

Performing local similarity searches with variable length seeds

Miklós Csűrös

Département d'informatique et de recherche opérationnelle
Université de Montréal

Seed-and-extend

Comparison of molecular sequences $S, T \in \Sigma^*$

Goal: find **local alignments**, i.e.,
 $S[i..i']$ and $T[j..j']$ with high similarity score

Dynamic programming (Smith-Waterman) $O(|S| \cdot |T|)$ too slow

Seed-and-extend heuristics
(BLAST, Fasta, SSAHA, SENSEI, BLAT, BLASTZ, PatternHunter, ...)

Hash function $h: \Sigma^\ell \mapsto \Sigma^k$
 k -mer: $\ell = k$ and $h(u) = u$

Hit at (i, j) if $h(S[i..i + \ell - 1]) = h(T[j..j + \ell - 1])$

Spaced seed

Spaced seed: ordered set $\mathcal{S} = \{s_1, \dots, s_k\}$; $s_i \in \{1, \dots, \ell\}$

Corresponding hash function: $h(u) = u[s_1] \cdot u[s_2] \cdots u[s_k]$

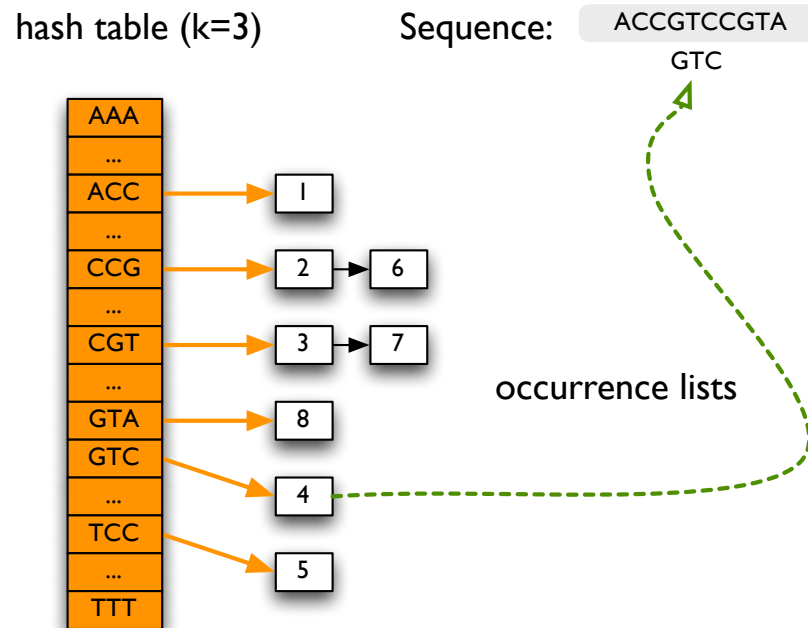
Gives higher sensitivity than a k -mer

Number of hits: $\frac{NM}{|\Sigma|^k}$ where $N = |S| - \ell + 1$ and $M = |T| - \ell + 1$

In order to find more similarities, you need to decrease k :
exponential increase in running time

Hash table

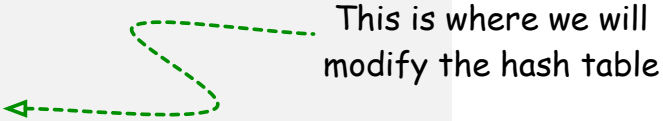
Table: **key occurrence** in S for each key



General seed-and-extend algorithm

List of positions in which key u occurs: $\text{Occ}(u)$

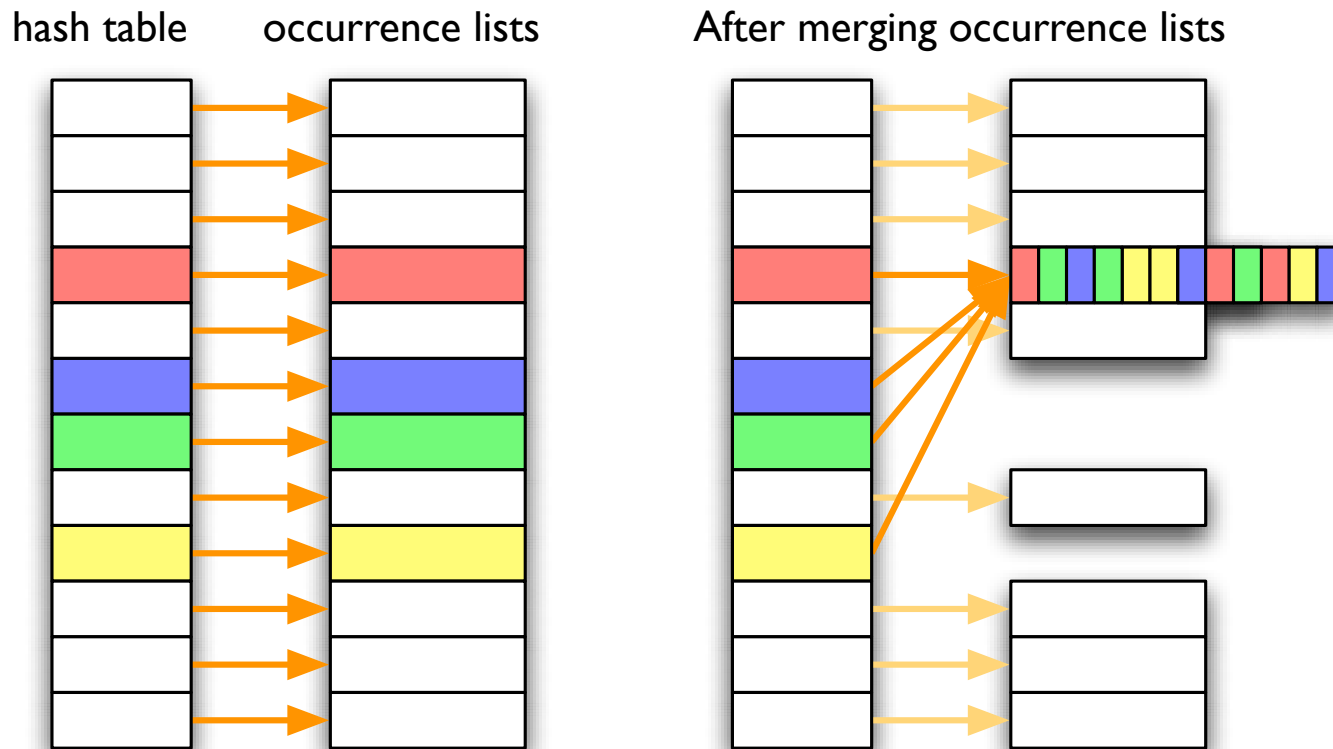
```
1. for  $i = 1, \dots, |S| - \ell + 1$  do
2.   set  $\text{key} \leftarrow h(S[i..i + \ell - 1])$ 
3.   add  $i$  to the list  $\text{Occ}(\text{key})$ 
4. end for
5. for  $j = 1, \dots, |T| - \ell - 1$  do
6.   set  $\text{key} \leftarrow h(T[j..j + \ell - 1])$ 
7.   process the hits  $(i, j) : i \in \text{Occ}(\text{key})$ 
8. end for
```



This is where we will modify the hash table

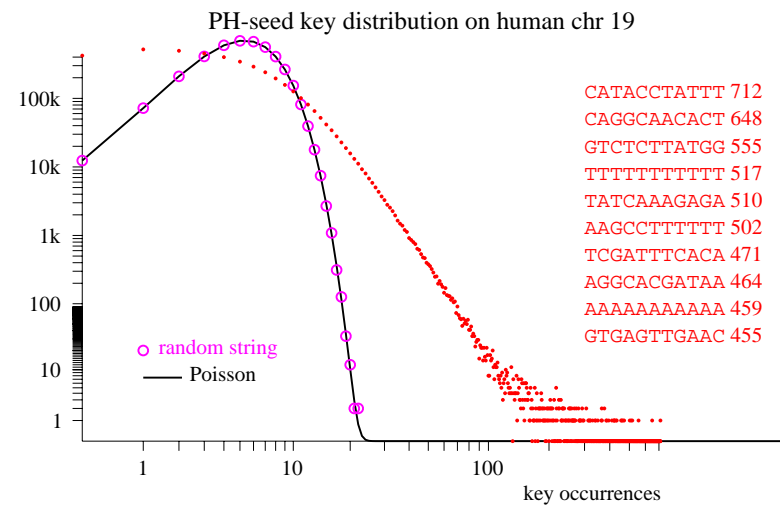
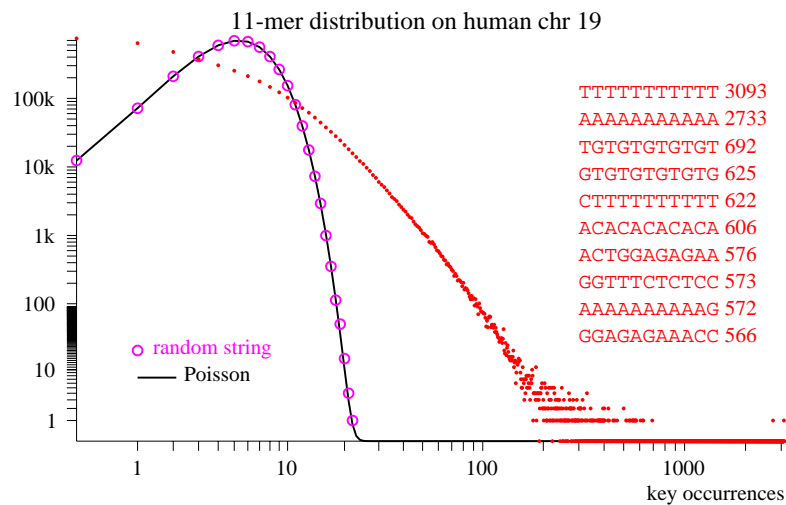
Modifying the hash table

merging occurrence lists (oblivious to second stage)



Not all keys are created equal

Repeated sequences, frequent signals, low complexity regions

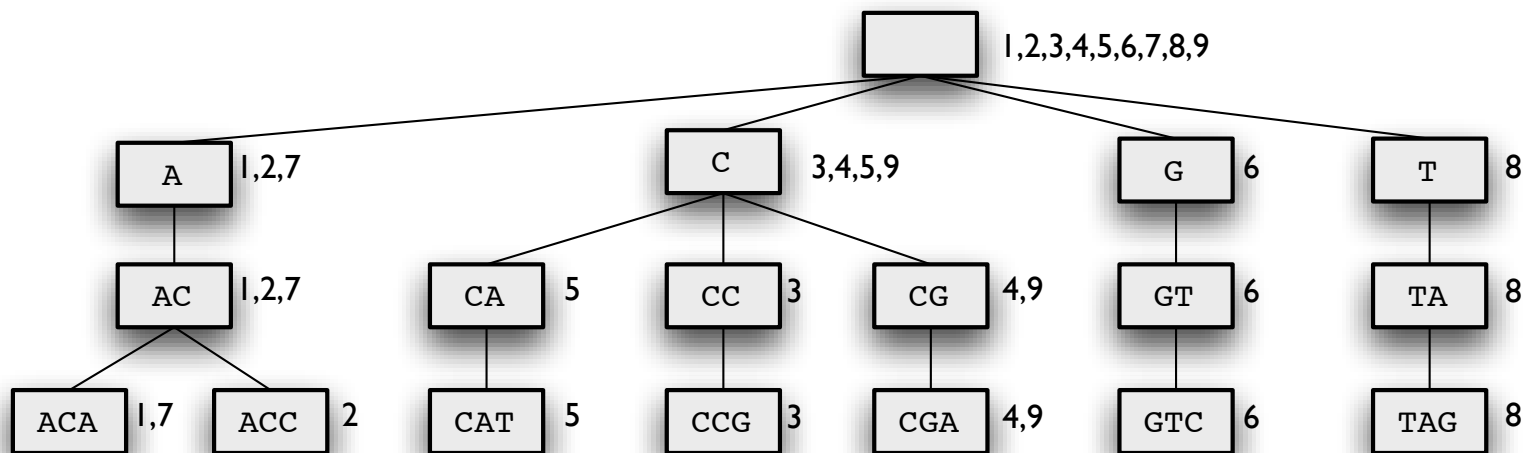


Seed tree

[only conceptual]: hash keys put in a trie

S = AACACGATCAGA

seed: {1,3,4}



Pruning the tree

Pruning: select a node v whose children are all leaves
remove the children \rightarrow merge their occurrence lists:

$$\text{Occ}(v) = \cup_{\sigma \in \Sigma} \text{Occ}(v \cdot \sigma)$$

Def. $n(v) = |\text{Occ}(v)|$, number of times key v occurs in S

Predicted number of hits for key v : $\text{hits}(v) = n(v) M \prod_{i=1}^{|v|} q_{v[i]}$,
where q_{σ} is the frequency of character σ in T

Increase in number of hits after pruning:

$$\text{hits}^+(v) = \text{hits}(v) - \sum_{\sigma \in \Sigma} \text{hits}(v \cdot \sigma) = M \left(\prod_{i=1}^{|v|} q_{v[i]} \right) \sum_{\sigma \in \Sigma} (1 - q_{\sigma}) n(v \cdot \sigma).$$

Predict the effect of pruning

Want to prune as many nodes as possible, as each pruning increases the sensitivity, while keeping the increase in hits low

Lemma. If $q_\sigma \leq 1/2$ for all $\sigma \in \Sigma$, then $\text{hits}^+(v) \geq \text{hits}^+(v \cdot x)$ for all $x \in \Sigma$.

\Rightarrow can greedily select nodes for pruning using $\text{hits}^+(v)$ alone, there is no need for keeping track of whether the children are all leaves

Procedure:

1. Calculate $\text{hits}^+(v)$ for every non-leaf node v
2. Keep selecting v with minimum $\text{hits}^+(v)$ and prune at v until a threshold on total hit increase is surpassed

First phase: estimation

Calculate $n(u)$ and $\text{hits}^+(u)$ level-by-level, upwards in the tree

Want greedy selection — use **binning**: put u in $\text{BinList}[b(\text{hits}^+(u))]$, where $b: \mathbb{R} \mapsto \{0, 1, \dots, B\}$ is a monotone binning function (avoids the need for sorting)

Second phase: merging occurrence lists

(We keep merging occurrence lists = pruning the seed tree)

Merge occurrence lists by going through each bin, from $\text{BinList}[0]$ to $\text{BinList}[B]$

Keep track of total increase in hits

Stop when number of hits increases by a factor of R where R is an input parameter

Optimizing the order of sampled positions

Spaced seed: **ordered set** $\mathcal{S} = \{s_1, \dots, s_k\}$

If pruning at level $(k - 1)$: as if hashing with spaced seed $\{s_1, \dots, s_{k-1}\}$

Define the seed chain $\mathcal{S}_k, \mathcal{S}_{k-1}, \dots, \mathcal{S}_0$ by $\mathcal{S}_k = \mathcal{S}$ and $\mathcal{S}_{t-1} = \mathcal{S}_t - \{s_t\}$

Problem: Given an unordered set of positions, find the “best” seed chain — select \mathcal{S}_{t-1} based on \mathcal{S}_t so that it maximizes sensitivity

[Sensitivity calculated as in Nicodème et al. (1999) or Buhler et al. (2003)]

Example: best ordering of PatternHunter’s seed:

2, 3, 8, 10, 13, 14, 5, 1, 16, 17, 18

Memory requirements

Number of non-leaf nodes in the seed tree: $\frac{|\Sigma|^k - 1}{|\Sigma| - 1}$

Every node is identified by an integer

Storing $n(u)$: $\frac{|\Sigma|^k - 1}{|\Sigma| - 1}$ integers

Binning by hits⁺: $(B+1) \cdot \frac{|\Sigma|^k - 1}{|\Sigma| - 1}$ integers

About 11 Mbytes for $k = 11$ and $|\Sigma| = 4$

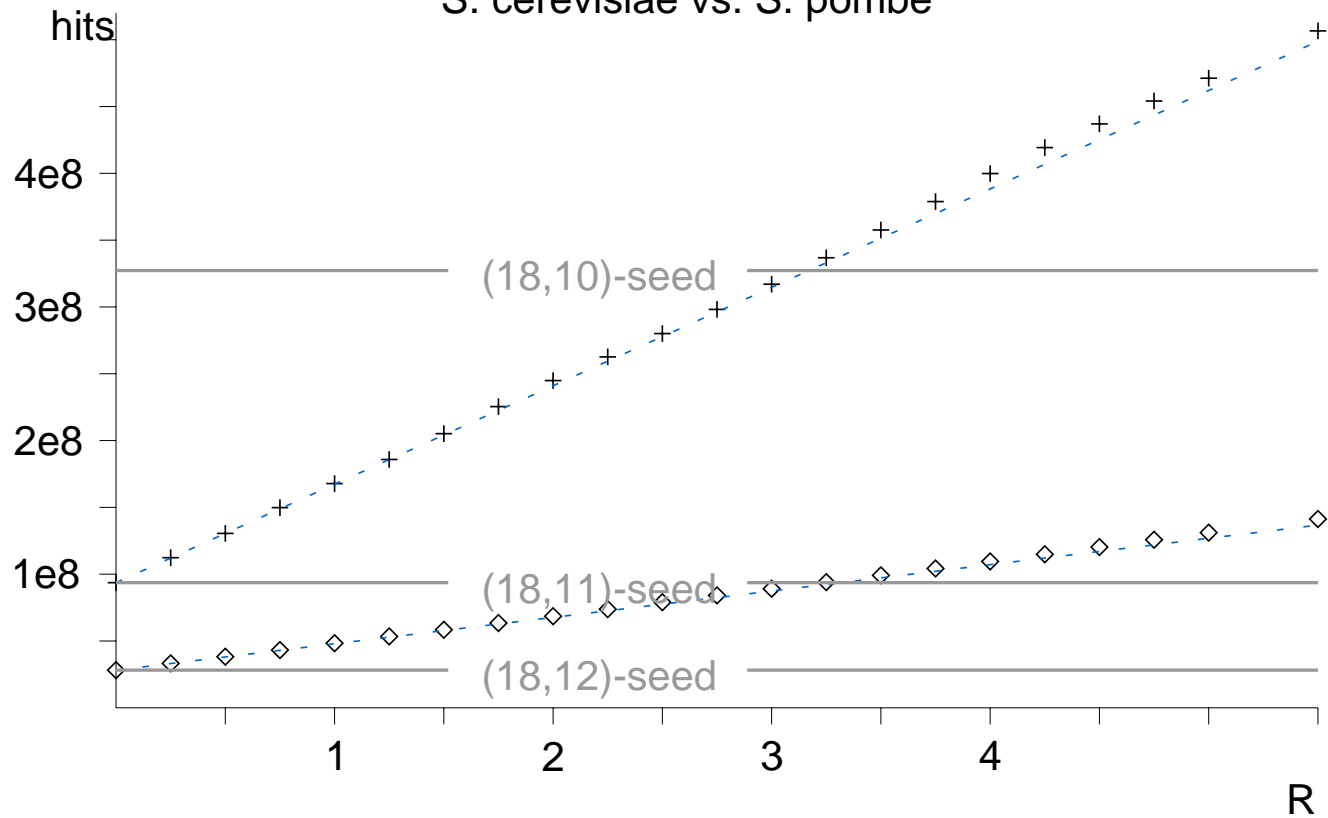
Running time

Takes $O(k|S| + k|\Sigma|^k)$ in worst-case (all nodes pruned), and
 $O(|S| + k|\Sigma|^{k-2})$ in practice

	Table	Pruning	Extensions
<i>H. influenzae</i> – <i>E. coli</i> , (18,12)-seed $R = 0$	2s	0s	8s
<i>H. influenzae</i> – <i>E. coli</i> , (18,12)-seed $R = 3.5$	2s	9s	15s
<i>H. influenzae</i> – <i>E. coli</i> , (18,11)-seed $R = 3.5$	2s	3s	31s
<i>S. cerevisiae</i> – <i>S. pombe</i> , (18,12)-seed $R = 0$	13s	0s	3m 31s
<i>S. cerevisiae</i> – <i>S. pombe</i> , (18,12)-seed $R = 3.5$	14s	21s	6m 27s
<i>S. cerevisiae</i> – <i>S. pombe</i> , (18,11)-seed $R = 3.5$	12s	13s	32m 24s
Human chrX–rat chrX, (20,13)-seed $R = 0$	82s	0s	31m
Human chrX–rat chrX, (20,13)-seed $R = 3.5$	76s	6m 34s	78m
Human chrX–rat chrX, (18,12)-seed $R = 3.5$	119s	105s	4h 21m

Yeasts

S. cerevisiae vs. S. pombe



Yeasts 2

HSP=High-Scoring Segment Pair, a high-scoring gapless alignment

S. cerevisiae vs. *S. pombe*

