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

„Effiziente Algorithmen“ in einer Bachelorarbeit

wissen.leben
WWU Münster

Joschka Strüber (j.st@wwu.de)

29. 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

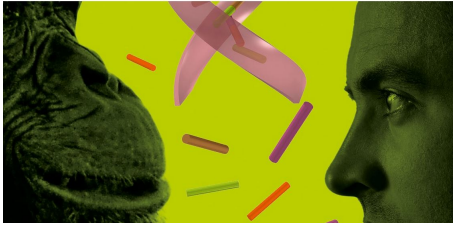
Wofür wird MSA benötigt?

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



[Reno18]

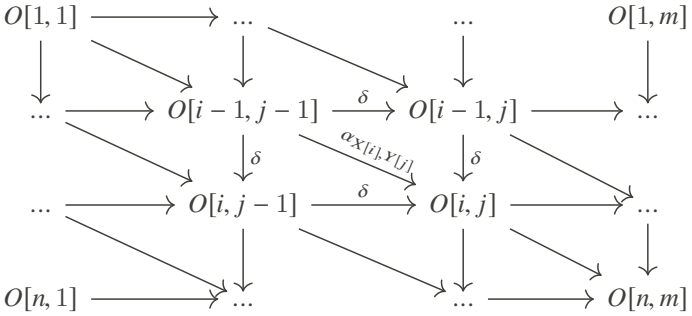
- Multiples Sequenzalignment wird hauptsächlich in der Bioinformatik angewendet. [MDW96]
- Verglichen werden üblicherweise DNA- oder Proteinsequenzen.
- Wird zum Beispiel benutzt, wenn man einen gemeinsamen evolutionären Ursprung zwischen mehreren Sequenzen vermutet oder allgemein, um mögliche Resultate 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. [WJ94]
- 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 berechnen: [NW70]

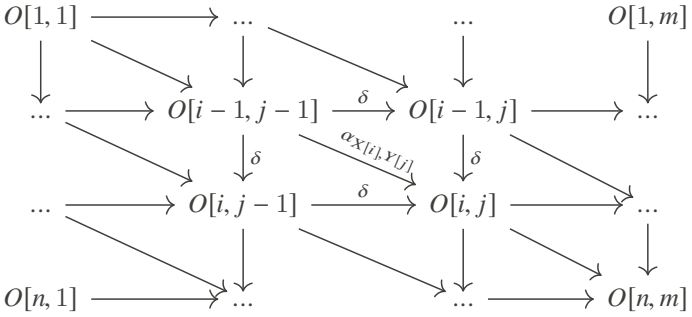
$$O[i,j] := min\{\alpha(X[i], Y[j]) + O[i-1, j-1], \delta + O[i-1, j], \delta + O[i, j-1]\} \quad (1)$$

Komplexität II



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

Komplexität II



- Alignment mit minimalen Kosten entspricht Pfad mit geringsten Kosten durch diese Matrix, der in $O(n * m)$ berechenbar ist.
- Vorgehen lässt sich auch mit n-vielen Sequenzen durchführen → Pfad durch n-dimensionale Matrix
- große multiple Alignments lassen sich nicht exakt, sondern nur heuristisch berechnen

1. Motivation

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

3. Umsetzung in der Bachelorarbeit

Grundidee hinter DIALIGN

1. Weise den zu alignierenden Abschnitten Gewichte zu.

Grundidee hinter DIALIGN

1. Weise den zu alignierenden Abschnitten Gewichte zu.
2. Maximiere die Summe der Gewichte bei paarweisen Alignments durch dynamische Programmierung.

Grundidee hinter DIALIGN

1. Weise den zu alignierenden Abschnitten Gewichte zu.
2. Maximiere die Summe der Gewichte bei paarweisen Alignments durch dynamische Programmierung.
3. Sortiere alle so entstandenen paarweisen Zuweisungen (Diagonalen in der Matrix von Needleman-Wunsch, daher DIALIGN für „DIagonal ALIGNment “) anhand ihrer Gewichte.

Grundidee hinter DIALIGN

1. Weise den zu alignierenden Abschnitten Gewichte zu.
2. Maximiere die Summe der Gewichte bei paarweisen Alignments durch dynamische Programmierung.
3. Sortiere alle so entstandenen paarweisen Zuweisungen (Diagonalen in der Matrix von Needleman-Wunsch, daher DIALIGN für „DIagonal ALIGNment “) anhand ihrer Gewichte.
4. Wähle gierig nacheinander Diagonalen aus, die zu allen bisher gewählten Diagonalen *konsistent* sind. [MDW96]

Beispielsequenzen und Gewichtsfunktionen

1. TCGTCTGCACG
2. AGTCGTCCGC
3. AATAGTCAT

- Wahrscheinlichkeit, dass es in einem Abschnitt der Länge l mindestens m viele Übereinstimmungen gibt:

$$P(l,m) := \sum_{i=m}^{l} \binom{l}{i} * p^i * (1-p)^{l-i} \tag{2}$$

- Gewichtsfunktion mit Korrekturterm $K := \ln l_1 + \ln l_2$:

$$w^* := -\ln(P(l,m)) - K \tag{3}$$

[MAHD98]

1. paarweises Alignment

1. TCGTCTGCACG
2. AGTCGTCGCG

GT GT } 0,5	GTCTGCACG GTCGTCGCG } 2,1	GTC GTC } 1,9	TCGTC TCGTC } 4,6	CACG CGCG } 0,5
GTCTGC GTCGTC } 0,7	GTCTGCA GTCGTCG } 0,2	GTCTGCAC GTCGTCGC } 1,1	CG CG } 0,5	TC TC } 0,5
CGTC AGTC } 0,5	GTCT GTCG } 0,5	TCGTCT TCGTCG } 2,8	TCGTCTG TCGTCGC } 1,8	TCGTCTGC TCGTCGCG } 1,1

1. paarweises Alignment

1. TCGTCTGCACG
2. AGTCGTCGCG

GT GT } 0,5	GTCTGCACG GTCGTCGCG } 2,1	GTC GTC } 1,9	TCGTC TCGTC } 4,6	CACG CGCG } 0,5
GTCTGC GTCGTC } 0,7	GTCTGCA GTCGTCG } 0,2	GTCTGCAC GTCGTCGC } 1,1	CG CG } 0,5	TC TC } 0,5
CGTC AGTC } 0,5	GTCT GTCG } 0,5	TCGTCT TCGTCG } 2,8	TCGTCTG TCGTCGC } 1,8	TCGTCTGC TCGTCGCG } 1,1

- Resultierendes Alignment:
  --TCGTCTgcaCG
  agTCGTCg---CG
- Score: 5,1

2. paarweises Alignment

1. TCGTCTGCACG
2. AATAGTCAT

TC TC } 0,5	GT GT } 0,5	CA CA } 0,5	GTC GTC } 1,9	TCGT TAGT } 0,5

TCGTC TAGTC } 1,6	GTCT GTCA } 0,5

2. paarweises Alignment

1. TCGTCTGCACG
2. AATAGTCAT

TC TC } 0,5	GT GT } 0,5	CA CA } 0,5	GTC GTC } 1,9	TCGT TAGT } 0,5

TCGTC TAGTC } 1,6	GTCT GTCA } 0,5

- Resultierendes Alignment:
  --tcGTCtgcacg
  aataGTCat----
- Score: 1,9

3. paarweises Alignment

1. AGTCGTCGCG
2. AATAGTCAT

TC TC } 0,5	GTC GTC } 1,9	AGTC AGTC } 3,2	AGTCG AGTCA } 1,6	AGTCGT AGTCAT } 2,8
GTCGT GTCAT } 1,6	TCGT TCAT } 0,5	AGT AGT } 1,9		

3. paarweises Alignment

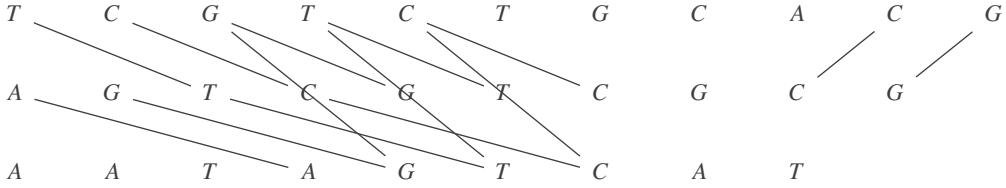
1. AGTCGTCGCG
2. AATAGTCAT

TC / TC } 0,5 GTC / GTC } 1,9 AGTC / AGTC } 3,2 AGTCG / AGTCA } 1,6 AGTCGT / AGTCAT } 2,8

GTCGT / GTCAT } 1,6 TCGT / TCAT } 0,5 AGT / AGT } 1,9

- Resultierendes Alignment:
  ```
  ---AGTCgtcgcg
  aatAGTCac----
  ```
- Score: 3,2

Konsistenz und Zwischenstand



- nicht *konsistent*, denn „TC“ aus der dritten Sequenz sind transitiv zwei „TC“-Abschnitte aus der ersten Sequenz zugeordnet und es gibt Überkreuzungen.

Gieriges multiples Alignment

- Sortiere alle Abschnitte aus den paarweisen Alignments nach ihrem Score.
- Wähle solange den Abschnitt mit dem höchsten Score, der mit allen bisher gewählten *konsistent* ist, bis es keinen solchen mehr gibt.

Gieriges multiples Alignment

- Sortiere alle Abschnitte aus den paarweisen Alignments nach ihrem Score.
- Wähle solange den Abschnitt mit dem höchsten Score, der mit allen bisher gewählten *konsistent* ist, bis es keinen solchen mehr gibt.

Sequenzen	Abschnitte	Score	zu den bisher gewählten *konsistent*?
1 2	TCGTC TCGTC	4,6	ja
2 3	AGTC AGTC	3,2	ja
1 3	GTC GTC	1,9	nein
1 2	CG CG	0,5	ja

Gieriges multiples Alignment

- Sortiere alle Abschnitte aus den paarweisen Alignments nach ihrem Score.
- Wähle solange den Abschnitt mit dem höchsten Score, der mit allen bisher gewählten *konsistent* ist, bis es keinen solchen mehr gibt.

Sequenzen	Abschnitte	Score	zu den bisher gewählten *konsistent*?
1 2	TCGTC TCGTC	4,6	ja
2 3	AGTC AGTC	3,2	ja
1 3	GTC GTC	1,9	nein
1 2	CG CG	0,5	ja

- Resultierendes gieriges Alignment:

```
-----TCGTCtgcaCG
---AGTCGTCg---CG
aatAGTCat-------
```

Erweiterung mit Min-Cut-Ansatz

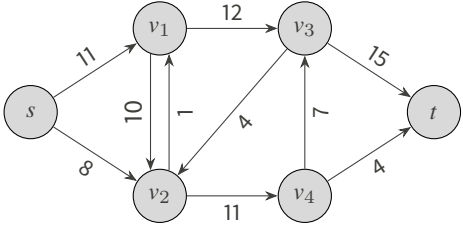
- Problem: Das gierige Konstruieren des multiplen Alignments zieht nur paarweise ähnliche Abschnitte in Betracht. [CPM10]
- Wir wollen aber Ähnlichenkeiten über möglichst viele Sequenzen hinweg.

Erweiterung mit Min-Cut-Ansatz

- Problem: Das gierige Konstruieren des multiplen Alignments zieht nur paarweise ähnliche Abschnitte in Betracht. [CPM10]
- Wir wollen aber Ähnlichenkeiten über möglichst viele Sequenzen hinweg.
- Alternativer Ansatz:
  1. Konstruiere Graph über alle aligierten *Sites* aus dem paarweisen Alignment (Inzidenzgraph).
  2. Entferne mit Hilfe von Flussnetzen Kanten aus den Zusammenhangskomponenten, um Inkosistenzen aufzulösen.
  3. Konstruiere Graph über die Zusammenhangskomponenten des Inzidenzgraphen (Sukzessionsgraph) und entferne geschickt Kanten und *Sites* aus dem Graph, bis alle Inkosistenzen aufgelöst sind.

Flussnetzwerke I

- Ein Flussnetzwerk ist ein gerichteter Graph $G = (V, E)$ mit zwei ausgewiesenen Knoten $s, t \in V$, genannt Quelle und Senke. Zusätzlich hat jede Kante $(u, v) \in E$ ein Kapazität genanntes, nichtnegatives Kantengewicht. [CLRS92]

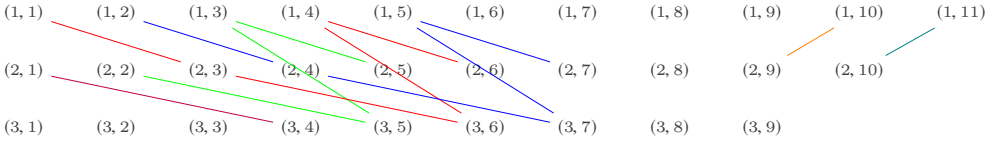


- Ziel: Berechnung des maximalen Flusses von der Quelle zur Senke.
- Jeder Kante wird ein Wert zugeordnet, der die Kapazität nicht überschreiten darf (Kapazitätsbeschränkung) und für jeden Knoten muss die Summe der einfließenden Flüsse der Summe der ausfließenden entsprechen (Flusserhaltung).

Flussnetzwerke II

- Ein *(s-t-)Schnitt* ist eine Partitionierung der Knotenmenge V in zwei Mengen A und B, bei der die Quelle $s \in A$ und die Senke $t \in B$ ist.
- Die Kapazität eines *(s-t-)Schnitts* ist die Summe der Kapazitäten aller Kanten, die von Knoten aus A nach B gehen.
- Der *Min-Cut-Max-Flow-Satz* besagt, dass die Kapazität des minimalen Schnitts dem Wert des maximalen Flusses entspricht.
- Es lässt sich zeigen, dass nach Berechnung des maximalen Flusses auch der minimale Schnitt bekannt ist. [CLRS92]

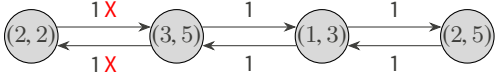
Inzidenzgraph



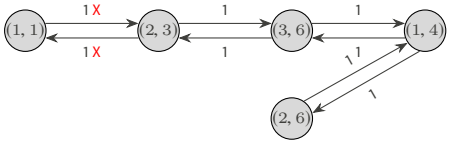
- Konstruiere Inzidenzgraph:
  Alle *Sites* entsprechen Knoten und zwischen zwei Knoten existiert genau dann eine Kante, wenn die entsprechenden Sites in einem der paarweisen DIALIGN-Fragmenten einander zugeordnet waren.
- Zusammenhangskomponenten sind farbig markiert.
- Eine Zusammenhangskomponente wird als *mehrdeutig* bezeichnet, wenn in ihr zwei oder mehr *Sites* aus der selben Sequenz existieren → Inkonsistenz.
  Genauso werden diese *Sites* als *mehrdeutig* bezeichnet. [CPM10]

Minimaler Schnitt auf Zusammenhangskomponenten I

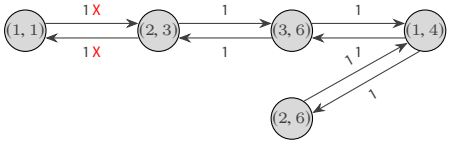
- Fasse Zusammenhangskomponenten des Inzidenzgraphen als Flussnetzwerke auf.
- Ersetze ungerichtete Kanten durch zwei gerichtete mit Kapazität 1.
- Setze jeweils zwei *mehrdeutige* Knoten aus der selben Sequenz als Quelle und Senke.
- Berechne den minimalen Schnitt und lösche die entsprechenden Kanten.
- Wiederhole dies, bis es keine *mehrdeutigen* Zusammenhangskomponenten mehr gibt.



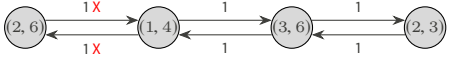
Minimaler Schnitt auf Zusammenhangskomponenten II



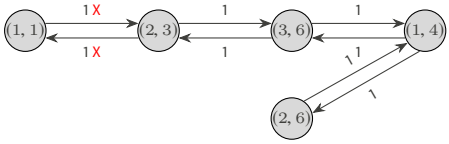
Minimaler Schnitt auf Zusammenhangskomponenten II



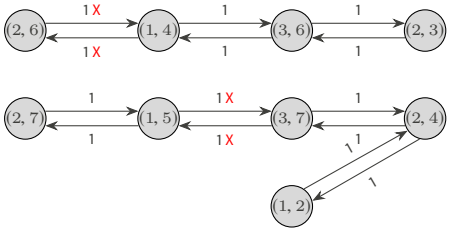
- Nach wie vor zwei *mehrdeutige* Sites in der Zusammenhangskomponente: (2,3) und (2,6).
- Berechne nächsten Min-Cut nach Entfernen der ersten Kante.



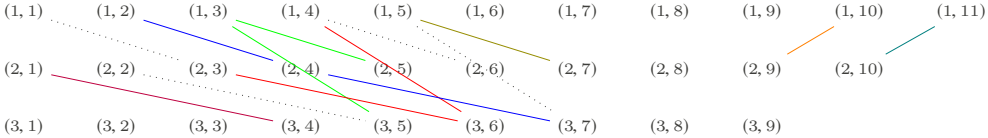
Minimaler Schnitt auf Zusammenhangskomponenten II



- Nach wie vor zwei *mehrdeutige* Sites in der Zusammenhangskomponente: (2,3) und (2,6).
- Berechne nächsten Min-Cut nach Entfernen der ersten Kante.

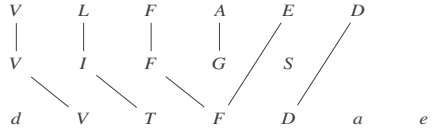


Unser Graph mit gelöschten Kanten

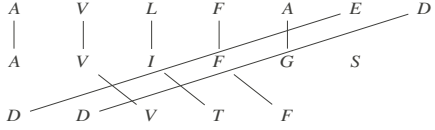


- Resultat: immer noch nicht *konsistent*.
- Grund: Überkreuzungen z.B. bei (1,2) → (2,4) → (3,7) und (1,3) → (3,5).

Zwei Arten nichtkonsistenter Zuweisungen



Implizite, transitive Mehrfachzuweisung von F: kann mit Flussnetzen und Inzidenzgraphen behoben werden.



Überkreuzungen: können wir noch nicht auflösen.

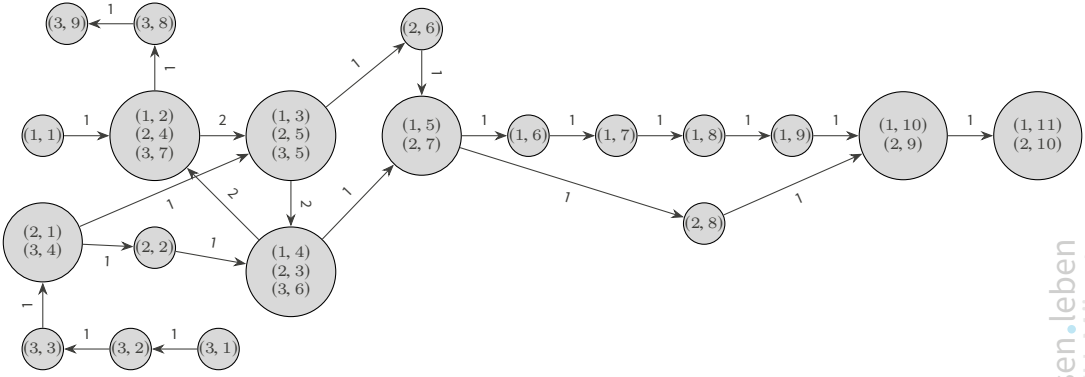
Sukzessionsgraph I

- Im Sukzessionsgraph sind die Zusammenhangskomponenten des Inzidenzgraphen Knoten. [PDC10]
- Es existiert eine Kante zwischen zwei Knoten C und C' genau dann, wenn eine Sequenz S_i exisitiert, sodass *Sites* $(i, p) \in C$ und $(i, p') \in C'$ existieren mit $p < p'$, ohne dass C'' existiert mit $(i, p'') \in C''$ und $p < p'' < p'$.
- Das Gewicht jeder Kante ist die Anzahl der Sequenzen für die die obige Bedingung gilt.

Sukzessionsgraph I

- Im Sukzessionsgraph sind die Zusammenhangskomponenten des Inzidenzgraphen Knoten. [PDC10]
- Es existiert eine Kante zwischen zwei Knoten C und C' genau dann, wenn eine Sequenz S_i exisitiert, sodass *Sites* $(i, p) \in C$ und $(i, p') \in C'$ existieren mit $p < p'$, ohne dass C'' existiert mit $(i, p'') \in C''$ und $p < p'' < p'$.
- Das Gewicht jeder Kante ist die Anzahl der Sequenzen für die die obige Bedingung gilt.
- Wenn der Sukzessionsgraph keine Zyklen enthält, ist das Alignment *konsistent*.
- Falls doch, müssen wir sukzessive *Sites* und Kanten entfernen, bis das der Fall ist.

Sukzessionsgraph II



Zyklen im Sukzessionsgraph und wie wir diese entfernen

- Herstellen von *Konsistenz* durch Löschen von Kanten und *Sites* aus dem Sukzessionsgraph. [PDC10]
- Optimal wäre Löschen von Kanten mit minimalem Kantengewicht, um alle Zyklen aufzulösen.
- Problem:

Zyklen im Sukzessionsgraph und wie wir diese entfernen

- Herstellen von *Konsistenz* durch Löschen von Kanten und *Sites* aus dem Sukzessionsgraph. [PDC10]
- Optimal wäre Löschen von Kanten mit minimalem Kantengewicht, um alle Zyklen aufzulösen.
- Problem: Das ist das NP-schwere *minimal weighted feedback arc set*-Problem.

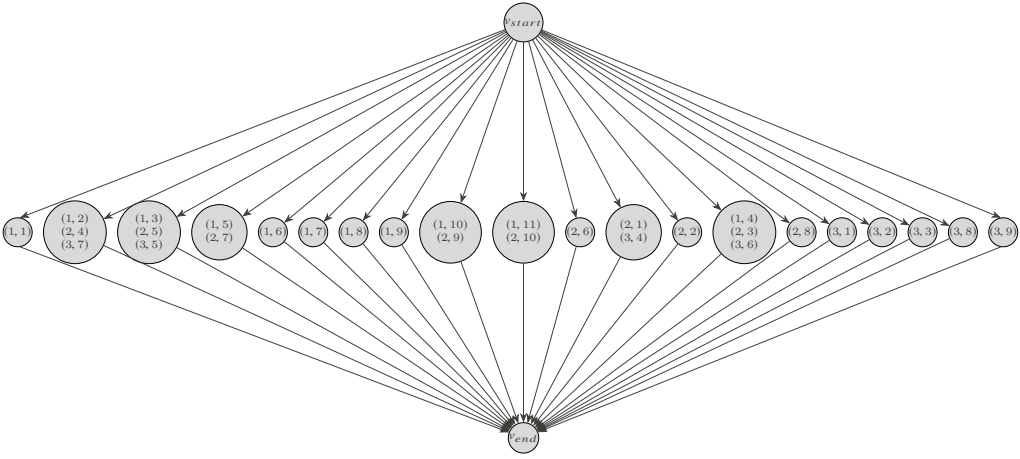
Zyklen im Sukzessionsgraph und wie wir diese entfernen

- Herstellen von *Konsistenz* durch Löschen von Kanten und *Sites* aus dem Sukzessionsgraph. [PDC10]
- Optimal wäre Löschen von Kanten mit minimalem Kantengewicht, um alle Zyklen aufzulösen.
- Problem: Das ist das NP-schwere *minimal weighted feedback arc set*-Problem.
- Stattdessen löschen wir solange die Kanten mit dem geringsten Kantengewicht, bis der Graph zyklenfrei ist:
  $E_k := \{(u,v) \in E | w(u,v) > k\}$ und $k^* := min\{k > 0 | (V, E_k) \text{ist azyklisch}\}$
- Zusätzlich fügen wir Start- und Endknoten ein, die mit den „Anfangs-“ und „Abschlussknoten“ aller Zusammenhangskomponenten nach dem Löschen verbunden sind.

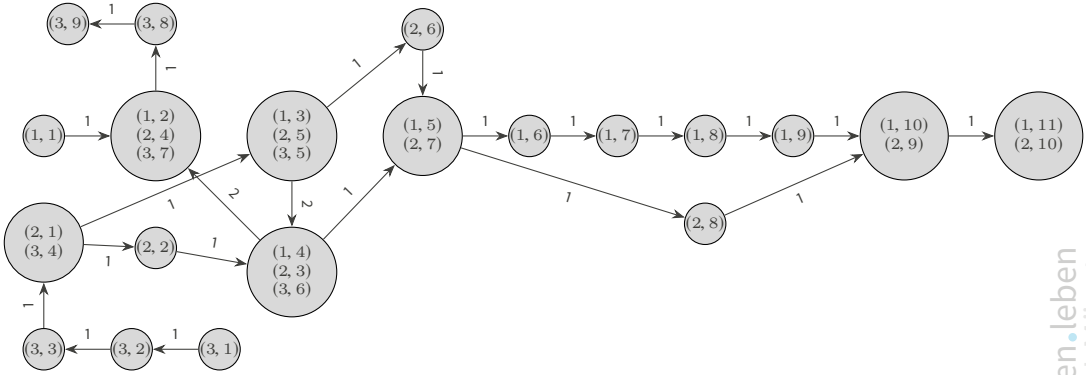
Zyklen im Sukzessionsgraph und wie wir diese entfernen

- Herstellen von *Konsistenz* durch Löschen von Kanten und *Sites* aus dem Sukzessionsgraph. [PDC10]
- Optimal wäre Löschen von Kanten mit minimalem Kantengewicht, um alle Zyklen aufzulösen.
- Problem: Das ist das NP-schwere *minimal weighted feedback arc set*-Problem.
- Stattdessen löschen wir solange die Kanten mit dem geringsten Kantengewicht, bis der Graph zyklenfrei ist:
  $E_k := \{(u,v) \in E | w(u,v) > k\}$ und $k^* := min\{k > 0 | (V, E_k) \text{ist azyklisch}\}$
- Zusätzlich fügen wir Start- und Endknoten ein, die mit den „Anfangs-“ und „Abschlussknoten“ aller Zusammenhangskomponenten nach dem Löschen verbunden sind.
- Leider gilt bei uns $k^* = 2 \rightarrow$ alle Kanten zwischen den ursprünglichen Knoten wurden gelöscht.

Das Dilemma



Erschummelter DAG

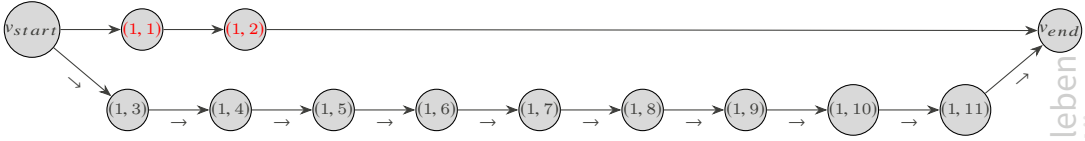


Folge der längsten Kette I

- Erstelle vom modifizierten Sukzessionsgraphen die transitive Hülle.
- Folge der längsten Kette für jede Sequenz S_i; dabei können nur Kanten (C, C') besucht werden, falls $(i, p) \in C$ und $(i, p') \in C'$ mit $p < p'$.
- Lösche alle *Sites* aus den Knoten, die nicht besucht wurden. [PDC10]

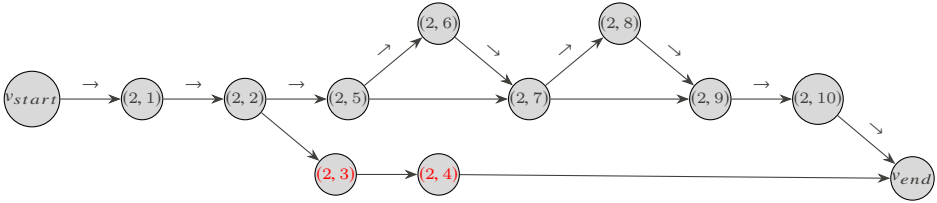
Folge der längsten Kette I

- Erstelle vom modifizierten Sukzessionsgraphen die transitive Hülle.
- Folge der längsten Kette für jede Sequenz S_i; dabei können nur Kanten (C, C') besucht werden, falls $(i, p) \in C$ und $(i, p') \in C'$ mit $p < p'$.
- Lösche alle *Sites* aus den Knoten, die nicht besucht wurden. [PDC10]



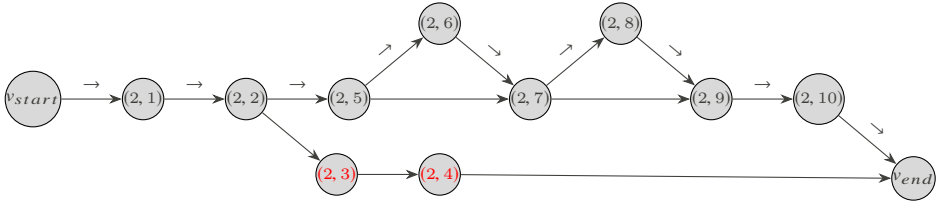
- Lösche $(1, 1)$ und $(1, 2)$ aus den Knoten im Sukzessionsgraph.

Folge der längsten Kette II



- Lösche $(2, 3)$ und $(2, 4)$.

Folge der längsten Kette II

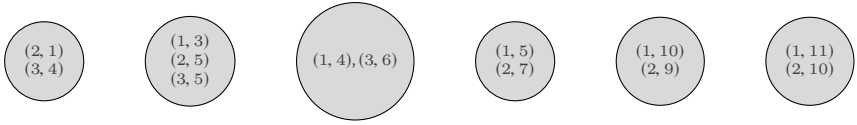


- Lösche $(2, 3)$ und $(2, 4)$.

- Nichts zu löschen.

Das resultierende multiple Alignment

- Damit bleiben nach dem Löschen die folgenden Alignments bestehen:



- Resultierendes Alignment:
  1. `-----tcGTCtgcaCG`
  2. `---AgtcGtCg---CG`
  3. `aatA---GTcat----`
- Schlechteres Alignment als gieriger Ansatz; funktioniert bei größeren Sequenzen hoffentlich besser.

Vereinfachungen, die ich vorgenommen habe

- Es werden nicht nur die Gewichte der Fragmente selbst betrachtet, sondern zusätzlich die Überschneidungen mit anderen (*Überlappgewichte*).
- Der Korrekturterm K wurde von mir halbiert. Er funktioniert wohl eher bei langen Sequenzen.
- Das aktualisieren der Konsistenzgrenzen ist recht kompliziert. Das Verfahren wurde hier nicht vorgestellt. [MDW96]
- Besserer Ansatz von Abdeddaim. [Abdeddaïm97]
- Nach dem Min-Cut-Ansatz werden diese Zuweisungsspalten als Ankerpunkte für das normale DIALIGN benutzt. [MPPS06]

1. Motivation

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

3. Umsetzung in der Bachelorarbeit

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:
  - Zur Berechnung des maximalen Flusses.
  - besitzt eine bessere Laufzeit, als der in der Veröffentlichung benutzte Edmonds-Karps-Algorithmus.

Algorithmus im Detail

- Gewichtsfunktionen für DNA- und Proteinsequenzen (BLOSUM62) mit Überlappgewichten.
- 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.

Fragen, Anregungen, Wünsche?

Vielen Dank für Ihre Aufmerksamkeit!

Quellenangaben

- *Introduction to Algorithms*. Cormen et all, MIT Press 1992
- *Multiple Sequence Alignment with user-defined achor points*. Morgenstern et all, Algorithms for Molecular Biology 2006
- *Automatic detection of anchor points for multiple sequence alignment*. Pitschi at all, BMC Bioinformatics 2010
- *Segment-based scores for pairwise and multiple sequence alignments*. Morgenstern at all, ISMB-98 Proceedings
- *On Incremental Computation of Transitive Closure and Greedy Alignment*. Abdeddaïm, in *Proceedings of the 8th Annual Symposium on Combinatorial Pattern Matching* 1997
- *A min-cut algorithm for the consistency problem in multiple sequence alignment*. Corel et all, Oxford University Press 2010
- *Multiple DNA and protein sequence alignment based on segment-to-segment comparison*. Morgenstern et all, 1996
- *Per DNA-Verlust zum Menschen?*. Reno, Spektrum der Wissenschaft Februar 2018
- *On the complexity of multiple sequence alignment*. Wang et all, Journal of Computational Biology 1994
- *A general method applicable to the search for similarities in the amino acid sequence of two proteins*. Needleman und Wunsch, Journal of Molecular Biology 1970