Development of Complex Curricula for Molecular Bionics and Infobionics Programs within a consortial* framework**

Consortium leader

PETER PAZMANY CATHOLIC UNIVERSITY

Consortium members

SEMMELWEIS UNIVERSITY, DIALOG CAMPUS PUBLISHER

The Project has been realised with the support of the European Union and has been co-financed by the European Social Fund ***

**Molekuláris bionika és Infobionika Szakok tananyagának komplex fejlesztése konzorciumi keretben

***A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

INTRODUCTION TO BIOINFORMATICS

(BEVEZETÉS A BIOINFORMATIKÁBA)

CHAPTER 11

Motif Finding Algorithms

(Motívum kereső algoritmusok)

András Budinszky

Motivation

To discover and understand the mechanisms that regulate gene expression is a major challenge in biology.

An important task in this challenge is to identify regulatory elements, especially the binding sites in DNA for transcription factors.

Definitions

Signals are DNA or RNA sequence patterns that are "recognized" proteins or other molecules.

A binding site of the transcription factor in the promoter region is such a signal.

A transcription factor can bind to several binding sites in the promoter regions of different genes to make these genes co-regulated, and such binding sites should have common patterns called motifs.

The task of motif finding is to locate these substrings of common patters in multiple sequences.

The Challenge of Motif Finding

The sequence of the motif is unknown.

Its location relative to the genes start is also unknown.

It can be located anywhere within the regulatory regions of the sequences, and the regulatory region typically stretching 100-1000 bp upstream of the transcriptional start site.

It may vary slightly across different regulatory regions since non-essential bases could mutate.

The Motif Finding Problem

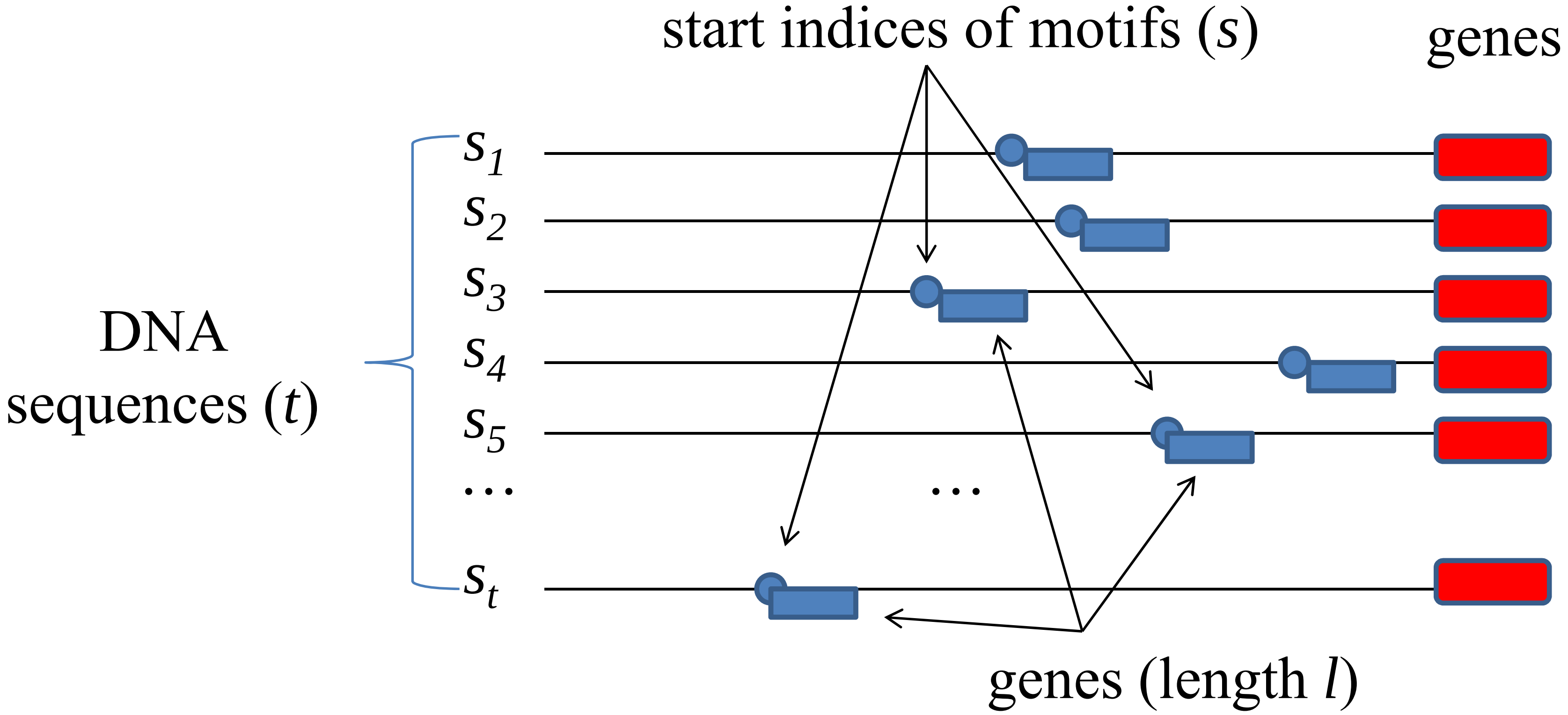
Given

- *t* number of DNA sequences, each contains *n* nucleotides,
- a fixed but unknown nucleotide sequence *M* (the motif) of length *l*,
- the integer value *d*.

Furthermore, each of the *t* DNA sequences contains a planted variant of *M* with at most *d* point substitution starting at unknown positions $s(s_1, s_2, \ldots, s_t)$.

The MFP is to determine these starting positions (*s*).

Motif Finding Problem, Example



Consensus Sequence

Once we know the starting positions, we can generate a consensus sequence; one that on each of its *l* position contains the nucleotide with the highest occurrence frequency in the corresponding positions in the *t* sequences .

We can think of the consensus sequence as an “ancestor” motif, from which the modified motifs in the sequences mutated.

Consensus Sequence, Example

T	A	G	T	G	A	T	T
A	A	A	T	G	A	T	G
C	T	A	T	G	A	C	T
C	A	A	T	A	G	T	T
C	A	A	T	T	C	T	T

alignment of motifs

A	1	4	4	0	1	3	0	0
C	3	0	0	0	0	1	1	0
G	0	0	1	0	3	1	0	1
T	1	1	0	5	1	0	4	4

frequency matrix

C A A T G A T T

consensus sequence

Evaluation of Motifs Alignment

We can introduce two evaluators as measure of alignment between the consensus sequence and the motifs in the sequences:

A. The sum of the highest occurrence frequencies (an indicator of the total similarities between the "ancestor" motif and the "implantations"):

$$\mathrm{TSim}(s, DNA) = \sum_{i=1}^{l} \max_{k \in \{A,T,C,G\}} count(k,i)$$

where *DNA* is the set of sequences,

l is the length of the motif, and

s is the vector of the starting points

Evaluation of Motifs Alignment (cont)

B. The sum of the Hamming distances (indicator of the total distances between the “ancestor” motif and the “implantations”):

$$\text{SDist}(s, DNA) = \sum_{i=1}^{t} d_H(c, s_i)$$

where *DNA* is the set of sequences,
s is the vector of the starting points,
t is the number of sequences, and
c is the consensus sequence

Note: Hamming distance is the number of point mutations (differences).

Evaluation Example

T	A	G	T	G	A	T	T
A	A	A	T	G	A	T	G
C	T	A	T	G	A	C	T
C	A	A	T	A	G	T	T
C	A	A	T	T	C	T	T

A	1	4	4	0	1	3	0	0
C	3	0	0	0	0	1	1	0
G	0	0	1	0	3	1	0	1
T	1	1	0	5	1	0	4	4

C A A T G A T T

$$\mathrm{TSim}(s, DNA) = 3 + 4 + 4 + 5 + 3 + 3 + 4 + 4 = 30$$

$$\mathrm{SDist}(s, DNA) = 2 + 2 + 2 + 2 + 2 = 20$$

Formal Definition of Motif Finding Problem

Given:

- a ***t*** x ***n*** matrix of ***DNA*** (a set of DNA sequences), and
- an integer ***l*** (the length of the pattern to find)

Find the starting points (a vector ***s***) of a set of subsequences of length ***l*** (***l***-mers), one in each sequence of ***DNA***, that maximizes the value of total similarity (TSim(***s***, ***DNA***)).

.

Another Formal Definition of Motif Finding

Given:

- a ***t*** x ***n*** matrix of ***DNA*** (a set of DNA sequences), and
- an integer ***ℓ*** (the length of the pattern to find)

Find a sequence of length ***ℓ*** (the so-called median string) that minimizes its total distance (SDist(***s***, ***DNA***)) to the set of sequences in ***DNA***.

Note: This problem is also known as Median String Problem.

Equivalence of MFP and MSP

The two problems (motif finding problem and median string problems) are computationally equivalent, because

$$\mathrm{TSim}(s, DNA) + \mathrm{SDist}(s, DNA) = t * l,$$

and thus maximizing TSim corresponds to minimizing SDist.

These problems are NP-hard and three categories of algorithms are known for solving them:

A. Brute-force

B. Clique search

C. Heuristic search

Motif Finding Problem, Brute-Force Algorithm

The idea of the algorithm is that it computes TSim for each possible starting point combination and picks the best.

```
A.  Initialize topSim to 0
B.  For each starting point combination
        B1. compute TSim
        B2. if TSim > topSim then
                save  TSim in topSim and
                      current start points in motif
C.  Return motif
```

.

Median String Problem, Brute-Force Algorithm

The idea of the algorithm is that it computes SDist for each possible median string (all combinations with length of *l)* and picks the best.

```
A.  Initialize minDist to ∞
B.  For each possible median string
        B1. compute SDist
        B2. if SDist < minDist then
                save  SDist in minDist and
                      current median string in median
C.  Return median
```

•

Run Time Analysis

Motif Finding:

- Varying $(n - \ell + 1)$ positions in each of $\boldsymbol{t}$ sequences, we have $(n - \ell + 1)^t$ starting positions.
- For each starting positions, TSim makes $\boldsymbol{\ell * t}$ operations, so complexity is

$$\ell t(\mathrm{n} - \ell + 1)^t = O(\ell \boldsymbol{tn}^t)$$

- For a good size problem (for example $t = 20$, $n = 600$, and $\ell = 15$) this would take unacceptable amount of time.

Run Time Analysis

Median String Search:

- Varying the median string candidates, we have 4^{ℓ} combinations
- For each median string candidate, minDist makes $t(n - \ell + 1)$ operations, so complexity is

$$t(n - \ell + 1)\ 4^{\ell} = O(\boldsymbol{nt}4^{\ell})$$

- For a good size problem this would also take unacceptable amount of time.

Branch and Bound Improvement

With the branch-and-bound technique, some improvement of the Motif Finding Algorithm can be achieved:

- While iterating the starting position combination, if the TSim value for a substring of length $k < \ell$ (a prefix of the motif being checked) is hopelessly small (that is, - with an optimistic assumption – even if the rest of the characters all match, and therefore they contribute the largest possible value to Tsim, it will be smaller than *topSim*), then the algorithm can trim the search space and directly skip to the next prefix combination.

Similarly, branch-and-bound can be used for MSP algorithm.

Motif Finding Problem, Greedy Algorithm

Steps:

A. Find the two closest *ℓ*-mers in the first 2 sequences and form a *2 x ℓ alignment matrix* with TSim(**s**,2,DNA).

B. Initialize *i* to 3.

C. For each of the rest of *t-2* sequences

   C1. find an *l*-mer in sequence *i* maximizing

   TSim(**s**,i,DNA) using the already constructed *(i-1) x l* alignment matrix for the first *(i-1)* sequences

   C2. increment *i* by 1

This is the CONSENSUS algorithm that sacrifices optimal solution for speed.

Motif Finding Problem, Clique Search Algorithm

Steps:

A. Construct a *t*-partite graph as

- Each partite belongs to one of the DNA sequences and contains vertices representing substring of length *l* (a total of (*n*-*l*+1).
- Two vertices in different partites are connected by an edge if the Hamming distance between the two corresponding substrings is at most 2d.

B. Find a clique of size *t* in this graph.

Motif Finding Problem, Heuristic Search Algorithm

The principle idea of these types of algorithms:

A. First they find a set of subsequences of length *l* with high probability of being the motif.

B. Then they refine these sequences by some local searching techniques.

Note: These algorithms may solve problems that would take access amount of time with a brute-force algorithm, there is no guarantee that the motif can be found.

Motif Finding Problem, Voting Algorithm

Steps:

```
A.  Create two hash tables Vote and Record with entry values 0.
B.  Create an empty set (this will collect the motifs found).
C.  For each sequence in DNA ($i$ runs from 1 to $t$)
        C1. For each $l$-length substring of it (from pos 1 to ($n$-$l$+1))
                C11. Create all variants that is at most $d$ distance
                        away from that substring
                C12. For each variant
                        C121. If Record[H(variant)] $\neq i$
                                then add 1 to Vote[H(variant)]
                                    set Record[H(variant)] to $i$
    (Note: H(s) is a hashing function)
```

Motif Finding Problem, Voting Algorithm (cont)

Steps (continued):

```
D.  For each l-length substring of sequence 1
          (from position 1 to (n-l+1))
      D1. Create all variants that is at most d distance
          away from that substring
      D2. For each variant
          D21. If Vote[H(variant)] = t
                  (that is, being a variant in all sequences)
                  then include this variant into result set C.
```

Note: It runs faster than the brute-force algorithm (O(***nt***(**3*l***)^{d}), but its space requirement increases exponentially with *d*.

Voting Algorithm with Projection

The voting algorithm discussed on the previous two slides exhaustive search algorithm time and space permits (see note at the end of previous slide).

The algorithm can be modified to a heuristic version which can work much larger *l* and *d* (when *l* > 15 and *d* > 5) as well.

Instead of all positions, it considers only *l'* of the *l* positions (a projection) of the motif. Based on the voting results on these l' positions, the motif of length *l* can with high probability be found.

Motif Databases

There are various databases that contain known motifs which can be searched for:

- Different JASPER databases (CORE, PHYLOFACTS, FAM, POLII, CNE, SPLICE). Collections of transcription factor DNA-binding preferences, modeled as matrices. The only databases where the data can be used with no restrictions (open-source).
- TRANSFAC. Contains data on transcription factors, their experimentally-proven binding sites, and regulated genes. Its broad compilation of binding sites allows the derivation of positional weight matrices.
- PROSITE. Primarily a protein database, but it contains protein motifs.

Motif Finding Programs

CONSENSUS	*Hertz, Stromo (1989)*
GibbsDNA	*Lawrence et al (1993)*
MEME	*Bailey, Elkan* (1995)
WINNOWER	*Pevzner, Sze (2000)*
RandomProjections	*Buhler, Tompa* (2002)
MULTIPROFILER	*Keich, Pevzner (2002)*
MITRA	*Eskin, Pevzner (2002)*
Pattern Branching	*Price, Pevzner (2003)*
Improved Voting	*Chin, Leung (2005)*