Implementation of KSEARCH and SANS Algorithms in the genometools-environment

Meike Bruns, Florian Markowsky, Michael Spohn

University of Hamburg
Center for Bioinformatics
Bundesstraße 43
20146 Hamburg, Germany

http://www.zbh.uni-hamburg.de
info@zbh.uni-hamburg.de

Gliederung

Konzept

Algorithmen
- Datenstrukturen
- KSEARCH
- SANS

Implementation
- Genometools
- Our implementation

Sequenzvergleiche

- Die Bestimmung von Ähnlichkeiten zwischen Sequenzen ist eine häufige Aufgabe bei der Sequenzanalyse
- Exakte Sequenzvergleiche sind oft zu zeitaufwändig

Sequenzvergleiche - Methoden

- Alignmentbasierte Methoden
  - SSEARCH
    - ermittelt optimale lokale Alignments mittels Smith-Waterman-Algorithmus
  - FASTA
    - verbindet HotSpots zu approximativen Alignments
  - BLAST
    - erweitert Hits zu Maximum Segment Pairs
- Ranking mittels Feature-Vektoren
  - USEARCH
    - Vorsortieren der Datenbank nach Anzahlen gemeinsamer Teilwörter, Abbruch der Suche nach erstem Hit
  - KSEARCH
    - bewertet Sequenzpaare aufgrund gemeinsamer k-mere
  - SANS - bewertet Sequenzpaare aufgrund potentieller gemeinsamer Substrings

Gliederung

Suffixarrays

- Enthalten alle Suffixe einer Sequenz in lexikographischer Ordnung

Beispiel

Sequenz = GYBLAAB$	Suffixarray: AAB$ AB$ BLAAB$ B$ GYBLAAB$ LAAB$ YBLAAB$ $

Gliederung

KSEARCH - Berechnen der Scores

$$KSEARCH(query, database) = \sum_{u \in \mathcal{A}^k} F(query, u) \cdot F(database, u)$$

Beispiel

query = BITQAPS

database = PQAPSQAP

kmer	query	database	Produkt
APS	1	1	1
BIT	1	0	0
ITQ	1	0	0
PQA	0	1	0
PSQ	0	1	0
QAP	1	2	2
SQA	0	1	0
TQA	1	0	0

Implementierungsansatz und Analyse

- Koskinen & Holm
  - Berechnung des KSEARCH-Scores mittels Suffixarrays:
    - Laufzeit: $O(|TXT| * |TXT_Q|)$ + lineare Konstruktion der Suffixarrays
    - Speicherplatzbedarf:
- Unsere Implementierung
  - Berechnung der KSEARCH-Scores mittels k-mer-Profilen (hier kurz erläutern, was das ist!)
    - Laufzeit: $O(|TXT| + |TXT_Q| + (m+n) \cdot |\mathcal{A}^k| + AnzahlQprots \cdot AnzahlSprots \cdot 1)$??? LAUFZEIT JETZT ANDERS!!!
    - Speicherplatzbedarf: $O(|AnzahlQprots| \cdot \mathcal{A}^k + AnzahlQprots \cdot AnzahlSprots)$

Gliederung

SANS - Berechnen der Scores

- Substrings von Sequenzen, die im Suffixarray nebeneinander liegen, haben mit hoher Wahrscheinlichkeit gemeinsame Präfixe
- SANS-Scores nutzen das aus, indem der Score aus den Positionen der Sequenzen im Suffixarray abgeleitet werden.

Beispielbild!

eventuell kann diese Folie raus. wenn nicht, Formel schöner machen!

$$SANS(qprot, sprot) = \sum_{s=START[qprot]}^{START[qprot+1]-1} \sum_{i=-w}^{+w} id(sprot, SAP[ISA_{mapped}[s] + i])$$

Mit der Identitätsfunktion

$$id(a, b) = \begin{cases} 1, & if\ a = b \\ 0, & if\ a \neq b \end{cases}$$

Implementierungsansatz und Analyse

- Koskinen & Holm
  - Ermitteln der Positionen durch Mergen der Suffixarrays
    - Laufzeit: $O(|TXT| + |TXT_Q|)$ + lineare Konstruktion der Suffixarrays
    - Speicherplatzbedarf:
- Unsere Implementierung
  - Ermitteln der Positionen durch for-Schleifen
    - Laufzeit: $O(|TXT| + |TXT_Q|)$ + Konstruktion der Suffixarrays
    - Speicherplatzbedarf: $O(m \cdot n)$ + Platz für Suffixarrays

Gliederung

Genometools

- Wir implementieren in der Softwareumgebung Genometools, aus der wir eine Anzahl an Tools nutzen:
  - encseq encode - ...
  - suffixerator -
  - kmer iterator -

Gliederung

KSEARCH

```
function KSEARCH(query-proteins, database-proteins)
    for each database-protein d do
        calculate kmer-profile
        for each query-protein q do
            calculate kmer-profile
            calculate KSEARCH(d,q) - score:
            score = 0
            for each possible kmer do
                score calculation??
            output KSEARCH(d,q) - score
```

Optimierungsmöglichkeiten KSEARCH

- verkettete Liste. dadurch keine 2 geschachtelten for-schleifen Zeitersparnis?

SANS

```
function SANS(query proteins, database proteins)
    calculate suffixarray for query proteins
    calculate suffixarray for database proteins
    initialize score matrix
    for each suffix entry from 0 - query suffixarray.length do
        find position in database suffixarray:
        current position = 0
        for i in current position - database suffixarray.length do
            if suffix entry fits in at position i then
                calculate scores for proteinpairs inside window BESSER FORMULIEREN!
                update score matrix
                current position = i
```

Optimierungsmöglichkeiten SANS

- mergen der Suffixarrays statt for-schleifen.
  Zeitersparnis?

Vergleich mit Koskinen & Holm

hier nochmal? Folie von oben runter?? oder ist das schon der Analyseteil??
wo kommt das mit den 1,2,3 bzw. 1,3,5 -repeats hin?

Laufzeiten

- 1000 Rattus rattus Proteine gegen Swissprot-Datenbank

	Holm SANS	ZBH SANS	Holm KSEARCH	ZBH KSEARCH	BLAST
Index-Erstellung					
Scoring					

Spezifität

Hier kommt eine Grafik wie die im Paper hin.