An Improved Branch and Bound Algorithm for Cyclopeptide Sequencing

Sangharsh Saini, Jagtar Singh, and Monika Saini

Abstract—The mass-production of antibiotics and medication has started an evolutionary race between antibiotics and bacteria. Pharmaceutical companies proved helpful to create new antibiotics, while pathogens developed a new level of resistance to these antibiotics. Growth of drug-resistant disease increases the challenge of searching for new, more effective antibiotics. The isolation and sequencing of cyclic peptide antibiotics is time-consuming and error-prone, compared with the linear peptides. Given these facts, there is a need for new tools to sequence cyclic non-ribosomal proteins (NRPs). In this paper we show how to improve the cyclic peptide sequencing method further to reduce its running time considerably in its expansion (branching) step. Our results suggest that instead of extending the k-mers over the full span of 18 to more than 100 amino acids, we could extend the k-mers just over the candidate amino acids measured in the first step of the cyclic peptide sequencing method. This strategy improves the running time and space requirement of the method significantly.

Index Terms — algorithms, amino acids, antibiotics, cyclopeptide sequencing, spectral convolution.

I. INTRODUCTION

Protein translation is carried out by a molecular machine called a ribosome. Nonribosomal peptides (NRPs) are a class of peptide secondary metabolites, usually produced by microorganisms like bacteria and fungi. Nonribosomal peptides are also found in higher organisms, but are thought to be made by bacteria inside these organisms [1]. Nobel laureate Edward Tatum in 1963 devised in an experiment for the production of tyrocidines and gramicidins peptides in Bacillus brevis that some yet unknown non-ribosomal mechanism must assemble these peptides and so Tatum reasoned that if he inhibited the ribosome, all protein production in Bacillus brevis should grind to a halt. To his amazement, all proteins did indeed shut down except for tyrocidines and gramicidins.

In 1969, Fritz Lipmann (another Nobel laureate) demonstrated that tyrocidines and gramicidins are non-ribosomal peptides (NRPs), synthesized not by the ribosome, but by a giant protein called NRP synthetase. This enzyme pieces together antibiotic peptides without any reliance on RNA or the genetic code [1]. Each nonribosomal peptide synthetase can synthesize only one type of peptide. Nonribosomal peptides often have a cyclic or branched structure. We now know that every NRP synthetase assembles peptides by growing them one amino acid at a time. Unfortunately, the cyclopeptide sequencing remains a difficult task today.

II. RELATED WORK

One of the techniques for sequencing cyclic peptides is by using a Branch and Bound algorithm. Since NRPs do not adhere to the Central Dogma, any method we use to sequence them cannot rely on genome analysis, which brings us back to where we started. What makes sequencing these peptides even more difficult is that many NRPs (including tyrocidines and gramicidins) are cyclic [4]. The following methods are used to sequence cyclic peptides.

A. Step-1 Shattering molecules into pieces

The workhorse of peptide sequencing is the mass spectrometer, an expensive molecular scale that shatters molecules into pieces and then weighs the resulting fragments [2]. The mass spectrometer measures the mass of a molecule in daltons (Da); 1 Da is approximately equal to the mass of a single nuclear particle (i.e., a proton or neutron) [6]. The mass of the molecule is approximated by simply adding the number of protons and neutrons found in the molecule’s constituent atoms, which yields the molecule’s integer mass. For example, the amino acid Gly, which has chemical formula C2H3ON, has an integer mass of 57, since 2·12 + 3·1 + 1·16 + 1·14 = 57 [4]. Yet 1 Da is not exactly equal to the mass of a proton/neutron, and we may need to account for different naturally occurring isotopes of each atom when weighing a molecule. Table I shows the mass of 20 amino acids in Dalton (Da).

As mentioned in [4] Tyrocidine B1, which is represented by VKLFPWFNQY in the single-letter amino acid alphabet, has total mass 1322 Da (99 + 128 + 113 + 147 + 97 + 186 + 147 + 114 + 128 + 163 = 1322). The collection of all the fragment masses generated by the mass spectrometer [7] is called an experimental spectrum. The theoretical spectrum of a cyclic peptide, denoted Cyclospectrum(Peptide), is the collection of all of the masses of its subpeptides, in addition to the mass 0
B. Step-2 Convolution of the spectrum:

Cyclic peptide sequencing presents a practical barrier, since mass spectrometers generate spectra that are far from ideal [6]. They are characterized by having both false masses and missing masses. A false mass is present in the experimental spectrum but absent from the theoretical spectrum; a missing mass is present in the theoretical spectrum but absent from the experimental spectrum [5].

Table II demonstrates the theoretical and experimental spectrum of the cyclic peptide NQEL, which displays three missing masses. A false mass is present in the experimental spectrum (with false and missing masses) and add these elements to this spectrum. After adding these elements we have all candidate 1-mers to start with our algorithm. Given an experimental spectrum, we first compute the convolution of an experimental spectrum in order to capture the missing candidate amino acids. We then select the M most frequent elements between 57 and 200 to form a putative alphabet of amino acid masses. These amino acids are the candidate amino acids to start with our algorithm of Branch and Bound.

C. Step-3 The Branch and Bound Algorithm:

Solving the cyclopeptide sequencing problem with branch and bound algorithm [8] works by slowly building candidate linear subpeptides whose masses are consistent with the experimental spectrum. Given an experimental spectrum, initially we form a collection List of candidate 1-mers or candidate amino acids. At the next step, we expand List to contain all linear peptides of length 2. This process is continued, creating 18 new peptides of length k + 1 for each amino acid string Peptide of length k in List by appending every possible amino acid mass to the end of Peptide.

To prevent the number of candidate peptides from increasing exponentially, every time we expand List, we trim it by keeping only those linear peptides that remain consistent with the experimental spectrum [9]. We then check to see if any of these new linear peptides, when circularized, provides a solution to the Cyclopeptide Sequencing Problem.

More generally, brute force algorithms that enumerate all candidate solutions but discard large subsets of hopeless candidates by using various consistency conditions are known as branch-and-bound algorithms [10]. Each such algorithm consists of a branching step to increase the number of candidate solutions, followed by a bounding step. In this...
branch-and-bound algorithm for the Cyclopeptide Sequencing Problem, the branching step will extend each candidate peptide of length k into 18 peptides of length k + 1, and the bounding step will remove inconsistent peptides from consideration [11].

Given an experimental spectrum of a cyclic peptide, a linear peptide is consistent with Spectrum if every mass in its theoretical spectrum is contained in the spectrum [4]. If a mass appears more than once in the spectrum of the linear peptide, then it must appear at least that many times in Spectrum in order for the linear peptide to be consistent with Spectrum.

The key to this algorithm is that every linear subpeptide of a cyclic peptide is automatically consistent with Cyclospectrum(Peptide). Thus, to solve the Cyclopeptide Sequencing Problem for Spectrum, we can safely ban all peptides that are inconsistent with Spectrum from the growing List, which powers the bounding step described above [4].

We can define the branching step as Expand(List). Given the current collection of linear peptides List, define Expand(List) as a new collection containing all possible extensions of peptides in List by a single amino acid mass. The pseudocode for the branch-and-bound algorithm suggested in [5] is as follows:

Algorithm for Cyclopeptide sequencing

Step1: Start
Step2: List = {0-peptide}
Step3: while List is nonempty
Step4: List = Expand(List)
Step5: for each peptide, Peptide in List
Step6: if Cyclospectrum(Peptide) == Spectrum
Step7: output Peptide
Step8: remove Peptide from List
Step9: else if Peptide is not consistent with Spectrum
Step10: remove Peptide from List
Step11: end

The potential problem with this algorithm is that it may generate incorrect k-mers at intermediate stages (i.e., k-mers that are not subpeptides of a correct solution) [5]. Also, the algorithm attempts to extend the k-mers to the whole span of 18 amino acids by appending every possible amino acid mass to the end of Peptide, which is unnecessary in order to be consistent with the spectrum.

We now perform a detailed analysis of this algorithm for sequencing a cyclic peptide of 5 amino acids PVCPT with 5 iteration as an example. In this analysis we will assume that we have performed our step 2 as described above and have created the convolution of the experimental spectrum. So, we now have our candidate amino acids {P, V, C, and T} to start our algorithm. We will analyze this algorithm with both strategies. First, extending k-mers to the whole span of 18 amino acids by appending every possible amino acid mass to the end of peptide. Second, extending k-mers to the span of only candidate amino acids. We observe that with our second strategy, this algorithm takes much less time and space compared to the first strategy.

III. PROPOSED METHOD

Strategy 1:

First Iteration The algorithm first captures the amino acids (1-mers) as the candidates for the second iteration from the spectrum that are present in the 18 amino acid table and are consistent with the spectrum (or also present in the spectrum). In this case the candidate amino acids are {P, V, C, and T}. See Table IV A.

Second Iteration The algorithm next appends each of the 18 amino acid masses to each of the 1-mers above. The resulting List containing 4 · 18 = 72 peptides of length 2 are then trimmed to keep only the 10 candidate peptides that are consistent with Spectrum. See Table IV B.

Third Iteration The algorithm next appends each of the 18 amino acid masses to each of the 2-mers above. The resulting List containing 10 · 18 = 180 peptides of length 3 are then trimmed to keep only the 15 candidate peptides that are consistent with Spectrum. See Table IV C.

Forth Iteration The algorithm next appends each of the 18 amino acid masses to each of the 3-mers above. The resulting List containing 15 · 18 = 270 peptides of length 4 is then trimmed to keep only the 10 candidate peptides that are consistent with Spectrum and we generate 10 consistent 5-mers. See Table IV D.

Fifth Iteration In the final iteration, the algorithm next appends each of the 18 amino acid masses to each of the 4-mers above. The resulting List containing 10 · 18 = 180 peptides of length 5 is then trimmed to keep only the 10 candidate peptides that are consistent with Spectrum and we generate 10 consistent 5-mers. See Table IV E.

Strategy 2:

First Iteration The algorithm first captures the amino acids (1-mers) as the candidates for the second iteration from the spectrum that are present in the 18 amino acid table and are consistent with the spectrum (or also present in the spectrum).

Second Iteration The algorithm next appends each of the candidate amino acid masses to each of the 1-mers above. The resulting List containing 4 · 18 = 72 peptides of length 2 are then trimmed to keep only the 10 candidate peptides that are consistent with Spectrum. See Table IV A.

Third Iteration The algorithm next appends each of the candidate amino acid masses to each of the 2-mers above. The resulting List containing 10 · 4 = 40 peptides of length 3 are then trimmed to keep only the 15 candidate peptides that are consistent with Spectrum. See Table IV B.

Forth Iteration The algorithm next appends each of the candidate amino acid masses to each of the 3-mers above. The resulting List containing 15 · 4 = 60 peptides of length 4 are then trimmed to keep only the 10 candidate peptides that are consistent with Spectrum. See Table IV C.

Fifth Iteration In the final iteration, the algorithm next appends each of the candidate amino acid masses to each of the 4-mers above. The resulting List containing 10 · 4 = 40
peptides of length 5 are then trimmed to keep only the 10 candidate peptides that are consistent with spectrum and we generate 10 consistent 5-mers. See Table IV E.

All these linear peptides correspond to the same cyclic peptide PVCPT, thus solving the Cyclopeptide Sequencing Problem. We can verify that Cyclopeptide Sequencing Algorithm quickly reconstructs Tyrocidine B1.

### IV. Results

In the first strategy the algorithm attempts to extend the k-mers to the whole span of 18 amino acid weights, which is unnecessary in order to be consistent with the spectrum. This consumes a lot of time and space in order to create and store the unnecessary extended k-mers in every iteration.

Until now, we have assumed that 20 amino acids form the building blocks of proteins; these building blocks are called proteinogenic amino acids. There are actually two additional proteinogenic amino acids, called selenocysteine and pyrrolsine, which are incorporated into proteins by special biosynthetic mechanisms. Yet in addition to the 22 proteinogenic amino acids, NRP contains non-proteinogenic amino acids, which increase the number of possible building blocks for antibiotic peptides from 20 to over 100. This problem of creating unnecessary k-mers gets worse if we use a span of over 100 amino acids.

In the second strategy we suggest a way to reduce the time and space for this algorithm. In order to be consistent with the theoretical spectrum, we do not need to extend the k-mer to the full span of 18 or 100 amino acids. We could just extend the k-mers in each iteration for only candidate amino acids captured from our step 2. This does not create unnecessary extra k-mers in our extension or branching step of each iteration. Furthermore, it also shortens our bounding step as the algorithm has to make much less comparisons as compared to the full extension in first strategy. Also, this does not have the risk of losing candidate k-mers for the next iteration, because we get the equal number of candidate k-mers after branching and bounding in each iteration from both strategies, having significant time and space improvement in second strategy.

Table V shows the detail of seeds produced in 5 iterations for both strategies. As we can see, in first strategy the total no of seeds produced in 5 iterations are 702 and in second strategy it is just 156, which is 4.5 times less than the former strategy. So, we can see a significant 4.5 fold increase in algorithm performance. The improved method demonstrated here shows the results for a very small and simple experiment (four candidate nucleotides). If we consider more than 100 amino acids to sequence cyclopeptides, where much more seeds are produced in each iteration, the improved method performs even better and the performance gain is more than 4.5 fold.

Fig 1 shows plot diagram of the data observed in Table V. It can be noted that there is a very sharp increase in the complexity of first method (red line) in each iteration. Note that these observations are for a set of 18 amino acid masses. The complexity increases even more sharply if we use a set of more than 100 amino acid masses. On the other hand, second method (green line) has very low complexity increase as we proceed with our iterations. The complexity of this method does not increase too much even if we use a set of more than 100 amino acids because this method selects candidate amino acids and extend the seeds only for those candidate amino acids.

### Table V

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Seeds Produced</th>
<th>Candidates for the next Iteration</th>
<th>Seeds Produced</th>
<th>Candidates for the next Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>2nd</td>
<td>4×18=72</td>
<td>10</td>
<td>4×4=16</td>
<td>10</td>
</tr>
<tr>
<td>3rd</td>
<td>10×18=180</td>
<td>15</td>
<td>10×4=40</td>
<td>15</td>
</tr>
<tr>
<td>4th</td>
<td>15×18=270</td>
<td>10</td>
<td>15×4=60</td>
<td>10</td>
</tr>
<tr>
<td>5th</td>
<td>10×18=180</td>
<td>10</td>
<td>10×4=40</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>702</td>
<td>156</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Seeds produced in each iteration for both strategies
ACKNOWLEDGMENT

The authors would like to thank Dr. Pradeep Kumar
Department of Biology, Kurukshetra University, Kurukshtra
for his valuable help in the experiment.

REFERENCES


Jagtar Singh obtained his undergraduate degree in medical in 2001, Masters in Systems Biology in 2005 and Ph.D. in 2009 from Kurukshetra University, Kurukshtra, India. Currently, he is working as Assistant Professor in Department of Biology, Government College Naraingarh, India. He has published papers in many national and international journals. His research interests are molecular biology, recombinant DNA, genomics and rational medicine and sequencing new antibiotics.

Sangharsh saini became an IEEE member in 2008. He obtained his B.Sc. in 2005 from Kurukshetra University, Kurukshtra, India and M.Sc. in Computing in 2008 from Oxford Brookes University, Oxford, United Kingdom. He obtained his M.Sc. in Computer Science and Applications from Kurukshetra University, Kurukshtra, India in 2013. He qualified UGC-NET (National Eligibility Test for Lectureship) in Computer Science and applications in June 2013. Currently, he is working as Assistant Professor in Computer Science Department, Government College, Naraingarh, India. He also has accomplished an Industry Program in Bioinformatics from BII (Bioinformatics Institute of India), New Delhi in 2013. His research interests include artificial intelligence, cloud computing, machine learning, bioinformatics algorithms and python programming.

Monika Saini obtained her undergraduate degree in 2009 and M.A. in English in 2011 from Kurukshetra University, Kurukshtra, India. Currently, she is a student of M.Sc. in Computer Science at Department of Computer Science and Applications, Kurukshtra University. She became an IEEE student member in 2011. She has worked as System Administrator for three years at MNSN Technologies, India. She is now working as Teaching Assistant with Almighty LeaningS, Haryana, India. Her research interests include programming in python, bioinformatics algorithms and DNA fingerprinting.

Fig. 1. Complexity of two strategies in each iteration (18 Amino Acids)