JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 23, Number 00, 2016
© Mary Ann Liebert, Inc.
Pp. 1–9
DOI: 10.1089/cmb.2015.0178

Research Article

A Complex Prime Numerical Representation of Amino Acids for Protein Function Comparison

DUO CHEN,[1] JIASONG WANG,[2] MING YAN,[3] and FORREST SHENG BAO[4]

ABSTRACT

Computationally assessing the functional similarity between proteins is an important task of bioinformatics research. It can help molecular biologists transfer knowledge on certain proteins to others and hence reduce the amount of tedious and costly benchwork. Representation of amino acids, the building blocks of proteins, plays an important role in achieving this goal. Compared with symbolic representation, representing amino acids numerically can expand our ability to analyze proteins, including comparing the functional similarity of them. Among the state-of-the-art methods, electro-ion interaction pseudopotential (EIIP) is widely adopted for the numerical representation of amino acids. However, it could suffer from degeneracy that two different amino acid sequences have the same numerical representation, due to the design of EIIP. In light of this challenge, we propose a complex prime numerical representation (CPNR) of amino acids, inspired by the similarity between a pattern among prime numbers and the number of codons of amino acids. To empirically assess the effectiveness of the proposed method, we compare CPNR against EIIP. Experimental results demonstrate that the proposed method CPNR always achieves better performance than EIIP. We also develop a framework to combine the advantages of CPNR and EIIP, which enables us to improve the performance and study the unique characteristics of different representations.

Key words: CPNR, EIIP, numerical representation, protein, sequence comparison.

1. INTRODUCTION

A PROTEIN SEQUENCE IS A SEQUENCE COMPOSED FROM 20 amino acids. One important task of bioinformatics research is computationally analyzing the functional similarity of protein sequences (Wen et al., 2005; Vacic et al., 2007). Properly representing amino acids is important to the accurate comparison of protein functions. Compared to symbolic representation of amino acids, numerically representing amino acids can largely expand our ability to analyze them. For example, resonant recognition model (RRM) (Cosic 1994; Keković et al., 2010), a widely adopted method to compare functional similarity between two proteins,

[1]School of Biological Science and Medical Engineering, Southeast University, Nanjing, China.
[2]Department of Mathematics, Nanjing University, Nanjing, China.
[3]Department of Biotechnology & Pharmaceutical Engineering, Nanjing Tech University, Nanjing, China.
[4]Department of Electrical and Computer Engineering, University of Akron, Akron, Ohio.

makes use of digital signal processing (DSP) techniques and hence requires amino acids to be represented numerically.

Among the state-of-the-art methods, electron-ion interaction potential (EIIP) is a *de facto* numerical representation approach (Cosic, 1994; Keković et al., 2010; Cosic and Pirogova, 2007). However, it only works well on identifying proteins of the same type of functions (Cosic, 1994; Cosic and Pirogova 2007), suffering from degeneracy (Yau et al., 2003) issues that two functionally different proteins may get very similar or even the same representation. The degeneracy issue is due to its design in which some amino acids are mapped to close or even same numbers.

To tackle this challenge, in this article, we propose a complex prime numerical representation (CPNR) of amino acids. The motivation behind CPNR is to consider the number of codons of amino acids. We observe a great similarity between the number of codons of amino acids and a differential pattern of prime numbers. Such observation inspires us to develop the new representation based on prime numbers. To make the representation more biologically meaningful, prime numbers are mapped into complex domain. Compared with previous work, CPNR has no degeneracy problem, and the numbers in CPNR are relatively independent from each other, which means that one number cannot be obtained from another by addition, multiplication, or exponentiation with a real number.

To validate the effectiveness of CPNR, we compare its accuracy against EIIP (Cosic, 1994). Results show that CPNR achieves higher accuracy of 69.18% than EIIP. We further develop a framework to combine the advantages of CPNR and EIIP. Modeled as an optimization problem, the framework achieves an accuracy of 84.25% and 81.53% when solved by particle swarm optimization (PSO) (Birge, 2003) and support vector machine (SVM) (Lu et al., 2004; Cheng et al., 2006), respectively.

2. DATA

Both protein sequences and annotations about their functions are from UniProt (Universal Protein Resource) where 520 protein sequences from 7 most systematically researched functional groups form our dataset (Table 1). Hence, there are $\binom{520}{2} = 134940$ pairs of proteins in total, including 29759 same-function pairs and 105181 different-function pairs.

3. METHODS

3.1. Complex prime numerical representation (CPNR)

A protein sequence is usually represented as a symbolic sequence based on the alphabet set $\Gamma = \{M, K, F, Y, P, C, T, H, V, L, Q, S, A, N, G, R, I, D, E, W\}$. Each of the element in the alphabet set Γ is an amino acid. A protein sequence could be transferred into a numerical sequence for convenience in digital signal processing. Electron-ion interaction potential (EIIP, Table 2) is the *de facto* approach for comparing protein functions numerically (Cosic, 1994; de Trad et al., 2000). However, EIIP suffers from its degeneracy: different amino acids can be mapped to close or same numbers (C and S, L and I).

To address degeneracy, in this article, we propose a prime number-based approach where we make use of the pattern on the numbers of amino acid codons. A codon is a tuple of three nucleic acids that can be encoded into one amino acid in gene expression. One amino acid can be encoded from more than one codon. This makes amino acid sequences more fault-tolerant for point mutations (Lehmann and Libchaber, 2008).

Table 1. Seven Protein Groups

ID	*Protein name*	*Gene name*	*Seq. No.*
1	Cytochrome C	CYC	34
2	Hemoglobin subunit beta	HBA	199
3	Lysozyme C	LYZ	36
4	Myoglobin	MB	100
5	Major prion protein	PRNP	62
6	Superoxide dismutase [Cu-Zn]	SOD1	35
7	Somatotropin	GH	54

Table 2. EIIP of 20 Amino Acids

Amino acid	*EIIP*	*Amino acid*	*EIIP*
M	0.0823	Q	0.0761
W	0.0548	S	0.0829
F	0.0946	A	0.0373
Y	0.0516	N	0.0036
P	0.0198	G	0.005
C	0.0829	R	0.0959
T	0.0941	I	0
H	0.0242	D	0.1263
V	0.0057	E	0.0058
L	0	K	0.0371

Table 3. Number of Codons for 20 Amino Acids

Number of codons	*Amino acid*
1	M, W
2	F, Y, C, H, Q, N, D, K, E
3	I
4	P, T, V, A, G
6	L, S, R

We notice that the numbers of amino acid codons share a great similarity with the differences between prime numbers in the interval [2, 67] (Tables 3 and 4). Table 3 shows the numbers of codons for all 20 amino acids, while Table 4 shows the difference between each noncomposable (i.e., 1 and prime) number in interval [1, 67] and the next one.

Take the five amino acids P, T, V, A, and G as an example. Each of the five amino acids has four codons. Correspondingly, there are five prime numbers (7, 13, 19, 37, and 43) in the interval [2, 67], each of which is less than the next noncomposable number by 4. Such similarity inspires us to develop a representation of amino acids based on prime numbers.

Table 4. Differential Pattern Between Each Non-composable Number and the Next

Difference with the next non-composable number	*Non-composable (1 or prime) numbers*
1	{1, 2}
2	{3, 5, 11, 17, 29, 41, 59}
3	$\emptyset$
4	{7, 13, 19, 37, 43}
6	{23, 31, 47, 53, 61}

Table 5. Mapping $f : \Gamma \rightarrow Q$

amino acid	*number*	*amino acid*	*number*
M	1	Q	29
W	2	S	31
F	3	A	37
Y	5	N	41
P	7	G	43
C	11	R	47
T	13	I	53
H	17	D	59
V	19	E	61
L	23	K	67

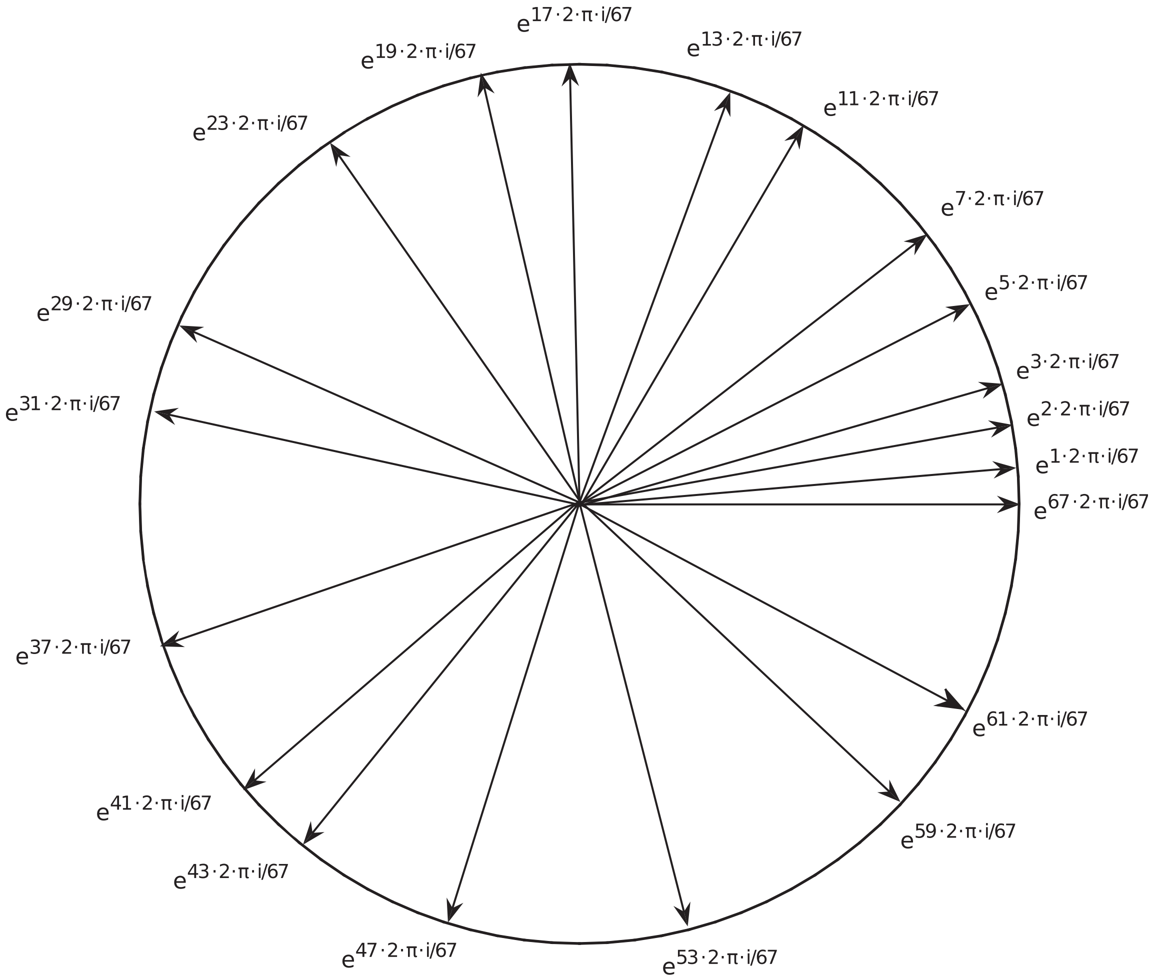
$e^{17\cdot 2\cdot\pi\cdot i/67}$
$e^{13\cdot 2\cdot\pi\cdot i/67}$
$e^{19\cdot 2\cdot\pi\cdot i/67}$
$e^{11\cdot 2\cdot\pi\cdot i/67}$
$e^{23\cdot 2\cdot\pi\cdot i/67}$
$e^{7\cdot 2\cdot\pi\cdot i/67}$
$e^{5\cdot 2\cdot\pi\cdot i/67}$
$e^{29\cdot 2\cdot\pi\cdot i/67}$
$e^{3\cdot 2\cdot\pi\cdot i/67}$
$e^{2\cdot 2\cdot\pi\cdot i/67}$
$e^{31\cdot 2\cdot\pi\cdot i/67}$
$e^{1\cdot 2\cdot\pi\cdot i/67}$
$e^{67\cdot 2\cdot\pi\cdot i/67}$
$e^{37\cdot 2\cdot\pi\cdot i/67}$
$e^{61\cdot 2\cdot\pi\cdot i/67}$
$e^{41\cdot 2\cdot\pi\cdot i/67}$
$e^{59\cdot 2\cdot\pi\cdot i/67}$
$e^{43\cdot 2\cdot\pi\cdot i/67}$
$e^{47\cdot 2\cdot\pi\cdot i/67}$
$e^{53\cdot 2\cdot\pi\cdot i/67}$

FIG. 1. The 20 complex numbers chosen in CPNR divide a unit circle on a complex plane.

Based on the observation above, we construct a set Q from prime numbers between 2 and 67 and the number 1. Then we define a one-to-one mapping $f : \Gamma \rightarrow Q$ such that the 20 amino acids are mapped to 20 noncomposable numbers in Q (Table 5).

In mapping f, the difference between two encoding numbers can be quite large. For example: 67–2 = 65. To avoid certain amino acids having much higher amplitudes than the others, we further map the 20 numbers in Q onto a unit circle (see Fig. 1) in complex domain.

Hence, a protein $\mathbf{a} = [a_1, \cdots, a_n]$ is represented by an array of complex numbers $\mathbf{x} = [x_1, \ldots, x_n]$ where

$$x_j = e^{-\frac{2\pi f(a_j)}{67}}, \forall j \in [1..n].$$

We call the complex number array $\mathbf{x}$ the complex prime numerical representation (CPNR) of the protein $\mathbf{a}$.

3.2. Functional comparison of proteins in resonant recognition model (RRM)

Here we briefly go over the RRM model (Cosic, 1994), which is the *de facto* method for functional comparison of proteins based on numerical representation in Algorithm 2. RRM begins by converting two symbolic amino acid sequences **a** and **b** into their numerical representation **x** and **y**. Then, RRM computes their respective Signal-to-Noise Ratios (SNRs), denoted as $\mathbf{SNR_x}$ and $\mathbf{SNR_y}$. Next, RRM computes their cross spectrum **C** and finds the digital frequency point **q** corresponding to the peak in cross spectrum. With the frequency point **q**, RRM goes back to the two SNRs and finds their values at frequency q. Denote those two values as SNR_{xq} and SNR_{yq}. If both of them are above a threshold t, RRM will consider that the two protein sequences are of the same function. Otherwise, different functions.

The steps to find SNR_{xq} and SNR_{yq} are abstracted into Algorithm 1.

Algorithm 1: SNR (**a**, **b**, *r*)

```
Input: two symbolically represented proteins a and b, and a symbol-to-number representation method r
       (e.g., EIIP or CPNR) for amino acids
1 Convert a and b into their numerical representation x and y, respectively.
2 Compute SNRs of x and y, $\mathbf{SNR}_x$ and $\mathbf{SNR}_y$, respectively.
3 Compute the cross spectrum $\mathbf{C} = [C_1, \cdots, C_m]$ .
4 Let
      $q = \arg\max_k C_k.$
5 return $(SNR_{xq}, SNR_{yq})$
```

Algorithm 2: RRM (**a**, **b**, *r*, *t*)

```
Input: two symbolically represented proteins a and b, a symbol-to-number representation method r
       (e.g., EIIP or CPNR), and a threshold t
1 $(SNR_{xq}, SNR_{yq}) =$ SNR (a, b, r)
2 if $SNR_{xq} > t$ and $SNR_{yq} > t$ then
3     return Same
4 else
5     return Different
```

3.3. Weighted RRM: Combining EIIP and CPNR for higher accuracy

From preliminary experiments, we notice that though CPNR performs better than EIIP, it still cannot give very high comparison accuracy. Hence, we propose a framework to combine the advantages of CPNR and EIIP representations.

3.3.1. Weighted SNR (wSNR).

Our solution to combining EIIP- and CPNR-based functional comparisons is to weight the SNRs obtained from both representations. It is illustrated in Algorithm 3.

Given two proteins **a** and **b**, two weights w_x and w_y, and a threshold t, if wSNR (**a**, **b**, w_x, w_y) $> t$, we say that the proteins are functionally same; otherwise, functionally different. This new framework that uses both numerical representations is called weighted RRM and is summarized in Algorithm 4.

Algorithm 3: wSNR (**a**, **b**, w_x, w_y)

```
Input: Two symbolically represented proteins a and b, two real number weights $w_x$ and $w_y$
1 Let $(SNR_{xq,\,\mathrm{EIIP}}, SNR_{yq,\,\mathrm{EIIP}}) =$ SNR (a, b, EIIP)
2 Let $(SNR_{xq,\,\mathrm{CPNR}}, SNR_{yq,\,\mathrm{CPNR}}) =$ SNR (a, b, CPNR)
3 Let $wSNR = w_x(SNR_{xq,\,\mathrm{CPNR}} + SNR_{yq,\,\mathrm{CPNR}}) + w_y(SNR_{xq,\,\mathrm{EIIP}} + SNR_{yq,\,\mathrm{EIIP}})$
4 return wSNR
```

Algorithm 4: wRRM (**a**, **b**, w_x, w_y, w_y t)

```
Input: two symbolically represented proteins a and b, two real number weights $w_x$ and $w_y$, and a threshold t
1 wSNR = wSNR (a, b, $w_x$,$w_y$)
2. if $wSNR > t$ then
3     return Same
4 else
5     return Different
```

3.3.2. Finding optimized parameters for weighted RRM.

Finding the proper values for w_x, w_y, and t can be modeled as an optimization problem. Denoting the comparison accuracy over a set A of proteins as a function $J(A, w_x, w_y, t)$, we can define the search for optimal weights w_x and w_y, and the threshold t as an optimization problem:

$$\begin{aligned} \max \quad & J(A, w_x, w_y, t) \\ s.t. \quad & -50 \leq w_x \leq 50, \ -50 \leq w_y \leq 50, \\ & 0 \leq t \leq 30. \end{aligned} \tag{1}$$

The boundaries for w_x, w_y, and t are chosen based on experience. The optimization problem in Equation 1 is solved by particle swarm optimization (PSO) (Eberhart and Shi 2001), an efficient optimization evolutionary algorithm. In our experiments, we use a set of proteins as the training set to find optimal parameters and use another set of proteins to test the accuracy. Cross-validation is used to test accuracy on different training and test sets.

3.3.3. Weighted RRM as a learning problem. The optimization problem in this paper could also be transferred into a binary classification problem (learning problem). In the learning problem, each pair of proteins has a 4-D feature vector, namely $SNR_{xq,\mathrm{CPNR}}$, $SNR_{yq,\mathrm{CPNR}}$, $SNR_{xq,\mathrm{EIIP}}$, and $SNR_{yq,\mathrm{EIIP}}$, and a binary label, ‘‘same’’ or ‘‘different.’’ Support vector machine (SVM) (Cortes and Vapnik, 1995) is used here to solve this learning problem for its generalization ability.

4. EXPERIMENTAL RESULTS

In order to empirically study the performance of CPNR, we compare it with EIIP. All proteins from the same group are considered as having the same function; Otherwise, different functions.

4.1. Metrics

We use the confusion matrix to measure the performance of functional comparison of proteins. Protein pairs having the same function is considered ‘‘positive.’’ Otherwise, it’s ‘‘negative.’’ Algorithm output has four possible outcomes of computational functional comparison: (1) True positive (*TP*); (2) false positive (*FP*); (3) true negative (*TN*); (4) false negative (*FN*). The metrics mentioned above are summarized in Table 6.

Four values are defined for later analysis of algorithm performance: (1) True positive rate (i.e., sensitivity), $TPR = \frac{TP}{TP+FN}$; (2) true negative rate (i.e., specificity), $TNR = \frac{TN}{FP+TN}$; (3) positive predictive value (i.e., precision), $PPV = \frac{TP}{TP+FP}$; and (4) negative predictive value, $NPV = \frac{TN}{FN+TN}$.

4.2. Results on individual representations

Results on using individual numerical representations are summarized in Table 7. Since the SNR threshold t contributes directly to the accuracy, we run tests on several different SNR thresholds. For the sake of space, we only report the results from the two best SNR thresholds, $t = 2$ and $t = 3$.

Table 7 shows the performance of the two numerical representations. CPNR has higher accuracy in both thresholds. As shown in the *TPR* column of Table 7, the two numerical representations perform well on same-function protein pairs. When using SNR threshold $t = 2$, CPNR and EIIP both reach *TPR* values above 95%.

However, both of them reach very low *TNR* (below 35%) when SNR threshold $t = 2$. Comparing the results of using two SNR thresholds, the two representations exhibit better overall performance when using SNR threshold $t = 3$.

Unfortunately, neither CPNR nor EIIP can achieve good accuracy (below 70%). This observation inspires us to develop the weighted RRM as discussed earlier in Section 3.3.

4.3. Results when combining representations

In Section 3.3, we develop a framework, weighted RRM, to combine the advantage of CPNR with EIIP. To evaluate the objective function that we established in Section 3.3.2, we use an exhaustive

TABLE 6. THE CONFUSION MATRIX

	Biologically same	*Biologically different*
Computationally same	True positive (TP)	False positive (FP)
Computationally different	False negative (FN)	True negative (TN)

Table 7. Overall Performances of CPNR and EIIP

Threshold (SNR)	*Representation*	*TPR*	*TNR*	*PPV*	*NPV*	*Accuracy*
$t=2$	CPNR (this work)	95.44%	32.81%	58.69%	87.80%	64.12%
	EIIP	98.15%	13.64%	53.19%	88.06%	55.89%
$t=3$	CPNR (this work)	75.37%	62.99%	67.07%	71.84%	69.18%
	EIIP	70.79%	63.67%	66.09%	68.50%	67.23%

cross-validation scheme. Each time, four groups of proteins are used as the training set, and the remaining three groups as the test set. No group is used in both training set and test set in any run. The number of proteins in each group is given in the right-most column of Table 1. Since each group has a small amount of samples, to avoid overfitting we carefully pick the combination of training set and test set such that the sizes of the two sets are balanced.

Results on weighted RRM (Algorithm 4) are given in the fourth column of Table 8. The accuracy of Algorithms 2 and 4 are listed in the same table for easy comparison. Boosting average accuracy to 84.25%, combining CPNR + EIIP shows significant performance improvement compared with using any numerical representation alone.

For the sake of space, we are not able to list weights for each run in the cross-validation. Weights for EIIP or CPNR converge to similar values (w_1: –0.92 for EIIP and w_2: 48.09 for CPNR) regardless of the training and test sets. The convergence shows that our approach for combining CPNR and EIIP is robust. In most cases, we observe that $w_2 > w_1$, meaning that CPNR is given a higher trust level than EIIP by our PSO-based optimization. This observation in turn supports that CPNR is better than EIIP.

As mentioned earlier, weighted RRM can be considered as a supervised learning problem as well. Besides Algorithm 4, we also use SVM as the classifier for the comparisons in this work (the SVM approach, Section 3.3.3). The result is given in the last column of Table 8. The overall accuracy of RBF-SVM is 81.53%, which is very similar to that of the PSO-based approach above.

5. DISCUSSION

5.1. On SNR threshold

In our experiments, we notice that the SNR threshold t affects the accuracy greatly. In Section 4.2, we compare the results of using two different SNR thresholds in Algorithm 2, respectively. When SNR threshold $t=2$, there is a big difference between *TPR/NPV* and *TNR/PPV* (high *TPR/NPV* and low *TNR/PPV*) for both numerical representations. Both representations perform well in the comparisons in same functional pairs (90% or even higher). However, both of them make too many false negative decisions for protein pairs of different functions. The highest *TNR* of the two representations is only 32.81% while the highest *PPV* is only 58.69% (both achieved by CPNR). The low *TNR* and *PPV* values lead to low accuracy, the highest of which is 64.12% when using CPNR.

Table 8. Result Summarization for Algorithm 2 and Algorithm 4

	Algorithm 2 EIIP	*Algorithm 2 CPNR*	*Algorithm 4*	*SVM*
TPR	70.79%	75.37%	82.08%	79.18%
TNR	63.67%	62.99%	86.41%	83.87%
PPV	66.09%	67.07%	85.79%	83.08%
NPV	68.50%	71.84%	82.83%	80.11%
PD	99.20%	106.18%	30.13%	36.18%
ND	28.07%	30.04%	8.52%	10.24%
Accuracy	67.23%	69.18%	84.25%	81.53%

Results of Algorithm 4 and SVM are indicated by cross-validation's average value.

When the SNR threshold increases from 2 to 3, there is an improvement of *TNR* and *PPV*. *TNR* increases for both numerical representations greatly (at least 20% improvement). No representation gives extremely low *TPR*, *TNR*, *PPV,* or *NPV*. We should also be aware that there is a side effect: While *TNR* and *PPV* increase with higher threshold, *TPR* and *NPV* both decrease. Various *SNR* thresholds have been tested in a series of pre-experiments and $t=2$ and $t=3$ are the two best ones.

5.2. *On algorithm evaluation*

Because proteins have different functional groups, any two proteins are more likely to be of different functions than of the same function. In such cases, it is very important to know the ratio or difference between testing outcome positive (i.e., *TP* + *FP*) and condition positive (i.e., *TP* + *FN*), as well as the ratio or difference between testing outcome negative (i.e., *TN* + *FN*) and condition negative (i.e., *TN* + *FP*). Based on this, we propose two new statistical metrics to measure the biases within positive and negative groups, respectively. We call them *PositiveDeviation* (*PD*) and *NegativeDeviation* (*ND*), which are defined as follows:

$$PD=\frac{|(TP+FP)-(TP+FN)|}{TP+FN}=\frac{|FP-FN|}{TP+FN},$$

$$ND=\frac{|(FN+TN)-(FP+TN)|}{FP+TN}=\frac{|FN-FP|}{FP+TN}.$$

In this paper, *PD* denotes the statistical difference between biological positive and computational positive, while *ND* denotes the statistical difference between biological negative and computational negative.

Table 8 summarizes values of all statistical measures on the performances of Algorithms 2 and 4. A good algorithm is expected to have higher values on *TPR*, *TNR*, *PPV*, *NPV*, and accuracy, but lower values on *PD* and *ND*.

According to Table 8, Algorithm 4 has improved *TNR* and *PPV* greatly than using individual representations in Algorithm 2. Compared to Algorithm 2, the statistics biases *PD* and *ND* decrease greatly by Algorithm 4. Lower *PD* and *ND* prove that the new algorithm can more precisely describe the distribution of protein sequences in group(s).

Also, Algorithm 4 overcomes the disadvantage of Algorithm 2, which is low accuracy on different functional groups. No matter how we change the representation and threshold, Algorithm 2 always give unacceptably low *TNR* and *PPV*. Though a higher *SNR* threshold gives better *TNR* and *PPV* for Algorithm 2, *TPR* and *NPV* will be inevitably decreased.

The final accuracy of Algorithm 4 based on PSO is 84.25%, almost 20% higher than that of Algorithm 2. Algorithm 4 gets much better overall performance, with *TPR*, *TNR*, *PPV*, and *NPV* all over 80%. It will give biologists credible protein comparison results and save precious time on benchwork.

6. CONCLUSION

Designing a numerical representation of amino acids is an important component toward applying numerical methods to compare the function similarities of two proteins. The state-of-the-art numerical representations of amino acids suffer from degeneracy due to their designs. Inspired by the similarity between the pattern of prime numbers and the number of codons of amino acids, we propose a new approach CPNR to the numerical representation of amino acids. Compared with EIIP, our method does not suffer from degeneracy and delivers significantly improved performance. We further introduce a framework to combine CPNR + EIIP, and promising performance gain is achieved. Future research could focus on new numerical representations. If more representations are considered in weighted RRM, similar to ensemble learning, it may achieve even better performance in different protein functional groups.

AUTHOR DISCLOSURE STATEMENT

The authors declare that no competing financial interests exist.

REFERENCES

Birge, B. 2003. PSOt—a particle swarm optimization toolbox for use with Matlab, 182–186. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, April 2003. IEEE, New York.

Cheng, J., Randall, A., and Baldi, P. 2006. Prediction of protein stability changes for single-site mutations using support vector machines. *Proteins Struct. Funct. Bioinform.* 62, 1125–1132.

Cortes, C., and Vapnik, V. 1995. Support-vector networks. *Mach. Learn.* 20, 273–297.

Cosic, I. 1994. Macromolecular bioactivity: Is it resonant interaction between macro-molecules? Theory and applications. *IEEE Trans. Biomed. Eng.* 41, 1101–1114.

Cosic, I., and Pirogova, E. 2007. Bioactive peptide design using the resonant recognition model. *Nonlin. Biomed. Phys.* 1, 1–11.

De Trad, C.H., Fang, Q., and Cosic, I. 2000. The resonant recognition model (RRM) predicts amino acid residues in highly conserved regions of the hormone prolactin (PRL). *Biophys. Chem.* 84, 149–157.

Eberhart, R.C., and Shi, Y. 2001. Particle swarm optimization: Developments, applications and resources, 81–86. In *Proceedings of the 2001 Congress on Evolutionary Computation*, 2001. IEEE, New York.

Keković, G., Raković, D., Tosić, B., et al. 2010. Quantum foundations of resonant recognition model. *Acta Phys Pol. A* 117, 756–759.

Lehmann, J., and Libchaber, A. 2008. Degeneracy of the genetic code and stability of the base pair at the second position of the anticodon. *RNA* 14, 1264–1269.

Lu, Z., Szafron, D., Greiner, R., et al. 2004. Predicting subcellular localization of proteins using machine-learned classifiers. *Bioinformatics* 20, 547–556.

Vacic, V., Uversky, V.N., Dunker, A.K., and Lonardi, S. 2007. Composition profiler: A tool for discovery and visualization of amino acid composition differences. *BMC Bioinform.* 8, 1–7.

Wen, Z.-N., Wang, K.-L., Li, M.-L., et al. 2005. Analyzing functional similarity of protein sequences with discrete wavelet transform. *Comput. Biol. Chem.* 29, 220–228.

Yau, S.S.-T., Wang, J., Niknejad, A., et al. 2003. DNA sequence representation without degeneracy. *Nucleic Acids Res.* 31, 3078–3080.

Address correspondence to:
Dr. Forrest Sheng Bao
Department of Electrical and Computer Engineering
University of Akron
Akron, Ohio, 44325

E-mail: forrest.bao@gmail.com

Prof. Ming Yan
Department of Biotechnology and Pharmaceutical Engineering
Nanjing Tech University
Nanjing, China, 211816

E-mail: yanming@njtech.edu.cn