

Error-Tolerant Read-Mapping to Burrows-Wheeler data

[redacted]

[redacted]

Columbia University

Overview

- What have we been doing?
 - Read Mapping with BWT
 - Most methods use exact “seed” segment
 - Inexact matching
 - Simulated traversal of the suffix tree using BWT with FM indexing and suffix arrays
 - adaptive substitution matrix, adaptive gap penalty weights, match rewards, match scoring, mismatch lower bound pruning, index-score pruning
 - Applications:
 - Alignment to reads or references with errors
 - Alignment to reads from another closely related species

Methods: Inexact Matching

- Deliverable: command-line tool for mapping reads, set of scripts for testing
 - Use of recursive program structure to bypass need to explicitly construct tree
 - Gives list of matches, scored, with their positions and suffixes
- Use of FM indexing:
 - Basically, store useful things for backward search
 - rank of each letter in the transform
 - number of lexicographically smaller symbols in the reference
 - Suffix array
- Search by keeping track of the error score on each suffix array interval that gets checked for a match

```

def inexact_recursion(s,i,diff,k,l, prev_type):
    '''search bwt recursively and tolerate errors'''

    global num_prunes

    #pruning based on estimated mistakes
    if diff < get_D(i):
        num_prunes += 1
        return set()

    #end of query condition
    temp = set()
    if i < 0:
        for j in range(k,l+1):
            temp.add((j,diff))
        return temp

    #search
    sa_idx = set() #set of suffix array indices at which a match starts

    if not NO_INDELS:
        # Insertion
        if prev_type == Type.INSERTION:
            sa_idx = sa_idx.union(inexact_recursion(s,i-1,diff-gap_ext,k,l,Type.INSERTION))
        else:
            sa_idx = sa_idx.union(inexact_recursion(s,i-1,diff-gap_ext-gap_open,k,l,Type.INSERTION))

    for char in alphabet:
        temp_k = C[char] + get_0(char, k-1) + 1
        temp_l = C[char] + get_0(char, l)

        if temp_k <= temp_l:
            if not NO_INDELS:
                # Deletion
                if prev_type == Type.DELETION:
                    sa_idx = sa_idx.union(inexact_recursion(s,i,diff-gap_ext,temp_k,temp_l,Type.DELETION))
                else:
                    sa_idx = sa_idx.union(inexact_recursion(s,i,diff-gap_ext-gap_open,temp_k,temp_l,Type.DELETION))
            if char == s[i]:
                # Match!
                sa_idx = sa_idx.union(inexact_recursion(s,i-1,diff+match,temp_k,temp_l,Type.MATCH))
            else:
                # Mismatch
                if sub_mat:
                    sa_idx = sa_idx.union(inexact_recursion(s,i-1,diff-mismatch*sub_mat[(s[i],char)],temp_k,temp_l, Type.MISMATCH))
                else:
                    sa_idx = sa_idx.union(inexact_recursion(s,i-1,diff-mismatch,temp_k,temp_l, Type.MISMATCH))

```

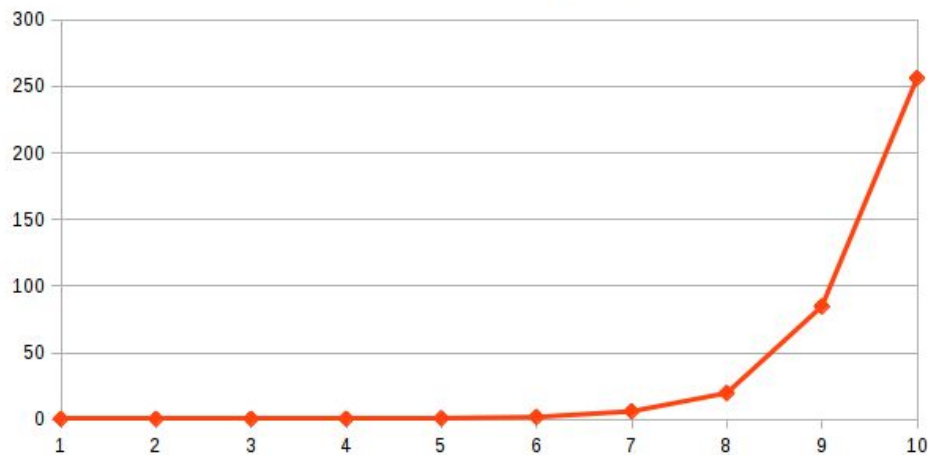
Methods: Scoring and Pruning Matches

- Accept substitutions and indels
- Estimate substitution matrix based on the likelihood of each replacement occurring for each index of the read
- Affine and linear gap penalties based on read and genome length
- Lower bound on number of mismatches of each substring $[1:i]$ of the read
 - Allows pruning when our threshold dips below that level
- Prune examples with the same index and have lower scores

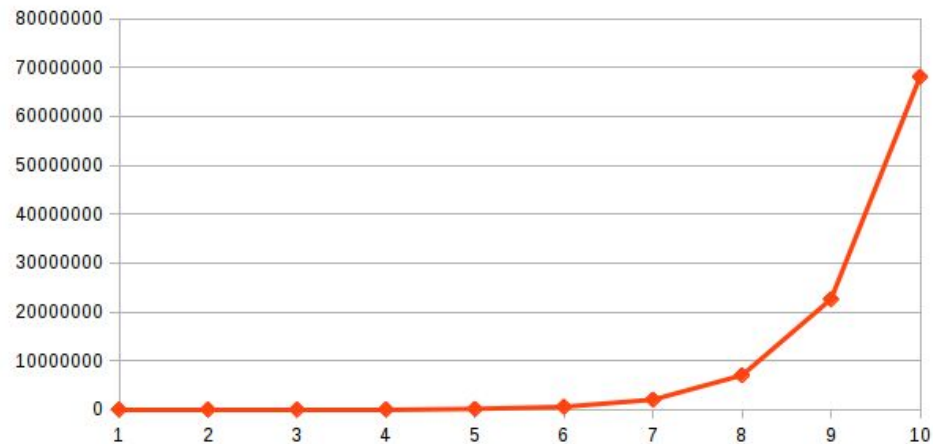
Results: Efficiency

- Algorithmic complexity:
 - $O(2^t)$ for Threshold t in the worst case
 - $O(r)$ time and space for read length r
- Test: Error Threshold Value, Runtime, Number of Nodes Pruned

Error Threshold vs Time (160Bp)



Threshold vs Number of Nodes Pruned



Results: Accuracy

- Although we only had time to test on a small set, our preliminary results are encouraging:

Program	Error Rate	Time (s)
Our Tool	.2	58
BWA-32	.3	65
BWA-125	.05	88

- 150bp reads mapped to Dengue virus genome with 83% of estimated read positions matching true read positions

[illegible]

Conclusions

- Promising results so far
 - Method seems to be viable for the ascribed problem
- Prospective future work
 - Use heap-like data structure to prioritize best partial alignments
 - Max differences change as a function of read length
 - Reduce memory needs by storing small amount of O array and suffix array; calculate remaining parts as the algorithm progresses