors via a kernel function for each classifier. The results are then linearly combined by using parameters \( l_j \) that are found by solving a quadratic optimization problem. The detailed process of training of each classifier \( C_1^k \) is illustrated in the algorithm I:

**Algorithm I:** Classifier \( C_1^k \)

**Inputs:** Training examples \( \{s_1, s_2, \ldots, s_n\} \) and class labels \( \{q_1^k, q_2^k, \ldots, q_n^k\} \) where \( q_i^k \in \{-1, 1\} \).

Maximize over \( l_j \):

\[
Q(l) = -(1/2) \sum_{i=1}^{n} \sum_{j=1}^{n} l_i l_j q_i^k q_j^k K(s_i, s_j) + \sum_{j=1}^{n} l_j
\]

subject to

\[
0 \leq l_j \leq g \quad \text{and} \quad \sum_{j=1}^{n} l_j q_j^k = 0
\]

**Outputs:** Parameters \( l_j \)

The algorithm I, when the cost function \( Q \) is optimized, yields a classifier for \( C_1^k \) with maximum margin of separation (Vapnik 1998). The summations of the maximizing function \( Q \) run over all training patterns. \( K(s_i, s_j) \) denotes the kernel function, \( q_i^k \) encodes the secondary structure such that a binary value +1 if the secondary of the residue \( r_i \) is the secondary structure \( k \) or -1 otherwise, and \( g \) is a positive constant used to decide the trade-off between training error and the margin. Once the parameters \( l_j \) are obtained from the above algorithm, the resulting discriminant function is known.

The resulting discriminant function of an input vector \( s_i \) of the above classifier is given by

\[
D^k(s) = \sum_{j=1}^{n} l_j K(s_i, s_j) + b^k
\]

where the bias \( b^k \) is chosen so that \( q_i^k D^k(s_i) = 1 \) for any \( j \) with \( 0 \leq l_j \leq g \). The secondary structural type \( t_i \) of the residue \( r_i \) is determined by the highest value of three discriminant function values, i.e. by taking

\[
t_i = \arg \max_{k} D^k(s)
\]

The SVM approach suffers from the drawback that the interactions between distant amino acids in the sequence and the dependencies of sliding the neighborhood window are not taken care of.

3 PHD Method

PHD method is two-stage technique where two MLP networks are cascaded. The first level neural network of PHD method, called sequence-to-structure network, attempts to predict secondary structure from an input is an amino acid sequence of length 13. The object is to train the sequence-to-structure neural network to respond correctly to a set of such flanking sequence fragments when the secondary structural features of the centrally located amino acid are known. The training is designed to recognize amino acid patterns associated with secondary structures. If the neural network has sufficient capacity for learning, these patterns may potentially include complex interactions.
among the flanking amino acids in determining secondary structures.

The architecture of the sequence-to-structure network is illustrated in Fig. 2. The sequence-to-structure network is trained to classify mutually independent segments of residues in terms of the state of a single residue. There is no explicit representation of the fact that consecutive patterns are correlated, like for an \( \alpha \)-helix consisting at least three consecutive patterns. The correlation can be taken into account in part, by using a second level, structure-to-structure network. In the PHD method, a window of 17 secondary structure predictions is used as input to fully connected structure-to-structure network. This network has three outputs and the predicted secondary structure for the central amino acid is chosen as the largest of the three outputs. In this way the prediction becomes dependent on the surrounding structures. The second network significantly improves the accuracy and makes the prediction more realistic in terms of predicted mean lengths of secondary structure segments (Rost et al. 1993).

4 General Framework for Two-Stage Approach

In single-stage methods, the probabilities of a particular site belonging to secondary structural elements are obtained and the secondary element selected based on an error minimizing criteria. In bioinformatics, however, sequences that need to be predicted are usually unseen before and therefore a procedure that had good generalization ability is more useful. Above all, the probabilities that obtained in the above technique may not be accurate: for instance, SVM approach may suffer from the limited window length. The tenet of the two-stage approach is that the risk involved in the single-stage approaches could be minimized by incorporating the contextual information of the output secondary structure sequences, by incorporating another prediction schemes. We propose SVMs to predict the secondary structure elements using the secondary structure sequence given by the existing single-stage techniques because of the ability of SVMs to guarantee the minimum risk in the detection and to generalize for unseen data.

Let \( p^k_i \), \([0, 1]\) denote the probability of \( i \) element in the sequence, belonging to the secondary structure \( k \), \( S_T \). Consider a window of \( 2w_2 + 1 \) size around the element \( i \) at the output of the first stage and the set of vectors \( p^k_i = (p^k_{i-w_2}, p^k_{i-w_2+1}, \ldots, p^k_{i+w_2}) \) where \( i = 1, \ldots, n \). Now, we use three binary classifiers, \( C_2^k \), \( k \), \( S_T \), that decide whether or not a particular element belongs to a particular, i.e. \( k \), secondary structure. For this purpose, we use three SVMs corresponding to the three structuring elements and their outputs are combined to obtain the decision of the secondary structure. Our approach is illustrated in Fig. 3: the first-stage predictor could be a SVM or a Bayesian or a GOR secondary structure prediction scheme.

Since the data points represented by the set \( \{p^k_i, i = 1, \ldots, n\} \), for the classifier \( C_2^k \), are usually not linearly separable, the SVM converts input vectors \( p^k_i \) into a high dimensional feature space, using the mapping \( \mathcal{F} : [0, 1]^{2w_2+1} \rightarrow \mathbb{F} \), with the following kernel function \( K \):

\[
K(p^k_i, p^k_j) = \mathcal{F}(p^k_i) \cdot \mathcal{F}(p^k_j)
\]

to make the separating hyperplane in the transformed space, say \( \mathcal{F} \), linearly separate the training set \( \mathcal{G}^k = \{ (p^k_i, q^k_i) \mid i = 1, \ldots, n \} \) where \( q^k_0 \) denotes the desired output of the SVM classifier, \( C_2^k \).

\[\begin{align*}
\text{(Input: Amino acid sequence)} \quad & \quad \cdots \text{LSWTKCYAAVGAYAPAFSV} \cdots \\
& \quad \downarrow \\
\text{Predictor} \quad & \quad \mathcal{C}_1^H \quad \mathcal{C}_1^E \quad \mathcal{C}_1^C \\
& \quad \downarrow \\
\text{SVM} \quad & \quad \mathcal{C}_2^H \quad \mathcal{C}_2^E \quad \mathcal{C}_2^C \\
& \quad \downarrow \\
\text{max argument} \quad & \quad \cdots \text{HHHHHHEEEEEECCCCCCCCHHHH} \cdots \\
\text{(Output: Secondary structure sequence)}
\end{align*}\]

Fig. 3. A general framework for two-stage method for protein secondary structure prediction: the classifier is obtained by cascading Bayesian, GOR, or SVM predictors with support vector machines.

The goal here is to find a function \( f(\mathcal{F}(p^k_i), \mathbf{w}) : \mathcal{F} \mapsto \{+1, -1\} \) where \( \mathbf{w} \) is a set of parameters, which provides the smallest value for the risk that measures the ability of the function to generalize well, by minimizing the following risk functional:

\[
R(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T K(\mathcal{F}(p^k_i), \mathcal{F}(p^k_j)) \mathbf{w} + \mathbf{w}^T q^k \cdot \mathbf{1}
\]

(3)
However, minimizing $R(w)$ is not a trivial problem because the form of the probability distribution $P(f(p^i), q^i)$ is unknown. Assuming that the training set is drawn randomly and independently based on the probability distribution $P(f(p^i), q^i)$ we can instead consider the following so-called empirical risk functional:

$$R_{\text{emp}}(w) = \frac{1}{n} \sum_{j=1}^{n} |q_j^i - f(p^i_j), w|$$  \hspace{1cm} (4)$$

The following bound Vapnik 1998) holds with probability $1-h$ where $0 \leq h \leq 1$

$$R(w) \leq R_{\text{emp}}(w) + \sqrt{\frac{1}{n} \log(2n/h) + 1 - \log(1/h)}$$  \hspace{1cm} (5)$$

where $h$ called the Vapnik Chervonenkis (VC) dimension is the maximum number of points that can be separated in all possible ways by the set of decision functions parameterized from $w$. The second term on the right hand side is called the confidence term. Thus, the smaller $h$ the better the generalization performance of function $f(p^i), w)$ chosen by empirical risk minimization.

**Theorem 1** (Vapnik 1998): Let $R$ be the radius of the smallest ball $B(a) = \{z : F : |z - a| \leq R \}$, $a \in F$ containing the points $f(p^i_1), \ldots, f(p^i_n)$ where $f:[0,1]^2 \rightarrow \{1,0\}$ and let

$$f(p^i), w) = \text{sgn}(w^T f(p^i) + b^i)$$

be canonical hyperplane decision functions defined on these points. Then the set $\{f(p^i), w), \|w\| < R\}$ has a VC dimension $h$ satisfying

$$h < R^2 A^2 + 1$$

From Theorem 1, we can minimize the confidence term by minimizing the term $\|w\|$. Once the minimum value of $\|w\|$ is obtained, the function $f(p^i), w)$ classifies a sample $p^i$ based on the sign of the resulting discriminant function, that measures the distance from this sample to the Optimal Separating Hyperplane (OSH), given by

$$D(p^i) = w^T f(p^i) + b^i$$  \hspace{1cm} (6)$$

Minimization of $\|w\|$ can be done by solving the following convex Quadratic Programming (QP) problem (Vapnik 1998): Maximize over $\lambda^i$:

$$Q = \sum_{j=1}^{n} \lambda_j^i - (1/2) \sum_{i=1}^{n} \sum_{j=1}^{n} q_j^i q_j^i \lambda_i^i \lambda_j^i K(p^i, p^j)$$

subject to $0 \leq \lambda_j^i \leq g$ and $\sum_{j=1}^{n} \lambda_j^i q_j^i = 0$. Here, $w = \sum_{j=1}^{n} q_j^i \lambda_j^i f(p^j)$. 

The resulting discriminant function of an input vector $p^i$ of the classifier $C^i_k$ is given by

$$D(p^i) = \sum_{j=1}^{n} q_j^i \lambda_j^i K(p^j, p^i) + b^i$$  \hspace{1cm} (7)$$

| Protein 3D structure prediction directly from amino acid sequences still remains as an open and important problem in life sciences. |

where $b^i$ is chosen so that $q_j^i D(p^i) = 1$ for any $j$ with $0 \leq \lambda_j^i \leq g$. The secondary structural type, $t_i$, of the residue $r_i$ is specified from the largest positive distance to the OSH of three discriminant function values, i.e. by taking

$$t_i = \arg \max_{k \in 3} D_k(p^i)$$  \hspace{1cm} (8)$$

The above equation is one way of combining the outputs of the three classifiers, which has been widely used to handle the multi-category problem (Scholkopf, Burges & Vapnik 1995).

**Despite the existence of many approaches, the current success rates of existing approaches are insufficient; further improvement of the accuracy is necessary.**

**Table 1**: Comparison of performances of two-stage SVM approach and single-stage approaches and PHD method in protein secondary structure prediction on RS126 dataset.
Fig. 4: Performance of secondary structure prediction by two-stage SVMs and GOR technique followed by SVMs on RS126 dataset. The window length indicates the size of neighborhood taken as the input for the second-stage SVMs.

5 Experiments and Results

The set 126 nonhomologous globular protein chains used in the experiment of Rost and Sandar (Rost et al. 1993), referred to as the RS126 set, was used to evaluate the accuracy of the classifiers. The dataset contained 24,395 residues with 32 percent \(\alpha\)-helix, 21 percent \(\beta\)-strand, and 47 percent coil. Many current generation secondary structure prediction methods have been developed and tested on this dataset.

The single-stage approaches and second-stage approaches were implemented and tested on the above data set, using a sevenfold cross validation technique to estimate the prediction accuracy. MLP and SVM approaches were implemented, with multiple alignment, and tested on RS126 dataset, using a seven-fold cross validation technique to estimate the prediction accuracy. With sevenfold cross validation approximately one-seventh of the database was left out while training and, after training, the left one-seventh of the dataset was used for testing. In order to avoid the selection of extremely biased partitions, the RS126 set was divided into seven subsets with each subset having similar size and content of each type of secondary structure. For SVMs, the kernel selected here was the radial basis function \(K(x, y) = e^{-\frac{||x-y||^2}{2s^2}}\) with the parameters: \(s = 1\) and \(g = 0.25\), determined empirically for optimal performance. The use of Gaussian kernel showed the best performance even though the dimension of feature space is infinite (Schmidler, Sung, Burges, Girosi, Niyogi, Poggio, & Vapnik 1997). The SVMs were implemented using sequential minimization algorithm (Platt 1999) which is simple to implement without needing storage for matrices or to invoke an iterative numerical routine for each sub-problem.

The tenet of the two-stage approach is that the risk involved in the single-stage approaches could be minimized.

We have used several measures to evaluate the prediction accuracy. The \(Q_3\) accuracy indicates the percentage of correctly predicted residues of three states of secondary structure (Cuff, & Barton 1999). Matthew's correlation coefficients \((r_H, r_E, r_C)\) provide the success of predicting residues for each type of secondary structure (Matthews 1975). Segment overlap measure (SOV) gives accuracy by counting predicted and observed segments, and measuring their overlap (Cuff et al. 1999). The experiment was also run on the SVMs and GOR IV method alone. Table 1 shows the performance of the different secondary structure predictors and the best results reported by Rost and Sander with PHD method (Rost et al. 1993) using two-level neural networks on the RS126 set with a single input sequence. The best algorithm was found to be the cascade of two SVMs, which achieved 72.7 percent accuracy \(Q_3\) while the accuracy of prediction by PHD method was 70.3 percent. By combining SVMs at the output of the GOR IV and Bayesian techniques, the accuracies of 68.2 percent was obtained. That is, by introducing SVMs at the output of SVM and GOR techniques, the prediction accuracy was improved by 4–5 percent.

The best algorithm was found to be the cascade of two SVMs, which achieved 72.7 percent accuracy \(Q_3\) while the accuracy of prediction by PHD method was 70.3 percent.

We experimented with different neighborhood sizes for the input of the second stage of the Bayesian and SVM first-stage techniques. Fig. 4 illustrates the
performance of the second-stage prediction against the neighborhood size (window length). As seen the window size of width seven gives the optimal accuracy for the second stage; at the present time, we are unable to reason this size of neighborhood from the biological point of view. The accuracies at window size of zero correspond to the accuracies obtained with the first stage alone.

Our experiments demonstrated that it is feasible to extend current single-stage approaches with a second-stage to improve the accuracy of prediction.

6 Conclusion

We introduced a general framework for two-stage approaches by using SVMs to predict protein secondary structure from the output from earlier single-stage techniques. Our experiments demonstrated that it is feasible to extend current single-stage approaches with a second-stage to improve the accuracy of prediction because secondary structure at a particular position of a sequence depends not only on the amino acid residue at a particular location but also on the structural formations of the rest of the sequence. This intrinsic relation cannot be captured by using only single-stage approaches alone. Therefore, another layer of classifiers, which predicts the output of single-stage methods, improves the accuracy of prediction. As seen in the experiments, SVMs were optimal classifiers for the second-stage because they minimized not only the empirical risk of known sequences but also the actual risk of unknown sequences.

SVMs were optimal classifiers for the second-stage because they minimized not only the empirical risk of known sequences but also the actual risk of unknown sequences.

References
