Scalable String Similarity Search/Join with Approximate Seeds and Multiple Backtracking

Enrico Siragusa, David Weese, Knut Reinert
FU Berlin, Algorithmic Bioinformatics

String Similarity Search/Join Competition, 22 March 2013
Overview

- This work is a variation of our new DNA read mapper\(^1\)
- We follow the seminal work of Navarro and Baeza-Yates\(^2\)
- We combine multiple methods
  - online Banded Myers bit-vector algorithm
  - indexed Multiple backtracking
  - filtering Approximate seeds
- We preprocess both database and query strings
- Our approach solves search and join

---


Online solution

- Check the edit distance between each query and any database string: **global k-differences problem**
- A band of $k + 1$ cells is sufficient

\[ c = \left\lfloor \frac{m-n+k}{2} \right\rfloor \]

- initial case
- recursive case
- ignored
- tracked score

\[ n - m + c \]
Banded Myers bit-vector algorithm

- The DP matrix has **diagonal** & **adjacency** properties

---

- The DP matrix has **diagonal & adjacency** properties.
- Myers algorithm\(^3\) encodes each column as **bit-vectors**.

---

Banded Myers bit-vector algorithm

- The DP matrix has **diagonal & adjacency** properties
- Myers algorithm\(^3\) encodes each column as **bit-vectors**
- Our banded version computes a column in time \(O(k/w)\)

\[\epsilon \ C \ G \ C \ A \ N \ A \ T \ A \ T \ A \ T \ C \ A \ G\]

\[
\begin{array}{cccccccc}
V_P & V_N & H_P & H_N & D_0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 \\
\end{array}
\]

Backtracking

- Index database strings using a **radix tree**
- Perform a **top-down traversal** on the radix tree
Backtracking

- Index database strings using a **radix tree**
- Perform a **top-down traversal** on the radix tree
- Compute the distance between query and edge labels
Backtracking

- Index database strings using a **radix tree**
- Perform a **top-down traversal** on the radix tree
- Compute the distance between query and edge labels
- **Cut** a branch when the minimum distance exceeds $k$
Backtracking

- Index database strings using a **radix tree**
- Perform a **top-down traversal** on the radix tree
- Compute the distance between query and edge labels
- **Cut** a branch when the minimum distance exceeds $k$

![Radix Tree Diagram]

**Database**
- $\epsilon$, $A$
- 0, 1
- $c$, $T$

**Query**
- $C$, $T$
- $\epsilon$, 1, 1, 1
Backtracking

- Index database strings using a **radix tree**
- Perform a **top-down traversal** on the radix tree
- Compute the distance between query and edge labels
- **Cut** a branch when the minimum distance exceeds $k$
- Backtracking takes time **exponential** in $k$
Multiple backtracking

- Any two queries sharing a common prefix also share part of their backtracking
- Index the queries using another radix tree
- Backtrack the query tree in the database tree
Multiple backtracking I

- Any two queries sharing a common prefix also share part of their backtracking
- Index the queries using another **radix tree**
- Backtrack the query tree in the database tree
Multiple backtracking I

- Any two queries sharing a common prefix also share part of their backtracking
- Index the queries using another *radix tree*
- Backtrack the query tree in the database tree
Multiple backtracking

- Any two queries sharing a common prefix also share part of their backtracking
- Index the queries using another **radix tree**
- Backtrack the query tree in the database tree
Parallelization of multiple backtracking is non-trivial
We collect pairs of subtrees at fixed depth $D$
Each pair can be processed independently

$$Q \leftarrow Q \cup \{(d_a, q_a), (d_a, q_c), (d_a, q_g), \ldots, (d_t, q_t)\}$$
Exact seeds

- Partition query $Q$ in $k+1$ seeds
- One seed occurs without errors
Exact seeds

- Partition query $Q$ in $k+1$ seeds
- One seed occurs without errors

1. Index database strings with a **generalized suffix tree**
2. Search all seeds
3. Verify seeds inside a $k$-band
Exact seeds

- Partition query $Q$ in $k + 1$ seeds
- One seed occurs without errors

1. Index database strings with a generalized suffix tree
2. Search all seeds
3. Verify seeds inside a $k$-band

- Seed length $L = \frac{|Q|}{k + 1}$
Approximate seeds

- Partition query \( Q \) in \( s \) seeds
- One seed is within \( \lfloor k/s \rfloor \) errors
Approximate seeds

- Partition query $Q$ in $s$ seeds
- One seed is within $\lfloor k/s \rfloor$ errors
- Seed length $L = |Q|/s$
- The seed length is a parameter
Implementation

- Implemented in C++ using SeqAn and OpenMP
- Indices implemented as generalized suffix arrays
- Index construction combines bucket sort and quicksort
- Backtracking implemented with banded DP
David Weese & Knut Reinert
The SeqAn team
