Ein graphtheoretischer Ansatz für das *multiple sequence Alignment*-Problem

„Effiziente Algorithmen“ in einer Bachelorarbeit

wissen.leben
WWU Münster

Joschka Strüber (j.st@wwu.de)

16. März 2018

1. Motivation

2. Das DIALIGN-Verfahren mit einem Min-Cut-Ansatz für das Konsistenzproblem

3. Umsetzung in der Bachelorarbeit

1. Motivation

2. Das DIALIGN-Verfahren mit einem Min-Cut-Ansatz für das Konsistenzproblem

3. Umsetzung in der Bachelorarbeit

Das *multiple sequence Alignment*-Problem

- Gegeben ist eine Menge von Zeichenketten.
- Beispiel:
  1. TCGTCTGCACGCGCTCTGCGAT
  2. AGTCGTCTGCACGGGATCTGCGA
  3. AATAGTCATGGACGCGTGCTCTA
  4. ATAGTCATGGACGCGTGCGCGAT
- Frage: Wie kann man einzelne Symbole oder ganze Abschnitte dieser Sequenzen einander zuordnen, sodass diejenigen Bereiche übereinander stehen, die sich möglichst ähnlich sind?

Das *multiple sequence Alignment*-Problem

- Gegeben ist eine Menge von Zeichenketten.
- Beispiel:
  1. `TCGTCTGCACGCGCTCTGCGAT`
  2. `AGTCGTCTGCACGGGATCTGCGA`
  3. `AATAGTCATGGACGCGTGCTCTA`
  4. `ATAGTCATGGACGCGTGCGCGAT`
- Frage: Wie kann man einzelne Symbole oder ganze Abschnitte dieser Sequenzen einander zuordnen, sodass diejenigen Bereiche übereinander stehen, die sich möglichst ähnlich sind?
- In unserem Beispiel könnte das beispielsweise so aussehen:
  1. `--TCGTC-TGCACGC--GCTCTGCGAT`
  2. `AGTCGTC-TGCACG-G-GATCTGCGA-`
  3. `AATAGTCATGGACGCGTGCTC---TA-`
  4. `-ATAGTCATGGACGCGTGCGC---GAT`

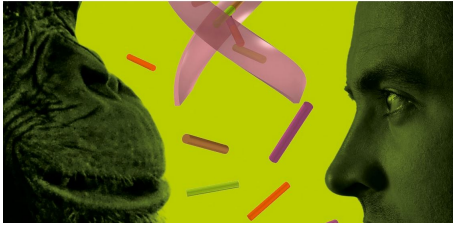
Wofür wird MSA benötigt?

PALÄOANTHROPOLOGIE

Per DNA-Verlust zum Menschen?

Das große Gehirn, der aufrechte Gang und unser Paarungsverhalten: Dies alles hätten wir vermutlich nicht ohne die Einbuße einiger DNA-Abschnitte, die für andere Primaten wichtig sind.

Philip L. Reno

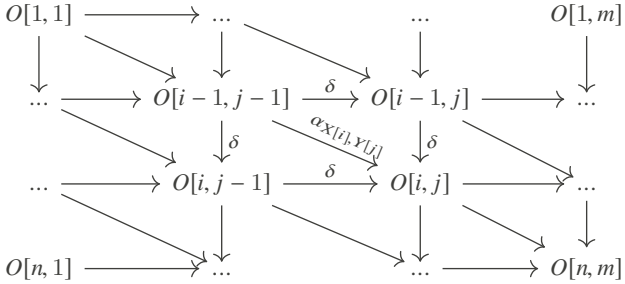


- Multiples Sequenzalignment wird hauptsächlich in der Bioinformatik benutzt.
- Verglichen werden üblicherweise DNA- oder Proteinsequenzen.
- Wird zum Beispiel benutzt, wenn man einen gemeinsamen evolutionären Ursprung zwischen mehreren Sequenzen vermutet, oder allgemein, um mögliche Resultate von Mutationen zu finden.
- Gefunden werden können beispielsweise Punktmutationen (Änderungen einzelner Basen oder Aminosäuren), sowie eingefügte oder gelöschte Abschnitte.

Komplexität I

- Problem: Unter allen auch nur halbwegs realistischen Annahmen ist das *multiple sequence Alignment*-Problem NP-schwer mit Laufzeit $O(length^n)$, wobei n die Anzahl der alignierten Sequenzen ist.
- Beweisskizze mit Hilfe des Algorithmus von Needleman-Wunsch:
- Kosten für alignieren von Zeichen a mit Zeichen b kostet $\alpha_{a,b}$, wobei typischerweise $\alpha_{a,a} = 0$ gilt.
- Gap Penalty δ, wenn wir die beiden Zeichen nicht miteinander alignieren und stattdessen in eine der Sequenzen eine Lücke einfügen.
- Gesamtkosten lassen sich mit dynamischer Programmierung und dieser Rekursionsgleichung lösen:
  $O[i,j] := min\{\alpha(X[i], Y[j]) + O[i-1, j-1], \delta + O[i-1, j], \delta + O[i, j-1]\}$

Komplexität II



- Alignment mit minimalen Kosten entspricht Pfad mit geringsten Kosten durch diesen Pfad, der in $O(n * m)$ berechnbar ist.
- Vorgehen lässt sich auch mit n-vielen Sequenzen durchführen → Pfad durch n-dimensionale Matrix

Grundidee hinter DIALIGN

Beispielsequenzen und Gewichtsfunktionen

1. paarweises Alignment

2. paarweises Alignment

3. paarweises Alignment

Konsistenz und Zwischenstand

Gieriges multiples Alignment

Flussnetzwerke

Inzidenzgraph

Minimaler Schnitt auf Zusammenhangskomponenten

Beispiele nichtkonsistenter Zuweisungen

Unser Graph mit gelöschten Kanten

Sukzessionsgraph

Zyklen im Sukzessionsgraph und wie wir diese entfernen

Das Dilemma

Erschummelter DAG

Folge der längsten Kette

Das resultierende multiple Alignment

Vereinfachungen, die ich vorgenommen habe

Gierige Algorithmen und Dynamische Programmierung

- Gierige Algorithmen:
  - Treffe lokal die bestmögliche Entscheidung und verwerfe diese danach nicht wieder
  - Gute Laufzeit, aber liefern oft nicht das optimale Ergebnis (→ gut geeignet für Heuristiken)
  - Wird in DIALIGN beim Zusammensetzen des multiplen Alignments benutzt
  - Beispiele: Algorithmus von Prim, Fractional Knapsack-Problem

Gierige Algorithmen und Dynamische Programmierung

- Gierige Algorithmen:
  - Treffe lokal die bestmögliche Entscheidung und verwerfe diese danach nicht wieder
  - Gute Laufzeit, aber liefern oft nicht das optimale Ergebnis (→ gut geeignet für Heuristiken)
  - Wird in DIALIGN beim Zusammensetzen des multiplen Alignments benutzt
  - Beispiele: Algorithmus von Prim, Fractional Knapsack-Problem
- Dynamische Programmierung:
  - Berechnung der optimalen Lösung durch Kombination von optimalen Lösungen sich überlappender Teilprobleme, wobei diese gespeichert werden und bei Bedarf abgerufen werden können
  - Beispiele: Algorithmus von Needleman-Wunsch, Knapsack-Problem

Flussnetze und der Push-Relabel-Algorithmus

- Flussnetze:
  - gerichteter, gewichteter Graph mit ausgewiesener Quelle und Senke; Gewichte der Kante entsprechen maximaler Kapazität an Fluss, die über diese fließen können
  - Ziel: Maximierung des Gesamtflusses eines Flussnetzwerkes von der Quelle zur Senke
  - Min-Cut-Max-Flow-Satz: der Wert eines maximalen Flusses entspricht dem Wert des minimalen Schnitts → benötigen wir zum Auflösen von Inkosistenzen im Inzidenzgraphen

Flussnetze und der Push-Relabel-Algorithmus

- Flussnetze:
  - gerichteter, gewichteter Graph mit ausgewiesener Quelle und Senke; Gewichte der Kante entsprechen maximaler Kapazität an Fluss, die über diese fließen können
  - Ziel: Maximierung des Gesamtflusses eines Flussnetzwerkes von der Quelle zur Senke
  - Min-Cut-Max-Flow-Satz: der Wert eines maximalen Flusses entspricht dem Wert des minimalen Schnitts → benötigen wir zum Auflösen von Inkosistenzen im Inzidenzgraphen
- Push-Relabel-Algorithmus:
  - Vorgehen:
  - besitzt eine bessere Laufzeit, als der in der Veröffentlichung benutzte Edmonds-Karps-Algorithmus

Algorithmus im Detail

- Gewichtsfunktionen für DNA- und Proteinsequenzen (BLOSUM62)
- Berechnung von paarweisen Alignments mit Hilfe von dynamischer Programmierung
- Aufbau des Inzidenzgraphen und Min-Cut auf den Zusammenhangskomponenten
- Aufbau des Sukzessionsgraphen und Löschen von Sites, um Konsistenz herzustellen
- Verwendung der so gefundenen alignierten Spalten als Ankerpunkte, um zwischen diesen das klassische DIALIGN laufen zu lassen
- Wenn mir am Ende noch langweilig ist: Blick auf DIALIGN TX, um dieses gegebenfalls zwischen den Ankerpunkten zu verwenden

Programmierung

- Programmiersprache C++
- Boost Graph Library, die bereits ausgeklügelte Graphen, Flussnetzwerke und Algorithmen zur Berechnung des maximalen Flusses bereitstellt
- variabelster Teil der Bachelorarbeit:
  - bei wenig Zeit am Ende: Ein- und Ausgabe über simple Textdateien
  - bei viel Zeit: Umsetzung mit Formaten wie Clustal und FASTA, sowie Visualisierung über externe Visualisierungssoftware

Beurteilung der Güte der Ergebnisse

- Bewertung des Algorithmus ist schwierig, weil er definitiv nicht das perfekte Alignment berechnet, ich aber nicht das Expertenwissen besitze, um die Güte eines Alignments wirklich beurteilen zu können
- Test auf BAliBase für global und (D)IRMBASE für lokal verwandte Sequenzen → bereits ausgewertete Sequenzen für die ein richtiges Alignment bekannt ist
- Erfordert noch Einarbeitung