

RCIS: A Common Interval Searching Algorithm for Amino Acid Sequences

Xiaolu Huang and Hesham Ali
Department of Computer Science
University of Nebraska at Omaha
Omaha, NE 68182-0116, USA
xiaoluhuang@mail.unomaha.edu

Abstract

Protein secondary structure prediction based merely on protein amino acid sequence is a very exciting fundamental challenge in protein structure study. Common interval searching provides a new angle for sequence pattern study. RCIS (Redundant element sequence Common Interval Searching) algorithm is designed specifically for protein amino acid sequence common interval searching.

1. Introduction

Recent developments in whole-genome sequencing have produced a huge amount of protein sequence data; unfortunately, protein structure determination technology (X-ray crystallography, NMR) has not kept up [1][2]. Protein structure prediction based purely on protein's amino acid sequence is a promising area of research, as it is known that nature has just such an algorithm. RCIS (Redundant element sequence Common Interval Searching) algorithm is a new approach that studies protein amino acid sequence information from a new angle, that is, through common interval (CI) searching. The ultimate goal of this algorithm is to understand the relationship between protein amino acid organization patterns and protein structure. The study is based on previous researches of CI searching among two or more permutations [3][4].

The common intervals (CI) for two permutations are a pair of intervals of the two permutations that have the same set of elements. In 2000, Uno and Yagiura developed an $O(n)$ time complexity RC (Reduced Candidate) algorithm that enumerated all CIs of two permutations [3]. The sequences the RC is capable of processing must comply with the following limitations: the two sequences must be permutations of each other and, as a permutation every element within the sequence must be unique.

P1: 1 2 3 4 5 6 7 8 9 10
P2: 7 9 8 2 4 5 3 10 6 1

Figure 1.1 P1 (3,5) and P2 (5,7) are a pair of CIs of two permutation P1 and P2.

RCIS is a CI searching algorithm that finds CIs within a protein amino acid sequence; it can be easily modified to handle CI searching between two or more sequences.

2. RCIS algorithm

The RCIS algorithm is designed to find all subsequences that form common intervals within a given sequence *Seq* of length *n* and with the following properties:

- $\forall a$ in *Seq*, $a \in E_{(Seq)}$; $E_{(Seq)} = \{e_1, e_2, \dots, e_m\}$; E is a set of finite *m* distinct elements;
- $\exists a$ in *Seq*, a could be redundant in *Seq*.

2.1 Data types and methods used in RCIS

The following are some data types and methods specifically designed for RCIS.

- **eCount [x] [y]**: is an object that stores the repeat times of all *m* elements in set $E_{(Seq)}$ from position *x* to *y*. Each array element represents the repeated times of an element *e*, $e \in E_{(Seq)}$. For subsequence *Seq* (4, 7) **vrgrt** in Figure 2.1, eCount of *Seq* (4, 7) - eCount [4] [7] is (0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0). The amino acid sequence represented in sequence of eCount integer array is {a, c, d, e, f, g, h, i, k, l, m, n, p, q, r, s, t, v, y, w}.
- **add (eCount eC, char e)**: adds 1 to an array element in eCount that represents *e*. In Figure 2.1, eCount [4] [8] = add (eCount [4] [7], a) results in eCount [4] [8] = {1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0};
- **equal (eCount eC₁, eCount eC₂)**: a function that returns true if every array element in eC₁ is equal to the corresponding array element in eC₂ and returns false otherwise. In Figure 2.1, as subsequence *Seq* (11, 14) is **gvtr**. eCount [11] [14] = (0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0); equal (eCount [11] [14], eCount [4] [7]) returns true;
- **element (x)**: returns the element symbol at position *x* of the tested sequence.

2.2 RCIS algorithm:

```

Line 1: initialize eCount [n][n];
Line 2: for x=1, ..., n-1 do
Line 3:   add (eCount [x][x], element (x));
Line 4:   for y=x+1, ..., n do
Line 5:     let eCount [x][y] = eCount [x][y-1];
Line 6:     add (eCount [x][y], element (y));
Line 7: for x1=1, ..., n-1 do
Line 8:   for y1=x1+1, ..., n do
Line 9:     int span = y1-x1;
Line 10:    for x2=1, ..., n-span do
Line 11:      if equal (eCount [x1][y1], eCount
Line 12:                [x2][x2+span]) then
Line 13:        add x1, y1 to CCList;
Line 14:        add x2, x2+span to CIList;
Line 14: print CIList.

```

Line 1 is to initialize eCount for all possible subsequences of *Seq*; lines 2 to 6 get all the eCount for all possible subsequence of *Seq*; Lines 7 to 14 is to do the eCount comparison between any equal length subsequences of *Seq*. In Line 9, span is an integer indicating the size of one subsequence, it is used to make sure that two subsequences are of the same length, and thus comparable. The time complexity of RCIS is $O(n^3)$. Figure 2.1 showed a pair of CIs of an amino acid sequences found by RCIS.

```

      1  2  3  4  5  6  7  8  9 10 11 12 13 14
Seq:  a  g  g  v  r  g  t  a  c  c  g  v  t  r
Figure 2.1. Seq (4, 7) and Seq (11, 14) are common
intervals to each other.

```

2.2 Some other issues of RCIS

The RCIS algorithm described above is just the outline. The following are some restrictions that are important to be included during RCIS implementation:

- Set integer L as the shortest length of a CI. In Figure 1, we are not interested in CIs of length 1, so $L=2$; CI (4, 7) has length 4; L set up is based on research questions;
- Adding a restriction that prevents two subsequences from intersecting will block a large number of unwanted CIs. In Figure 2, *Seq* (1, 7) and *Seq* (2, 8) are a pair of legitimate CIs according to the above algorithm outline. But they are unwanted for obvious reasons.

3. RCIS implementation and results

RCIS algorithm was coded in C++ and run under Linux. Four protein amino acid sequences with four different protein secondary structure features were tested for their inner amino acid CI groups. Table 3.1 provides a summary of the results.

Table 3.1 CI statistics for four amino acid sequences

AA #	Name	2 nd Struc	CI Size				
			2*	3	4	5	6
192	Myoglobin	α helix	50**	9	2	0	0
181	Chymotrypsin	β sheet	38	12	2	1	0
255	Papain	α and β	49	15	5	3	1

- * Indicated how many elements in a CI
- ** Shows number of groups of a certain CI size

4. Conclusion

The testing has shown that RCIS algorithm is applicable for CI searching for actual protein amino acid sequences. However, these CI results are not yet explainable biologically. Our explanation is that since in this preliminary CI searching application, we have arbitrarily used the amino acid name classification as our finite element set, it may not be the best element set. The study of protein folding pathways indicates that protein structure forming is more characterized by several features consistent with low free energy: dense packing of residues in the interior, satisfaction of hydrogen-bonding potential of polar groups, and burying of hydrophobic surface, etc. So an alternate element set needs to be designed to include all known chemical forces that play roles in protein structure forming [3], such as:

- Negatively charged versus positively charged amino acids
- Polar versus hydrophobic amino acids
- Small versus medium and large amino acids
- Hydrogen bond forming versus non-hydrogen bond forming amino acids

We are currently working on organizing all of this information into one finite element set.

5. References

- [1] Pierre Baldi, et al. "Bioinformatics the machine learning approach", The MIT Press (1998).
- [2] Arthur M. Lesk, "Introduction to Protein Architecture", Oxford University Press (2001).
- [3] T. Uno et al., "Fast Algorithms to Enumerate all common Intervals of Two Permutations", *Algorithmica* (2000) 26: 290-309.
- [4] S. Heber, J. Stoye. "Finding all Common Interval of k Permutations", *Proceedings of CPM 2001, Lecture Notes in Computer Science 2089(2001), 207-218.*