18.417 Introduction to Computational Molecular Biology

Lecture 13: October 21, 2004 — Scribe: Eitan Reich
Lecturer: Ross Lippert — Editor: Peter Lee

13.1 Introduction

We have been looking at algorithms to find optimal scoring alignments of a query text Q with a database T. While these algorithms have reasonable asymptotic runtimes, namely $O(|Q||T|)$, they are impractical for larger databases such as genomes. To improve the runtime of our algorithms, we will sacrifice the optimality requirement and instead use smart heuristics and statistical significance data to find "close to optimal" alignments and quantify how significant such alignments are.

13.2 Inexact Matching

The inexact matching problem is to find good alignments while allowing for limited discrepancies in the form of substitutions, insertions and deletions. The problem can be stated in the following two ways, which are slight variations of each other:

Definition 1 (k-mismatch l-word problem) *Given distance limit k, word length l, query string q, and text string t, return all pairs of integers (i, j) such that $d(q_i \cdots q_{i+l}, t_j \cdots t_{j+l}) <= k$, where d is the Hamming distance function.*

Definition 2 (S-scoring l-word problem) *Given score requirement S, scoring function δ, word length l, query string q, text string t, return all pairs of integers (i, j) such that $\Delta(q_i \cdots q_{i+l}, t_j \cdots t_{j+l}) >= S$ where $\Delta(x, y) := \sum_i \delta(x_i, y_i)$.*

13.3 Pigeonhole Principle

A key insight in the inexact matching problem is that wherever there is a good approximate alignment, there will be smaller exact alignments. For example, if there

is an alignment of 9 bases with at most 1 mismatch, there must be an exact alignment of at least 4 bases. The pigeonhole principle is used to generalize this idea and quantify exactly what type of exact alignments we can be guaranteed to find, given the type of inexact alignment. To locate good approximate matches, we instead look for the

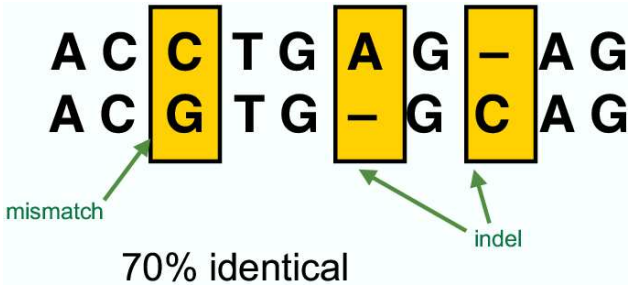

Figure 13.1: Good matches must contain exact matches.

exact matches that we would expect to find in longer inexact alignments. These exact matches can then be extended to produce longer matches that are may no longer be exact, but may still be within the given discrepancy threshold. More specifically, we can locate exact matches and extend them, using the k-mismatch algorithm:

1. We know that wherever there is a k-mismatch l-word, there is at least an exact match of length s where $s = \lfloor l/(k+1) \rfloor$.
2. We can look for potential alignment locations by finding all s-words that match exactly between the query string and the text.
3. These s-words can then be extended to the left and the right to find an l-word within k mismatches. To do this extension correctly takes $O(l^2)$ time, although many methods dont actually achieve this run-time.

or using the using the S-scoring algorithm as follows for the k-mismatch problem or the s-scoring problem:

1. We know that wherever there is an S-scoring l-word match, there must be some s-word match with score threshold T (from exact matching)
2. To locate potential high scoring locations, we form T-scoring neighborhoods of the s-words in the query text.
3. All neighborhood words in T are then found by exact matching.

4. Each of these s-words can then be extended to the left and right to find an l-word with score at least S.

To extend exact matching s-words to the left and right to produce approximately matching l-words, we can use either ungapped extension, or gapped extension.

Using ungapped extension, the exact matches are extended to the left and right without allowing any insertions or deletions. If two bases do not match, there is simply a mismatch that counts towards the k-mismatch limit or will penalize the score in the S-scoring problem. Using gapped extension, the exact matches are extended

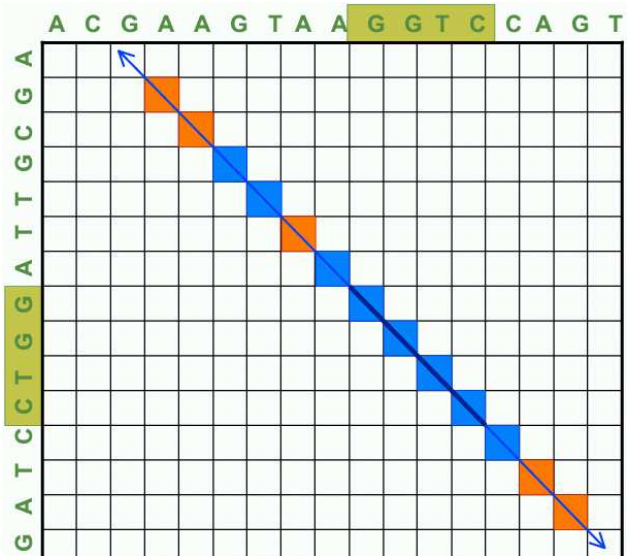

Figure 13.2: Ungapped extension.

to the left and right allowing mismatches, insertions and deletions. While the exact matching s-word that we begin with has no insertions or deletions, the approximate match that is produced by extension allows for such imperfections which will count toward the mismatch limit or penalize the score in the S-scoring problem:

13.4 BLAST

BLAST, or Basic Local Alignment Search Tool, is the successor to two simpler tools: FASTA, a nucleotide alignment tool, and FASTP, a protein sequence alignment tool.

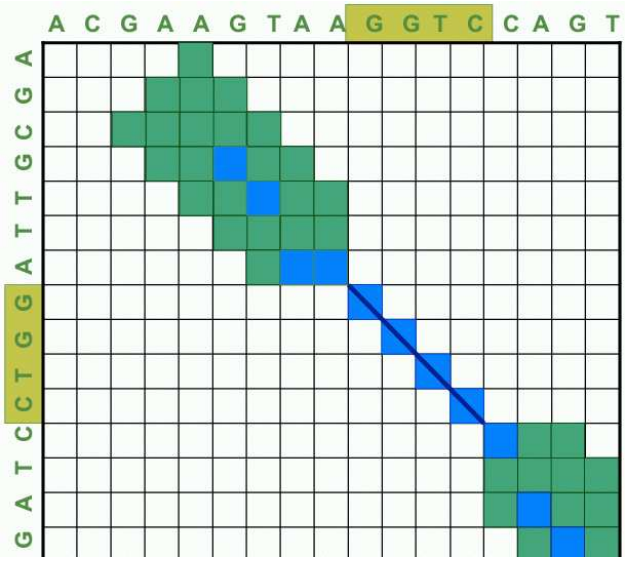

Figure 13.3: Gapped extension.

Like its predecessors, BLAST works by starting with a seed, and then, using a extension heuristic, finds approximate matches from shorter exact matches. What is so innovative about BLAST, however, is its incorporation of statistical measures along with alignment results to tell how statistically significant an alignment is. In addition, given a query and text string of any length, BLAST can find maximal scoring pairs (MSPs) very efficiently. While BLAST originally just returned MSPs, it now also returns alignments after extension.

Fact 1 (Altschul-Karlin statistical result) *If our query string has length n and our text has length m, then the expected number of MSPs with score greater than or equal to S is:*

$$E(S) = Kmne^{-\lambda S}$$

where K is a constant and λ is a normalizing factor that is a positive root of the following equation:

$$\sum_{x,y \in \alpha} p_x p_y e^{\lambda \delta(x,y)} = 1$$

where p_x is the frequency of character x from our alphabet α and δ is our scoring function.

Fact 2 (Chen-Stein statistical result) *The number of MSPs forms a Poisson distribution with average value $E(S)$.*

13.4.1 Variations

- BLASTn: nucleotide to nucleotide database alignment
- BLASTp: protein to protein database alignment
- BLASTx: translated nucleotide to protein database alignment
- tBLASTn: protein to translated nucleotide database alignment
- tBLASTx: translated nucleotide to translated nucleotide database alignment

13.5 Filtration

The idea behind filtration is to find seeds that will locate every MSP.

Given a proposed alignment, we can produce a binary string where there is a 1 whenever the two strings match and a 0 if they do not. This binary string gives us the matching rate of the alignment. An MSP is an alignment whose associated binary string has a high proportion of 1s. For example, here is an MSP of length 20 where more than 70% of the bases are aligned as matches:

```
Q a a t c t t g c g a g a c c a a t g g c a c t t
T c t t c c t g c g g g a c c t a t c c c a c a a
= 0 0 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 0 0 1 1 1 0 0
```

If to locate MSPs, we look for an alignment that would produce an exact match of 4 bases (as we might have derived using the pigeonhole principle), we would be using the seed 1111, which corresponds to finding a location in the binary string where there are 4 consecutive 1s. We can see that if we choose our seed to be too small (for example choosing 11 as our seed), we would hit too many locations in the string to be a statistically significant alignment, since many short exact matches can occur by chance and have nothing to do with the existence of an MSP. On the other hand, if we choose our seed to be too long, we will miss many MSPs because there may still be slight mismatches in an MSP that would cause the seed to reject that location.

A creative idea to allow the seed to account for slight mismatches, but at the same time not pick up too many alignments that are occurring merely due to chance is to used gap seeds.

Instead of looking for 1111, we can look for 11011. We can see the effectiveness of the gapped seed as opposed to the consecutively spaced seed in the example because the consecutively spaced seed misses a lot of locations in the MSP and almost fails to hit it altogether. With the gapped seed however, a small amount of error is allowed by introducing a dont care bit, and there are now many more hits on the MSP to ensure that it isn't missed. The idea of using gapped seeds rather than simple consecutively spaced seeds increases the effectiveness of MSP search methods by not allowing random noise to produce too many hits while at the same time ensuring that MSPs are hit.