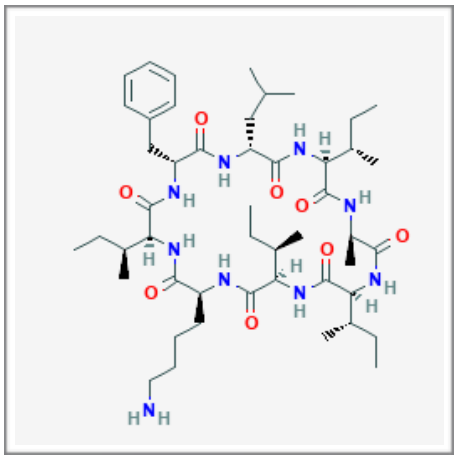
Daniel Hakim
BNFO 298
December 5, 2018

NSeq Cyclofamily Sequencing



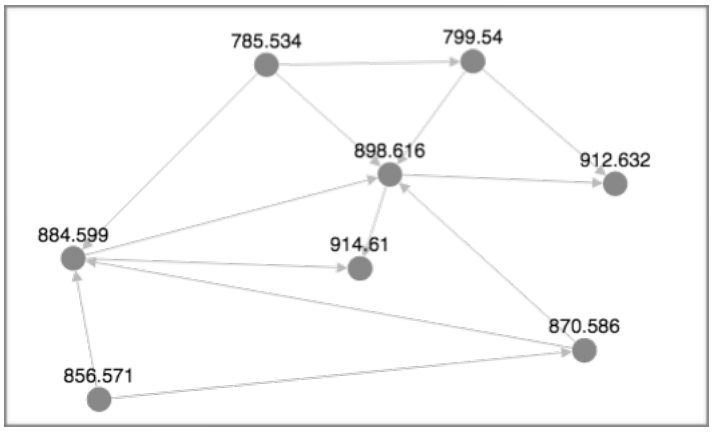
Surugamide A
PubChem CID: 71764189
Molecular Weight: 912.231 g/mol
IUPAC Condensed:
cyclo[D-Ala-Ile-D-Ile-Lys-Ile-D-Phe-D-Leu-Ile]

National Center for Biotechnology Information. PubChem Compound Database; CID=71764189, https://pubchem.ncbi.nlm.nih.gov/compound/71764189 (accessed Dec 7, 2018).

Cyclopeptides exhibit both a strong resistance to digestion in the human gut and number of desirable biological activities including both antibiotic and tumor suppressant capabilities. These properties have made cyclopeptides prime candidates in the search for new pharmaceuticals. This search is complicated by the structure and composition of these cyclic peptides. Due to their non ribosomal nature, mass spectrometry based sequencing approaches must be tailored for their cyclic structure, and must support non proteinogenic amino acids. Further, genome mining based approaches cannot directly use standard techniques to predict protein sequence from genomic sequence.

The CycloNovo and CycloMiner tools, tailored for de novo spectral sequencing and genome mining based prediction of cyclopeptides respectively, have been demonstrated to address these issues with considerable success.

Here we will develop the new Network Sequencing Tool (NSeq), which combines the candidate sequence generation of the CycloNovo and CycloMiner tools with externally identified spectral networks.



Estimated Surugamide Spectral Network

Nodes are labeled by precursor mass of spectra. Edges are assigned by Co-Sign similarity metric across experimental spectra.

Investigated spectra available for download from GNPS ProteoSAFe. ID: MSV000078604

We begin with three goals to guide the investigation:

- **Goal 1:** Given spectra of related cyclopeptides (with lengths from 5 to 12), their spectral network *G*, and sets of candidate cyclopeptides for each spectra, find all cyclopeptides that generated the spectra.
- **Goal 2:** Given spectra of related cyclic peptides (with lengths from 5 to 12), their spectral network *G*, and sets of *candidate* cyclopeptides for some of the spectra, find all cyclic peptides that generated the spectra.
- **Goal 3:** Investigate scoring metrics that take advantage of the spectral network to sort candidates at each node for further evaluation.

Goal 1 is designed to identify and strip away inconsistent candidate sequences by cross checking them with other candidates across a spectral network. **Goal 2** is designed to fill in the gaps in a network when existing tools are unable to generate satisfactory candidates. **Goal 3** is designed to simplify further validation of candidate sequences by reducing the number of sequences for test before the correct sequence is identified.

To achieve **Goal 1**, we must first rigorously define our cross checking procedure. For a given spectral network, we will assume that each node can be explained by a single true cyclic candidate sequence. We will further assume that the true sequences associated with adjacent nodes in the network will be related by a small number of additions, deletions or substitutions of amino acids.

		K	I	F	L	A	I	I
	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*
K	*1*	0	1	2	3	4	5	6
I	*2*	1	0	1	2	3	4	5
F	*3*	2	1	0	1	2	3	4
L	*4*	3	2	1	0	1	2	3
V	*5*	4	3	2	1	1	2	3
A	*6*	5	4	3	2	1	2	3
V	*7*	6	5	4	3	2	2	3
I	*8*	7	6	5	4	3	2	2

Levenshtein Distance Computation

Calculation of Levenshtein distance between two Surugamide variants.

Levenshtein distance has long been used a measure of edit distance between strings. It defines the minimum number of additions, deletions, and substitutions of characters to transform one string into another.

Cyclic Edit Distance (CED) is the minimum of Levenshtein distances between one candidate sequence and all rotations of another candidate sequence.

We will define cyclic edit distance (CED) as the minimum number of additions, deletions, and substitutions relating two cyclic candidate sequences. We will define a G-Consistent Network as an assignment of individual candidate sequences to each node of a spectral network such that all pairs of sequences associated with adjacent nodes are related with a CED below a given threshold. By our assumptions, the true assignment of sequences must form a G-Consistent Network.

Given a spectral network, assigned candidates, and a maximum CED threshold, the NSeq tool enumerates the set of all G-Consistent Networks.

CANDIDATE FORMAT	CANDIDATE DATA
Raw	`57.0214, 71.0371, 71.0371, 85.0524, 87.0324`
Canonical Form	`Gly, Ala, Ala, Abu, Ser`
Spin Invariant Canonical Form	`Abu, Ser, Gly, Ala, Ala`

Input candidates are converted from raw masses to a canonical form written as amino acids. The language of amino acids can be chosen dependent on the network being investigated and should include non proteinogenic amino acids.
The canonical form will allow computation of the cyclic edit distance.
The spin invariant canonical form is the lexicographic minimum of all rotations of the canonical form. It enables equality, hashing, and lexicographic comparison of candidate sequences. All computation involving candidate sequences uses this format.

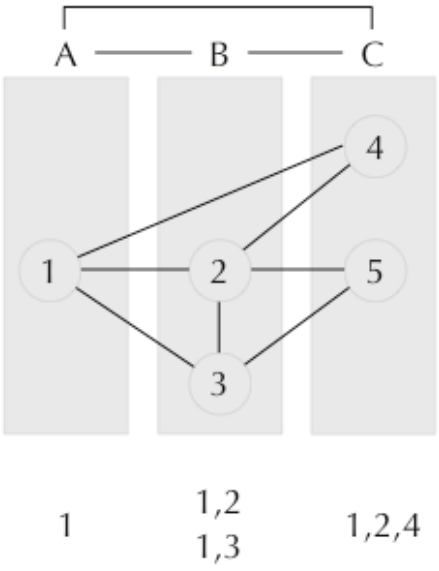
NSeq Algorithm

1. Ingest spectral network (SG) and assigned candidates at each node. Remove duplicate candidates. Optionally filter by score or mass tolerance.

2. Across each edge of SG, connect all candidate pairs passing the target CED threshold. These connections between candidates form the edges of a new graph called the Candidate Graph.

3. As an optimization step, we now simplify the Candidate Graph. For each candidate, count the associated SG nodes of its neighbors in the Candidate Graph. This count is that candidate's coverage. Candidates with lower coverage than the degree of their associated SG node cannot be part of any G-Consistent Network. Iteratively remove these candidates, recomputing coverage at each step.

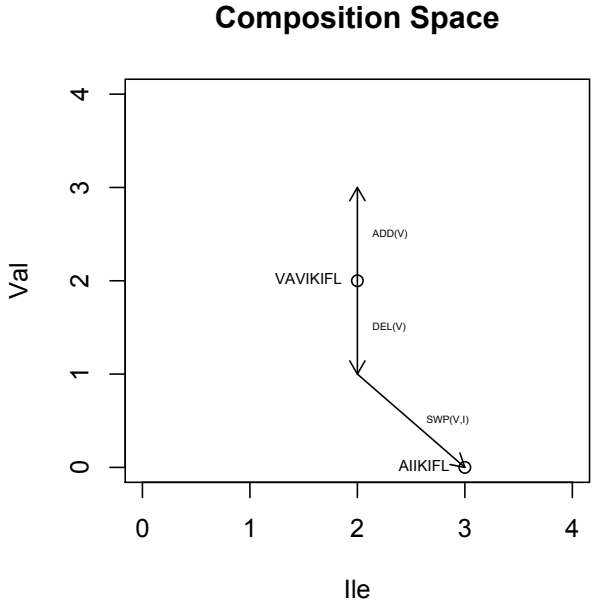
4. For each connected component of SG, pick an arbitrary start node. Initialize a wavefront of partial G-Consistent Networks with the set of candidates at this start node. Depth first search over SG from that start node. Upon visiting each node, construct the next wavefront by branching each partial G-Consistent Network out to encompass each candidate of this new node that is consistent (connected in the candidate graph) with its existing members.
   Note that some partial G-Consistent Networks in the wavefront will not be consistent with any candidates and can be removed. Others may be consistent with multiple candidates, causing the wavefront to grow exponentially.



NSeq Step 4

The three fully connected nodes of SG: A, B and C, have associated candidate sets {1}, {2,3}, and {4,5} as indicated. The Candidate Graph is overlaid.

A is chosen as the start node, and the wavefront is initialized with candidate 1.
B is visited. Both candidates 2 and 3 are consistent with 1, so the new wavefront contains both 1,2 and 1,3.
C is visited. 4 is consistent with 1,2 but is not consistent with 1,3. 5 is neither consistent with 1,2 nor with 1,3, as it is inconsistent with 1. Thus the final wavefront consists of the single G consistent network 1,2,4.



Composition Space - Ile/Val

Composition space projects each candidate sequence to a point located by the counts of the candidate sequence's constituent amino acids.

Here we show a cross section of Ile and Val counts of related Surugamide variants with CED 2. The three classes of transitions, ADD, DEL, SWP, (addition, deletion, substitution) are illustrated as well. These transitions define paths between compositions. The minimum path length between compositions corresponds to a lower bound on CED between candidate sequences.

Goal 2 is focused on generation of candidate sequences at nodes in the spectral graph for which candidate sequences are unavailable. Here we will again assume that the true assignment across a spectral network is a G-Consistent Network of sequences for a given threshold CED. From this assumption, if a true assignment is within a neighboring node's candidate set, the true assignment at the unannotated node is within the threshold CED of the candidates at that neighbor.

The Candidate Assignment module will enumerate all candidate sequences that are both within a threshold CED of a set of candidate sequences and whose total mass is within a given tolerance of the precursor mass at the unannotated node.

Let's first introduce several new concepts:

- *Composition Space*: A projection from candidate sequence to a vector of counts of each constituent amino acid. The dimension of this space equals the number of amino acids in the language.
- Transition: ADD(x), DEL(x) or SWP(x,y) where x,y are amino acids in the language. These transform a composition into an adjacent composition in the Composition Space.
- Transition Path: A list of transitions that transforms one composition into another.

- Delta Mass: The delta mass of a composition is the difference in mass of its ending and starting compositions. The delta mass of a path is the sum of the delta masses of its transitions.

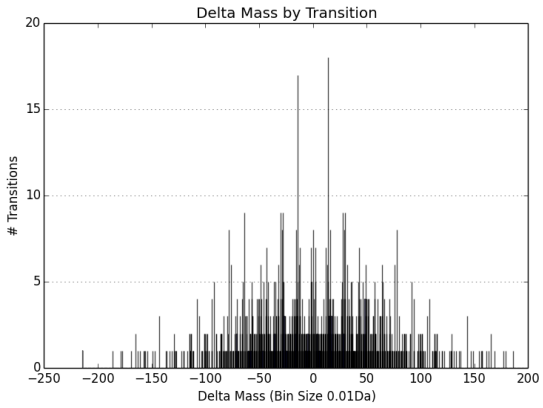
Candidate Assignment Algorithm

1. Construct the set of all transitions in the composition space (ADD(x), DEL(y), SWP(x,y) for x,y as elements of the amino acid language)

2. Construct the set of all paths of transitions of length less than or equal to the threshold CED. Sort these paths by their delta mass

3. Across each edge of SG around the unannotated node, subtract precursor masses to determine the required delta mass. Filter the set of transition paths for a tolerance around this value.

4. For each candidate of each neighbor of the unannotated node, apply the necessary transition paths to construct candidates at the annotated node.
   Note that each transition in the path can be applied at multiple indices to each candidate.



Delta Mass Filtering of Paths

Given a language of 40 amino acids:
There are 1640 transitions. There are 2691241 paths consisting of up to two transitions.

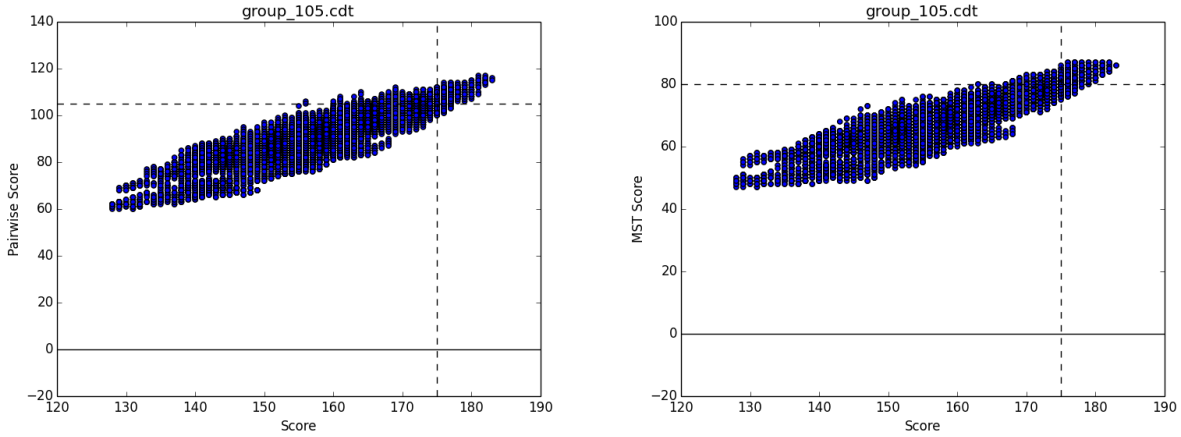
As we generate candidate sequences, we will only apply paths that result in the proper precursor mass to within a tolerance of 0.02Da. This reduces the number of paths to evaluate from a few million to a few thousand.

Goal 3 is the creation of a scoring metric used to sort and filter candidates. A well designed scoring metric enables branch and bound heuristics to improve run time of these algorithms and sorts output candidates in the order they should be evaluated further. A new metric, the Pairwise Score, was evaluated against the existing scoring metric used in CycloNovo.

Let S1, S2 be experimental spectra: {s: s corresponds to an m/z peak}
Let T1, T2 be theoretical spectra of two candidate sequences: {t: t corresponds to the mass of some cyclic substring of the candidate sequence}

PairwiseScore(S1, S2, T1, T2, Δ, ε):
Define relation a~b: |b-a| ≤ ε
Pairwise score is the cardinality of
{(t1, t2):t1 ∈ T1 ∧ t2 ∈ T2 ∧ t2-t1~Δ ∧ ∃s1 ∈ S1 st s1~t1 ∧ ∃s2 ∈ S2 st s2~t2}

For each G-Consistent Network, the Pairwise Score for the network is computed as a summation of Pairwise Scores computed at multiple edges. Two variants were evaluated, one computed along all edges of the spectral network, another was built by weighting these same edges and summing only those edges forming a maximal spanning tree (MST) of the network.



Pairwise Score and MST Score Variants
Pairwise score and MST score are compared against baseline scoring metrics for all G-Consistent Networks of Surugamide. Location of "true" candidate network for Surugamide network as identified in prior work is marked by dashed lines.

The Pairwise Score correlates highly with the existing scoring metric. Examining the lower right quadrant formed by the dashed lines, we see there can only be a slight benefit to additional filtering by these metrics.

NODE MASS:	785	799	856	870	884	898	912	914
# Input Candidates	706	732	691	698	694	698	712	1183
# High Scoring Candidates	29	8	89	121	66	87	2	12
G-Consistent High Scoring Candidates	28	3	63	39	18	5	1	6

Goal 1 Validation:

The number of candidates at each node of the Surugamide network at input to the NSeq tool, after score threshold filtering, and after G-Consistency across high scorers.
By cross validation of the candidates at node 898, adjacent to nodes 912 and 914, we were able to dramatically reduce the set of candidate sequences requiring further evaluation.

NODE MASS:	785	799	856	870	884	898	912	914
# Candidates After Candidate Assignment	6136	4073	70239	31488	30625	630	*	61693
“True” Candidate Ranking	556	2	>1000	202	170	2	*	110

Goal 2 Validation:

Candidate Assignment was seeded with 712 candidates at node 912. The tool was used recursively to annotate the rest of the network as though CycloNovo, CycloMiner had failed to annotate these nodes. Candidates were computed at each node, sorted by baseline scoring metric, and the top 1000 were used to seed future Candidate Assignment calls. Starting from the candidates at 912 and the experimental spectral data, the Candidate Assignment algorithm regenerated externally validated candidate sequences for all but one of the nodes of the Surugamide network.

Discussion

The NSeq tool shows significant utility in its ability to filter candidates by G-Consistency. One must take care however, to ensure all of its required assumptions can be met. As spectral networks are estimated on the basis of similarity of spectral data, edges of these networks may not correspond exactly to CED relationships as is assumed by NSeq. Should a single edge of the input network correspond to a CED larger than the threshold in use, NSeq may cull the entire network of correct answers from consideration. In this sense, it is better to add edges to the network only where it is absolutely certain that the CED threshold will not be exceeded.

The CED calculations based on Levenshtein distances lend themselves to a number of potential optimizations. These include both cyclic variants of the Levenshtein distance dynamic programming algorithm, and tree based structures that allow faster computation of 1 to many relations. Due to this, it may be feasible to increase the CED threshold used by NSeq, allowing its use in less well characterized spectral networks, but doing so will generate numerous additional G-Consistent networks.

The same cannot be said for the CED threshold of the Candidate Assignment module. The number of transition paths to evaluate expands exponentially on the CED with a factor greater than the square of the number of amino acids in the language. Without a significant algorithmic improvement, this tool will be limited to a CED threshold <= 3.

The Candidate Assignment tool itself shows significant promise, both in its ability to generate candidate sequences, and as a side effect, in its ability to generate putative compositions. While it generates many thousands of spurious candidates, most of which can be safely eliminated by the baseline scoring metric, it generates far fewer compositions. As putative composition generation is a complex computation with non negligible chance of failure in the CycloNovo tool, it may be that this composition generation side effect is a useful module in and of itself. It may also be that the composition generation problem can be optimized more heavily.

The PairwiseScore metric did not capture the relationship between candidates as well as we had hoped. This is potentially due to its construction assuming a CED of 1. More than half of the edges of the Surugamide network investigated corresponded to a CED of 2. It should therefore be expected that the PairwiseScore metric as defined cannot fully distinguish the best candidates.

We introduced this investigation with the allure of cyclopeptides as potential pharmaceuticals. However, not all spectra we wish to apply these techniques to

correspond to cyclopeptides. It is important therefore to identify the gaps in extending these algorithms to generalized peptide structure. Some groundwork has already been laid in candidate definition in the form of a graph. To extend NSeq for this definition, we must build a new definition of CED (perhaps the Graph Edit Distance/GED built on A*) , and ideally, an compact format that will allow optimized hashing, equality, and lexicographic comparison. Extending Candidate Assignment will similarly require a mechanism to generate all graphs within some distance of some seed graph. Calculations up through the enumeration of transition paths will remain identical, it is only the application of these transitions to the new candidate format that must be modified.

The NSeq and Candidate Assignment tools have shown significant promise, but their testing thus far has been limited to a single network. These techniques must now be attempted on additional datasets, and their results validated by parallel techniques. Only once these tools have been proven can we confidently apply them in the search for effective pharmaceuticals.

Full source of the NSeq Tool is available on Github:

https://github.com/dhakim87/netseq.git