Efficient algorithms for motif discovery in ranked lists

Limor Leibovich1, Yael Mandel-Gutfreund2, Zohar Yakhini1,3
1 Faculty of Computer Science, Technion-Israel Institute of Technology, Haifa, Israel
2 Faculty of Biology, Technion-Israel Institute of Technology, Haifa, Israel
3 Agilent Laboratories, Tel Aviv, Israel

Introduction to motif discovery in ranked lists

A transcription factor (TF) is a protein that binds to specific DNA sequences, thereby controlling the flow (through transcription) of genetic information from DNA to mRNA.

Transcription factors typically bind specific short sequences of DNA. By analyzing sequences that bind a transcription factor of interest, we want to determine the sequence (or motif) of its binding site.

How do we calculate enrichment?

Given:
- # Ranked list of N sequences
- Motif of interest
- Cutoff n

P(X > b) = HGT(b)(N, B, n) = \sum_{i=b+1}^{N} \binom{n}{i} \frac{(N-n)^i}{B}

P(X > b) = HGT(b)(N, B, n) = \sum_{i=b+1}^{N} \frac{n! (N-n)!}{i! (N-B)!}

mHGT = \min_{b=1, n \leq B} \text{HGT}_{b}(N, B, n)

How to define the motif search space?

Given that we are interested in finding motifs of length \( k \), we can calculate the enrichment of all \( k \)-mers over the alphabet \( \Sigma \) and then choose the most enriched motif.

- Exponential in \( k \): there are \( |\Sigma|^k \) possible \( k \)-mers
- Not feasible for long motifs
- Not feasible for large alphabets

A better approach would be to test only \( k \)-mers that appear at some sequence in the list.

Our algorithmic approach (DRIMUST)

1. Build a generalized suffix tree for \( S_1,\ldots, S_P \)
2. Given that we search for motifs of length \( k \), traverse the tree to find paths of length \( k \), and for each path \( P \) do:
   - Calculate \( P \)'s enrichment using the following process:
     - Get the list \( C \) of strings containing \( P \)
     - \( C = \{8, 14, 31, 36\} \) in the example
     - Calculate the mHG score for \( P \):
       - \( N \) is the total number of sequences
       - \( B \) is the size of \( C \)
       - \( b \) is the number of members in \( C \) not greater than \( n \)
       - Assume \( N = 100 \), then from the example \( B = 4 \)
       - \( n = 8 \), \( b = 4 \)
       - HGT = 0.29, HGT = 0.09
       - \( n = 31 \), \( b = 5 \)
       - HGT = 0.08
     - \( mHGT = mHGT_{b}(N, B, n) \)

Comment on time complexity:

- Denote \( |C(P)| \) as the number of strings containing \( P \) and \( m = |\Sigma|^i \).
- Given \( P \), HGT is calculated for \( |C(P)| \) candidates, and each calculation takes \( O(|C(P)|) \) time (when the HG assessment is done in \( O(1) \)).

Total complexity, given \( P \), is \( O(|C(P)|) \).

- We perform this process for all paths of length \( k \).

Total time complexity: \( O(\sum_{P} C(P)|P|) \); it holds that \( \sum_{P} C(P) \leq m \).

References

Gusfield D: Algorithms on strings, trees, and sequences, Cambridge Univ. Press; 2007, chapters 5-6