Prezentácie

9.5. Hozza, Zajacová, Anderle

15.5. Ruhalovský, Hajdin, Gibaštíková

16.5. Janočko, Herencsár, Žák

Prineste si pdf na USB kľúči alebo vlastný počítač,

rady k prezentácii na stránke

Shortest common superstring

Input: $S_1, S_2, \ldots, S_k \in \Sigma^*$

Goal: Shortest string S such that each S_i is a substring of S.

Greedy approximation algorithm

1. remove words which are substring of another word
2. find words S_i and S_j with longest overlap,
   i.e. longest α such that $S_i = \beta\alpha$, $S_j = \alpha\gamma$ for some $i \neq j$
3. connect S_i and S_j to $\beta\alpha\gamma$
4. repeat steps 2 and 3 until only one word left

Efficient implementation

- Given S_i and its suffix tree and given S_j,
  find longest prefix of S_j which is a sufix of S_i
- Given a suffix tree for $S_1, \ldots, S_k$, and S_j
  find longest prefix of S_j which is a suffix of some S_i, $i \neq j$
- Find largest overlap
- Running time of the whole algorithm?

Number of strings k

Total length of string $n = \sum_{i=1}^{k} |S_i|$

Multiple sequence alignment (viacnásobné zarovnanie)

Align several sequences

```
     Human  ctccatagcaatgt-cagagatagggcagagcggat------ggtggtgac
    Rhesus  ctccatggcaatgt-cagagatagggcagagcggat------gctggtgac
     Mouse  ttt--tgacaaca--tagagac-tgagatagaaaat-------atgctgac
       Dog  -tccccgctaatgtacaaagatggggcag-gaaga--a----tgtgctgaa
     Horse  -tccacggcaatac-tggagatggggcagagcaga--agat-ggtgatgaa
 Armadillo  ctgcatagaaatct-cagagatgggggaaagcaga------agacattcat
   Opossum  atccatggaaacat-cagaagtgggagaaatagaaga----tggcaatga-
  Platypus  acccggggaagggg-aagaggaagggccggccg-----------------
```

- Hypothesis about evolution:

  all letters in a column evolved from a common ancestor

- Is gap insertion or a deletion?

- Positions that do not change perhaps important

Score of a multiple alignment

For pairwise alignment, assign score to each column

using matrix $w(a, b)$, $a, b \in \Sigma \cup \{-\}$.

Different ways how to extend to k sequences

Sum-of-pairs scoring: $w(A) = \sum_{i<j} d_A(S_i, S_j)$

$d_A(S_i, S_j)$ is cost of alignment of S_i, S_j induced by A

(skip columns with two -)

```
S1: ACAGT-A
S2: A-ACT-C
S3: GCACTGA
```

$$w(A) = d_A(S_1, S_2) + d_A(S_1, S_3) + d_A(S_2, S_3) = 3 + 3 + 4 = 10$$

Multiple alignment in practice

- Pruning parts of dynamic programming matrix
  – skip parts that cannot (or are unlikely to) belong to the shortest path
- Progressive alignment
  – successively align pairs of sequences or pairs of alignments
- Anchors
  – find string with approximate occurrences in all input strings
  – align with each other
  – align left and right parts of sequences
- Approximation algorithms

Motif finding (hľadanie motívov)

Input: Strings $S_1, \ldots, S_k$, motif length L

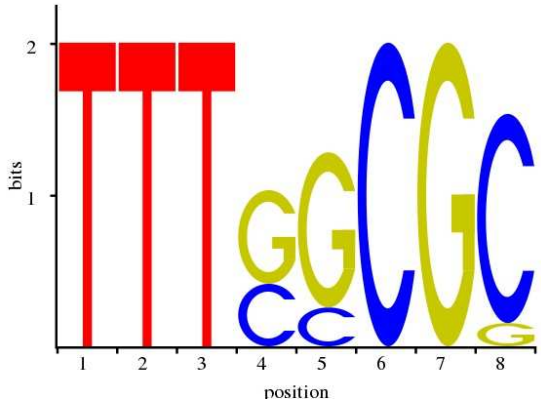
Goal: Motif of length L that has (exact/approximate) occurrences in all (or as many as possible) strings S_i

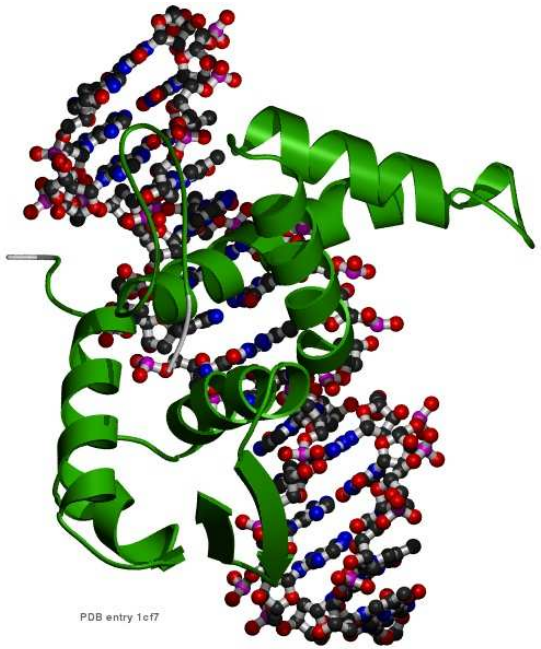
Many formulations depending on definition of

- motif (string, regular expression, scoring matrix)
- motif occurrence (exact, Hamming distance, edit distance, score)
- goal (occurrences in all/many sequences, strength of the motif)

Motif finding in bioinformatics

- Motif occurrences: places where proteins bind DNA
- Motif corresponds to binding preferences of a protein
- Example: transcription factor E2F1
  – regulates cell cycle
  – binds e.g. TTTCCCGC, TTTCGCGC





- Experimentally get approximate positions of binding sites
  Discover motif common to many such sites

Problem 1

Motif = string of length L, exact occurrences

Input: Strings $S_1, \ldots, S_k$, motif length L

Output: A string S of length L that occurs as a substring in the highest number of sequences from $\{S_1, S_2, \ldots, S_k\}$.

How to do this efficiently?

Problem 2: Consensus Pattern Problem

Allow approximate occurrences, consider Hamming distance, require an occurrence in each S_i.

Input: Strings $S_1, \ldots, S_k$, motif length L

Output: A string S and occurrences $i_1, \ldots, i_k$ minimizing $c(S, i_1, \ldots, i_k) = \sum_{j=1}^{k} d_H(S, S_j[i_j \ldots i_j + L - 1])$.

NP-hard problem

We can compute in polynomial time:

For given $i_1, \ldots, i_k$, compute $\min_S c(S, i_1, \ldots, i_k)$

For a given S, compute $\min_{i_1,\ldots,i_k} c(S, i_1, \ldots, i_k)$

Problem 2: Consensus Pattern Problem

Simple algorithm

For each substring S of length L in $S_1, \ldots, S_k$:

compute $c_S = \min_{i_1,\ldots,i_k} c(S, i_1, \ldots, i_k)$

Choose S with the smallest c_S

Theorem. This algorithm has approximation ratio at most 2.

Heuristic iterative improvement:

for a set of occurrences $i_1, \ldots i_k$ find best S

for this S find best occurrences $i_1, \ldots i_k$

Problem 2: Consensus Pattern Problem

Polynomial-time approximation scheme (PTAS):

For each r-tuple $(\alpha_1, \ldots, \alpha_r)$

where α_j is a substring of length L of some S_i:

– compute $S = \arg\min_S \sum_j d_H(S, \alpha_j)$

– compute $c_S = \min_{i_1,\ldots,i_k} c(S, i_1, \ldots, i_k)$

Choose S with the smallest c_S

Theorem. This algorithm has approximation ratio at most $1 + \frac{4\sigma-4}{\sqrt{e}(\sqrt{4r+1}-3)}$ and running time $O(n^{r+1}k^{r+1}L)$.

Problem 3: Closest Substring

As problem 2, change scoring.

Input: Strings $S_1, \ldots, S_k$, integers L, k

Output: Is there a string S and occurrences $i_1, \ldots, i_k$ such that $d_H(S, S_j[i_j \ldots i_j + L - 1]) \leq k$?

NP-hard problem.

The same 2-approximation algorithm (if each occurrence of distance $\leq k$ from S, distance to each other $\leq 2k$)

Closest string problem:

Input: Strings $S_1, \ldots, S_k$ of the same length L, integer k

Output: Is there a string S such that $d_H(S, S_j) \leq k$?

This problem is also NP-hard.