CycloAnt: Sequencing Cyclic Peptides Using Hybrid Ants

Sujata Baral*
United International University
Dhaka, Bangladesh
sujata689@gmail.com

Swakkhar Shatabda
United International University
Dhaka, Bangladesh
swakkhar@cse.uiu.ac.bd

Mahmood A Rashid
The University of the South Pacific
Suva, Fiji
mahmood.rashid@gmail.com

ABSTRACT

Non ribosomal cyclic peptides have long been used as effective antibiotics in drug industry. Reconstruction of these peptide sequences extracted from natural elements remain a challenge till today. Introduction of mass spectrometry in this regard created scope for computer scientists to develop efficient algorithms to interpret a mass spectrum into a peptide sequence. Mass spectrum have a well known limitation of missing peaks which misleads the *de novo* sequencing process of cyclic peptides. In this paper, we present CycloAnt, a computational method that can reproduce correct cyclic amino acid sequence from distorted mass spectrum in an efficient way. We have used hybrid ants those construct the solution first and then try to improve quality of the solution using subsequent local search. We proposed a set of novel scoring functions which emphasize on the presence of sub-sequences of amino acids rather than approving equal contribution of all partial mass and precursor mass present in the spectrum. Moreover, we proposed a novel set of operators for the local search and refinement. Experiments show the effectiveness of our method on a standard set of benchmark and improvement over other methods.

CCS CONCEPTS

•**Computing methodologies** → *Discrete space search;*

KEYWORDS

Hybrid ants, local search, heuristic, operators

ACM Reference format:
Sujata Baral, Swakkhar Shatabda, and Mahmood A Rashid. 2017. CycloAnt: Sequencing Cyclic Peptides Using Hybrid Ants. In *Proceedings of GECCO '17, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3071178.3071239

1 INTRODUCTION

It was in 1939 when René J. Dubos [11] made an extra ordinary discovery by finding out that *Tyrothricin* collected from the soil microbe *Bacillus brevis* has the ability to stop the growth of *Streptococcus pneumoniae* and thus started the mass use of antibiotics. During World War II, Soviet hospitals utilized this antibiotic successfully. However, it was not until 1960's when Edward Tatum and Fritz Lipmann's discovery found that synthesis of all antibiotic peptides do not follow the usual genetic pathway via protein translation. These peptides are called non-ribosomal peptides (NRP). NRP cyclic peptides are widely used in Pharmaceutical industry to produce antibiotic, anti-fungal, anti-asthma, anti-cancer, anti-viral medicines [23]. Other than that, NRPs also have potentials to be utilized in crop protection, food industry and as bio-surfactants for environment restoration purpose [20].

In spite of having such important applications, finding amino acid sequences of cyclic antibiotic peptides is a difficult task even today. Nuclear magnetic resonance (NMR) and mass spectrometry (MS) are the two dominant methods used to sequence cyclic peptides. NMR methods are often time-consuming, error prone, and requires milligrams (large amount) of highly purified samples, which is almost impossible to collect from natural resources [23]. On the other hand, mass spectrometry is able to progress with minimal amount of materials and allows a certain range of impurity in the sample. Considering above situation, introduction of mass spectrometry to sequence peptide in 1990s was a breakthrough [29]. Cyclic (or linear) peptides are shattered into pieces into mass spectrometer and a spectrum of mass is produced. The basic job of peptide sequencing is - translate this spectrum into a sequence. One way of translation can be, looking for matching sequences into database, which is termed as peptide identification problem [2]. The other way is, doing everything from scratch, *de novo* sequencing problem. In *de novo* sequencing of cyclic NRP, we are only left with the spectrum since there are no such matches found in the database. Moreover, the spectrum of mass is ionized and sometimes erroneous due to missing and false peaks.

A number of computational approaches have been applied to solve the cyclic peptide sequencing problem. These include branch and bound algorithms [17], probabilistic models [14], dynamic programming [6, 19] etc. Ant Colony Algorithms have also been used recently to solve the cyclic peptide sequencing problem [16]. Most of these algorithms have some limitations. Firstly, they often deal only with a small set of amino-acid alphabet consisting of 20 amino-acids. Secondly, they perform poorly with erroneous spectrum, which is a common scenario in mass spectrometry. And thirdly these methods are not often scalable and do not perform well for larger sequences.

In this paper, we propose CycloAnt, a novel *de novo* approach to find the amino acid sequence of cyclic peptides from noisy MS spectrum data. CycloAnt is a hybrid algorithm that uses ants to construct cyclic peptide sequences from pheromone trail and subsequent local search to improve the quality of the solutions. We have proposed a set of novel scoring scheme or heuristic to guide the search in both phases of the algorithm. We have also proposed

*corresponding author.

GECCO '17, Berlin, Germany

DOI: http://dx.doi.org/10.1145/3071178.3071239

a novel set of operators for the local search phase. CycloAnt is capable of handling distorted spectrum with missing peaks, can handle a large amino-acid alphabet ranging from 57 to 200 and also work for relatively large sequences. Experimental results have shown that CycloAnt significantly improves over other state-of-the-art algorithms used in the literature. The algorithm was tested on standard benchmark peptides and synthetically prepared spectrum data.

Rest of the paper is organized as follows: Section 2 presents the problem model; Section 4 describes the basic ACO algorithm; Section 3 narrates the related work in the literature; Section 5 presents our proposed algorithm; the experimental results and discussion are given in Section 6 and the paper concludes in Section 7 with a summary and indication of possible future work.

2 CYCLIC PEPTIDE SEQUENCING PROBLEM

A protein or peptide is a polymer or chain of amino-acid monomers connected by peptide bonds. The peptide bond starts with a nitrogen (N) and ends with a carbon (C). The standard set of alphabets of genetic amino-acids is limited to 20. On the other hand, antibiotic peptides are built with extended alphabet set consisting more than hundreds of amino-acids. Let, Σ=$\{a_0, a_1, \cdots, a_{|\Sigma|}\}$ be the set of amino acids, each with molecular masses $m(a_i)$ ranging from m_{low} to m_{high}. A cyclic (linear) peptide P=$p_1, p_2, \cdots, p_n$ is a circular (linear) string of symbols drawn from the alphabet of amino-acids Σ. The length of the peptide P is n. In case of a circular peptide, a peptide bond connects the n^{th} amino acid p_n with the first one p_1. All the symbols $p_i \in P$ is drawn from the alphabet, i.e., $p_i \in \Sigma$. The sum of the mass of all the amino-acids in a peptide P is called the *parent mass*, denoted by $m(P)$. Formally,

$$m(P) = \sum_{i=1}^{n} m(p_i)$$

A partial N-*terminal* peptide of an original peptide P is denoted by $P_i = p_1, p_2, \cdots, p_i$ of mass, $m(P_i)$. Formally,

$$m(P_i) = \sum_{j=1}^{i} m(p_j)$$

Similarly, a partial C-*terminal* peptide of an original peptide P is denoted by $P_i^- = p_{i+1}, p_{i+2}, \cdots, p_n$ of mass, $m(P_i^-)$.

$$m(P_i^-) = m(P) - m(P_i) = \sum_{j=i+1}^{n} m(p_j)$$

A mass spectrometer breaks a peptide P at different peptide bonds that results into partial N-terminal and C-terminal peptides. For example, the peptide *NQELV* may be broken into N-terminal peptides N, NQ, NQE, NQEL and C-terminal peptides QELV, ELV, LV, V. However, this breaking into small peptides may result into fragments of lower masses called *ions* (water, amonia). We denote the set of *ions* as $\Delta = \{\delta_1, \delta_2, \cdots, \delta_k\}$. For tandem mass spectrometry, the theoretical spectrum $\Delta(P)$=$\{d_1, d_2, \cdots, d_r\}$ of a peptide P can be calculated by subtracting the masses of all possible ion types from the masses of all possible partial peptides of P. In this paper, we considered a model with an empty set of ions, i.e., $|\Delta|$=0.

Hence, the theoretical spectrum is a set containing the mass of all possible partial peptides. Formally,

$$\Delta(P) = \{d : d = m(P_i)\} \cup \{d : d = m(P_i^-)\}, 1 \leq i \leq n$$

An experimental spectrum, $S(P) = s_1, s_2, \cdots, s_q$ is a set of masses obtained in a tandem mass spectrometry experiment. The difference between theoretical and experimental spectrum of a peptide is that the experimental spectrum contains some fragmented small ion masses and chemical noise that includes missing masses of partial peptides and false masses. The *score* of a cyclic peptide P for a given experimental spectrum $S(P)$ is defined as the number of masses in $S(P)$ are equal to the masses in the theoretical spectrum $\Delta(P)$. Formally,

$$score(P, S(P)) = \sum_{k=1}^{|S(P)|} match(s_k, P) \quad (1)$$

Here, $match(s, P)$ is a binary function defined as follows:

$$match(s, P) = \begin{cases} 1, \text{ if } s \in \Delta(P) \\ 0, \text{ else} \end{cases} \quad (2)$$

Here, one of the difficulties is that the length of the peptide P might not be known. Let's assume the length is n. We will represent the peptide by the masses of the amino-acids. We denote this by $\mathcal{P} = \{m_1, m_2, \cdots, m_n\}$ such that $m_i = m(p_i)$. Now the problem becomes to construct a set of variables $\mathcal{P}$. Here, each variable m_i has a domain, $[m_{low}, m_{high}]$. In this paper, we are considering all the amino-acid masses from 57 to 200. Therefore, $m_{low} = 57, m_{high} = 200$.

Given the experimental spectrum $S(P)$, the task in *de novo* cyclic peptide sequencing is to find the peptide P that has maximum match or score to the experimental spectrum $S(P)$ defined in Equation 1.

3 RELATED WORK

Noiseless cyclic peptide sequence problem is equivalent to solving the beltway problem of mathematics [13]. Several variants of this problem is proved to be NP-Complete [27]. However, addition of noise makes the problem harder [31]. The linear equivalent of this problem is known as turnpike problem or the partial digest problem [9, 28]. Noisy version of this problem is NP-hard [8].

A large variety of algorithms have been applied to solve the *de novo* cyclic peptide sequencing problem [2]. As the length of a peptide grows, number of candidate peptides grows exponentially. Therefore the main focus of researches in this area has always been on the techniques where this growth rate can be curtailed and correct peptides are found. Branch and bound algorithm [3] has been long used in this regard with limitations of not working correctly for longer peptides and deviated spectrum. Mohimani et al. [22] experimented with multi stage mass spectrometry to gather more data about peptide fragments. They had gathered MS^3, MS^4 data in this purpose. The result of those experiments shows that the method was not tested for peptides over length 10 and also, algorithms had mistaken amino acid's position in few cases, returned wrong masses for Etamycin group. Ng et al. [23] employed spectral alignment algorithms after MS, which was further researched by [21] by implementing spectral network. Both of those papers limited their experiments on peptides, whose number of amino acids were within the range of 10. Furthermore, Ng et al. [23] had

reported of returning erroneous masses in few cases. Some of the work in literature [21] requires spectra of multiple related peptides. Fomin et al. [13] have addressed the challenge from a mathematical point of view. He reconstructed the problem as building a sequence of finite set of positive numbers from a finite set of circular partial sums of those positive numbers. He had built an efficient algorithm which can handle peptides of length 160 elements on real time spectrums. The only limitation of the whole concept was, the algorithm is based on the finite set of numbers of standard 20 amino acids.

Among other methods in literature are probabilistic models [14], dynamic programming [6], hidden markov models [12] etc. A recent work has proposed solutions for branched peptides [24]. Ant Colony Algorithms are also used recently to solve the cyclic peptide sequencing problem [16]. However, most of these algorithms are applicable only to smaller peptide sequences; often deal with only a small size of amino-acid alphabet and fail to address the chemical noise in the experiments. Our motivation is to overcome these difficulties and propose a novel and robust method.

4 ANT COLONY OPTIMIZATION

Ant colony optimization is a metaheuristic [10] inspired from ants in nature where ant's food searching skill to build a shortest path between nest and food source is used as an optimization technique. Ant colony provides a general algorithmic framework which can guide a search problem to converge in a promising region of search space where high quality solutions exist. In this method, a colony of artificial ants is used. Each of these ants creates different path from source to destination and indirectly cooperate with each other by laying pheromone on the path. Pheromone evaporation takes place with time, making the longer paths less favorable to ants. Depending on the accumulation of pheromone on the paths ants get biased to use one path over the others. Although a handful number of ants will always explore the longer paths which ensure exploration of new paths throughout the searching procedure. The probability calculation of ants to move from one node to another is defined by:

$$\pi_k(r,s) = \frac{\tau(r,s)^\alpha \eta(r,s)^\beta}{\sum_{u \epsilon M_k} \tau(r,u)^\alpha \eta(r,u)^\beta}$$

Here, $\pi_k(r,s)$ is the probability to move from node r to s by ant k, $\tau(r,s)$ is the amount of accumulated pheromone on that path and $\eta(r,s)$ is the heuristic that measures visibility of s from r. This mean, if s falls to the neighborhood of r and when positive, than the cost of reaching s from r. Here, $u \epsilon M_k$ is set of valid nodes where an ant can go from r to s and α is the degree of importance of pheromone while β is the degree of importance of visibility. Amount of pheromone on a segment i,j, passed by ant k is calculated by

$$\tau(i,j) \leftarrow \tau(i,j) + \Delta\tau^k$$

Here $\Delta\tau^k = Q.L_k$, L_k denotes the way of measuring quality of a solution (value of objective function). Q is a constant. A specific amount of pheromone ρ is evaporated after an interval. Pheromone evaporation is given as:

$$\tau_{i,j} = (1-\rho)\tau_{i,j}$$

Ant colony optimization technique has been successfully used to solve many optimization problems in the literature [4, 15, 26, 30].

5 OUR METHOD

In this section, we describe different components of CycloAnt. Figure 1 shows an block diagram of the proposed algorithm. As spectrum is generally prone to errors, our first step is to perform auto convolution (details given in Section 5.1) on the given spectrum and select a number of highest scored masses in a separate list. This number depends on the length of the spectrum or the peptide. Next, the pheromone matrix is initialized and we let the ants build peptides in a probabilistic manner using masses from former step. Details of ant's path building mechanism is given in Section 5.2. Those paths are further modified using a subsequent local search (details in Section 5.4) to achieve a better score. Pheromone is calculated for the modified peptides and stored in a global Pheromone matrix. Pheromone is laid on each of the amino acids explored by the ants. Pheromone evaporation also occurs at this stage to ensure exploration by ants. The search is guided by a set of heuristic functions described in Section 5.3. List of generated peptides form local search is compared with previous results and only highest scored peptides are saved into a global list. This whole procedure executes for a number of iterations and restarted once to reach better region of search space. Completion of these cycle triggers a refinement stage operation, where different operators are applied on the best scored peptides saved in the global list. A pseudo-code of CycloAnt is given in Algorithm 1. Rest of this section describes details of the components of the algorithm.

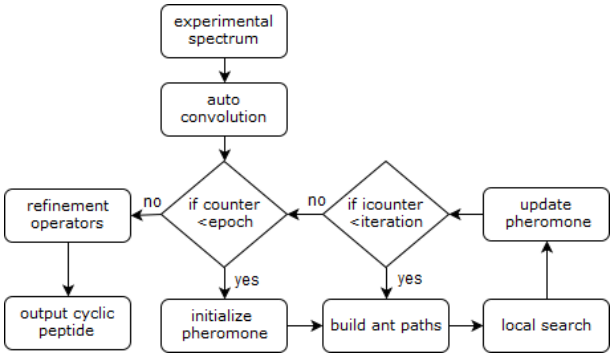

Figure 1: Architecture of CycloAntAlgorithm

5.1 Auto Convolution

Spectral convolution was first proposed in [25] to find similarities between two different but related spectrum. In auto-convolution [23] of a spectrum S with offset x is defined as the set of masses $s \in S$ such that $s-x \in S$. In this step, we take positive differences between all masses of given spectrum. The most frequent differences are stored in a list. We will refer to this list as *candidateMassList* in rest of this section. In forthcoming steps, amino acids from this list will work as building blocks for peptide generation. This self convolution procedure was applied and experimented by [7] and later treated in mass spectrometry [23] as a widely used mechanism to avoid false peaks of spectrum. An example auto-convolution for a 8 length peptide is given in Figure 2. Selected most frequent masses are colored in red.

Algorithm 1: CycloAnt(*spectrum*)

```
candidateMassList ← autoConvolution(spectrum)
highestScoredPeptideList={}
foreach epoch do
    peptideList={}
    initializePheromone()
    foreach iterations do
        foreach ants do
            newPeptide=constructPeptide(pheromone)
            peptideList.add(newPeptide)
        modifiedList=localSearch(peptideList)
        updatePheromone(modifiedList)
        updatehighestScoredPeptideList()
refinement(highestScoredPeptideList)
```

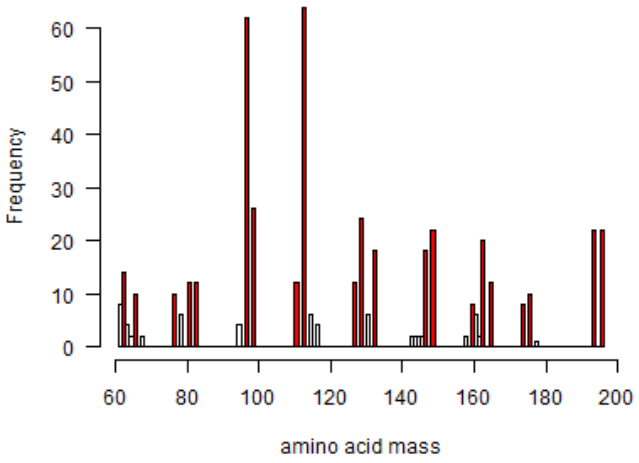

Figure 2: Auto convolution of an experimental spectrum for a peptide with length 8.

5.2 Guiding Ants

Once we have the list of amino acids from auto-convolution, we work with ants to move forward and construct solutions. We initialize a Pheromone matrix of $n \times l$ dimension, where l is the number of mass present in candidateMassList and n is possible peptide length. As stated earlier, it is one of the difficulties to estimate the length of the peptide. We assume that the parent mass $m(P)$ for a particular peptide is the highest mass in the spectrum and thus we estimate the sequence length n by $\frac{m(P)}{\xi}$, where ξ is the smallest mass in the *candidateMassList*. Therefore, each row in the pheromone matrix represents position of an amino-acid in the sequence and the column represents the different amino-acid symbols as selected from auto-convolution. Pheromone is stored in the cells of this matrix that guide the ants during peptide construction. Initially all the cells get equal pheromone value indicating each cell has got the same chance to be selected, i.e., each amino-acid symbol has the same chance to be selected at any position in the peptide. Ants read pheromone accumulation data from the matrix and create a probability list to select element for a position. As the number of iteration grows, influence of pheromone matrix in making decisions increases. In each iteration, pheromone matrix is updated according to the score of the generated peptides following Equation 1. Amount of pheromone to be laid on is calculated by the following equation,

$$\Delta\tau^k = Q.\zeta^k(P)$$

Here, Q is a constant and $\zeta^k(P)$ is the score of quality of the peptide P generated by ant k in that iteration. The best quality peptides will thus increase the pheromone levels of the amino-acids used to build that peptide. We have proposed novel heuristic function as quality function of the peptides generated. We have deliberately skipped considering heuristic here as score calculation of a peptide is a complex task and will significantly increase over all computational cost of the algorithm. As repetition of amino acid is allowed and all amino acids from candidateMassList is accepted, visibility checking is not required here.

5.3 Heuristic Functions

We have introduced two heuristic functions in our proposed algorithm. First one, *BreakDownScore* is employed at initial stage of the program, during peptide generation by ants and local search. The second one, $BreakDownScore^+$ is utilized at refinement stage to work with operators. In this section, we formally define these two heuristic functions. First of all we create two maps from any given peptide: *sub-sequence map* and *break-down map*.

A *sub-sequence map*, $\mathcal{M}_S(P)$ is a data structure similar to hash map, where each of the amino-acid present in the peptide P works as a key and as values it stores a list of all the partial peptides where that amino acid is present. Formally,

$$\mathcal{M}_S(P) : \mathcal{A} \rightarrow \Delta(P)$$

Here, $\mathcal{A}$ is the set of all amino-acid mass present in the peptide P and $\Delta(P)$ is the theoretical spectrum. For example, *sub-sequence map* for peptide 99-114-113-163 is {99:[213, 326, 489], 114:[227,390], 113:[276], 163:[]}, where 213=99 + 114, 326=99 + 114 + 113, 489=99 + 114 + 113 + 163.

The *break-down map* is the opposite mapping of the *sub-sequence map*. Here, mass of each partial peptide is mapped to the amino-acids masses present in the given peptide. Formally,

$$\mathcal{M}_B(P) : \Delta(P) \rightarrow \mathcal{A}$$

For the same peptide, *break-down map* is given as: {489:[99, 114, 113, 163], 390:[114, 113, 163], 326:[99, 114, 113], 276:[113, 163], 227:[114, 113], 213:[99, 114]}. Combination of *break-down map* and *sub-sequence map* is utilized in the local search, while *sub-sequence map* is utilized in calculating heuristic functions.

Now, we define the first function *BreakDownScore* with the help of *sub-sequence map*. *BreakDownScore* sums up contributions of a small value ϵ to each of the key (amino acids) in *sub-sequence map* found in the experimental spectrum and a value γ for each of the partial peptide mass as values in the *sub-sequence map* found in the

experimental spectrum. Formally,

$$BreakDownScore(P) = \sum_{i=1}^{n} f(i)\epsilon + \sum_{j=1}^{|\Delta(P)|} g(j)\gamma \quad (3)$$

In equation 3, $f(i) = 1$ if any mass $m_i \in \mathcal{A}$,$m_i \in S(P)$, $S(P)$ is the experimental spectrum, otherwise 0. Similarly, $g(j) = 1$ if for any, $m_i \in \mathcal{A}$, if there is a partial mass $m(P) \in \Delta(P)$ mapped by the *sub-sequence map* and $m(P) \in S(P)$, otherwise zero.

Likewise, $BreakDownScore^+$ not only sums up the presence of keys and values of the *sub-sequence map* in the experimental spectrum but also penalizes any absence or missing values. For the missing keys (amino-acid), it subtracts a value ϵ and for the missing values (mass of partial peptides) it subtracts another value γ.

$$BreakDownScore^+(P) = \sum_{i=1}^{n} f^+(i)\epsilon + \sum_{j=1}^{|\Delta(P)|} g^+(j)\gamma \quad (4)$$

Here, $f^+(i) = 1$ if any mass $m_i \in \mathcal{A}$,$m_i \in S(P)$, $S(P)$ is the experimental spectrum, otherwise -1. Similarly, $g^+(j) = 1$ if for any, $m_i \in \mathcal{A}$, if there is a partial mass $m(P) \in \Delta(P)$ mapped by the *sub-sequence map* and $m(P) \in S(P)$, otherwise -1.

5.4 Local Search

Local Search looks into the neighborhood for a better scored peptide produced by ants. A local search operator *mutate* is implemented to generate neighbor peptides or candidates. Mutation uses the *sub-sequence map* and *break-down map*. The definition of these maps have already been discussed in short in Section 5.3. Mutate retrieves key value pairs from *break-down map* one by one and processes them. For any key, it further searches into the subsequence map using the amino-acid masses stored in the value list. It now calculates a ratio λ which is the ratio of matched partial peptide mass from the value list of *sub-sequence map* with the experimental spectrum to the total number of partial peptides in that list. If that ratio is smaller than a threshold value th, *mutate* changes this value in the original peptide by selecting from one of the v values from candidateMassList. This greedy selection gives a hill-climbing nature to the local search.

Let's assume the first pair in the *break-down map* is 489: [99, 114, 113, 163]. Next, mutate will go to *sub-sequence map* and extract values [213,326,489] for 99. Then number of matches between spectrum S(P) and the list [213, 326, 489] is counted. If $\lambda \geq th$, mutate keeps 99 unchanged. Otherwise 99 will be replaced with three random masses taken from candidateMassList. This process continues for each element present in key value pair 489: [99, 114, 113, 163] and then for each pair of *break-down map*. Pseudo-code for the local search is described in Algorithm 2.

Algorithm 2: LocalSearch(*peptideList*)

```
modifiedPeptideList={}
for i ← 1 to len(peptideList) do
    upDatedPeptide=mutate(peptideList_i)
    modifiedPeptideList.add(upDatedPeptide)
return modifiedPeptideList
```

5.5 Refinement

After completion of peptide generation and modification in ant colony optimization and local search phase for a certain number of epochs, a refinement stage takes into action. Refinement cycle executes for a number of iterations decided by a parameter called ϕ. At this step, local search utilizes six operators namely: swap, transposition, mutate, fusion, fission and fusion$^+$ for neighborhood generation. All these operators except mutate were introduced after observation of the peptides created in the initial greedy stage. At refinement phase, for each candidate peptide, local search randomly selects an operator and applies. It accepts the resultant peptide if it comes with a higher score than the original candidate peptide. The behavior is similar to that of randomized hill-climbing. Our first operator, swap interchanges position of two amino acids in a peptide. For a given peptide representation, $\mathcal{P}$={m_1, m_2, $\cdots$, m_{i-1}, m_i, m_{i+1}, $\cdots$, m_{j-1}, m_j, m_{j+1}, m_n}, a swap of amino acids at position i and j would transform the peptide into $\mathcal{P}'$={m_1, m_2, $\cdots$, m_{i-1}, m_j, m_{i+1}, $\cdots$, m_{j-1}, m_i, m_{j+1}, m_n}. Transposition selects an element from position i and then place it after another position j in peptide. For a given peptide representation, $\mathcal{P}$={m_1, m_2, $\cdots$, m_{i-1}, m_i, m_{i+1}, $\cdots$, m_{j-1}, m_j, m_{j+1}, m_n}, a transposition of amino acid mass at position i after position j transforms it into $\mathcal{P}'$={m_1, m_2, $\cdots$, m_{i-1}, m_{i+1}, $\cdots$, m_{j-1}, m_j, m_i, m_{j+1}, m_n}. Fusion operator adds the masses of two amino acids at adjacent positions when the added value is available in candidateMassList. For a peptide representation, $\mathcal{P}$={m_1, m_2, $\cdots$, m_{i-1}, m_i, m_{i+1}, m_{i+2}, m_n}, it adds m_i, m_{i+1} and thus converts it into $\mathcal{P}'$={m_1, m_2, $\cdots$, m_{i-1}, $(m_i + m_{i+1})$, m_{i+2}, m_n}. Fission does the opposite of fusion. It breaks a amino acids into two to seek for improved score. Another operator fusion$^+$, sums up two adjacent amino acids, then tries to break that into a combination of other valid amino acids from candidateMassList. For a peptide, $\mathcal{P}$={m_1, m_2, $\cdots$, m_{i-1}, m_i, m_{i+1}, m_{i+2}, m_n}, it adds m_i and m_{i+1}, then break it into two new amino acids q_i and q_{i+1}, where {$m_i + m_{i+1}$=$q_i + q_{i+1}$}. The resultant peptide becomes, $\mathcal{P}'$={m_1, m_2, $\cdots$, m_{i-1}, q_i, q_{i+1}, m_{i+2}, m_n} for fusion$^+$.

All the operators are implemented in a greedy manner. It depends on a parameter r for mutate, swap and transposition. Each operator selected at random is applied r times and the best among the generated candidates are returned. For fusion, fission and fusion$^+$ operators, the greediness is depended on availability of appropriate amino acids in candidateMassList. They try to operate on adjacent elements and replace those with valid amino acids. If multiple numbers of peptides are created, the best scored peptide is returned. At the last step of refinement, we have employed a reverse operator, which reverses the whole sequence. Main objective of introducing this operator is to ensure generation of right sequence instead of the reverse one.

6 EXPERIMENTAL RESULTS

In this section, we provide the details of the implementation of our algorithm, dataset for testing and experiments with results and analysis.

6.1 Implementation

Total four algorithms including CycloAnt have been implemented in python to test and compare performance. The first algorithm is

our proposed algorithm CycloAnt, a combination of ant colony optimization and local search. Second algorithm is bare ant colony optimization without any local neighborhood search. Third algorithm implements local search with refinement cycle. Fourth algorithm applies branch and bound as described in [21]. We created a batch script to execute each of the above algorithms five times for a given peptide. As score defined in Equation 1 is the only way to compare the quality of produced peptides, we calculated cyclo score of all end peptides to create a common platform of comparison among algorithms. For CycloAnt, the parameters of the algorithm are given in Table 1. Among other parameters, $iteration$ number depends on the length of the peptide ranging from 1000-2500 and the refinement stage iteration ϕ is in the range of $100 - 200$ depending on the length of the peptide. The details of our implementation can be accessed at https://github.com/sb689/CyclonAnt2.

$epoch$	Q	ϵ	γ	r	th	ϕ
2	0.1	1	2	5	2/3	100-200

Table 1: Parameters on CycloAntAlgorithm

6.2 Datasets

Peptide sequences to test CycloAnt are collected from [5] and used to generate theoretical spectrum. We acquired sequences of *Axinastatin 5, Gramicidin S, Tyrocidine A, Gratisin, Mycobacillin* from Norine database. A 15 length peptide was randomly generated to test algorithm's behavior for longer peptide. For each of these peptides, we created an ideal theoretical spectrum, termed them as original spectrum. This is our noiseless dataset. In reality, mass spectrums are far from being accurate and remain noisy. Missing peaks is a well known noise [1, 18], which makes data analysis from mass spectrums very complex. To add noise to the spectrum, we randomly deleted 10% and 25% data from the noiseless dataset. This process fulfilled the requirement of creating two deviated dataset for test purpose. All the peptides and length of the sequences are added in result tables.

6.3 Environment

All experiments were performed on a laptop computer. Hardware specification of the machine was - Intel(R) Core(TM)i5-2450M CPU @2.50 GHz,4.00 GB Ram(2.94 GB usable), Operating system- Windows7 Ultimate(32 bit). For python development, we worked with Anaconda3-4.1.1-Windows-x86 and atom editor, version - 1.13.0 ia32.

6.4 Results

The results of the experiment are reported in Table 2 and Table 3. Each of these table has three parts, each part showing the results of the spectrum for one dataset as discussed in section 6.2. From the five independent runs, we report best and average score achieved by each of the four algorithms. The first part of the tables contain results for noiseless spectrum. Results from noisy spectrum are given in second part and third part of Table 2 and Table 3.

For table 2, algorithms were executed for same number of iterations. Best score, average score of experimented peptides from four different algorithms' are shared. A column to express number of correct peptide generation among 5 runs is also added denoted as *match*. Table 3 describes the result of same experiments with a difference that, algorithms were executed for equal number of fitness evaluation instead of equal number of iterations.

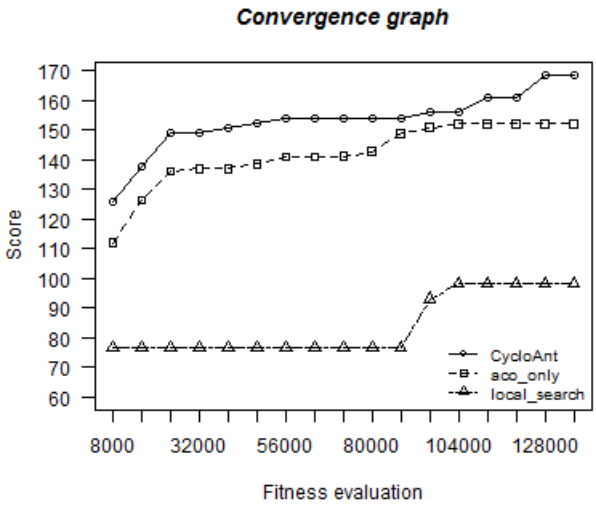

Figure 3: convergence graph.

6.5 Analysis

What we can observe from Table 2 is, CycloAnt has always produced better result than bare aco and local search. Therefore, the main competition exists between Branch and bound and CycloAnt. Branch and bound (BnB) algorithm works perfectly for original theoretical spectrum although the scenario changes as distortion in spectrum rises. For 10% distorted spectrum, BnB is not able to generate right sequence for the peptide *Gramicidine S* for a single time out of 5 runs. On the contrary, CycloAnt produced correct sequence for all given peptides. In case of 25% deviated spectrum, BnB's performance decline significantly. Apart from *Gratisin* and *Tyrocidine A*, all other sequences are wrong. A further investigation reveals that BnB is able to generate correct sequence in reverse order for length 13 and length 15 peptides which resulted in same cyclo score as CycloAnt's but match score to 0. For 25% distorted spectrum of *Axinastatin5* (8 length peptide), BnB generates the output sequence as 113-97-97-163-196-113-113, whereas the original sequence is 113-97-97-163-99-97-113-113. As masses 210, 260, 339, 553, 666 get deleted from theoretical spectrum in a random manner, BnB is not able to differentiate between amino acid mass 196 and 99-97. But different operators of CycloAnt can handle the same scenario perfectly and return the correct peptide. Same situation occurs for *Gramicidin S* in case of 10% and 25% distorted spectrum.

In table 3, performance of algorithms almost stays same with a slight improvement in case of local search and aco. Despite of the

Peptide info			Our algorithm			ACO			Local search			BnB		
sl no	name	size	best	avg	match	best	avg	match	best	avg	match	best	avg	match
1	Axinastatin5	8	**58**	**58**	5	**58**	49.6	2	**58**	44.6	1	**58**	**58**	5
2	GramicidinS	10	**92**	**92**	5	**92**	75.6	1	**92**	78.8	3	**92**	**92**	5
3	TyrocidineA	10	**92**	**92**	5	74	64.4	0	**92**	57.8	2	**92**	**92**	5
4	Gratisin	12	**134**	**134**	5	102	81.2	0	**134**	121.4	4	**134**	**134**	5
5	Mycobacillin	13	**158**	**158**	5	104	94.8	0	**158**	104	2	**158**	**158**	5
6	random generated peptide	15	**212**	202.4	3	92	78.4	0	**212**	173.6	2	**212**	**212**	5
10% distorted spectrum														
7	Axinastatin5	8	**53**	**53**	5	**53**	42.6	1	**53**	32.8	1	**53**	**53**	5
8	GramicidinS	10	**83**	**83**	5	**83**	60.6	1	**83**	67.4	3	68	68	0
9	TyrocidineA	10	**83**	**83**	5	68	57.4	0	**83**	70.8	2	**83**	**83**	5
10	Gratisin	12	**121**	**121**	5	**121**	90.6	1	**121**	109	4	**121**	**121**	5
11	Mycobacillin	13	**143**	**143**	5	101	89.8	0	**143**	105.2	3	**143**	**143**	5
12	random generated peptide	15	**191**	181	4	111	92.2	0	127	82.4	0	**191**	**191**	5
25% distorted spectrum														
13	Axinastatin5	8	**44**	**44**	5	35	32.8	0	**44**	39.6	3	39	39	0
14	GramicidinS	10	**69**	**69**	5	57	49.2	0	**69**	49.6	2	60	60	0
15	TyrocidineA	10	**69**	67	4	**69**	42.4	1	37	32.6	0	**69**	**69**	5
16	Gratisin	12	**101**	**101**	5	92	72.2	0	**101**	83.8	2	**101**	**101**	5
17	Mycobacillin	13	**119**	**119**	5	**119**	85.4	1	**119**	105.8	3	**119**	**119**	0
18	random generated peptide	15	**160**	**160**	5	63	56.6	0	**160**	83.4	1	**160**	**160**	0

Table 2: Results achieved by four different algorithms for different datasets based on equal iteration number in each algorithm.

Peptide info			Our algorithm			ACO			Local search			BnB		
sl no	name	size	best	avg	match	best	avg	match	best	avg	match	best	avg	match
1	Axinastatin5	8	**58**	**58**	5	**58**	55.6	4	**58**	47	3	**58**	**58**	5
2	GramicidinS	10	**92**	**92**	5	**92**	75.6	1	**92**	**92**	5	**92**	**92**	5
3	TyrocidineA	10	**92**	**92**	5	62	58.4	0	**92**	74	3	**92**	**92**	5
4	Gratisin	12	**134**	**134**	5	98	80	0	**134**	127.6	4	**134**	**134**	5
5	Mycobacillin	13	**158**	**158**	5	122	102.8	0	134	77.8	0	**158**	**158**	5
6	random generated peptide	15	**212**	206.4	4	106	95.2	0	**212**	193.6	3	**212**	**212**	5
10% distorted spectrum														
7	Axinastatin5	8	**53**	**53**	5	**53**	**53**	5	**53**	45	2	**53**	**53**	5
8	GramicidinS	10	**83**	**83**	5	69	64	0	**83**	**83**	5	68	68	0
9	TyrocidineA	10	**83**	**83**	5	58	54.6	0	**83**	61.8	2	**83**	**83**	5
10	Gratisin	12	**121**	**121**	5	101	94.4	0	**121**	115.2	4	**121**	**121**	5
11	Mycobacillin	13	**143**	**143**	5	128	108.4	0	**143**	88.4	2	**143**	**143**	5
12	random generated peptide	15	**191**	178.8	3	113	98.2	0	**191**	146.8	2	**191**	**191**	5
25% distorted spectrum														
13	Axinastatin5	8	**44**	**44**	5	35	29.6	0	**44**	30	1	39	39	0
14	GramicidinS	10	**69**	**69**	5	62	52.2	0	**69**	50.8	3	60	60	0
15	TyrocidineA	10	**69**	**69**	5	60	44.8	0	**69**	62.2	3	**69**	**69**	5
16	Gratisin	12	**101**	**101**	5	86	71.2	0	**101**	86	3	**101**	**101**	5
17	Mycobacillin	13	**119**	**119**	5	**119**	75.4	1	**119**	81.2	2	**119**	**119**	0
18	random generated peptide	15	**160**	141.2	2	71	64.6	0	130	84.4	0	**160**	**160**	0

Table 3: Results achieved by four different algorithms for different datasets based on equal fitness evaluation number in each algorithm.

improvement, local search and aco is not in a state to compete with CycloAnt.

To find the convergence rate among CycloAnt, local search and ant colony optimization technique, we have prepared a convergence graph from their result set. Branch and bound algorithm has been kept out of this experiment, as its base mechanism is quite different than that of other three. We executed each of the algorithms 5 times for length 10 peptide and recorded their global best score at a specific interval of fitness evaluation count. Average score from 5 runs are calculated and plotted against Fitness evaluation in graph 3. The graph clearly shows convergence rate of CycloAnt is better than that of aco and local search.

We performed statistical Wilcoxon rank sum test for independent samples from CycloAnt and Branch and bound algorithm with 95% level of significance to verify difference in performance. In case of deviated spectrum, performance differed significantly.

7 CONCLUSION

In this paper, we have presented CycloAnt, a hybrid ant colony optimization algorithm that uses ants to construct solutions for *de novo* cyclic peptide sequencing problem from a noisy experimental spectrum. Our algorithm uses a subsequent local search after each ant constructs a solution. The algorithm is guided by two novel heuristic function and a novel set of operators for the local search part. On standard benchmark sets of data, CycloAnt significantly improves the results achieved by state-of-the-art algorithms. One of the limitations of this work is not considering the presence of fragment masses for the ion types and false peaks. In a future work, we would like to extend our method to incorporate that. We would also like to develop a web based stand-alone application or tool so that this method can be used online by the researchers working in Proteomics.

REFERENCES

[1] Ruedi Aebersold and David R Goodlett. 2001. Mass spectrometry in proteomics. *Chemical reviews* 101, 2 (2001), 269–296.

[2] Jens Allmer. 2011. Algorithms for the de novo sequencing of peptides from tandem mass spectra. *Expert review of proteomics* 8, 5 (2011), 645–657.

[3] Nuno Bandeira, Julio Ng, Dario Meluzzi, Roger G Linington, Pieter Dorrestein, and Pavel A Pevzner. 2008. De novo sequencing of nonribosomal peptides. In *Annual International Conference on Research in Computational Molecular Biology*. Springer, 181–195.

[4] Benjamín Barán and Matilde Schaerer. 2003. A Multiobjective Ant Colony System for Vehicle Routing Problem with Time Windows.. In *Applied informatics*. 97–102.

[5] Ségolène Caboche, Maude Pupin, Valérie Leclère, Arnaud Fontaine, Philippe Jacques, and Gregory Kucherov. 2008. NORINE: a database of nonribosomal peptides. *Nucleic acids research* 36, suppl 1 (2008), D326–D331.

[6] Ting Chen, Ming-Yang Kao, Matthew Tepel, John Rush, and George M Church. 2001. A dynamic programming approach to de novo peptide sequencing via tandem mass spectrometry. *Journal of Computational Biology* 8, 3 (2001), 325–337.

[7] Keng Wah Choo and Wai Mun Tham. 2007. Tandem mass spectrometry data quality assessment by self-convolution. *BMC bioinformatics* 8, 1 (2007), 1.

[8] Mark Cieliebak, Stephan Eidenbenz, and Paolo Penna. 2003. Noisy data make the partial digest problem NP-hard. In *International Workshop on Algorithms in Bioinformatics*. Springer, 111–123.

[9] Tamara Dakic. 2000. *On the turnpike problem.* Ph.D. Dissertation. Simon Fraser University.

[10] Marco Dorigo, Mauro Birattari, Christian Blum, Maurice Clerc, Thomas Stützle, and Alan Winfield. 2008. *Ant Colony Optimization and Swarm Intelligence: 6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008, Proceedings*. Vol. 5217. Springer.

[11] René J Dubos. 1939. Studies on a bactericidal agent extracted from a soil bacillus: I. Preparation of the agent. Its activity in vitro. *The Journal of experimental medicine* 70, 1 (1939), 1.

[12] Bernd Fischer, Volker Roth, Franz Roos, Jonas Grossmann, Sacha Baginsky, Peter Widmayer, Wilhelm Gruissem, and Joachim M Buhmann. 2005. NovoHMM: a hidden Markov model for de novo peptide sequencing. *Analytical chemistry* 77, 22 (2005), 7265–7273.

[13] Eduard Fomin. 2015. Reconstruction of sequence from its circular partial sums for cyclopeptide sequencing problem. *Journal of bioinformatics and computational biology* 13, 01 (2015), 1540008.

[14] Ari Frank and Pavel Pevzner. 2005. PepNovo: de novo peptide sequencing via probabilistic network modeling. *Analytical chemistry* 77, 4 (2005), 964–973.

[15] Muztaba Hasanat, Mehadi Hasan, Istiak Ahmed, Mehedi Iqbal Chowdhury, Jannatul Ferdous, and Swakkhar Shatabda. 2016. An ant colony optimization algorithm for load shedding minimization in smart grids. In *Informatics, Electronics and Vision (ICIEV), 2016 5th International Conference on*. IEEE, 176–181.

[16] Jan A Hiss, Michael Reutlinger, Christian P Koch, Anna M Perna, Petra Schneider, Tiago Rodrigues, Sarah Haller, Gerd Folkers, Lutz Weber, Renato B Baleeiro, and others. 2014. Combinatorial chemistry by ant colony optimization. *Future medicinal chemistry* 6, 3 (2014), 267–280.

[17] Guohui Lin, Dong Xu, Zhi-Zhong Chen, Tao Jiang, Jianjun Wen, and Ying Xu. 2002. An efficient branch-and-bound algorithm for the assignment of protein backbone NMR peaks. In *Bioinformatics Conference, 2002. Proceedings. IEEE Computer Society*. IEEE, 165–174.

[18] Bin Ma. 2010. Challenges in computational analysis of mass spectrometry data for proteomics. *Journal of Computer Science and Technology* 25, 1 (2010), 107–123.

[19] Bin Ma, Kaizhong Zhang, and Chengzhi Liang. 2005. An effective algorithm for peptide de novo sequencing from MS/MS spectra. *J. Comput. System Sci.* 70, 3 (2005), 418–430.

[20] Mario Alberto Martínez-Núñez and Víctor Eric López y López. 2016. Nonribosomal peptides synthetases and their applications in industry. *Sustainable Chemical Processes* 4, 1 (2016), 13.

[21] Hosein Mohimani, Wei-Ting Liu, Yu-Liang Yang, Susana P Gaudêncio, William Fenical, Pieter C Dorrestein, and Pavel A Pevzner. 2011. Multiplex de novo sequencing of peptide antibiotics. *Journal of Computational Biology* 18, 11 (2011), 1371–1381.

[22] Hosein Mohimani, Yu-Liang Yang, Wei-Ting Liu, Pei-Wen Hsieh, Pieter C Dorrestein, and Pavel A Pevzner. 2011. Sequencing cyclic peptides by multistage mass spectrometry. *Proteomics* 11, 18 (2011), 3642–3650.

[23] Julio Ng, Nuno Bandeira, Wei-Ting Liu, Majid Ghassemian, Thomas L Simmons, William Gerwick, Roger Linington, Pieter Dorrestein, and Pavel Pevzner. 2009. Dereplication and de novo sequencing of nonribosomal peptides. *Nature methods* 6, 8 (2009), 596.

[24] Jiří Novák, Karel Lemr, Kevin A Schug, and Vladimír Havlíček. 2015. CycloBranch: de novo sequencing of nonribosomal peptides from accurate product ion mass spectra. *Journal of The American Society for Mass Spectrometry* 26, 10 (2015), 1780–1786.

[25] Pavel A Pevzner, Vlado Dančík, and Chris L Tang. 2000. Mutation-tolerant protein identification by mass spectrometry. *Journal of Computational Biology* 7, 6 (2000), 777–787.

[26] Alena Shmygelska and Holger H Hoos. 2005. An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem. *BMC bioinformatics* 6, 1 (2005), 30.

[27] Steven S Skiena, Warren D Smith, and Paul Lemke. 1990. Reconstructing sets from interpoint distances. In *Proceedings of the sixth annual symposium on Computational geometry*. ACM, 332–339.

[28] Steven S Skiena and Gopalakrishnan Sundaram. 1994. A partial digest approach to restriction site mapping. *Bulletin of Mathematical Biology* 56, 2 (1994), 275–294.

[29] Hanno Steen and Matthias Mann. 2004. The ABC's (and XYZ's) of peptide sequencing. *Nature reviews Molecular cell biology* 5, 9 (2004), 699–711.

[30] Marc Oliver Wagner, Bernard Yannou, Steffen Kehl, Dominique Feillet, and Jan Eggers. 2003. Ergonomic modelling and optimization of the keyboard arrangement with an ant colony algorithm. *Journal of Engineering Design* 14, 2 (2003), 187–208.

[31] Changjiang Xu and Bin Ma. 2006. Complexity and scoring function of MS/MS peptide de novo sequencing. In *Comput. Syst. Bioinformatics Conf*, Vol. 5. 361–369.