Prediction of Disulfide Bonding Pattern Based on Support Vector Machine with Parameters Tuned by Multiple Trajectory Search

HSUAN-HUNG LIN[1,2], LIN-YU TSENG[3,4]
[1]Department of Applied Mathematics, National Chung Hsing University, Taichung, Taiwan 402, ROC
[2]Department of Management Information Systems, Central Taiwan University of Science and Technology, Taichung 40601, Taiwan, R.O.C
[3]Institute of Networking and Multimedia, National Chung Hsing University, Taichung, Taiwan 402, ROC
[4]Department of Computer Science and Engineering, National Chung Hsing University, Taichung, Taiwan 402, ROC
shlin@ctust.edu.tw, lytseng@cs.nchu.edu.tw

Abstract: - The prediction of the location of disulfide bridges helps solving the protein folding problem. Most of previous works on disulfide connectivity pattern prediction use the prior knowledge of the bonding state of cysteines. In this study an effective method is proposed to predict disulfide connectivity pattern without the prior knowledge of cysteins' bonding state. To the best of our knowledge, without the prior knowledge of the bonding state of cysteines, the best accuracy rate reported in the literature for the prediction of the overall disulfide connectivity pattern (*Qp*) and that of disulfide bridge prediction (*Qc*) are 48% and 51% respectively for the dataset SPX. In this study, the cystein position difference, the cystein index difference, the predicted secondary structure of protein and the PSSM score are used as features. The support vector machine (SVM) is trained to compute the connectivity probabilities of cysteine pairs. An evolutionary algorithm called the multiple trajectory search (MTS) is integrated with the SVM training to tune the parameters for the SVM and the window sizes for the predicted secondary structure and the PSSM. The maximum weight perfect matching algorithm is then used to find the disulfide connectivity pattern. Testing our method on the same dataset SPX, the accuracy rates are 54.5% and 60% for disulfide connectivity pattern prediction and disulfide bridge prediction when the bonding state of cysteines is not known in advance.

Key-Words: - Disulfide bonding pattern, SVM, multiple trajectory search

1 Introduction

Disulfide bonds play an important structural role in stabilizing protein conformations. The prediction of disulfide bonding pattern helps to a certain degree the prediction of the three-dimensional protein structure and hence its function because disulfide bonds impose geometrical constraints on the protein backbones. Some recent research works do have shown the close relation between the disulfide bonding patterns and the protein structures [1, 2]. As shown in Fig. 1, the structures of (a) the tick anticoagulant peptide (1TAP), a protease inhibitor, and (b) the bovine pancreatic trypsin inhibitor (1QLQ), a serine protease inhibitor. Their sequence identity is only 18.2%, the absence of significant sequence identity between 1TAP and 1QLQ was noted by Antuch *et al*. [3] and PSI-BLAST searches in the SwissProt/TrEMBL and NR database were unsuccessful in identifying the similarity between these two proteins. In disulfide-bonding base classification, these two proteins have the same disulfide-bonding connectivity pattern (1-6, 2-3, 4-5), and all of which are classified in the BPTI-like superfamily in SCOP [4].

In the realm of the disulfide bond prediction, two problems are addressed. The first is the prediction of the disulfide bonding states and the second is the prediction of the disulfide bonding pattern. Recently, significant progress has been made in the prediction of the disulfide bonding states. Several methods based on statistical analysis [5], neural networks [6, 7], or support vector machines [8] had been proposed. They are quite effective in predicting the bonding state of cysteines with the accuracy rates around 81%-90%.

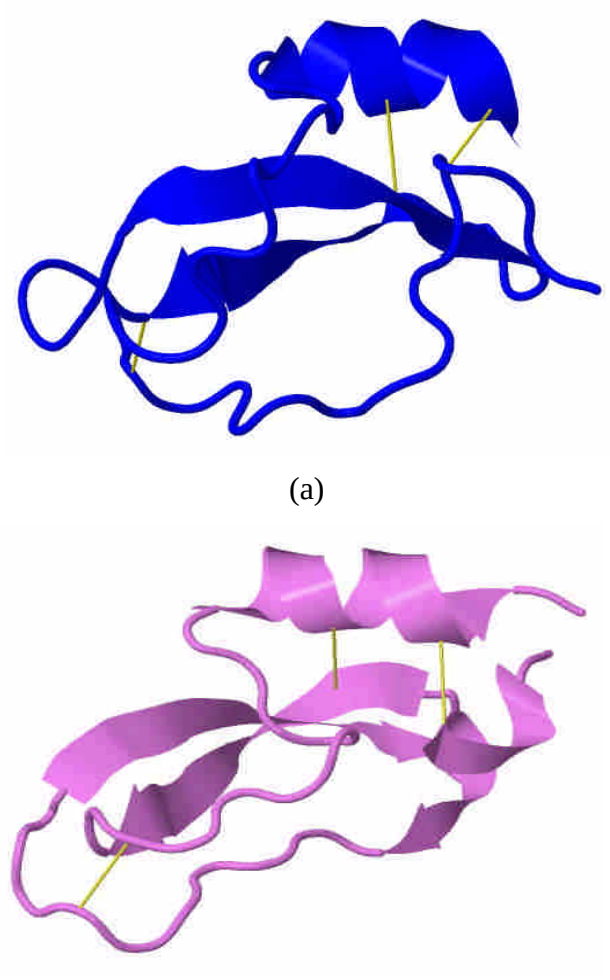

Fig. 1 The structures of (a) the tick anticoagulant peptide (1TAP), a protease inhibitor, and (b) the bovine pancreatic trypsin inhibitor (1QLQ), a serine protease inhibitor.

Recently, several methods were proposed for the prediction of the disulfide bonding pattern. The first method was presented by Fariselli and Casadio [9]. They reduced disulfide connectivity to the graph matching problem in which vertices are oxidized cysteines and edges are labeled by the strength of interaction (contact potential) in the associated pair of cysteines. The Monte Carlo simulated annealing method is used to find the optimal values of contact potentials and finally the disulfide bridges are located by finding the maximum weight perfect matching. Fariselli *et al.* [10] improved their previous results by using neural networks to predict the cysteine pairwise interactions. Vullo and Frasconi [11] developed an ad-hoc recursive neural network for scoring labeled undirected graphs that represent the connectivity pattern and they improved the accuracy rate of bonding pattern prediction significantly from 34% to 44%. Cheng *et al.* [12] improved the prediction accuracy by using two-dimensional recursive neural networks to predict connectivity probabilities between cysteine pairs. Ferrè and Clote [13] designed the diresidue neural network to predict connectivity probabilities between cysteine pairs. They also used secondary structure information and diresidue frequencies in their training. Tsai *et al.* [14] used the support vector machine to predict connectivity probabilities between cysteine pairs. The features used in training the support vector machine are local sequence profiles and the linear distance of cysteines. All above mentioned methods are based on the reduction of the connectivity pattern prediction to the maximum weight perfect matching problem. The following four methods are not based on this reduction. Chen and Hwang [15] used the support vector machine to predict the bonding pattern directly. The features they used in training the support vector machine are the coupling between the local sequence environments of cysteine pairs, the cysteine separations, and the amino acid content. Zhao *et al.* [16] used a simple feature called cysteine separations profiles (CSP) to predict the connectivity patterns. Chen *et al.* [17] proposed a two-level model. Lu *et al.* [18] obtained the accuracy of 73.9% by using GA to optimize feature selection for the SVM. Song *et al.* [19] obtained the accuracy of 74.4% by using multiple sequence vectors and secondary structure. This accuracy rate is the best one found in the literature. Rubinstein *et al.* [20] analyzes the correlated mutation patterns in multiple sequence alignments to predict the disulfide bond connectivity. All these methods except Cheng *et al.* [12] and Ferrè *et al.* [13] assume that the bonding states are known. The method proposed by Cheng *et al.* [12] and Ferrè *et al.* [13] can be applied whether the bonding states are known or not.

1.1 Support vector machine

Support vector machine (SVM) is a supervised learning method used for classification [21]. It is believed to be superior to traditional statistical and neural network classifiers. However, it is critical to determine suitable combination of SVM parameters regarding classification performance. A special property of SVM is that it can simultaneously minimize the empirical classification error and maximize the geometric margin. It is a useful technique for data classification and regression and has become an important tool for machine learning and data mining. It is a powerful methodology for

solving problems in nonlinear classification, function estimation and density estimation which has also led to many applications, such as the image interpretation, data mining and other biotechnological fields [22-24]. SVM is generally used for data which can be classified into two clusters, however, classification of multiple clusters can also be easily expanded [25].

In general, SVM has better performance when competed with existing methods, such as neural networks and decision trees [26-28]. Recently, application of SVM in medicine has grown rapidly. For examples, it has been applied in prediction of RNA-binding sites in proteins [29], protein secondary structure prediction [30]. SVM has also been applied to biological problems [31, 32], electrical energy consumption forecasting [33], remote sensing image classification [34], and pattern recognition and data classification [35].

The goal of support vector machine (SVM) is to separate multiple clusters by constructing a set of separating hyperplanes with greatest margin to the boundary of each cluster. For a two-class classification example, let us view the input data as in n-dimensional space. SVM will construct the hyperplane (Eq. 1) in the space to separate two classes that leaves the maximum margins from both classes. [36, 37]

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = 0 \qquad (\mathbf{1})$$

The distance of a point from a hyperplane is given by

$$z = \frac{g(\mathbf{x})}{\|\mathbf{w}\|} \qquad (\mathbf{2})$$

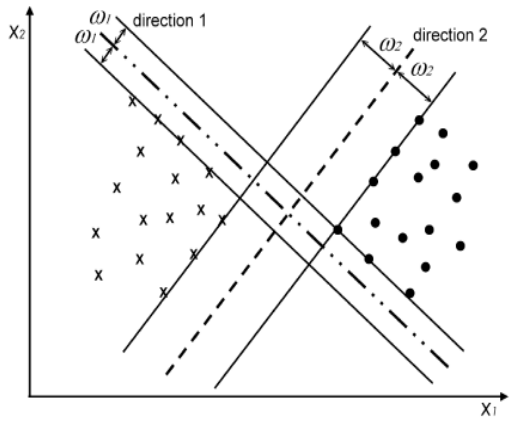

Fig. 2. The hyperplane with greatest margin. Note that the margin of direction 2 is larger than the margin of direction 1

As shown in Fig. 2, the values of **w** and w_0 in Eq. (1) are scaled so that the values of g(x) at the nearest points in class 1 and class 2 equal to 1 and -1 respectively. Therefore, finding the hyperplane becomes a nonlinear quadratic optimization problem, which can be formulated as:

$$\text{Minimize } J(w) = \frac{\|\mathbf{w}\|^2}{2} \qquad (\mathbf{3})$$
$$\text{Subject to } y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1, \quad i = 1, 2, ..., N$$

The above minimizer must satisfy Karush-Kuhn-Tucker (KKT) condition, and it can be solved by considering Lagrangian duality. The problem can be stated equivalently by its Wolfe dual representation form:

$$\text{Maximize} \quad L(\mathbf{w}, w_0, \boldsymbol{\lambda}) = \frac{w^T w}{2} - \sum_{i=1}^{N}\lambda_i[y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1]$$
$$\text{Subject to} \quad \mathbf{w} = \sum_{i=1}^{N}\lambda_i y_i \mathbf{x}_i, \quad \sum_{i=1}^{N}\lambda_i y_i = 0, \quad \text{and} \quad \boldsymbol{\lambda} \geq 0 \qquad (\mathbf{4})$$

where $L(\mathbf{w}, w_0, \boldsymbol{\lambda})$ is the Lagrangian function and $\boldsymbol{\lambda}$ is the vector of Lagrangian multipliers. By comparing Eqs. (3) and (4), it is noted that the first two constraints in Eq. (4) become equality constraints and this makes the problem easier to be solved. After a little bit algebra manipulation, Eq. (4) becomes

$$\max_{\boldsymbol{\lambda}} \left(\sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j\right) \qquad (\mathbf{5})$$
$$\text{Subject to } \sum_{i=1}^{N}\lambda_i y_i = 0 \text{ with } \boldsymbol{\lambda} \geq 0$$

As soon as the Lagrangian multipliers are obtained by maximizing the above equation, the optimal hyperplane can be obtained from $\mathbf{w} = \sum_{i=1}^{N}\lambda_i y_i \mathbf{x}_i$ in Eq. (4).

Once the optimal hyperplane is obtained, classification of a sample is performed based on the sign of the following equation:

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = \sum_{i=1}^{Ns}\lambda_i y_i \mathbf{x}_i^T\mathbf{x}_i + w_0 \qquad (\mathbf{6})$$

where *Ns* is the number of support vectors. For a vector $\mathbf{x} \in R^l$ in the original feature space, assume that there exists a mapping ϕ from $\mathbf{x} \in R^l$ to $\mathbf{y} = \phi(\mathbf{x}) \in R^k$, where k is usually much higher than l. Then it is always true that

$$\sum_r \phi_r(\mathbf{x})\phi_r(\mathbf{z}) = K(\mathbf{x}, \mathbf{z}) \qquad (\mathbf{7})$$

where $\phi_r(\mathbf{x})$ is the rth component of the mapping and the kernel function $K(\mathbf{x},\mathbf{z})$ is a symmetric function satisfying the following condition

$$\int K(\mathbf{x},\mathbf{z})g(\mathbf{z})d\mathbf{x}d\mathbf{z} \geq 0, \text{ and } \int g(\mathbf{x})^2 d\mathbf{x} \leq \infty \quad (\mathbf{8})$$

For a nonlinear classifier, various kernels, including polynomial, radial basis function, and hyperbolic tangent, as shown in Eq. (9) can be used for mapping the original sample space into a new Euclidian space in which Mercer's conditions are satisfied. The linear classifier can then be designed for classification.

$$K(\mathbf{x},\mathbf{z}) = (\mathbf{x}^T\mathbf{z}+1)^q, \quad q>0 \quad (\mathbf{9a})$$

$$K(\mathbf{x},\mathbf{z}) = \exp(-\|\mathbf{x}-\mathbf{z}\|^2/\sigma^2) \quad (\mathbf{9b})$$

$$K(\mathbf{x},\mathbf{z}) = \tanh(\beta\mathbf{x}^T\mathbf{z}+\gamma) \quad (\mathbf{9c})$$

n-fold cross-validation of the SVM model is achieved by dividing the dataset into n folds. When some fold is reserved for testing, the other n-1 folds are used for training the model.

1.2 Multiple Trajectory Search

The multiple trajectory search (MTS) had been presented for large scale global optimization [38]. The MTS had also been used to solve the multi-objective optimization problems and obtained satisfactory results [39]. It uses multiple agents to search the solution space concurrently. Each agent does an iterated local search using one of three candidate local search methods. By choosing a local search method that best fits the landscape of a solution's neighborhood, an agent may find its way to a local optimum or the global optimum.

1.2.1 Orthogonal Array and Simulated Orthogonal Array

The concept of orthogonal arrays which are used in experimental design methods is briefly introduced. Suppose in an experiment, there are k factors and each factor has q levels. In order to find the best setting of each factor's level, qk experiments must be done. Very often, it is not possible or cost effective to test all qk combinations. It is desirable to sample a small but representative sample of combinations for testing. The orthogonal arrays were developed for this purpose. In an experiment that has k factors and each factor has q levels, an orthogonal array OA(n,k,q,t) is an array with n rows and k columns which is a representative sample of n testing experiments that satisfies the following three conditions. (1) For the factor in any column, every level occurs the same number of times. (2) For the t factors in any t columns, every combination of q levels occurs the same number of times. (3) The selected combinations are uniformly distributed over the whole space of all the possible combinations. In the notation OA(n,k,q,t), n is the number of experiments, k is the number of factors, q is the number of levels of each factor and t is called the strength.

The orthogonal arrays exist for only some specific n's and k's. So it is not appropriate to use the OA in some applications. Tseng *et al.* [38] proposed the simulated OA (SOA). The SOA satisfies only the first of the above mentioned three conditions, but it is easy to construct an SOA of almost any size. Suppose there are k factors and each factor has q levels, an m×k simulated orthogonal array $SOA_{m\times k}$ with m being a multiple of q can be generated as follows. For each column of $SOA_{m\times k}$, a random permutation of 0,1, …, q-1 is generated and denoted as sequence C. Then the elements in C are picked one by one sequentially and filled in a randomly chosen empty entry of the column. If all elements in C were picked, the process picks elements again from the beginning of C. So in every column of $SOA_{m\times k}$, each of q elements will appear the same number of times (condition 1).

1.2.2 The Multiple Trajectory Search

The MTS generates M initial solutions by utilizing the simulated orthogonal array $SOA_{M\times N}$, where the number of factors corresponds to the dimension N and the number of levels of each factor is taken to be M. So each of 0, 1, …, M-1 will appear once in every column. Using SOA tends to make these M initial solutions uniformly distributed over the feasible solution space. The initial search range for local search methods is set to half of the difference between the upper bound and the lower bound. Afterwards, local search methods will change the search range.

The MTS consists of iterations of local searches until the maximum number of function evaluations is reached. In the first iteration, the MTS conducts local searches on all of M initial solutions. But in the following iterations, only some better solutions are chosen as foreground solutions and the MTS conducts local searches on these solutions. Three local search methods are provided for the MTS. The MTS will first test the performance of three local search methods and then choose the one that performs best, that is, the one that best fits the landscape of the neighborhood of the solution, to do the search. After conducting the search on foreground solutions, the MTS applies Local Search 1 to the current best solution trying to improve the current best solution. Before the end of an iteration, some better solutions are chosen as the foreground

solutions for the next iteration. The multiple trajectory search algorithm is described in the following.

```
Multiple Trajectory Search
/*Generate M initial solutions */
Build simulated orthogonal array SOA_{M×N}
For i = 1 to M
  For j = 1 to N
    X_i[j] = l_i + (u_i-l_i) * SOA[i, j] / (M-1)
  End For
End For
Evaluate function values of X_i's
For i = 1 to M
  Enable[i] ← TRUE
  Improve[i] ← TRUE
  SearchRangeX_i = (UPPER_BOUND-LOWER_BOUND)/2
End For
While ( #ofEvaluation ≤ predefined_max_evaluation)
  For I =1 to M
    If Enable[i] = TRUE
    Then GradeX_i ← 0
      LS1_TestGrade ← 0
      LS2_TestGrade ← 0
      LS3_TestGrade ← 0
      For j = 1 to #ofLocalSearchTest
        LS1_TestGrade ← LS1_TestGrade+
            LocalSearch1(X_i , SearchRangeX_i)
        LS2_TestGrade ← LS2_TestGrade+
            LocalSearch2(X_i , SearchRange X_i)
        LS3_TestGrade ← LS3_TestGrade+
            LocalSearch3(X_i , SearchRange X_i)
      End For
      Choose the one with the best TestGrade and
      let it be LocalSearchK   /* K may be 1, 2, or 3 */
      For j = 1 to #ofLocalSearch
        GradeX_i ← GradeX_i+
            LocalSearchK(X_i , SearchRangeX_i)
      End For
    End If
  End For
  For I = 1 to #ofLocalSearchBest
    LocalSearch1(BestSolution,SearchRangeBestSolution)
  End For
  For i = 1 to M
    Enable[i] ← FALSE
  End For
  Choose #ofForeground X_i's whose GradeX_i are best among the
  M solutions and set their corresponding
  Enable[i] to TRUE
End While
```

In the MTS, three local search methods are used for searching different landscape of the neighborhood of a solution. Local Search 1 searches along one dimension from the first dimension to the last dimension. Local Search 2 is similar to Local Search 1 except that it searches along a direction derived from about one-fourth of dimensions. In both local search methods, the search range (SR) will be cut to one-half until it is less than 1×10^{-15} if the previous local search does not make improvement. In Local Search 1, on the dimension concerning the search, the solution's coordinate of this dimension is first subtracted by SR to see if the objective function value is improved. If it is, the search proceeds to consider the next dimension. If it is not, the solution is restored and then the solution's coordinate of this dimension is added by 0.5*SR, again to see if the objective function value is improved. If it is, the search proceeds to consider the next dimension. If it is not, the solution is restored and the search proceeds to consider the next dimension. Local Search 1 and Local Search 2 are listed in the following.

```
Function LocalSearch1(X_k, SR)
If Improve[k]=FALSE
  Then SR = SR /2
      If SR < 1e-15
      Then SR←(UPPER_BOUND-LOWER_BOUND) *0.4
    End If
End If
Improve[k] ← FALSE
For i = 1 to N
  X_k[i] ←X_k[i]- SR
  If X_k is better than current best solution
    Then grade ← grade + BONUS1
        Update current best solution
  End If
  If function value of X_k is the same
    Then restore X_k to its original value
  Else
    If function value of X_k degenerates
      Then restore X_k to its original value
        X_k[i]_X_k[i] + 0.5* SR
        If X_k is better than current best solution
          Then grade ← grade + BONUS1
              Update current best solution
        End If
        If function value of X_k has not been improved
          Then restore X_k to its original value
        Else
          grade ← grade + BONUS2
          Improve[k] ← TRUE
        End If
    Else
      grade ← grade + BONUS2
      Improve[k] ← TRUE
    End If
  End If
End For
return grade
```

```
Function LocalSearch2(X_k, SR)
If Improve[k] = FALSE
  Then SR = SR/2
    If SR < 1e-15
      Then SR ← (UPPER_BOUND-LOWER_BOUND) *0.4
    End If
End If
Improve[k] ← FALSE
For I = 1 to N
  For i = 1 to N
    r[i] ← Random{0,1,2,3}
    D[i] ← Random{-1,1}
  End For
  For i = 1 to N
    If r[i] = 0
      Then X_k[i] ← X_k[i] – SR * D[i]
    End If
  End For
  If X_k is better than current best solution
    Then grade ← grade + BONUS1
         Update current best solution
  End If
  If function value of X_k is the same
    Then restore X_k to its original value
  Else
    If function value of X_k degenerates
      Then restore X_k to its original value
        For i =0 to N
          If r[i]=0
            Then X_k[i]_X_k[i] + 0.5 * SR * D[i]
          End If
        End For
        If X_k is better than current best solution
          Then grade ← grade + BONUS1
               Update current best solution
        End If
        If function value of X_k has not been improved
          Then restore X_k to its original value
        Else
          grade ← grade + BONUS2
          Improve[k] ← TRUE
        End If
    Else
      grade ← grade + BONUS2
      Improve[k] ←TRUE
    End If
  End If
End For
return grade
```

Local Search 3 is different from Local Search 1 and Local Search 2. Local Search 3 considers three small movements along each dimension and heuristically determines the movement of the solution along each dimension. In Local Search 3, although the search is along each dimension from the first dimension to the last dimension, the evaluation of the objective function value is done after searching all the dimensions, and the solution will be moved to the new position only if the objective function has been improved at this evaluation. Local Search 3 is described in the following.

```
Function LocalSearch3(X,SR)
For i = 1 to N
  X_1 ← X's ith coordinate is increased by 0.1
  Y_1 ← X's ith coordinate is decreased by 0.1
  X_2 ← X's ith coordinate is increased by 0.2
  If X_1 is better than current best solution
    Then grade ← grade + BONUS1
         Update current best solution
  End If
  If Y_1 is better than current best solution
    Then grade_grade + BONUS1
         Update current best solution
  End If
  If X_2 is better than current best solution
    Then grade ← grade + BONUS1
         Update current best solution
  End If
  D_1 = F(X) - F(X_1)
  If D_1 > 0        // X_1 is better than X
    Then grade ← grade + BONUS2
  End If
  D_2 = F(X) - F(Y_1)
  If D_2 > 0        //Y_1 is better than X
    Then grade ← grade + BONUS2
  End If
  D_3 = F(X) - F(X_2)
  If D_3 > 0        //X_2 is better than X
    Then grade ← grade + BONUS2
  End If
  a ← Random[0.4, 0.5]
  b ← Random[0.1, 0.3]
  c ← Random[0, 1]
  X[i] = X[i] + a(D_1 - D_2) + b(D_3-2D_1) + c
End For
If function value of X has not been improved
  Then restore X to its original value
Else
  grade ← grade + BONUS2
End If
return grade
```

2 Materials and Methods

2.1 Dataset

In order to compare the prediction accuracy rates with previously reported method by Cheng et al. [12], the same dataset SPX used by them was employed in the experiment. In SPX, all proteins were extracted from the PDB on May 17, 2004 that contain at least one intrachain disulfide bond and all

proteins that contain less than 12 amino acids were removed. Furthermore, to reduce overrepresentation of particular protein families, Cheng et al. used the UniqueProt, a protein redundancy reduction tool based on the HSSP distance [40], to choose 1018 proteins by setting the HSSP cut-off distance to 10. In SPX, the protein sequences were randomly divided into 10 subsets with roughly equal size for 10-fold cross-validation experiment.

2.2 Methodology

Cheng *et al.* [12] predicted the bonding state of cysteines first, and then they predicted the disulfide bond pattern for the predicted oxidized cysteines. Our method, instead of predicting bonding state first, directly predicts the bonding probability of all pairs of cysteines. Our method uses the cystein position difference, the cystein index difference, the predicted secondary structure of the protein and the PSSM score as features. The SVM is trained to compute the connectivity probabilities of all the cysteine pairs. The MTS [38] is used to evolve the parameters C and γ for SVM and window sizes for the predicted secondary structure and the PSSM. The maximum weight perfect matching algorithm is then used to find the disulfide connectivity pattern without the prior knowledge of the bonding state of cysteines.

2.3 Features

(1) ***NCPD*** (***Normalized Cysteine Position Difference)***: Let $\{c_1,c_2,\ldots,c_n\}$ be the positions of the cysteines in ascending order. The normalized cystein position difference between c_i and c_j is defined as $|c_i-c_j|/(c_n-c_1)$.

(2) ***NCID (Normalized Cysteine Index Difference)***: Let $\{c_1,c_2,\ldots,c_n\}$ be the positions of the cysteines in ascending order. The normalized cystein ordering index between c_i and c_j is defined as $|i-j|/n$.

(3) ***PSSM***: PSI-BLAST [41] is used to obtain the local sequence profiles. The output file contains four parts. The first part is the position-specific scoring matrix (PSSM). The PSSM score is used as one of features.

(4) ***PSS (Predicted Secondary Structure)***: The predicted secondary structure obtained by applying Jones' prediction method [42] is used as a feature. In practice, the PSIPRED program is used to predict the secondary structure information.

2.4 Construction of Prediction Model Based on SVM

It is noted that SVM is superior to traditional statistical and neural network classifiers in many applications. However, it is critical to determine proper combination of SVM parameters (C and γ) in order to achieve good classification performance. The SVM implementation used in this work is LIBSVM [43]. Since the prediction rate is highly influenced by the value of the parameters C and γ, the multiple trajectory search [38] is used for finding good settings of parameter values for the SVM and the window sizes for the PSS and the PSSM.

2.5 Multiple Trajectory Search for Selecting SVM Parameters and Window Sizes

The multiple trajectory search can find optimal or near-optimal solution within an acceptable time, and is faster than the dynamic programming or the branch-and-bound strategy. Previously, some research works applied the evolutionary algorithms to select features in the first phase and then used the selected features to train the SVM in the second phase. In this study, the MTS and the SVM training are tightly integrated. Since the values of parameters C and γ for the SVM are critical to classification accuracy of the SVM, selecting proper values of C and γ becomes an important task. Traditionally, the regular grid search strategy was used to perform the parameter value selection. However, it is very time-consuming. In this work, the MTS is integrated with the SVM training to select not only the value of parameters C and γ but also the window sizes of the PSS and the PSSM. As shown in Fig. 3, a chromosome is coded as $S_i=(C_i, \gamma_i, S_{i1}, S_{i2})$ where C_i and γ_i are the log values of the parameters C and γ, and S_{i1}, S_{i2} are the window sizes for the PSS and the PSSM respectively. The fitness function is defined as the accuracy of the SVM on disulfide connectivity prediction. The flowchart of the integration of the MTS and SVM training is shown in Fig. 4.

<table>
<tr><td>$\log_2 C$</td><td>$\log_2 \gamma$</td><td>S_1</td><td>S_2</td></tr>
<tr><td colspan="2">SVM Parameters</td><td colspan="2">Window size for the PSS and the PSSM</td></tr>
</table>

Fig. 3. Chromosome of the proposed method.

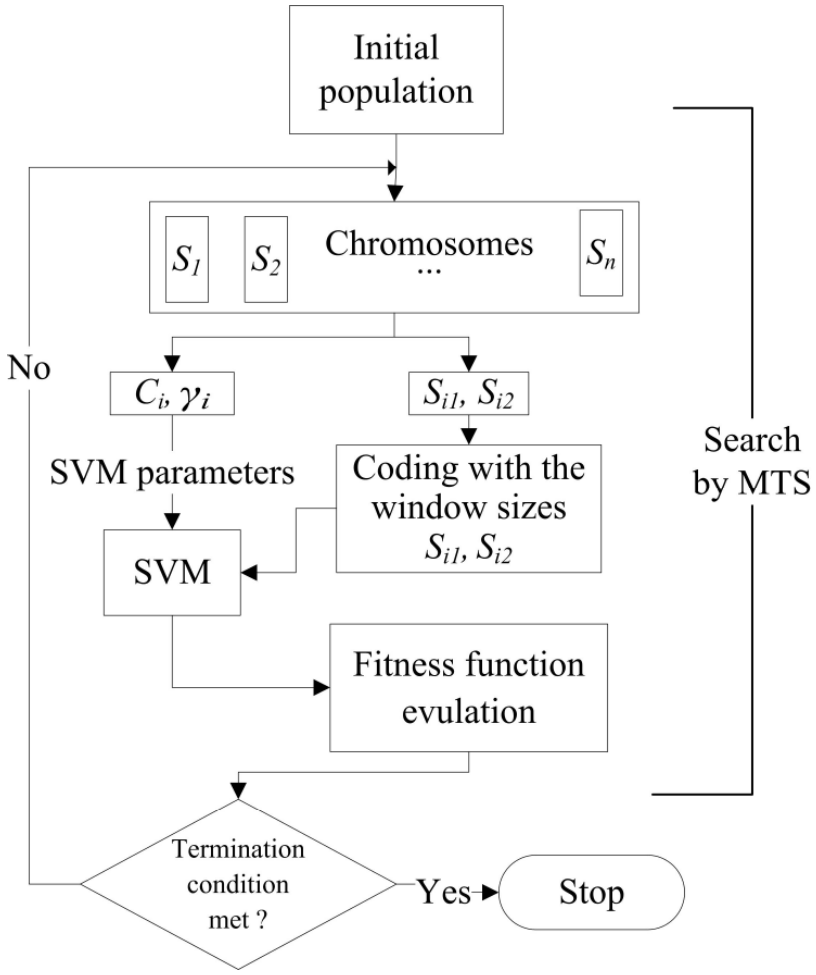

Fig. 4. Flowchart of the proposed method.

2.6 Maximum weight perfect matching

When a test protein is given, the connectivity probability of each cysteine pair will be computed by the trained SVM. A complete graph is then constructed with all cysteines as nodes and the weight associated with each edge is the disulfide connectivity probability of the pair of cysteines that are incident to this edge. The Gabow's algorithm [44] is then applied to find the maximum weight perfect matching. Because the Gabow's algorithm can only be applied with integer edge labels, the disulfide connectivity probability of the pair of cysteines is multiplied by 10000 and then truncated into an integer to represent the weight associated with the edge. This matching represents the prediction result of the disulfide connectivity pattern.

We set a probability threshold, when the disulfide connectivity probability of two cysteines is greater than the probability threshold, then there is a bond between the two cysteines, otherwise there is no bond between the two cysteines. For dataset SPX with 10-fold cross-validation, the results of parameter value selection by the proposed method are as follows: $\log_2 C = 7.4$ and $\log_2 \gamma = -4.6$ for SVM, the window sizes are 1 and 23 for the PSS and the PSSM. Moreover, the probability threshold is set to 0.22 for a bond to be existed between two cystiens. This value is determined empirically.

3 Experiment Results

In order to evaluate the performance of the prediction, two accuracy indices Q_P and Q_C are used:

$$Q_P = \frac{C_P}{T_P}$$

$$Q_C = \frac{C_C}{T_C}$$

where C_P is the number of proteins whose bonding patterns are correctly predicted; T_P is the total number of proteins in the test set; C_C is number of disulfide bridges that are correctly predicted and T_C is the total number of disulfide bridges in test proteins.

Tables 1 and 2 show the results for bridge classification and the prediction of the disulfide bonding pattern of the dataset SPX. For bridge classification, Cheng *et al*. [12] listed sensitivity and specificity instead of accuracy. From the definition of sensitivity and specificity, in general the value of accuracy lies between them. From Table 1, it is noted that the overall sensitivity is 52% for Cheng's method and the prediction accuracy (*Qc*) is 60% for our method. There is an increase of prediction accuracy from 52% to 60%.

As for the disulfide connectivity prediction, Cheng's method with true secondary structure (SS) and solvent accessibility (SA) information in the inputs has the accuracy rate 51%. And Cheng's method with predicted secondary structure (PSS) and predicted solvent accessibility (PSA) information in the inputs has the prediction accuracy 48% only. In this study, using predicted secondary structure in the inputs, the prediction accuracy is 54.5%. There is an increase of prediction accuracy from 48% to 54.5%.

Table 1. Bridge classification result for dataset SPX without the prior knowledge of the bonding state of cysteines.

# of bonds	Cheng et al. (2006)		This work
	Sensitivity	Specificity	Accuracy(Q_c)
1	71%	48%	72.9%
2	59%	60%	73.2%
3	55%	61%	69.7%
4	44%	48%	59.7%
5	32%	35%	42.0%
6	32%	36%	33.3%
7	29%	32%	27.1%
8	20%	22%	22.5%
9	44%	52%	44.4%
10	33%	36%	23.3%
12	38%	39%	58.3%
14	79%	85%	100.0%
16	13%	13%	15.6%
17	53%	60%	55.9%
25	32%	53%	20.0%
26	31%	51%	30.8%
All	52%	51%	60.0%

Table 2. Disulfide connectivity prediction result for dataset SPX without the prior knowledge of the bonding state of cysteines.

# of bonds	Cheng et al. (2006)		This work
	Q_p with SS and SA	Q_p with PSS and PSA	Accuracy(Q_p)
1	59%	59%	60.6%
2	59%	56%	65.9%
3	50%	47%	59.8%
4	34%	22%	36.4%
5~26	20%	13%	11.8%
All	51%	48%	54.5%

4 Conclusion

Recently, the progress in the prediction of the oxidation states of cysteines in protein sequence is significant. But for the prediction of the bonding pattern of cysteines, much research effort is still needed to improve the prediction accuracy. To these authors' knowledge, all previous approaches except those presented by Ferrè *et al*. [13] and Cheng *et al*. [12] assume that the oxidation states of cysteines were known in advance. To practically solve the prediction of the bonding pattern of cysteines, this assumption eventually should be removed. In this work, without the prior knowledge of the oxidation states of cysteines, by integrating the MTS and the SVM training to tune parameters of SVM and the window sizes of the PSS and the PSSM, the proposed method achieves the accuracy of 54.5% on the bonding pattern prediction, which improves the accuracy of 51% obtained by Cheng *et al*. [12].

References.

[1] C. C. Chuang, C. Y. Chen, J. M. Yang, P. C. Lyu, J. K. Hwang, Relationship between protein structures and disulfide-bonding patterns, *Proteins*, Vol. 55, 2003, pp.1-5.

[2] H. W. T. van Vlijmen, A. Gupta, L. S. Narasimhan, J. Singh, A novel database of disulfide patterns and its application to the discovery of distantly related homologs, *J. Mol. Biol.*, Vol. 335, 2004, pp. 1083-1092.

[3] W. Antuch, P. Guntert, M. Billeter, T. Hawthorne, H. Grossenbacher, K. Wuthrich, NMR solution structure of the recombinant tick anticoagulant protein (rTAP), a factor Xa inhibitor from the tick Ornithodoros moubata, *FEBS Letters*, Vol. 352, 1994, pp. 251-257.

[4] L. C. Loredana, E. B Steven, J. P. H. Tim, C. Chothia, A. G. Murzin, SCOP database in 2002: refinements accommodate structural genomics, *Nucleic Acids Res.* Vol. 30, 2002, pp.264-267.

[5] A. Fiser, I. Simon, Predicting the oxidation state of cysteines by multiple sequence alignment, *Bioinformatics*, Vol. 16, 2000, pp. 251-256.

[6] P. Fariselli, P. Riccobelli, R. Casadio, Role of evolutionary information in predicting the disulfide-bonding state of cysteine in proteins, *Proteins*, Vol. 36, 1999, pp. 340-346.

[7] P. L. Martelli, P. Fariselli, L. Malaguti, R. Casadio, Prediction of the disulfide-bonding state of cysteines in proteins with hidden neural networks, *Protein Engineering*, Vol. 15, 2002, pp. 951-953

[8] Y. C. Chen, Y. S. Lin, C. J. Lin, J. K. Hwang, Prediction of the bonding states of cysteines using the support vector machines based on multiple feature vectors and cysteine state sequences, *Proteins*, Vol.55, 2004, pp. 1036-1042.

[9] P. Fariselli, R. Casadio, Prediction of disulfide connectivity in proteins, *Bioinformatics*, Vol. 17, 2001, pp. 957-964.

[10] P. Fariselli, P. L. Martelli, R. Casadio, A neural network base method for predictiing the disulfide connectivity in proteins, *In Damiani,E. et al., eds. Knowledge based Intelligent Information Engineering Systems and Allied Technologies KES 2002, IOS Press, Amsterdam*, 1, pp. 464-468.

[11] A. Vullo, P. Frasconi, Disulfide connectivity prediction using recursive neural networks and evolutionary information, *Bioinformatics*, Vol. 20, 2004, pp. 653-659.

[12] J. Cheng, H. Saigo, P. Baldi, Large-scale prediction of disulphide bridges using kernel methods, two-Dimensional recursive neural networks, and weighted graph matching, *Proteins*, Vol. 62, 2006, pp. 617-629.

[13] F. Ferrè, P. Clote, Disulfide connectivity prediction using secondary structure information and diresidue frequencies, *Bioinformatics*, Vol. 21, 2005, pp. 2336-2346.

[14] C. H. Tsai, B. J. Chen, H. H. Chan, H. L. Liu, C.Y. Kao, Improving disulfide connectivity prediction with sequential distance between oxidized cysteines, *Bioinformatics*, Vol. 21, 2005, pp. 4416-4419

[15] Y. C. Chen, J. K. Hwang, Prediction of disulfide connectivity from protein sequences, *Proteins*, Vol. 61, 2005, pp. 507-512.

[16] E. Zhao, H. L. Liu, C. H. Tsai, H. K. Tsai, C. H. Chan, C. Y. Kao, Cysteine separations profiles on protein sequences infer disulfide connectivity, *Bioinformatics*, Vol. 21, 2005, pp. 1415-1420.

[17] B. J. Chen, C. H. Tsai, C. H. Chan, C.Y. Kao, Disulfide connectivity prediction with 70% accuracy using two-level models, *Proteins*, Vol. 55(1), 2006, pp. 246-252.

[18] C. H. Lu, Y. C. Chen, C. S. Yu, J. K. Hwang, Predicting disulfide connectivity patterns, *Proteins*, Vol. 67, 2007, pp. 262-270.

[19] J. Song, Z. Yuan, H. Tan. T. Huber. K. Burrage, Predicting disulfide connectivity from protein sequence using multiple sequence feature vectors and secondary structure, *Bioinformatics*, Vol. 23, 2007, pp. 3147-3154.

[20] R. Rubinstein, A. Fiser, Predicting disulfide bond connectivity in proteins by correlated mutations analysis, *Bioinformatics*, Vol. 24, 2008, pp. 498-504.

[21] C. C. Chang, C. J. Lin, LIBSVM: a library for support vector machines, 2001, Retrieved from http://www.csie.ntu.edu.tw/~cjlin/libsvm

[22] S. Idicula-Thomas, A. J. Kulkarni, B. D. Kulkarni, V. K. Jayaraman, P. V. Balaji, A support vector machine-based method for predicting the propensity of a protein to be soluble or to form inclusion body on overexpression in Escherichia coli, *Bioinformatics*, Vol. 22, 2006, pp. 278-84.

[23] P. M. Kasson, J. B. Huppa, M. M. Davis, A. T. Brunger, A hybrid machine-learning approach for segmentation of protein localization data, *Bioinformatics*, Vol. 21, 2005, pp. 3778-86.

[24] M. E. Mavroforakis, H. V. Georgiou, N. Dimitropoulos, D. Cavouras, S. Theodoridis, Mammographic masses characterization based on localized texture and dataset fractal analysis using linear, neural and support vector machine classifiers, *Artif Intell Med*, Vol. 37, 2006, pp. 145-62.

[25] C. W. Hsu, C. J. Lin, A comparison of methods for multiclass support vector machines, *IEEE Trans Neural Network*, Vol. 13, 2002, pp. 415-425.

[26] M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. S. Furey, J. M. Ares, et al., Knowledge-based analysis of microarray gene expression data using support vector machines, *PNAS*, Vol. 97, 2000, pp. 262–267.

[27] D. DeCoste, B. Schuolkopf, Training invariant support vector machines, *Machine Learning* Vol. 46, 2002, pp. 161-190.

[28] Y. Lecun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, *et al.*, Comparison of learning algorithms for handwritten digit recognition, *International Conference on Artificial Neural Networks*, 1995, pp.53-60.

[29] J. Tong, P. Jiang and Z. Lu, RISP: A web-based server for prediction of RNA-binding sites in proteins, *Computer Methods and Programs in Biomedicine*, Vol. 90, 2008, pp. 148-153.

[30] Y. Qi, F. LIN, K. K. Wong, A Networked multi-class SVM for protein secondary structure prediction, *WSEAS International Conference on: Mathematical Biology and Ecology (MABE '05)*, Udine, Italy, January 20-22, 2005.

[31] M. Brown, W. Grundy, D. Lin, N. Cristianini, C. Sugnet, M. Ares, D. Haussler, Support vector machine classification of microarray gene expression data, *Technical Report UCSCCRL-99-09*, University of California, Santa Cruz, 1999.

[32] C. H. Yeang, S. Ramaswamy, P. Tamayo, S. Mukherjee, R. M. Rifkin, M. Angelo, M. Reich, E. Lander, J. Mesirov, T. Golub, Molecular classification of multiple tumor types, *Bioinformatics*, Vol. 17, 2001, pp.316S-322S.

[33] X. P. Zhang, R. Gu, Electrical energy consumption forecasting based on cointegration and a support vector machine in China, WSEAS *Transactions on* Mathematics, Issue 12, Vol. 6, 2007, pp. 878-883.

[34] H. Xiao, X. Zhang, Comparison studies on classification for remote sensing image on data mining method, *WSEAS Transactions on Computers*, Issue 5, Vol. 7, 2008, pp. 552-558.

[35] G. Pajares, Support vector machines for shade identification in urban areas, *WSEAS Transactions on Information Science and Applications*, Vol. 2, No. 1, 2005, pp. 38-41.

[36] S. Theodoridis, K. Koutroumbas, *Pattern Recognition. 2nd edn. Academic Press, San Dieago*, 2003.

[37] N. Cristianini, J. Shawe-Taylor, An introduction to support vector machines and other kernel-based methods, C*ambridge University Press*, Cambridge UK. 2000.

[38] L. Y. Tseng, C. Chen, Multiple trajectory search for large scale global optimization, *Proceedings of the 2008 IEEE Congress on Evolutionary Computation CEC 2008*, pp. 3057-3064.

[39] L. Y. Tseng, C. Chen, Multiple trajectory search for multiobjective optimization, *Proceedings of 2007 IEEE Congress on Evol. Comput.*, 2007, pp. 3609-3616.

[40] C. Sander, R. Schneider, Database of homology-derived protein structures and the structural meaning of sequence alignment, *Proteins*, Vol. 9, 1991, pp. 56-68.

[41] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, D. J. Lipman, Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, *Nucleic Acids Res.*, Vol. 25, 1997, pp. 3389-3402.

[42] D. T. Jones, Protein secondary structure prediction based on position-specific scoring matrices, *J. Mol. Biol.*, Vol. 292, 1999, pp. 195-202.

[43] C. C. Chang, C. J. Lin, LIBSVM : a library for support vector machines, 2001.

[44] H. N. Gabow, Implementation of algorithms for maximum matching on nonbipartite graphs, *Phd Thesis, Stanford University*, CA. 1973.