Finding subtle motifs by branching from sample strings [1]

Alkes Price Sriram Ramabhadran Pavel A. Pevzner

SUMMARY OF THE PAPER

A. Sequence Motifs

Sequence motifs are the various different sub sequences that appear more frequently in biological networks when compared to the random networks. There are different approaches that can be used to find these motifs. One such approach involves spanning over all possible starting positions of all motif occurrences. The main short coming of this approach is that the possible starting positions may be typically large. Also, going for greedy or random approach may never give an approximate solution that would be close to the global optima. The other approach involves describing all possible motifs first and then spanning the space of the defined motifs. By using this approach, the search can be restricted to small neighbourhoods as globally optimum motifs have an approximate occurrence in the sample. These three approaches have been termed as sample-driven, extended sample-driven and branching from sample strings respectively (Fig. 1). The two algorithms that have been introduced as a part of this paper are - The pattern branching algorithm and the profile branching algorithm.

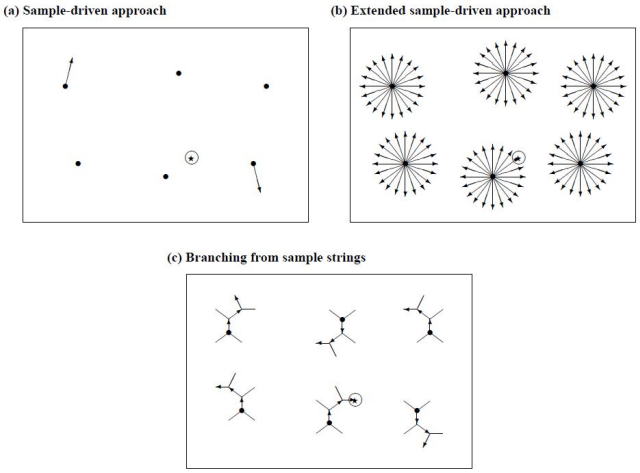

Fig. 1. Comparison of sample-driven, extended sample-driven and branching approaches to searching in motif space. Bullets represent sample strings, circled stars represent the globally optimal motif. (a) Sample-driven algorithms often fail to find the global optimum. (b) Extended sample-driven algorithms typically find the global optimum, but with high computational cost. (c) Branching from sample strings efficiently finds the global optimum.

B. The pattern branching algorithm

Given a set of sequences(S) along with the length of the motif(l) and the number of allowed mutations(k), we wish to find the sequence motif by starting from an arbitrary random motif. A path of patterns would be constructed as $A_0 \rightarrow A_1 \rightarrow A_2 ... A_k$ where A_j represents the sequence at a hamming distance of j from A_0. The distance of a sequence from the set (S) is defined as

$$d(A, S) = \sum_{Si \in S} d(A, S_i)$$

$$d(A, S_i) = min(d(A, P) | P \in S_i)$$

where P denotes a l-mer. After every iteration we wish to update the motif such that the sequence that is closest to the set S is retained. The algorithm is as per Fig.2.

```
PatternBranching(S, l, k)
Motif ← arbitrary motif pattern
For each l-mer A_0 in S
    For j ← 0 to k
        If d(A_j, S) < d(Motif, S)
            Motif ← A_j
    A_{j+1} ← BestNeighbor(A_j)
Output Motif
```

Fig. 2. The pattern branching algorithm

ProfileBranching($\mathcal{S}, l, k$)
$Motif \leftarrow$ arbitrary motif profile
For each l-mer A_0 in $\mathcal{S}$
$X_0 \leftarrow X(A_0)$
For $j \leftarrow 0$ to k

If $e(X_j, \mathcal{S}) > e(Motif, \mathcal{S})$
$Motif \leftarrow X_j$
$X_{j+1} \leftarrow$ BestNeighbor(X_j)
Run EM algorithm with $Motif$ as seed

Fig. 3. The profile branching algorithm

Algorithm	Success Rate	Running Time
PROJECTION	about 100%	2 minutes
MITRA	100%	5 minutes
MULTIPROFILER	99.7%	1 minute
PatternBranching	99.7%	3 seconds

Algorithm	Perf. Coeff.	Running Time
CONSENSUS	0.20	40 seconds
GibbsDNA	0.32	40 seconds
MEME	0.14	5 seconds
ProfileBranching	0.57	80 seconds

Fig. 4. Algorithmic comparison

C. The profile branching algorithm

The profile branching algorithm works on similar lines as the pattern branching algorithm but with a few modifications which are:
1. Each and every sample string is converted to its corresponding profile
2. The scoring method is modified to score profiles
3. The branching algorithm is also modified and applied to profiles
4. The best profile is used as the seed for the expectation maximization (EM) algorithm.
Let w be the set of nucleotides and A be the sample to be profiled. Then, X(A) represents the profile of A and is a 4xl matrix in which the column w has a probability 1/2 for $v = a_w$ and 1/6 otherwise. The distance is modified to capture entropy score for profiles and is given by:

$$e(X, P) = \sum_{w=1}^{l} log(x_{p_w w})$$

For each l-mer in the sample S, the path of profiles is constructed as $X_0 \rightarrow X_1 \rightarrow X_2 ... X_k$. The best neighbour strategy is applied iteratively to select the best neighbour on the basis of entropy and after going for k iterations, EM algorithm is run over the top scoring profile. The algorithm is as per Fig.3. A comparative analysis is also provided among various algorithms as per Fig.4.

IMPLEMENTATION

The above stated algorithms were implemented using Python v2.7. The source code is available in files MT16121_problem_4_algo1.py and MT16121_problem_4_algo2.py. The various methods that were defined included get_all_l_mers, get_hamming_distance, get_distance, best_neighbour, pattern_branching, get_entropy_x_si, get_entropy_x_s, get_idx_within_atgc, get_entropy_distance, get_profile and profile_branching. The code was written using a modular approach. Two external libraries were used which were 'random' and 'math'. The inputs were taken from console and subtle motif was printed in part1 and seed in part2. Screenshots of the run as shown in Fig. 5 and Fig.6.

```
C:\Python27\python.exe C:/Users/ankit/PycharmProjects/ACB_Assignment_3_4/MT16121_problem_4_algo1.py
Enter the number of sequences in the sample 2
Enter the sequence ATGCATGC
Enter the sequence CGTACGTA
Enter the length of the pattern 3
Enter the number of mutations (k) 2
Subtle motif is TTC

Process finished with exit code 0
```

Fig. 5. Pattern branching run

```
C:\Python27\python.exe C:/Users/ankit/PycharmProjects/ACB_Assignment_3_4/MT16121_problem_4_algo2.py
Enter the number of sequences in the sample 2
Enter the sequence ATGCATGC
Enter the sequence CGTACGTA
Enter the length of the pattern 3
Enter the number of mutations (k) 2
Seed is GCC

Process finished with exit code 0
```

Fig. 6. Profile branching run

REFERENCES

[1] A. Price, S. Ramabhadran, and P. A Pevzner, "Finding subtle motifs by branching from sample strings," vol. 19 Suppl 2, pp. ii149–55, 11 2003.