Speichereffiziente Methoden zur Repräsentation von paarweisen Sequenzalignments

Thorben Wiese

9. Juni 2016

Einleitung

Ein Sequenzalignment wird in der Bioinformatik dazu verwendet, zwei oder mehrere Sequenzen von zum Beispiel DNA-Strängen oder Proteinsequenzen miteinander zu vergleichen und die Verwandtschaft zu bestimmen. Ein Alignment ist das Ergebnis eines solchen Vergleichs. Bei einem globalen Alignment wird jeweils die gesamte Sequenz betrachtet, bei einem lokalen Alignment lediglich Teilabschnitte der beiden Sequenzen. Um die verschiedenen Sequenzen vergleichen zu können, berechnet man einen Score oder die Kosten, um den Aufwand, den man betreiben muss, um die gegebenene Sequenz in die Zielsequenz umzuwandeln, beschreiben zu können. Hierbei wird jeweils das Optimum, also entweder der maximale Score oder die minimalen Kosten gesucht. Die verschiedenen Schritte, um die Symbolde der Strings zu verändern, sind bei Gleichheit ein 'match', bei der Substitution ein 'mismatch', bei der Löschung eine 'deletion' und bei der Einfügung eine 'insertion', welche je nach Verfahren unterschiedlich gewichtet werden können. Hierbei haben ähnliche Sequenzen einen hohen Score und geringe Kosten und unterschiedliche Sequenzen analog einen kleinen Score und hohe Kosten.

Die Edit-Operationen

Die in diesem Dokument eingeführten Begriffe werden in [1, S. 5-7, 14-16] definiert.

Sei $\mathcal{A}$ eine endliche Menge von Buchstaben, die man Alphabet nennt. Für DNA-Sequenzen verwendet man üblicherweise die Menge der Basen, also $\mathcal{A} = \{a, c, g, t\}$. $\mathcal{A}^i$ sei die Menge der Sequenzen der Länge i aus $\mathcal{A}$ und ε sei die leere Sequenz. Formal ausgedrückt ist eine Edit-Operation ein Tupel

$$(\alpha, \beta) \in (\mathcal{A}^1 \cup \{\varepsilon\}) \times (\mathcal{A}^1 \cup \{\varepsilon\}) \backslash \{(\varepsilon, \varepsilon)\},$$

Eine äquivalente Schreibweise von (α, β) ist $\alpha \to \beta$. Es gibt drei verschiedene Edit-Operationen

$$a \to \varepsilon \text{ist eine Deletion für alle } a \in \mathcal{A}$$
$$\varepsilon \to b \text{ist eine Insertion für alle } b \in \mathcal{A}$$
$$a \to b \text{ist eine Substitution für alle } a, b \in \mathcal{A}$$

Dabei ist zu beachten, dass $\varepsilon \to \varepsilon$ keine Edit-Operation darstellt.

Ein Alignment von zwei Sequenzen u und v lässt sich nun als eine Sequenz $(\alpha_1 \to \beta_1, ..., \alpha_h \to \beta_h)$ von Edit-Operationen definieren, sodass $u = \alpha_1 ... \alpha_h$ und $v = \beta_1 ... \beta_h$ gilt.

Die Edit-Distanz

Sei eine Kostenfunktion δ mit $\delta(a \to b) \geq 0$ für alle Substitutionen $a \to b$ und $\delta(\alpha \to \beta) > 0$ für alle Einfügungen und Löschungen $\alpha \to \beta$ gegeben. Die Kosten für ein Alignment $A = (\alpha_1 \to \beta_1, ..., \alpha_h \to \beta_h)$ ist die Summe der Kosten aller Edit-Operationen des Alignments.

$$\delta(A) = \sum_{i=1}^{h} \delta(\alpha_i \to \beta_i)$$

Ein Beispiel einer Kostenfunktion ist die Einheitskostenfunktion

$$\delta(\alpha \to \beta) = \begin{cases} 0, & \text{wenn } \alpha, \beta \in \mathcal{A} \text{ und } \alpha = \beta \\ 1, & \text{sonst.} \end{cases}$$

Die Edit-Distanz von zwei Sequenzen ist wie folgt definiert:

$$edist_\delta(u, v) = \min\{\delta(A) \mid A \text{ ist Alignment von } u \text{ und } v\}$$

Ein Alignment A ist optimal, wenn $\delta(A) = edist_\delta(u, v)$ gilt.

Wenn δ die Einheitskostenfunktion ist, so ist $edist_\delta(u, v)$ die Levenshtein Distanz [1, S. 19-21].

Ein Alignment kann für eine Edit-Distanz e mit der Einheitskostenfunktion in $O(e)$ Zeit berechnet werden [1, S. 41-42].

Codierung als Cigar-String im SAM/BAM-Format

Ein Dateiformat, welches zur Speicherung von Alignments verwendet wird, ist das SAM-Format oder die komprimierte Version BAM. Dieses codiert ein Alignment in einem sogenannten Cigar-String der aus einzelnen Zeichen besteht, die jeweil eine Edit-Operation bezeichnen, also M für eine Substitution, I für eine Insertion und D für eine Deletion. Gleiche aufeinanderfolgende Operationen werden als Kombination von Quantität und Symbol geschrieben.

Beispiel 1: Sei $u =$ `actgaact`, $v =$ `actagaat` und das Alignment $A = (a \to a, c \to c, t \to t, ...)$ gegeben.

```
a c t - g a a c t
| | |   | | |   |
a c t a g a a - t
```

Ein Alignment wird üblicherweise in drei Zeilen geschrieben, wobei in der ersten Zeile die Sequenz u und in der dritten Zeile die Sequenz v geschrieben wird. In der mittleren Zeile symbolisiert das Zeichen '|' eine Substitution, wobei üblicherweise nur ein Match markiert wird. Außerdem wird ein ε aus der Edit-Operation in diesem Fall durch das Zeichen '-' dargestellt.

Dieses Alignment wird duch den Cigar-String `3M1I3M1D1M` repräsentiert. Das Format benötigt wenig Speicher für Alignments mit einer kleinen Edit-Distanz und deutlich mehr Speicher für Alignments mit einer großen Edit-Distanz [2].

Die Idee der Trace-Points

Ein neuer Ansatz der speichereffizienten Repräsentation von Alignments wurde von Gene Myers in [3] beschrieben und basiert auf dem Konzept der Trace Points.

Sei A ein Alignment von $u[i...j]$ und $v[k...\ell]$ mit $i < j$ und $k < l$ und sei $\Delta \in \mathbb{N}$. Sei $p = \lceil \frac{i}{\Delta} \rceil$. Man unterteilt $u[i...j]$ in $\tau = \left\lceil \frac{j}{\Delta} \right\rceil - \lfloor \frac{i}{\Delta} \rfloor$ Substrings $u_0, u_1, ..., u_{\tau-1}$ mit

$$u_q = \begin{cases} u[i...p \cdot \Delta] & \text{falls } q = 0 \\ u[(p+q-1) \cdot \Delta + 1...(p+q) \cdot \Delta] & \text{falls } 0 < q < \tau - 1 \\ u[(p+\tau-2) \cdot \Delta...j] & \text{falls } q = \tau - 1 \end{cases}$$

Für alle q mit $0 \leq q < \tau - 1$ sei t_q der letzte Index des Substrings von v, der in A mit u_q aligniert. t_q nennt man Trace Point. Für $q = 0$ aligniert u_0 mit $v_0 = v[k...t_0]$. Für alle q mit $0 < q < \tau - 1$ aligniert u_q mit $v_q = v[t_{q-1} + 1...t_q]$.

Seien i, j, k, ℓ, Δ und die Trace-Points eines Alignments von u und v gegeben. Dann kann ein Alignment A' von u und v mit $\delta(A') \leq \delta(A)$ konstruiert werden. Danach bestimmt man aus den Trace-Points die Substring-Paare u_q und v_q, berechnet hierfür ein optimales Alignment und konkateniert die Alignments von den aufeinanderfolgenden Substring-Paaren zu A'.

Beispiel 2:

```
Sequenz 1: gagcatgttgcctggtcctttgctaggtactgtagaga
Sequenz 2: gaccaagtaggcgtggaccttgctcggtctgtaagaga
Delta: 15

Gesamtalignment:

0    5    0    5    0    5    0    5    0
gagc-a-t-gttgcc-tggtcctttgctaggtactgta-gaga
||  |  |  |  |   | |  |||  ||||  |||  |||  |||||  ||||
gaccaagtag--g-cgtggacctt-gctcggt-ctgtaagaga
0    5    0    5    0    5    0    5    0

seq1[0...14] aligniert mit seq2[0...15]
gagc-a-t-gttgcc-tgg
||  |  |  |  |   | |  |||
gaccaagtag--g-cgtgg

seq1[15...29] aligniert mit seq2[16...28]
tcctttgctaggtac
 ||||  |||  |||  |
acctt-gctcggt-c

seq1[30...37] aligniert mit seq2[29...37]
tgta-gaga
||||  ||||
tgtaagaga

Trace Points: [15, 28]
```

```
Berechnung der Intervalle anhand der Trace Points:

seq1[0...14] aligniert mit seq2[0...15]
gagc-a-t-gttgcc-tgg
|| | | | |  | | |||
gaccaagtag--g-cgtgg
Score: 11.0

seq1[15...29] aligniert mit seq2[16...28]
tcctttgctaggtac
 |||| ||| ||| |
acctt-gctcggt-c
Score: 11.0

seq1[30...37] aligniert mit seq2[29...37]
tgta-gaga
|||| ||||
tgtaagaga
Score: 8.0

0    5    0    5    0    5    0    5    0
gagc-a-t-gttgcc-tggtcctttgctaggtactgta-gaga
|| | | | |  | | ||| |||| ||| ||| ||||| ||||
gaccaagtag--g-cgtggacctt-gctcggt-ctgtaagaga
0    5    0    5    0    5    0    5    0
```

Bewertung

Für die Trace-Point Repräsentation wird für eine Edit-Distanz e mit Einheitskosten als Kostenfunktion δ wie oben beschrieben lediglich $O(e^2)$ Zeit pro Teilalignment benötigt, wobei bei einer erwarteten Fehlerrate ε des Alignments die Edit-Distanz immer höchstens so groß ist wie die Anzahl der Fehler im Teilalignment. [1, S.41-42]

Je größer der vorher definierte positive Parameter Δ ist, desto weniger Trace-Points werden gespeichert und umso länger dauert die Berechnung, um die Teil-Alignments zu rekonstruieren. Bei einem kleinen Δ werden analog mehr Trace-Points gespeichert, aber die Rekonstruktionszeit der Teil-Alignments ist geringer.

Mithilfe von Δ lässt sich somit ein Trade-Off zwischen dem Speicherplatzverbrauch und dem Zeitbedarf für die Rekonstruktion der Teil-Alignments einstellen.

Ziel dieser Bachelorarbeit soll die Umsetzung und Implementierung dieses neuen Verfahrens der Repräsentation von paarweisen Sequenzalignments mithilfe der Trace-Points sein, um die oben genannten Vorteile zu realisieren.

Literatur

[1] S. Kurtz: *Foundations of Sequence Analysis*. Lecture notes for a course in the Wintersemester 2015/2016, 14.02.2016

[2] The SAM/BAM Format Specification Group: *Sequence Alignment/Map Format Specification*. Samtools Github, 18.11.2015 - `https://samtools.github.io/hts-specs/SAMv1.pdf` (Stand: 18.04.2016)

[3] G. Myers: *Recording Alignments with Trace Points*. dazzlerblog, 05.11.2015 - `https://dazzlerblog.wordpress.com/2015/11/05/trace-points/` (Stand: 18.04.2016).