

Die Komplexität des Multiple Sequence
Alignment Problems bei nichtmetrischen
Kostenfunktionen

Bodo Siebert

Januar 2000
Bericht A-00-05

Zusammenfassung

Multiple Sequence Alignment ist ein wichtiges Problem in der molekularen Genetik. Dynamisches Programmieren dauert in der Praxis meist zu lange, andere (schnellere) Algorithmen zur Berechnung der optimalen Lösung sind bislang nicht bekannt.

Gesucht sind also schnelle Approximations-Algorithmen mit garantierter Approximationsgüte. Es wird die k -Stern-Methode vorgestellt. Für diese werden Laufzeitverbesserungen und Verallgemeinerungen gezeigt.

Für den Fall, daß die Kostenfunktion variabel ist und die Dreiecksungleichung nicht erfüllt sein muß, wird ein alternativer Beweis für die \mathcal{NP} -Vollständigkeit des korrespondierenden Entscheidungsproblems gegeben und die $MAX-S\mathcal{NP}$ -Härte sowie eine untere Schranke für die Approximationsgüte bewiesen.

Inhaltsverzeichnis

1	Einführung	3
2	Notationen und Definitionen	4
3	Dynamisches Programmieren	8
4	Komplexität und Approximationshärte	11
4.1	\mathcal{NP} -Vollständigkeit von MSA	11
4.2	Nichtapproximierbarkeit von MSA	15
4.2.1	Approximation von Optimierungsproblemen	15
4.2.2	Die Klasse $MAX-S\mathcal{NP}$	16
4.2.3	$MAX-S\mathcal{NP}$ -Härte von MSA	16
4.2.4	Nichtapproximierbarkeitsschranke für MSA	19
5	Approximationsalgorithmen	21
5.1	Die Center-Star-Methode	21
5.2	Verallgemeinerung auf k -Sterne	24
5.2.1	Approximation mit k -Sternen	25
5.2.2	Ausgewogene Mengen	26
5.3	Beschleunigung der 3-Stern-Methode	27
5.3.1	Kleine ausgewogene Mengen von 3-Sternen	27
5.3.2	Die Algorithmen	31
5.4	Verallgemeinerung der Fehlerabschätzung	33
	Literatur	37

1 Einführung

Multiple Sequence Alignment ist eine wichtige Aufgabe der Bioinformatik. Dabei stehen zwei Anwendungen im Vordergrund: zum einen das Finden gleicher oder ähnlicher Regionen in DNA-, RNA- oder Aminosäuresequenzen, zum anderen die Konstruktion phylogenetischer Bäume anhand der genetischen Informationen der Lebewesen.

Leider sind keine effizienten Algorithmen zur Berechnung eines optimalen Alignments einer variablen Anzahl von Sequenzen bekannt. Viele verschiedene Ansätze zur Konstruktion eines möglichst guten Alignments wurden bereits veröffentlicht (Altschul, Lipman [1], Feng, Doolittle [6], Gotoh [10], Vingron, von Häseler [19]). Eine ausführliche Auflistung würde den Rahmen an dieser Stelle sprengen. Für eine Übersicht empfehle ich Gusfield [12] und Waterman [22].

Viele der vorgestellten Methoden versuchen ein Alignment zu konstruieren, das eine gegebene Kostenfunktion optimiert. Bisher hat sich aber kein Maß zur Bewertung von Alignments durchsetzen können. Es gibt sogar Ansätze, die ganz auf eine Bewertung der konstruierten Alignments verzichten.

Ich werde hier die sogenannten SP-Kosten (sum of pair-Kosten, siehe Carrillo [5]) zur Bewertung der Alignments verwenden.

Kapitel 3 hat zum Inhalt, daß eine Greedy-Strategie mittels dynamischen Programmierens im Allgemeinen kein optimales Alignment liefert. In Kapitel 4 zeige ich die \mathcal{NP} -Vollständigkeit für die Berechnung eines optimalen Alignments sowie eine untere Schranke für die Approximierbarkeit. Kapitel 5 beschäftigt sich mit der Approximation des optimalen Alignments.

Ich werde hier nur eigene Sätze beweisen. Bei Ergebnissen von Anderen verweise ich für den Beweis auf die Literatur.

2 Notationen und Definitionen

Sei Σ ein Alphabet, dann bezeichnet $\Sigma_\Delta := \Sigma \cup \{\Delta\}$ mit $\Delta \notin \Sigma$. Δ heißt Gap. ϵ bezeichnet das leere Wort.

Mit $[i : j]$ wird die Menge $\{n \in \mathbb{N} | i \leq n \leq j\}$ bezeichnet. Ist x ein Vektor, so sind x_1, \dots, x_n die einzelnen Elemente von x , d. h. $x = (x_1, \dots, x_n)$.

Im folgenden ist \mathcal{S} eine Menge von Sequenzen S_1, \dots, S_n über dem Alphabet Σ . S_i hat die Länge l_i und es gilt $S_i = s_{i1} \dots s_{il_i}$. Dabei ist ein mehrfaches Auftreten gleicher Sequenzen erlaubt. \mathcal{M} sei die Menge aller endlichen Mengen von Sequenzen.

Die Definitionen 2.1, 2.2 und 2.4 sind im wesentlichen aus Pevzner [18] übernommen.

Definition 2.1 (Alignment-Graph)

Der Alignment-Graph $G=(V,E)$ von \mathcal{S} ist ein gerichteter azyklischer Graph und definiert durch

$$V = \{v \in \mathbb{N}^n | v_i \in [0 : l_i]\}$$

und

$$E = \{(v, w) \in V^2 | 0 \leq w_i - v_i \leq 1 \forall i \in [1 : n] \text{ und } v \neq w\} .$$

In einem Alignment-Graph repräsentiert eine Kante $e = (v, w)$ ein n -Tupel $f(e)$ aus Σ_Δ^n mit

$$f(e)_i := \begin{cases} s_{iw_i} & \text{falls } v_i < w_i \\ \Delta & \text{falls } v_i = w_i \end{cases} .$$

Ein Pfad (e_1, \dots, e_L) von der Quelle $(0, \dots, 0)$ zur Senke (l_1, \dots, l_n) des Alignment-Graphen kann als $n \times L$ -Matrix aufgefaßt werden. Dabei ist der Eintrag in der i -ten Zeile und der j -ten Spalte gerade $f(e_j)_i$.

Definition 2.2 (Alignment)

Sei (e_1, \dots, e_L) ein Weg von der Quelle zur Senke in dem Alignment-Graphen von \mathcal{S} . Dann heißt $\mathcal{A} = \{A_1, \dots, A_n\}$ mit $A_i = f(e_1)_i \dots f(e_L)_i$ Alignment von \mathcal{S} .

Beispiel 2.3

Sei $S_1 = \text{AT}$, $S_2 = \text{ACA}$ und $S_3 = \text{T}$. In Abb. 2 ist der zugehörige Alignmentgraph abgebildet. Es sind nicht alle Kanten eingezeichnet, um den Graph übersichtlich zu halten.

Der eingezeichnete Pfad steht für das Alignment

$$\mathcal{A} := \begin{pmatrix} \text{AT}\Delta \\ \text{ACA} \\ \Delta\text{T}\Delta \end{pmatrix}$$

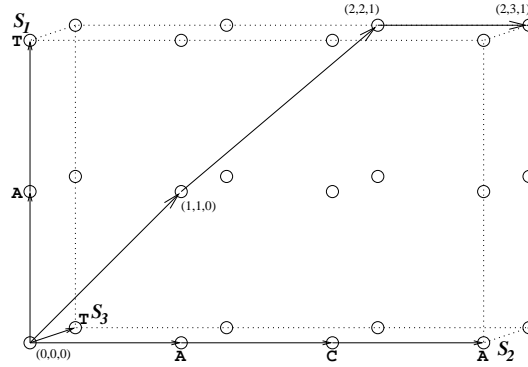


Abbildung 1: Der Alignmentgraph für die Sequenzen AT, ACA, T

\mathcal{A} steht, wenn nicht anders angegeben, für ein Alignment $\{A_1, \dots, A_n\}$ der Sequenzen aus \mathcal{S} . Dabei haben alle A_i ($i = 1, \dots, n$) die Länge L und es gilt $A_i = a_{i1} \dots a_{iL}$. a_{ij} ist durch einen Pfad (e_1, \dots, e_L) durch den Alignment-Graphen gegeben, $a_{ij} = f(e_j)_i$.

Definition 2.4 (Kosten, optimales Alignment)

Eine Kostenfunktion ist eine Funktion $d : \Sigma_\Delta^2 \rightarrow \mathbb{N}$ mit $d(x, x) = 0$ für alle $x \in \Sigma_\Delta$. Die SP-Kosten (SP: sum of pairs) $D_d(\mathcal{A})$ eines Alignments $\mathcal{A} = (A_1, \dots, A_n)$, wobei L die Länge der A_i ($i = 1, \dots, n$) ist, sind definiert durch

$$D_d(\mathcal{A}) = \sum_{1 \leq i < j \leq n} \sum_{p=1}^L d(a_{ip}, a_{jp}).$$

Für eine Menge von Sequenzen \mathcal{S} bezeichnet $\mathcal{A}_d(\mathcal{S})$ ein optimales Alignment von \mathcal{S} bezüglich der Kostenfunktion d , das heißt es gilt $D_d(\mathcal{A}_d(\mathcal{S})) \leq D_d(\mathcal{A})$ für alle Alignments \mathcal{A} von \mathcal{S} .

Die gewichteten Kosten eines Alignments für eine Gewichtsmatrix $C = (c_{ij})_{i,j=1}^n$ sind definiert durch

$$D_d(\mathcal{A}, C) = \sum_{1 \leq i < j \leq n} \sum_{p=1}^L c_{ij} d(a_{ip}, a_{jp})$$

mit $c_{ij} \geq 0$.

Für Sequenzen \mathcal{S} und Gewichtsmatrix C bezeichnet $\mathcal{A}_d(\mathcal{S}, C)$ ein optimales gewichtetes Alignment, das heißt $D_d(\mathcal{A}_d(\mathcal{S}, C), C) \leq D_d(\mathcal{A}, C)$ für alle Alignments \mathcal{A} von \mathcal{S} .

Die Aufgabe ist nun, einen Algorithmus zu finden, um für eine gegebene Menge von Sequenzen ein optimales Alignment zu berechnen. Dieses ist im Allgemeinen nicht eindeutig bestimmt.

Beispiel 2.5

Sei d eine Kostenfunktion mit $d(x, y) = 1$ für $x \neq y$. Die paarweisen Kosten sind dann:

$$2 \left\langle \begin{array}{c} \text{A T } \Delta \\ \text{A C A} \\ \Delta \text{T } \Delta \end{array} \right\rangle 1$$

Damit ist $D(\mathcal{A}) = 6$. Aber auch das Alignment

$$\begin{pmatrix} \text{A } \Delta \text{T} \\ \text{A C A} \\ \Delta \Delta \text{T} \end{pmatrix}$$

hat Kosten 6. Alignments mit geringeren Kosten sind nicht möglich. Es gibt also mehrere optimale Alignments.

Ich werde nur SP-Kosten betrachten. Es sind aber auch andere Bewertungen der Alignments möglich. Eine Verallgemeinerung der SP-Kosten sind Funktion der Form $d : \Sigma_{\Delta}^n \rightarrow \mathbb{N}$, wobei nicht nur Paare von Zeichen, sondern eine ganze Spalte eines Alignments bewertet wird.

Im folgenden werde ich statt D_d und $\mathcal{A}_d(\mathcal{S})$ auch D und $\mathcal{A}(\mathcal{S})$ schreiben, wenn klar ist, auf welche Kostenfunktion d sich D bzw. $\mathcal{A}(\mathcal{S})$ bezieht. Statt $D(\mathcal{A}(\mathcal{S}))$ schreibe ich auch $D(\mathcal{S})$.

$D(S_i, S_j)$ bzw. $D(S_i, S_j, S_k)$ bezeichnen die Kosten eines optimalen Alignments der zwei bzw. drei Sequenzen. $D(S_i, S_j, S_k, C)$ sind die Kosten eines optimalen gewichteten Alignments der drei Sequenzen mit Gewichten aus C .

Für eine Teilmenge S von \mathcal{S} bezeichne mit $D(S|\mathcal{A})$ die Kosten des Alignments, das sich durch Einschränken eines Alignments \mathcal{A} von \mathcal{S} auf Sequenzen aus S ergibt. Dieses Alignment heißt das durch \mathcal{S} induzierte Alignment von S . Offensichtlich ist $D(S) \leq D(S|\mathcal{A})$. \mathcal{A} heißt S -konsistent, falls $D(S) = D(S|\mathcal{A})$.

Beispiel 2.6

Betrachte wieder das Alignment aus Beispiel 2.3. Dann sind $D(\{S_1, S_3\}|\mathcal{A}) = 1$ die Kosten des Alignments

$$\mathcal{A}_{13} = \begin{pmatrix} \text{A T } \Delta \\ \Delta \text{T } \Delta \end{pmatrix}.$$

In \mathcal{A}_{13} taucht eine reine Gap-Spalte auf. Streng genommen ist \mathcal{A}_{13} also kein Alignment, da dieser Fall in der Definition nicht enthalten ist. Eine reine Gap-Spalte würde einer Schlinge im Alignment-Graphen entsprechen. Aufgrund der Bedingung $d(x, x) =$

0 erzeugen Gap-Spalten aber keine zusätzlichen Kosten, können also toleriert werden.

Aus den oben genannten Definitionen ergibt sich für Komplexitätsuntersuchungen das folgende Entscheidungsproblem.

Definition 2.7 (MSA)

Sei \mathcal{S} eine Menge von Sequenzen über einem gegebenen Alphabet Σ , $k \in \mathbb{N}$ eine Konstante und d eine Kostenfunktion. Dann ist

$$MSA := \{(\mathcal{S}, d, k) \mid D_d(\mathcal{A}_d(\mathcal{S})) \leq k\}$$

Desweiteren wird die folgende Funktion benötigt.

Definition 2.8 (MIN-MSA)

MIN-MSA ist das Optimierungsproblem zu einer Menge von Sequenzen und einer Kostenfunktion die Kosten eines optimalen Alignments zu berechnen.

3 Dynamisches Programmieren

Die übliche Methode zum Berechnen eines Alignments von n Sequenzen ist dynamisches Programmieren in n Dimensionen. Dabei wird die Methode, den Editierabstand zwischen zwei Sequenzen zu bestimmen, einfach auf höhere Dimensionen übertragen. Für eine genaue Beschreibung des Verfahrens empfehle ich Gusfield [12] und Waterman [22]. Für die Grundlagen dynamischer Programmierung von zwei Sequenzen empfehle ich Ottmann, Widmayer [16].

Der Nachteil des Verfahrens ist, daß sich für n Sequenzen der Länge l eine Laufzeit von $O(2^{nl})$ ergibt. Für eine konstante Anzahl Sequenzen ist die Laufzeit polynomiell in der Länge l der Sequenzen, wobei der Exponent die Anzahl der Sequenzen ist. Das macht diese Methode nur für kleine Anzahl von Sequenzen praktikabel.

Das folgende erste Theorem sagt aus, daß eine Greedy-Strategie im Allgemeinen nicht zu einem optimalen Alignment führt.

Dazu verwende ich Alignments von Gruppen von Sequenzen (siehe Gotoh [10]). Dabei wird in einer Dimension der Editiermatrix nicht eine einzelne Sequenz abgetragen, sondern ein bereits berechnetes Alignment A von mehreren Sequenzen S . Das bereits berechnete Alignment wird nicht mehr verändert. Lediglich komplette Δ -Spalten dürfen eingefügt werden. Das so entstehende gesamte Alignment \mathcal{A} ist im Allgemeinen nicht optimal. Es gilt aber $D(A) = D(S|\mathcal{A})$ und damit ist \mathcal{A} S -konsistent, falls $D(A) = D(S)$ erfüllt ist, also A ein optimales Alignment für S ist.

Ich schränke die Kostenfunktionen in Theorem 3.1 stark ein. Man kann aber davon ausgehen, daß durch allgemeinere Kostenfunktionen die Berechnung eines optimalen Alignments nicht einfacher wird.

Theorem 3.1

Seien $g, m > 0$ und $k \in \mathbb{N}$, $k \geq 4$ mit $m > g \cdot \frac{2}{k+1}$ und $m < 2 \cdot g$ gegeben. Ferner sei Σ ein Alphabet mit $|\Sigma| \geq 4$. d sei eine Kostenfunktion und gegeben durch

$$d(x, y) = \begin{cases} 0 & \text{falls } x = y, \\ m & \text{falls } x \neq y \text{ und } x, y \neq \Delta, \\ g & \text{sonst.} \end{cases}$$

d erfüllt die Dreiecksungleichung.

Dann gibt es $n + k$ Sequenzen S_1, \dots, S_{n+k} über Σ mit folgender Eigenschaft: Wird zunächst das optimale Alignment von n beliebigen Sequenzen berechnet und danach, unter Beibehaltung des berechneten optimalen Alignments, das optimale Alignment der n Sequenzen mit den restlichen k Sequenzen berechnet, so ist das resultierende Alignment nicht optimal bzgl. d .

Beweis: Sei $\Sigma := \{a, b, c, d\}$. Numeriere alle k -elementigen Teilmengen von $[1 : n+k]$ mit $1, 2, \dots, p$ durch, $p = \binom{n+k}{k}$. Bezeichne mit T_j die j -te Teilmenge.

Erstelle Sequenzen S_1, \dots, S_{n+k} wie folgt:

$$S_i = FM_{i_1}FM_{i_2}F \dots FM_{i_p}F$$

mit $F = d(ccd)^{n+k}$ und

$$M_{ij} = \begin{cases} ab & \text{falls } i \in T_j \\ b & \text{falls } i = \min([1 : n+k] \setminus T_j) \\ a & \text{falls } i = \max([1 : n+k] \setminus T_j) \\ \epsilon & \text{sonst} \end{cases}$$

Nach Konstruktion der Sequenzen kann man nun aus Symmetriegründen ohne Einschränkung zunächst das Alignment von S_1, \dots, S_n berechnen. Sei $T_{j_0} = [n+1 : n+k]$. Dann ist nach Konstruktion $M_{ij_0} \in \{a, b, \epsilon\}$ für alle $i \in [1 : n]$ und $M_{ij_0} = ab$ für $i \geq n+1$.

Bedingt durch die in jeder Sequenz gleich häufig auftretenden Teilsequenzen F müssen bei einem optimalen Alignment einer beliebigen Teilmenge von Sequenzen diese F -Teilstücke exakt untereinander ausgerichtet sein. Die Kosten des Alignment ergeben sich daher nur aus dem Aufeinandertreffen von a , b und Δ .

Da die Kosten für $\begin{smallmatrix} a \\ b \end{smallmatrix}$ nach Wahl der Kostenfunktion günstiger sind als für $\begin{smallmatrix} \Delta & a \\ b & \Delta \end{smallmatrix}$ sieht die Spalte j_0 des entstanden Alignment wie folgt aus:

$$\left. \begin{array}{c} b \\ \Delta \\ \vdots \\ \Delta \\ a \end{array} \right\} (n-2) \text{ mal}$$

Wird nun das Alignment mit den restlichen k Sequenzen berechnet (durch ein $k+1$ -dimensionales Alignment, bei dem in einer Dimension das bereits berechnete Alignment steht), so entsteht einer der beiden folgenden Fälle in der Spalte j_0 :

$$\left. \begin{array}{c} \Delta b \\ \Delta \Delta \\ \vdots \\ \Delta \Delta \\ \Delta a \\ a b \\ \vdots \\ a b \end{array} \right\} (n-2) \text{ mal} \quad \text{oder} \quad \left. \begin{array}{c} b \Delta \\ \Delta \Delta \\ \vdots \\ \Delta \Delta \\ a \Delta \\ a b \\ \vdots \\ a b \end{array} \right\} (n-2) \text{ mal}$$

$$\left. \begin{array}{c} a b \\ \vdots \\ a b \end{array} \right\} k \text{ mal} \quad \left. \begin{array}{c} a b \\ \vdots \\ a b \end{array} \right\} k \text{ mal}$$

Beide Möglichkeiten haben identische Kosten. Daher betrachte ich o. E. die linke

Möglichkeit A_1 . Die Kosten für diese Spalte berechnen sich wie folgt:

$$\begin{aligned} D(A_1) &= \underbrace{nk g}_{\text{linke Spalte}} + \underbrace{(k+1)m + (n-2)(k+2)g}_{\text{rechte Spalte}} \\ &= (k+1)m + [(2n-2)k + (2n-4)]g \\ &= 2(n-1)(k+1)g \end{aligned}$$

Ein optimales Alignment hätte aber stattdessen folgende Spalte A_2 :

$$\left. \begin{array}{c} \text{a } \Delta \\ \Delta \text{ b} \\ \Delta \Delta \\ \vdots \\ \Delta \Delta \\ \text{a b} \\ \vdots \\ \text{a b} \end{array} \right\} \begin{array}{l} (n-2) \text{ mal} \\ \\ k \text{ mal} \end{array}$$

mit Kosten

$$\begin{aligned} D(A_2) &= \underbrace{(k+1)(n-1)g}_{\text{linke Seite}} + \underbrace{(k+1)(n-1)g}_{\text{rechte Seite}} \\ &= [2(n-1)k + 2(n-1)]g \end{aligned}$$

Es gilt nämlich:

$$\begin{aligned} D(A_1) &> D(A_2) \\ \Leftrightarrow (k+1)m + [(2n-2)k + (2n-4)]g &> [2(n-1)k + 2(n-1)]g \\ \Leftrightarrow (k+1)m - 2g &> 0 \\ \Leftrightarrow m &> \frac{2}{k+1}g \end{aligned}$$

Nach Voraussetzung ist $m > \frac{2}{k+1}g$, daher gilt die Behauptung. ■

Der Fall $m = 2g$ wird hier nicht betrachtet, da in diesem Fall die Kosten für $\begin{smallmatrix} \Delta & \text{a} \\ \text{b} & \Delta \end{smallmatrix}$ und $\begin{smallmatrix} \text{a} \\ \text{b} \end{smallmatrix}$ gleich wären. Über die in obigem Beweis betrachtete Spalte j_0 könnte in diesem Fall keine Aussage gemacht werden.

4 Komplexität und Approximationshärte

Für einige mit *MSA* verwandte Probleme wurde \mathcal{NP} -Vollständigkeit bereits nachgewiesen.

Beim Multiple Sequence Tree Alignment geht es darum, zu einer gegebenen Menge von Sequenzen neue Sequenzen so hinzuzufügen, daß der minimal spannende Baum des vollständigen Graphen mit den Sequenzen als Knoten und den Kosten des paarweisen Alignments der adjazenten Sequenzen als Kantengewichte minimale Kosten hat. Für dieses Problem wurde \mathcal{NP} -Vollständigkeit und $MAX - \mathcal{SNP}$ -Härte nachgewiesen (siehe Wang, Jiang [20] und Wareham [21]).

Ebenfalls wurde \mathcal{NP} -Vollständigkeit für Multiple Sequence Alignment mit Kostenfunktionen der Form $d : \Sigma_{\Delta}^n \rightarrow \mathbb{N}$ nachgewiesen (siehe Maier [15] und Pevzner [18]).

In [20] beweisen Wang und Jiang die \mathcal{NP} -Vollständigkeit für Multiple Sequence Alignment mit SP-Kosten durch Reduktion von Shortest Common Superstring, wobei die Kostenfunktion d die Dreiecksungleichung erfüllt.

Ich werde zeigen, daß *MSA* \mathcal{NP} -vollständig ist, wenn die Dreiecksungleichung nicht erfüllt sein muß. In Abschnitt 4.2 werde ich dann die in Abschnitt 4.1 benutzte Reduktion dazu verwenden, um $MAX - \mathcal{SNP}$ -Härte und eine untere Schranke für die Approximierbarkeit von *MSA* nachzuweisen.

4.1 \mathcal{NP} -Vollständigkeit von *MSA*

Zum Beweis der \mathcal{NP} -Vollständigkeit werde ich $1-2,3SAT$ auf *MSA* reduzieren.

$1-2,3SAT$ ist die Menge aller Booleschen Formeln F in konjunktiver Normalform, deren Klauseln aus 2 oder 3 positiven Literalen bestehen, und es eine Belegung der Variablen gibt, so daß jede Klausel durch genau ein Literal erfüllt wird. Eine Klausel, die durch genau ein Literal erfüllt wird, nenne ich eindeutig erfüllt.

$1-3SAT$ ist die Teilmenge von $1-2,3SAT$, in der jede Klausel genau 3 Literale enthält. $1-3SAT$ ist nach Garey, Johnson [9] \mathcal{NP} -vollständig. $1-3SAT$ ist ein Teilproblem von $1-2,3SAT$ und offensichtlich ist $1-2,3SAT$ auch in \mathcal{NP} , daher ist auch $1-2,3SAT$ \mathcal{NP} -vollständig.

Theorem 4.1

MSA ist \mathcal{NP} -vollständig für $|\Sigma| \geq 9$.

Beweis: Ich werde zeigen: $1-2,3SAT \leq_{\text{pol}} MSA$.

Sei $F = F_1 \wedge \dots \wedge F_r$ eine Formel in konjunktiver Normalform bestehend aus $|F| = r$ Klauseln, wobei alle F_j 2 oder 3 Literale enthalten, die alle nichtnegiert sind. Seien

	Δ	\bar{p}	p	a	b	c	d	f	1	0
Δ	0	0	13r	13r	13r	13r	13r	13r	13r	13r
\bar{p}	0	0	0	13r	13r	13r	13r	13r	13r	13r
p	13r	0	0	0	0	0	0	0	0	0
a	13r	13r	0	0	3	3	3	0	1	1
b	13r	13r	0	3	0	3	3	0	1	1
c	13r	13r	0	3	3	0	3	0	1	1
d	13r	13r	0	3	3	3	0	0	1	1
f	13r	13r	0	0	0	0	0	0	0	0
1	13r	13r	0	1	1	1	1	0	0	0
0	13r	13r	0	1	1	1	1	0	0	0

Abbildung 2: Die Kostenfunktion d_F für eine Formel F mit r Klauseln

x_1, \dots, x_n die in F vorkommenden Variablen. Erstelle $n+1$ Sequenzen S_1, \dots, S_n, C über dem Alphabet $\Sigma := \{a, b, c, d, f, p, \bar{p}, 1, 0\}$.

$$S_i = \bar{p}pX_{i1}ppX_{i2}pp \dots ppX_{ir}p\bar{p}$$

für alle $i = 1, \dots, n$. O. E. seien F_1, \dots, F_m die Klauseln mit jeweils 3 Literalen und F_{m+1}, \dots, F_r die Klauseln mit je 2 Literale. Dann ist für $1 \leq j \leq m$

$$X_{ij} = \begin{cases} a10 & \text{falls } x_i \text{ das erste Literal in } F_j \text{ ist} \\ b10 & \text{falls } x_i \text{ das zweite Literal in } F_j \text{ ist} \\ c10 & \text{falls } x_i \text{ das dritte Literal in } F_j \text{ ist} \\ fff & \text{sonst} \end{cases}$$

und für $m+1 \leq i \leq r$

$$X_{ij} = \begin{cases} aaffaaaffa & \text{falls } x_i \text{ das erste Literal in } F_j \text{ ist} \\ 11fbbbbbf1 & \text{falls } x_i \text{ das zweite Literal in } F_j \text{ ist} \\ ffffffff & \text{sonst} . \end{cases}$$

Desweiteren ist

$$\begin{aligned} C &= \bar{p}pp \underbrace{d0pppd0ppp \dots pppd0ppp}_{m \text{ mal } d0} \underbrace{ffffffffppp \dots pppffffffff}_{(r-m) \text{ mal } f^9} p\bar{p} \\ &= \bar{p}pp(d0ppp)^{m-1} d0(pppf^9)^{r-m} p\bar{p} . \end{aligned}$$

Sei $\mathcal{S}_F = \{S_1, \dots, S_r, C\}$. Die Kostenfunktion d_F ist in Abbildung 2 dargestellt.

Definition 4.2 (variablen-konsistentes Alignment)

Ein Alignment $\mathcal{A} = \{\hat{C}, \hat{S}_1, \dots, \hat{S}_n\}$ heißt variablen-konsistent zu einer Belegung $\alpha_1, \dots, \alpha_n$ der Variablen x_1, \dots, x_n wenn

1. $\hat{C} = C$,

$$2. \hat{S}_i = \begin{cases} S_i \Delta & \text{falls } \alpha_i = \text{false}, \\ \Delta S_i & \text{falls } \alpha_i = \text{true}. \end{cases}$$

Zunächst ein Beispiel zur Verdeutlichung der Konstruktion.

Beispiel 4.3

Sei $F = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee x_4 x_5) \wedge (x_1 \vee x_3)$. Daraus ergeben sich folgende Sequenzen:

$$\begin{aligned} S_1 &= \bar{p}pa10ppa10ppffffppaaffaaaffap\bar{p} \\ S_2 &= \bar{p}pb10ppffffppa10ppffffffffffffp\bar{p} \\ S_3 &= \bar{p}pc10ppb10ppffffpp11fbbbbbf1p\bar{p} \\ S_4 &= \bar{p}pffppc10ppb10ppffffffffffffp\bar{p} \\ S_5 &= \bar{p}pffppffffppc10ppffffffffffffp\bar{p} \\ C &= \bar{p}ppd0pppd0pppd0pppffffffffffffpp\bar{p} \end{aligned}$$

Wähle $x_1 = x_5 = 1$ und $x_2 = x_3 = x_4 = 0$. Bei dieser Belegung sind alle Klauseln eindeutig erfüllt. Das zu dieser Belegung gehörende variablen-konsistente Alignment

$$\begin{aligned} \hat{S}_1 &: \Delta \bar{p}pa10ppa10ppffffppaaffaaaffap\bar{p} \\ \hat{S}_2 &: \bar{p}pb10ppffffppa10ppffffffffffffp\bar{p}\Delta \\ \hat{S}_3 &: \bar{p}pc10ppb10ppffffpp11fbbbbbf1p\bar{p}\Delta \\ \hat{S}_4 &: \bar{p}pffppc10ppb10ppffffffffffffp\bar{p}\Delta \\ \hat{S}_5 &: \Delta \bar{p}pffppffffppc10ppffffffffffffp\bar{p} \\ \hat{C} &: \bar{p}ppd0pppd0pppd0pppffffffffffffpp\bar{p} \end{aligned}$$

hat Kosten 40, jeweils 10 pro Klausel.

Behauptung 1: Es gibt ein optimales Alignment, das variablen-konsistent ist.

Beweis: Man sieht (siehe auch Beispiel 4.3), daß es ein Alignment \mathcal{A} von \mathcal{S}_F mit $D(\mathcal{A}) \leq 12r$ gibt. Dies ist gerade dann der Fall, wenn in \mathcal{A} variablen-konsistent zu einer beliebigen Belegung ist.

Da $D(\mathcal{A}) \leq 12r$ können Gaps nur in Spalten auftreten, wo sich außerdem nur \bar{p} befinden. Ansonsten würden sich aufgrund der gewählten Kostenfunktion sofort Kosten $\geq 13r$ ergeben. Daher können Gaps nur vorne oder hinten in einer Sequenz eingefügt werden.

Sei \hat{S}_i die aus S_i entstandene Sequenz in \mathcal{A} , \hat{S}_0 die aus C entstandene Sequenz in \mathcal{A} . Jede Sequenz \hat{S}_i aus \mathcal{A} hat also ein Präfix der Form $\Delta^{j_{i1}}\bar{p}\Delta^{j_{i2}}$ für $j_{i1}, j_{i2} \in \mathbb{N}$. Konstruiere ein Alignment \mathcal{A}' wie folgt:

1. Ersetze für jedes i den Präfix von \hat{S}_i durch $\Delta^{j_{i1}+j_{i2}}\bar{p}$.

2. Entferne alle reinen Δ -Spalten.

In dem so konstruierten Alignment hat jedes \hat{S}_i , $i \geq 1$, maximal ein führendes Gap. \hat{S}_0 hat kein führendes Gap. Es gilt außerdem $D(\mathcal{A}') \leq D(\mathcal{A})$. Verfahre analog mit den Postfixen. Daraus folgt die Behauptung.

Behauptung 2: $D(\mathcal{S}_F) = 12r - 2k$, wobei k die maximale Anzahl gleichzeitig eindeutig erfüllbarer Klauseln von F ist.

Beweis: Nach Behauptung 1 genügt es variablen-konsistente Alignments zu betrachten.

Da $d(\mathbf{p}, x) = 0$ für alle $x \neq \Delta$ und $d(\mathbf{p}, x) = 0$ für alle $x \neq \Delta, \bar{\mathbf{p}}$ genügt es, die Kosten für einzelne Klauseln anhand der X_{ij} der beteiligten Variablen zu ermitteln.

Es bleibt noch zu zeigen, daß eine erfüllte Klausel Kosten 10 und eine nicht erfüllte Klausel Kosten 12 hat. Betrachtet man die repräsentierenden Spalten für eine Klausel mit 2 Literalen x und y , so ergeben sich diese zwei Möglichkeiten (aus Symmetriegründen ist es ausreichend zwei der vier Möglichkeiten zu betrachten):

$$\begin{array}{cc} \text{aaffaaaffa} & \text{aaffaaaffa} \\ \text{11fbbbbbf1} & \text{11fbbbbbf1} \\ x = y & x \neq y \end{array}$$

Im linken Fall entstehen Kosten 12, beide Variablen haben den gleichen Wert, die entsprechende Klausel ist also nicht eindeutig erfüllt. Im rechten Fall entstehen Kosten 10, die Variablen sind unterschiedlich belegt, die Klausel ist eindeutig erfüllt.

Für Klauseln mit 3 Literalen sind 4 Fälle zu betrachten (insgesamt sind es 8 Fälle, aufgrund der Symmetrie sind 4 Fälle ausreichend):

$$\begin{array}{cccc} \text{a10} & \text{a10} & \text{a10} & \text{a10} \\ \text{b10} & \text{b10} & \text{b10} & \text{b10} \\ \text{c10} & \text{c10} & \text{c10} & \text{c10} \\ \text{d0} & \text{d0} & \text{d0} & \text{d0} \\ 0 \times \text{true} & 1 \times \text{true} & 2 \times \text{true} & 3 \times \text{true} \end{array}$$

Man sieht, daß sich in allen Fällen bis auf $1 \times \text{true}$ Kosten 12 ergeben. In Fall $1 \times \text{true}$ ergeben sich Kosten 10, dies ist auch gerade der Fall, in dem eine Variablenbelegung diese Klausel eindeutig erfüllt.

Es ergibt sich also:

$$\begin{array}{l} \text{In } F \text{ sind max. } k \text{ von } r \text{ Klauseln gleichzeitig eindeutig erfüllbar} \\ \Leftrightarrow \text{Es gibt ein optimales Alignment von } \mathcal{S}_F \text{ mit Kosten } 12r - 2k \end{array}$$

Folglich gilt:

$$F \in 1-2, 3SAT \Leftrightarrow (\mathcal{S}_F, 10 \cdot |F|, d_F) \in MSA$$

Die Reduktion läßt sich in polynomieller Zeit berechnen, MSA ist also \mathcal{NP} -hart.

Man kann nichtdeterministisch leicht überprüfen, ob (S, k, d) in MSA ist: Rate eine mögliche Belegung der maximal $n \times nl$ großen Alignment-Matrix, überprüfe, ob diese Belegung ein gültiges Alignment darstellt, und berechne die Kosten. Akzeptiere, wenn die Kosten kleiner oder gleich k sind. Überprüfung und Berechnung der Kosten ist deterministisch in polynomieller Zeit möglich, raten nichtdeterministisch in polynomieller Zeit.

Daraus folgt die Behauptung. ■

4.2 Nichtapproximierbarkeit von MSA

4.2.1 Approximation von Optimierungsproblemen

MSA ist als Sprache eingeführt worden, es gibt also für jede Eingabe nur die Entscheidung, ob sie in MSA ist oder nicht. Bei der Approximation eines Problems geht es darum, eine Lösung zu finden, die zwar nicht unbedingt optimal ist, aber höchstens um einen gewissen Faktor vom Optimum abweicht.

Dazu muß MSA zunächst als Optimierungsproblem formuliert werden. Anstatt zu einer Menge von Sequenzen und einer Zahl k zu fragen, ob ein Alignment mit höchstens Kosten k existiert, kann man das Problem auch direkt als Funktion, und damit als Optimierungsproblem, auffassen. Dazu betrachte ich die Funktion $MIN-MSA$ (Def. 2.8), die eine Eingabe von Sequenzen und einer Kostenfunktion auf die Kosten des optimalen Alignments abbildet. In den folgenden Abschnitten werde ich untersuchen, wie gut $MIN-MSA$ approximiert werden kann.

Für eine Instanz I eines Optimierungsproblems Q seien $OPT(I)$ die Kosten der optimalen Lösung von I .

Definition 4.4 (Approximation)

Sei Q ein Optimierungsproblem (Minimierung oder Maximierung) und $\delta > 1$ eine Konstante. Dann heißt Q δ -approximierbar, wenn es eine in polynomieller Zeit berechenbare Funktion f gibt, so daß für jede Instanz I von Q gilt:

$$\frac{1}{\delta} OPT(I) \leq f(I) \leq \delta \cdot OPT(I)$$

Ist Q ein Maximierungsproblem, so gilt $f(I) \leq OPT(I)$. Es ist also nach einem δ zu suchen, so daß $OPT(I) \leq \delta f(I)$. Analog ist für ein Minimierungsproblem nach einem δ zu suchen mit $OPT(I) \leq \delta^{-1} f(I)$.

Gibt es für ein Optimierungsproblem Q für jedes $\delta > 1$ einen polynomiellen δ -approximierenden Algorithmus, so besitzt Q ein $PTAS$ (*polynomial time approximation scheme*, siehe [3]). Ich werde zeigen, daß es für $MIN-MSA$ kein $PTAS$ gibt.

4.2.2 Die Klasse $MAX - \mathcal{SNP}$

Papadimitriou und Yannakakis führten in [17] die Klasse $MAX - \mathcal{SNP}$ ein. Unter anderem gibt es für diese Klasse auch den Begriff der Härte und Vollständigkeit. Diese sind über sogenannte L-Reduktionen definiert.

Definition 4.5 (L-Reduktion [17])

Seien P_1 und P_2 Optimierungsprobleme (Minimierungs- oder Maximierungsprobleme). Dann heißt P_1 L-reduzierbar auf P_2 , wenn es zwei in polynomieller Zeit berechenbare Funktionen f, g und Konstanten $\alpha, \beta > 0$ gibt, so daß für jede Instanz I_1 von P_1 gilt:

1. f erzeugt eine Instanz $f(I_1) = I_2$ von P_2 , so daß für die Optima von I_1 und I_2 $OPT(I_2) \leq \alpha OPT(I_1)$ gilt.
2. Für eine Lösung von I_2 mit Kosten c_2 erzeugt g eine Lösung von I_1 mit Kosten c_1 , so daß $|c_1 - OPT(I_1)| \leq \beta |c_2 - OPT(I_2)|$

Ist P_1 L-reduzierbar auf P_2 , so schreibe ich $P_1 \leq_L P_2$. L-Reduktionen sind transitiv (siehe [17]). Im Gegensatz zu many-one-Reduktionen, bei denen eine Sprache auf eine andere reduziert wird, wird bei L-Reduktionen ein Optimierungsproblem auf ein anderes reduziert. Ist $P_1 \leq_L P_2$, so führt ein Approximationsalgorithmus für P_2 sofort zu einem Approximationsalgorithmus für P_1 .

Für jedes $MAX - \mathcal{SNP}$ -vollständige Problem Q gibt es (siehe Arora [2]) zwei Konstanten $\delta_Q > \epsilon_Q > 1$, so daß Q δ_Q -approximierbar ist und daß aus einer ϵ_Q -Approximation sofort $\mathcal{P} = \mathcal{NP}$ folgen würde. Insbesondere kann es also unter der Voraussetzung $\mathcal{P} \neq \mathcal{NP}$ kein PTAS für Q geben.

Ich werde in diesem Abschnitt die $MAX - \mathcal{SNP}$ -Härte für $MIN - MSA$ zeigen.

$MAX - 3SAT$ ist das Problem, zu einer Booleschen Formel F in 3CNF die maximale Anzahl von Klauseln zu finden, die gleichzeitig erfüllt werden können. Analog ist $MAX - 1 - 2, 3SAT$ das Problem, zu einer Formel F in konjunktiver Normalform, die nur nichtnegierte Literale enthält und deren Klauseln aus entweder 2 oder 3 Literalen bestehen, die maximale Anzahl von Klauseln zu finden, die gleichzeitig eindeutig erfüllt werden können.

Nach [17] und Hochbaum [14] ist $MAX - 3SAT$ vollständig für $MAX - \mathcal{SNP}$.

4.2.3 $MAX - \mathcal{SNP}$ -Härte von MSA

Ich zeige zunächst, daß $MAX - 1 - 2, 3SAT$ $MAX - \mathcal{SNP}$ -hart ist. Für die Untersuchung von $MIN - MSA$ ist aber nur die Reduktion von Interesse.

Lemma 4.6

$MAX-1-2, 3SAT$ ist $MAX-SNP$ -hart.

Beweis: Ich zeige $MAX-3SAT \leq_L MAX-1-2, 3SAT$.

Sei $F = F_1 \wedge \dots \wedge F_r$ eine Formel in 3CNF über den Variablen x_1, \dots, x_n , $F_j = x_{j_1}^{\alpha_{j_1}} \vee x_{j_2}^{\alpha_{j_2}} \vee x_{j_3}^{\alpha_{j_3}}$.

Bilde daraus \tilde{F} wie folgt: Führe für $i = 1, \dots, n$ die neuen Variablen b_{i1}, b_{i2}, b_{i3} ein sowie für $j = 1, \dots, r$ die neuen Variablen a_{j1}, \dots, a_{j6} . Ersetze x_i durch x_i^+ und x_i^- . Setze

$$\beta_{j_i} = \begin{cases} + & \text{falls } \alpha_{j_i} = 1 \text{ (positives Literal),} \\ - & \text{falls } \alpha_{j_i} = 0 \text{ (negiertes Literal).} \end{cases}$$

Ersetze F_j durch:

$$\begin{aligned} & x_{j_1}^{\beta_{j_1}} \vee a_{j_1} \vee a_{j_4} \\ & x_{j_2}^{\beta_{j_2}} \vee a_{j_2} \vee a_{j_4} \\ & x_{j_3}^{\beta_{j_3}} \vee a_{j_3} \\ & a_{j_1} \vee a_{j_2} \vee a_{j_5} \\ & a_{j_3} \vee a_{j_4} \vee a_{j_6} \end{aligned} \quad (1)$$

sowie für $i=1,2,3$:

$$\begin{aligned} & x_{j_i}^+ \vee x_{j_i}^- \vee b_{j_i 1} \\ & x_{j_i}^+ \vee x_{j_i}^- \vee b_{j_i 2} \\ & b_{j_i 1} \vee b_{j_i 2} \vee b_{j_i 3} \end{aligned} \quad (2)$$

Pro Auftreten der Variable x_i in F befindet sich in \tilde{F} ein Block wie (2). Dadurch wird die konsistente Belegung der Variablen gesichert ($x_i^+ \neq x_i^-$ für alle $i = 1, \dots, n$).

Behauptung: In F sind maximal k Klauseln gleichzeitig erfüllbar \Leftrightarrow In \tilde{F} sind maximal $13r + k$ Klauseln gleichzeitig eindeutig erfüllbar.

Beweis: Man sieht, daß die konsistente Übertragung einer Belegung der x_i , die in F k Klauseln erfüllt, auf x_i^+ und x_i^- gerade $13r + k$ Klauseln erfüllt.

Es können in \tilde{F} auch immer mindestens $13r$ Klauseln erfüllt werden können. Zu zeigen bleibt also, daß, wenn in \tilde{F} $13r + k$ Klauseln gleichzeitig eindeutig erfüllt werden können, es immer auch eine konsistente Belegung der x_i^+ , x_i^- gibt, in der $13r + k$ Klauseln eindeutig erfüllt sind. Diese Belegung kann auf die x_i übertragen werden, so daß in F k Klauseln erfüllt sind.

Angenommen es können $13r + k$ Klauseln gleichzeitig eindeutig erfüllt werden und es gibt ein i_0 mit $x_{i_0}^+ = x_{i_0}^-$. Taucht x_{i_0} in p Klauseln auf, dann sind wg. (2) p Klauseln nicht erfüllt. Es können aber durch die Inkonsistenz maximal p Klauseln aus (1) zusätzlich erfüllt werden. Wird x_i^+, x_i^- konsistent belegt, so können also maximal p Klauseln aus (1) nicht mehr eindeutig erfüllt werden, dafür sind aber die p nicht eindeutig erfüllten Klauseln aus (2) zusätzlich erfüllt. Auf diese Weise kann sukzessive eine konsistente Belegung erzeugt werden, bei der mindestens $13r + k$ gleichzeitig eindeutig erfüllt werden können.

In jeder Formel in 3CNF ist mindestens die Hälfte aller Klauseln erfüllbar. Sind bei einer beliebigen Belegung der Variablen weniger als die Hälfte der Klauseln erfüllt, so invertiere die Belegung. Dann sind mindestens die Klauseln erfüllt, die vorher nicht erfüllt waren.

Wähle $\alpha = 28$ und $\beta = 1$. Dann gilt:

$$OPT(\tilde{F}) \leq 14r = 28 \frac{r}{2} \leq \alpha OPT(F)$$

Sei \tilde{c} das errechnete Resultat für \tilde{F} , dann gilt

$$|c - OPT(F)| \leq \beta |\tilde{c} - OPT(\tilde{F})|$$

wobei sich c aus $\tilde{c} = 13r + c$ ergibt, also $c = \tilde{c} - 13r$.

$$\Rightarrow MAX-3SAT \leq_L MAX-1-2,3SAT$$

$$\Rightarrow MAX-1-2,3SAT \text{ ist } MAX-SNP\text{-hart.} \quad \blacksquare$$

Theorem 4.7

MIN-MSA ist MAX-SNP-hart für $|\Sigma| \geq 9$.

Beweis: Ich werde $MAX-3SAT \leq_L MIN-MSA$ zeigen. Dazu verwende die Reduktionen aus Theorem 4.1 und Lemma 4.6. Aus technischen Gründen reduziere ich nicht direkt von $MAX-1-2,3SAT$.

Sei F eine Formel in 3CNF mit r Klauseln. Berechne dazu die zugehörigen Sequenzen \mathcal{S} wie in Theorem 4.1 und Lemma 4.6. Sind in F k Klauseln erfüllbar, so sind in der nach 4.6 berechneten Formel $13r + k$ von $14r$ Klauseln eindeutig erfüllbar und damit ergeben sich für ein optimales Alignment von \mathcal{S} die Kosten

$$D(\mathcal{S}) = 12(14r) - 2(13r + k) = 142r - 2k.$$

Seien \tilde{D} eine Lösung, d. h. die Kosten eines Alignments, von \mathcal{S} und \tilde{c} die daraus zu errechnende Lösung, d. h. Anzahl gleichzeitig zu erfüllender Klauseln, von F . Dabei gilt $142r - 2\tilde{c} = \tilde{D}$, also $\tilde{c} = 71r + \frac{\tilde{D}}{2}$.

Wähle $\alpha = 284$ und $\beta = \frac{1}{2}$, dann gilt

$$OPT(\mathcal{S}) \leq 142r \leq 284r \frac{r}{2} \leq \alpha OPT(F)$$

und

$$\begin{aligned} |\tilde{c} - OPT(F)| &= \left| 71r + \frac{\tilde{D}}{2} - \left(71r + \frac{OPT(\mathcal{S})}{2} \right) \right| \\ &\leq \beta |\tilde{D} - OPT(\mathcal{S})|. \end{aligned}$$

Es gilt also $MAX-3SAT \leq_L MIN-MSA$. Da $MAX-3SAT$ $MAX-\mathcal{SNP}$ -vollständig ist, ist folglich $MIN-MSA$ $MAX-\mathcal{SNP}$ -hart. ■

Korollar 4.8

Unter der Voraussetzung $\mathcal{P} \neq \mathcal{NP}$ gibt es kein PTAS für $MIN-MSA$.

Beweis: Die Behauptung folgt aus der $MAX-\mathcal{SNP}$ -Härte von $MIN-MSA$ und der Aussage in Arora [2], daß kein $MAX-\mathcal{SNP}$ -hartes Problem ein PTAS besitzt, wenn $\mathcal{P} \neq \mathcal{NP}$ ist. ■

4.2.4 Nichtapproximierbarkeitsschranke für MSA

Da $MIN-MSA$ $MAX-\mathcal{SNP}$ -hart ist muß es eine Konstante γ geben, so daß eine $(\gamma-\delta)$ -Approximation für beliebige $\delta > 0$ nicht möglich ist außer es gilt $\mathcal{P} = \mathcal{NP}$. In diesem Abschnitt zeige ich, daß, unter der Annahme $\mathcal{P} \neq \mathcal{NP}$, $MIN-MSA$ nicht besser approximiert werden kann als $\frac{561}{560} \approx 1.00178$. Dafür verwende ich die Aussage aus Hastad [13], daß aus einer δ -Approximation für $MAX-3SAT$ mit $\delta < \frac{8}{7}$ $\mathcal{P} = \mathcal{NP}$ folgen würde.

Theorem 4.9

Unter der Voraussetzung $\mathcal{P} \neq \mathcal{NP}$ gibt es keine ϵ -Approximation für $MIN-MSA$ für beliebige $\epsilon < \frac{561}{560}$.

Beweis: Ich verwende die Reduktion aus Theorem 4.7. Angenommen es gibt einen ϵ -approximativen Algorithmus für $MIN-MSA$, $\epsilon > 1$. Seien in einer Formel in 3CNF mit r Klauseln k Klauseln gleichzeitig erfüllbar. Dann hat das zugehörige Alignment Kosten $142r - 2k$. Es gibt also einen Approximationsalgorithmus für $MIN-MSA$, der ein Alignment mit Kosten kleiner oder gleich $\epsilon \cdot (142r - 2k)$. Da $MAX-3SAT$, unter der Annahme $\mathcal{NP} \neq \mathcal{P}$, keinen δ -Approximationsalgorithmus für $\delta < \frac{8}{7}$ besitzt (siehe [13]), gibt es keinen Approximationsalgorithmus für $MAX-3SAT$, der garantiert eine

Belegung der Variablen liefert, die mindestens $\frac{7}{8}k$ Klauseln simultan erfüllt. Es gilt also folgende Ungleichung für beliebige $k \in [0 : r]$.

$$\begin{aligned} (142r - 2k)\epsilon &\geq 142r - \frac{7}{8}2k \\ \Leftrightarrow \epsilon &\geq \frac{142r - \frac{7k}{4}}{142r - 2k} \end{aligned}$$

Für $k = r$ ergibt sich damit

$$\epsilon \geq \frac{561}{560}$$

Da $MAX-3SAT$ nicht besser als $\frac{8}{7}$ approximiert werden kann, gilt folglich $\epsilon \geq \frac{142 - \frac{7}{4}}{140} = \frac{561}{560}$. Gäbe es eine ϵ -Approximation für $MIN-MSA$ mit $\epsilon < \frac{561}{560}$, so würde diese eine δ -Approximation für $MAX-3SAT$ liefern mit $\delta < \frac{8}{7}$. Daraus würde $\mathcal{P} = \mathcal{NP}$ folgen. ■

5 Approximationsalgorithmen

Thema dieses Kapitels sind schnelle Algorithmen zur Berechnung eines Alignments und Abschätzung des maximal möglichen Fehlers.

Im folgenden sei für die Kostenfunktion d die Dreiecksungleichung vorausgesetzt, d. h. $d(x, z) \leq d(x, y) + d(y, z)$ für alle $x, y, z \in \Sigma_\Delta$.

5.1 Die Center-Star-Methode

Grundlage dieses Abschnittes liefert das folgende Theorem. Darauf basiert die Idee Approximationsalgorithmen mit Hilfe von k -Sternen zu konstruieren.

Theorem 5.1 (Feng, Doolittle [6], Gusfield [12])

Für jede Menge von Sequenzen \mathcal{S} und jeden Baum $T = (\mathcal{S}, E)$ gibt es ein Alignment \mathcal{A} , so daß für alle $\{S_i, S_j\} \in E$ gilt:

$$D(\{S_i, S_j\}|\mathcal{A}) = D(\{S_i, S_j\})$$

Ein solches Alignment werde ich kompatibel zu T nennen.

Beispiel 5.2

Betrachte den Baum in Abb. 3. Mit der Kostenfunktion $d(x, y) = 1$ für $x \neq y$ ergeben sich die folgenden paarweise optimalen Alignments:

$$S_1, S_2: \begin{pmatrix} \text{AGC AT} \\ \text{AGC}\Delta\text{T} \end{pmatrix} \quad S_2, S_3: \begin{pmatrix} \text{AGCT} \\ \text{ATAT} \end{pmatrix} \quad S_3, S_4: \begin{pmatrix} \text{ATAT}\Delta \\ \text{ATCTG} \end{pmatrix} \quad S_3, S_5: \begin{pmatrix} \text{AT}\Delta\text{AT} \\ \text{ATC AT} \end{pmatrix}$$

Links ist das daraus zu konstruierende zu dem Baum in Abb. 3 kompatible Alignment \mathcal{A}_{comp} , rechts das optimale Alignment \mathcal{A}_{opt} der 5 Sequenzen dargestellt:

$$\mathcal{A}_{comp} = \begin{pmatrix} \text{AG}\Delta\text{C AT}\Delta \\ \text{AG}\Delta\text{C}\Delta\text{T}\Delta \\ \text{AT}\Delta\text{A}\Delta\text{T}\Delta \\ \text{AT}\Delta\text{C}\Delta\text{TG} \\ \text{ATC A}\Delta\text{T}\Delta \end{pmatrix} \quad \mathcal{A}_{opt} = \begin{pmatrix} \text{AGC AT}\Delta \\ \text{AGC}\Delta\text{T}\Delta \\ \text{AT}\Delta\text{AT}\Delta \\ \text{ATC}\Delta\text{TG} \\ \text{ATC AT}\Delta \end{pmatrix}$$

Es gilt $D(\mathcal{A}_{comp}) = 24$ und $D(\mathcal{A}_{opt}) = 18$. Ein kompatibles Alignment ist also nicht unbedingt optimal. Andererseits ist $D(\{S_2, S_3\}|\mathcal{A}_{opt}) = 3$ und $D(\{S_2, S_3\}|\mathcal{A}_{comp}) = D(S_2, S_3) = 2$. Es kann also durchaus vorkommen, daß ein durch ein optimales Alignment induziertes Alignment nicht optimal ist.

In [11] und [12] beschreibt Gusfield den sogenannten center-star-Algorithmus zur Berechnung eines Alignments mit garantierter Fehlerschranke. Ich werde sie hier vorstellen, da sie die Grundidee für weitere Überlegungen liefert. Es werden dafür sogenannte 2-Sterne benötigt (in [11]: center-star). Ich definiere eine Verallgemeinerung, die sogenannten k -Sterne, die in den weiteren Abschnitten gebraucht werden.

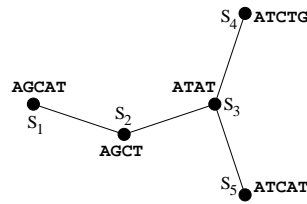
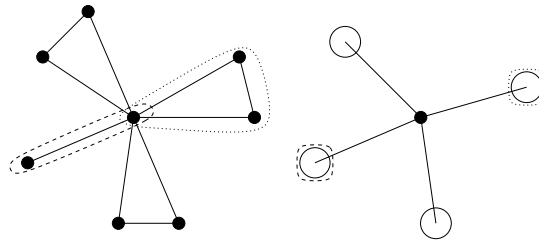


Abbildung 3: Ein Baum mit Sequenzen als Knoten

Abbildung 4: Ein Graph G (3-Stern mit 2-Clique, links) und der daraus resultierende Graph \tilde{G} (rechts). Zwei maximale Cliques in G sind umrandet.**Definition 5.3 (k -Stern)**

Ein Graph $G = (V, E)$ heißt k -Stern, wenn sich V in paarweise disjunkte Teilmengen V_1, \dots, V_j, C zerlegen läßt mit

1. $|C| = 1$
2. $|V_i| = k - 1$ für alle $i \in [1 : j]$
3. $E = E_1 \cup E_2 \cup \dots \cup E_j$, wobei $E_i = \{\{v, w\} | v \neq w \text{ und } v, w \in V_i \cup C\}$.

G heißt k -Stern mit q -Clique für $2 \leq q \leq k-1$, wenn 1 und 3 gelten und $|V_j| = q-1$, $|V_i| = k-1$ für $i = 1, \dots, j-1$.

Anschaulich hat ein k -Stern G ein Zentrum C mit einem Knoten, welcher in jeder der j k -Cliques $(V_i \cup C, E_i)$ des Graphen enthalten ist. Abgesehen von dem Knoten aus C sind die Cliques paarweise disjunkt.

Es ist klar, daß ein k -Stern nur existiert, wenn $|V| \equiv 1 \pmod{k-1}$. Ist dies nicht der Fall, so wird eine zusätzliche q -Clique mit den restlichen $q-1$ Knoten angefügt, es entsteht ein k -Stern mit q -Clique. In Abb. 4 ist links ein 3-Stern mit 2-Clique dargestellt.

2-Sterne existieren für jede Anzahl Knoten, 3-Sterne nur für eine ungerade Anzahl Knoten. Für eine gerade Anzahl Knoten existiert aber immer ein 3-Stern mit 2-Clique.

Basis für den center-star-Algorithmus ist Theorem 5.4.

Theorem 5.4 (Gusfield [11, 12])

Sei \mathcal{A}_c ein zu dem 2-Stern mit Zentrum S_c kompatibles Alignment von \mathcal{S} . Dann gilt:

$$\min_{c \in [1:n]} \frac{D(\mathcal{A}_c)}{D(\mathcal{A}(\mathcal{S}))} \leq 2 - \frac{2}{n}$$

Die Funktion $\mathcal{A}, D = \text{compatible_alignment}(G)$ liefert ein zu einem Graphen G kompatibles Alignment \mathcal{A} und dessen Kosten D . G ist dabei eine Konfiguration mit Knotenmenge \mathcal{S} . Bäume und k -Sterne sind ein Spezialfall der Konfigurationen. Konfigurationen werden in Abschnitt 5.2 benötigt (siehe Def. 5.7). 2-Sterne sind auch Bäume.

Aus Theorem 5.4 ergibt sich der folgende Algorithmus zur Berechnung eines Alignments von n Sequenzen mit garantierter Fehlerschranke $2 - \frac{2}{n}$.

Algorithmus 1 (center-star-Algorithmus)

Input: Sequenzen $\mathcal{S} = \{S_1, \dots, S_n\}$

Output: Alignment \mathcal{A}^* mit Kosten D^*

- 1: **for** $i = 1$ to n **do** {Berechne alle paarweisen Alignments}
- 2: **for** $j = 1$ to n **do**
- 3: $\mathcal{A}_{ij} = \mathcal{A}(S_i, S_j)$
- 4: $D_{ij} = D(\mathcal{A}_{ij})$
- 5: **end for**
- 6: **end for**
- 7: **for** $j = 1$ to n **do**
- 8: $D_j = \sum_{i=1}^n D_{ij}$
- 9: **end for**
- 10: $c \in [1:n]$ mit $D_c \leq D_i \forall i \in [1:n]$
- 11: Erstelle 2-Stern G_c mit Knoten \mathcal{S} , Mittelpunkt S_c
- 12: $\mathcal{A}^*, D^* = \text{compatible_alignment}(G_c)$

Algorithmus 1 hat eine Laufzeit von $O(n^2 l^2)$. Der Beweis der Fehlerschranke beruht im wesentlichen auf den folgenden beiden Lemmata und Theorem 5.1.

Lemma 5.5 (Gusfield [11, 12])

Erfüllt die Kostenfunktion d die Dreiecksungleichung, so gilt für beliebige Sequenzen S_i, S_j, S_k :

$$D(S_i, S_k) \leq D(S_i, S_j) + D(S_j, S_k)$$

Lemma 5.6 (Gusfield [11, 12])

Die Kosten eines optimalen Alignments von \mathcal{S} sind mindestens so groß wie die Summe der paarweise optimalen Alignments der einzelnen Sequenzen:

$$D(\mathcal{S}) \geq \sum_{1 \leq i < j \leq n} D(S_i, S_j)$$

5.2 Verallgemeinerung auf k -Sterne

In diesem Abschnitt wird die 2-Stern-Methode (Theorem 5.4) auf k -Sterne (Def. 5.3) verallgemeinert. Dazu führt Pevzner in [18] den Begriff der Konfiguration ein.

Definition 5.7 (Konfiguration)

Sei $V = [1 : n]$ und $G = (V, E)$ ein Graph mit den maximalen Cliques $\Omega_1, \dots, \Omega_t$. Sei $W_1 = [1 : t]$ und $W_2 = \{v | v \in \Omega_i \cap \Omega_j \text{ für } 1 \leq i < j \leq t\}$. Definiere einen Graphen $\tilde{G} = (W_1 \cup W_2, \tilde{E})$ mit $\tilde{E} = \{(i, v) | v \in \Omega_i \cap \Omega_x \text{ für ein } \Omega_x\}$.

Ist \tilde{G} ein Baum, so heißt G Konfiguration.

Bäume sind Konfigurationen, die ausschließlich aus 2-Cliquen bestehen. Bei k -Sternen und k -Sternen mit q -Clique handelt es sich ebenfalls um Konfigurationen (siehe Abb. 4).

Analog zu der Definition eines kompatiblen Alignments für einen Baum ist die Kompatibilität eines Alignments zu einer Konfiguration G definiert.

Definition 5.8 (Kompatibilität)

Ein Alignment \mathcal{A} von \mathcal{S} heißt kompatibel zu einer Konfiguration $G = (\mathcal{S}, E)$, falls für alle maximalen Cliques $\Omega \subseteq \mathcal{S}$ von G

$$D(\Omega | \mathcal{A}) = D(\Omega)$$

gilt.

\mathcal{A}^G bezeichnet ein optimales zu einer Konfiguration G kompatibles Alignment.

Die Existenz eines solchen Alignments ist eine direkte Verallgemeinerung von Theorem 5.1.

Für die Fehlerabschätzung wird der Begriff der Verbindungskosten (in [18]: communication cost) benötigt.

Definition 5.9 (Verbindungskosten)

Sei $G = (V, E)$ ein Graph, dann sind die Verbindungskosten von G

$$c(G) := \sum_{v, w \in V} l(v, w),$$

wobei $l(v, w)$ die Anzahl der Kanten des kürzesten Weges zwischen v und w ist. Die Länge eines Weges ist dabei die Anzahl der Kanten des Weges.

$$b(G) := \frac{c(G)}{|V|(|V| - 1)}$$

sind die normalisierten Verbindungskosten von G .

Für den vollständigen Graphen H_n mit n Knoten gilt: $c(H_n) = n(n-1)$ und $b(H_n) = 1$. Für jeden anderen Graphen mit n Knoten $G \neq H_n$ gilt: $b(G) > b(H_n)$.

Insbesondere gilt für k -Sterne G mit n Knoten $b(G) = 2 - \frac{k}{n}$.

\mathcal{G}^G bezeichnet die Menge aller zu einem Graphen G isomorphen Graphen mit Knotenmenge \mathcal{S} .

Unter allen zu Graphen aus \mathcal{G}^G kompatiblen Alignments erfüllt dasjenige mit minimalen Kosten die folgende Abschätzung:

Theorem 5.10 (Pevzner [18])

Sei \mathcal{S} eine Menge von n Sequenzen und $G = (V, E)$ eine Konfiguration mit $|V| = n$. Dann gilt:

$$\min_{G' \in \mathcal{G}^G} \frac{D(\mathcal{A}^{G'})}{D(\mathcal{S})} \leq b(G)$$

Das Alignment mit minimalen Kosten unter allen zu Graphen aus \mathcal{G}^G kompatiblen Alignments heißt optimales kompatibles Alignment zu \mathcal{G}^G .

5.2.1 Approximation mit k -Sternen

Mit Theorem 5.10 ergibt sich für das optimale kompatible Alignment zu einem k -Stern eine Fehlerabschätzung von $2 - \frac{k}{n}$. Für $k = 2$ ergibt sich Theorem 5.4. Ist $k = 3$, so gilt für eine gerade Anzahl Sequenzen, also einen 3-Stern mit 2-Clique eine Fehlerabschätzung von $2 - \frac{3}{n} + \frac{1}{n(n-1)}$ (siehe Pevzner [18]).

Für eine gegebene Konfiguration, in dem die Größe der maximalen Cliques durch eine Konstante beschränkt ist, läßt sich ein kompatibles Alignment effizient berechnen.

Das Problem ist die große Menge der isomorphen Graphen zu einer Konfiguration, die alle untersucht werden müssen. Für $k = 2$ sind nur n viele Fälle zu untersuchen (Gusfield [11, 12]), für $k = 3$ ist das Problem durch perfektes minimales Matching zu lösen (siehe Gabow [7], Galil [8], Pevzner [18]).

Für $k = 4$ entspricht das Finden des optimalen Graphen einem perfektem minimalen 3-dimensionalen Matching. Diese Problem ist \mathcal{NP} -vollständig (siehe Garey, Johnson [9]).

Es ist aber für die Fehlerabschätzung nicht erforderlich, das optimale kompatible Alignment zu G zu finden. Ausreichend ist ein Alignment mit Kosten kleiner oder gleich dem Durchschnitt der Kosten über alle kompatiblen Alignments zu Graphen aus \mathcal{G} .

5.2.2 Ausgewogene Mengen

Bafna et al. führen dazu den Begriff der ausgewogenen Menge ein (in [4]: balanced set).

Definiere für einen k -Stern G mit Knoten \mathcal{S} und Zentrum $S_c \in \mathcal{S}$ die Gewichtsmatrix $C_G = (c_{ij})_{i,j=1}^n$ mit

$$(C_G)_{ij} = c_{ij} := \begin{cases} n - (k - 1) & \text{für } i \neq j \text{ und } i = c \text{ oder } j = c \\ 1 & \text{für } i, j \neq c \text{ und } S_i, S_j \text{ in der gleichen Clique} \\ 0 & \text{sonst} \end{cases} .$$

Für einen k -Stern G mit q -Clique Ω und Zentrum S_c ist die Gewichtsmatrix definiert durch:

$$(C_G)_{ij} = c_{ij} := \begin{cases} n - (k - 1) & \text{für } i \neq j = c \text{ und } S_i \notin \Omega \text{ oder umgekehrt} \\ n - (q - 1) & \text{für } i \neq j = c \text{ und } S_i \in \Omega \text{ oder umgekehrt} \\ 1 & \text{für } i, j \neq c \text{ und } S_i, S_j \text{ in der gleichen Clique} \\ 0 & \text{sonst} \end{cases} .$$

$D(\mathcal{A}, C_G)$ sind die gewichteten Kosten des Alignments \mathcal{A} in Bezug auf G . Dann gilt folgendes Lemma aus [4].

Lemma 5.11 (Bafna et al. [4])

Für jedes Alignment \mathcal{A} von \mathcal{S} und k -Stern oder k -Stern mit q -Clique G mit Knoten \mathcal{S} gilt:

$$D(\mathcal{A}) \leq D(\mathcal{A}, C_G)$$

Bei dieser Abschätzung kommt die gleiche Idee zum Tragen wie beim Beweis der Fehlerschranken in Theorem 5.4 und 5.10:

Für zwei benachbarte Sequenzen, d. h. zwei Sequenzen aus einer Clique, werden direkt die Kosten des optimalen Alignments genommen. Für zwei Sequenzen, die nicht in einer Clique liegen, werden die Kosten mit Hilfe der Dreiecksungleichung über die Summe der Kosten zur Sequenz im Zentrum abgeschätzt. Auf diese Weise ergeben sich die oben aufgeführten Gewichte, da es genau $n - (k - 1)$ Sequenzen für eine bestimmte Sequenz S_i gibt, die nicht in der gleichen Clique wie S_i liegen. Das optimale Alignment wird nach unten durch die Summe der Kosten der paarweise optimalen Alignments abgeschätzt.

Definition 5.12 (Ausgewogene Menge)

Sei \mathcal{G} eine Menge von k -Sternen oder k -Sternen mit q -Clique mit Knoten \mathcal{S} . Gilt für eine Konstante $p \in \mathbb{N}$

$$\sum_{G \in \mathcal{G}} (C_G)_{ij} = \begin{cases} p & \text{für } i \neq j \\ 0 & \text{sonst} \end{cases}$$

so heißt \mathcal{G} ausgewogen mit Parameter p .

Ein Beispiel für eine ausgewogene Menge ist die Menge aller zu einem k -Stern isomorphen Graphen mit Knoten \mathcal{S} . Auch die Menge aller zu einem k -Stern mit q -Clique isomorphen Graphen mit Knoten \mathcal{S} ist ausgewogen. Für ausgewogene Mengen zeigen Bafna et al. das folgende Lemma.

Lemma 5.13 (Bafna et al. [4])

Ist \mathcal{G} eine ausgewogene Menge von k -Sternen oder k -Sternen mit q -Clique mit Parameter p , dann gilt:

$$\min_{G \in \mathcal{G}} D(\mathcal{A}^G, C_G) \leq \frac{p}{|\mathcal{G}|} D(\mathcal{S})$$

Die Menge \mathcal{G}^G für eine Konfiguration G ist ausgewogen und es ergibt sich die Fehlerabschätzung aus Theorem 5.10. Damit wäre aber nichts gewonnen, da wiederum alle k -Sterne betrachtet werden müßten.

Die Aufgabe ist nun, möglichst kleine ausgewogene Mengen von k -Sternen zu finden. Leider sind solche kleinen ausgewogenen Mengen nicht für alle n und k leicht zu finden.

5.3 Beschleunigung der 3-Stern-Methode

Pevzner stellt in [18] einen Algorithmus vor, der mit Hilfe von 3-Sternen und perfektem minimalen Matching ein Alignment von n Sequenzen mit Fehlerschranke $2 - \frac{3}{n}$, falls n ungerade ist, und $2 - \frac{3}{n} + \frac{1}{n(n-1)}$, falls n gerade ist, berechnet. Die Laufzeit des Algorithmus ist $O(n^3 l^3 + n^4)$.

Ich werde diesen hier mit Hilfe ausgewogener Menge beschleunigen, so daß sich eine Laufzeit von $O(n^2 l^3)$ und eine garantierte Fehlerschranke von $2 - \frac{3}{n}$ für eine ungerade Anzahl n von Sequenzen und eine Laufzeit $O(n^3 l^3)$ und eine garantierte Fehlerschranke von $2 - \frac{3}{n} + \frac{1}{n(n-1)}$ für eine gerade Anzahl n von Sequenzen ergibt.

Im folgenden ist $\mathcal{S} = \{S_0, \dots, S_{n-1}\}$.

5.3.1 Kleine ausgewogene Mengen von 3-Sternen

Die Beschleunigung der 3-Stern-Methode basiert auf der Konstruktion von kleinen ausgewogenen Mengen.

Sei zunächst n ungerade. Ich konstruiere eine Menge \mathcal{G}_n von 3-Sternen, bei der jede Sequenz S_i Zentrum von genau einem 3-Stern G_i ist und jedes Paar von Sequenzen S_i, S_j in genau einem 3-Stern $c \neq i, j$ in der gleichen Clique liegt.

Sei $\mathcal{G}_n = \{G_0, \dots, G_{n-1}\}$ mit $G_c = (\mathcal{S}, E_c)$ und $E_c = E'_c \cup E''_c$, wobei $E'_c = \{\{S_c, S_i\} | i \neq c\}$ und $E''_c = \{\{S_i, S_j\} | i, j \neq c \text{ und } i + j \equiv 2c \pmod{n}\}$. G_c ist ein 3-Stern mit Zentrum S_c .

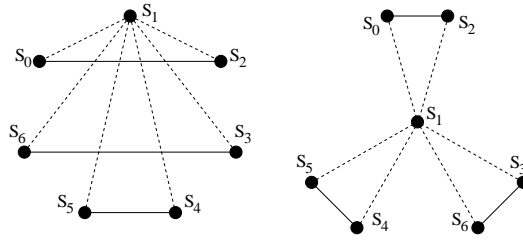


Abbildung 5: Eine möglicher 3-Stern: G_1 aus \mathcal{G}_7 . Die Kanten aus E_1' sind mit gestrichelten, die Kanten aus E_1'' mit durchgezogenen Linien dargestellt.

Ein 3-Stern G_c kann anschaulich dadurch konstruiert werden, daß alle Sequenzen S_0, \dots, S_{n-1} auf einem Kreis abgetragen werden, alle Sequenzen mit S_c verbunden werden und dann zwei Sequenzen verbunden werden, wenn sie den gleichen Abstand zu S_c haben (siehe Abb. 5).

Lemma 5.14

Sei $n \in \mathbb{N}$ ungerade. Dann gibt es für $i, j \in [0 : n-1]$, $i \neq j$, genau ein $c \in [0 : n-1]$, $c \neq i, j$, so daß sich S_i und S_j in der gleichen Clique in G_c befinden.

Beweis: Seien $i, j \in [0 : n-1]$ beliebig gewählt, $i \neq j$. Da $0 \leq i, j \leq n-1$ gilt $i + i \not\equiv i + j \not\equiv j + j \pmod{n}$. Es gibt also ein $\tilde{c} \in [0 : n-1]$ mit $\tilde{c} \not\equiv 2i, 2j \pmod{n}$, so daß $\tilde{c} \equiv i + j \pmod{n}$.

Ist \tilde{c} gerade, so wähle $c = \frac{\tilde{c}}{2}$. Dann ist $c \neq i, j$ und S_i, S_j befinden sich in G_c in der gleichen Clique.

Ist \tilde{c} ungerade, so wähle $c = \frac{\tilde{c}+n}{2} \pmod{n}$. $\tilde{c} + n$ ist gerade, da n ungerade ist. Es ist wiederum $c \neq i, j$. Wäre o. E. $i = c$, dann würde $2i \equiv 2c \pmod{n}$, also $2i \equiv \tilde{c} + n \pmod{n}$ und damit $\tilde{c} = 2i \pmod{n}$ gelten.

Es gibt also für jedes Paar von Sequenzen S_i, S_j ein $c \neq i, j$, so daß sich S_i, S_j in G_c in einer Clique befinden.

Zu zeigen bleibt noch die Eindeutigkeit.

Angenommen es gibt $c, \hat{c} \in [0 : n-1]$ mit $c, \hat{c} \neq i, j$, so daß sich S_i und S_j in G_c und $G_{\hat{c}}$ in einer Clique befinden und $c \neq \hat{c}$.

Dann muß nach Konstruktion der Graphen $2c \equiv 2\hat{c} \pmod{n}$ gelten. Da n ungerade und somit teilerfremd zu 2 ist, folgt daraus $c = \hat{c}$. Dies ist ein Widerspruch zu $c \neq \hat{c}$.

■

Für zwei Sequenzen S_i, S_j gibt es also nach Konstruktion von \mathcal{G}_n jeweils genau einen Graphen in \mathcal{G}_n mit S_i als Zentrum, einen mit S_j als Zentrum und einen, indem sich S_i, S_j in der gleichen Clique befinden, aber nicht Zentrum sind.

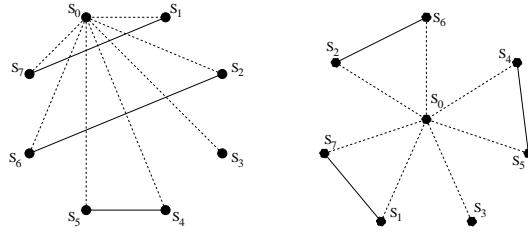


Abbildung 6: Eine möglicher 3-Stern mit 2-Clique: G_{03} aus \mathcal{G}_8 . Die Kanten aus E'_{03} sind mit gestrichelten, die Kanten aus E''_{03} mit durchgezogenen Linien dargestellt.

Damit kann nun folgendes Theorem bewiesen werden:

Theorem 5.15

Sei $n \in \mathbb{N}$ ungerade. Dann ist \mathcal{G}_n eine ausgewogene Menge von 3-Sternen und es gilt:

$$\min_{G \in \mathcal{G}_n} D(\mathcal{A}^G) \leq \left(2 - \frac{3}{n}\right) D(\mathcal{A}(\mathcal{S}))$$

Beweis: Es gilt $|\mathcal{G}_n| = n$. Damit bleibt zu zeigen, daß \mathcal{G}_n eine ausgewogene Menge mit Parameter $2n - 3$ ist. Dann gilt mit Lemma 5.11 und 5.13:

$$\min_{G \in \mathcal{G}_n} D(\mathcal{A}^G) \stackrel{5.11}{\leq} \min_{G \in \mathcal{G}_n} D(\mathcal{A}^G, C_G) \stackrel{5.13}{\leq} \frac{2n-3}{|\mathcal{G}_n|} D(\mathcal{A}(\mathcal{S})) = \left(2 - \frac{3}{n}\right) D(\mathcal{A}(\mathcal{S}))$$

Sei $C = \sum_{G \in \mathcal{G}_n} C_G$, $C = (c_{ij})_{s_i, s_j \in \mathcal{S}}$. Es ist $c_{ii} = 0$. Für beliebige $i \neq j$ gilt:

$$c_{ij} = \underbrace{n-2}_{\text{Zentrum } S_i} + \underbrace{n-2}_{\text{Zentrum } S_j} + \underbrace{1}_{S_i, S_j \text{ in gleicher Clique}} = 2n - 3 \quad (5.14)$$

\mathcal{G}_n ist also ausgewogen mit Parameter $2n - 3$. ■

Für eine gerade Anzahl von Sequenzen ist die Konstruktion einer ausgewogenen Menge wegen der zusätzlichen 2-Clique schwieriger.

Sei nun $n \in \mathbb{N}$ gerade. Ich konstruiere eine ausgewogene Menge $\mathcal{G}_n = \{G_{ce} | c, e \in [0 : n-1], c \neq e\}$ von 3-Sternen mit 2-Clique. Dazu seien $\varphi_e : [0 : n-1] \setminus \{e\} \rightarrow [0 : n-2]$ bijektive Funktionen für $e \in [0 : n-1]$ definiert durch

$$\varphi_e(x) = \begin{cases} x & \text{für } x < e \\ x - 1 & \text{für } x > e \end{cases}$$

Damit ist $G_{ce} = (\mathcal{S}, E_{ce})$ und $E_{ce} = E'_{ce} \cup E''_{ce}$, wobei $E'_{ce} = \{\{S_i, S_c\} | i \neq c\}$ und $E''_{ce} = \{\{S_i, S_j\} | i, j \neq c, e \text{ und } \varphi_e(i) + \varphi_e(j) \equiv 2\varphi_e(c) \pmod{n-1}\}$.

Anschaulich ist G_{ce} ein 3-Stern mit Zentrum S_c und 2-Clique $\{S_e, S_c\}$ (Abb. 6). Die 3-Cliquen sind, nach auslassen von S_e und "zusammenschieben" der restlichen Sequenzen mit der Funktion φ_e , genauso konstruiert wie für eine ungerade Anzahl von Sequenzen.

Zunächst eine Aussage über die Zusammensetzung von \mathcal{G}_n .

Lemma 5.16

Sei $n \in \mathbb{N}$ gerade. Dann gibt es für $i, j \in [0 : n - 1]$, $i \neq j$, genau $n - 2$ Graphen in \mathcal{G}_n , so daß S_i, S_j in der gleichen Clique liegen und $i, j \neq c$.

Beweis: Zu zeigen ist, daß es genau $n - 2$ Paare $c, e \in [0 : n - 1]$, $c \neq e$, gibt, so daß $c, e \neq i, j$ und $\varphi_e(i) + \varphi_e(j) \equiv 2\varphi_e(c) \pmod{n}$.

Sei $e \in [0 : n - 1] \setminus \{i, j\}$ beliebig gewählt. Dann gibt es genau ein $\tilde{c} \in [0 : n - 2]$ mit $\varphi_e(i) + \varphi_e(j) \equiv 2\tilde{c} \pmod{n - 1}$ und es gilt $\varphi_e(i), \varphi_e(j) \neq \tilde{c}$. Der Beweis ist analog zu dem Beweis von Lemma 5.14.

Da φ_e bijektiv ist gibt es folglich genau ein $c \in [0 : n - 1]$ mit $\varphi_e(c) = \tilde{c}$. Nach Definition von φ_e ist $c \neq e$. Würde $c = i$ oder $c = j$ gelten, so würde $\tilde{c} = \varphi_e(i)$ oder $\tilde{c} = \varphi_e(j)$ sein. Es gilt also $c \in [0 : n - 1] \setminus \{i, j, e\}$ und $\varphi_e(i) + \varphi_e(j) \equiv 2\varphi_e(c) \pmod{n - 1}$. Nach Konstruktion der Graphen befinden sich S_i, S_j in G_{ce} in der gleichen Clique. Da φ_e bijektiv ist und \tilde{c} eindeutig bestimmt ist, ist auch c eindeutig bestimmt.

Es gibt also für jedes $e \in [0 : n - 1] \setminus \{i, j\}$ genau ein $c \neq i, j$ mit $\{S_i, S_j\} \in E_{ce}$. Aus $|[0 : n - 1] \setminus \{i, j\}| = n - 2$ folgt die Behauptung. ■

Theorem 5.17

Sei $n \in \mathbb{N}$ gerade. Dann ist \mathcal{G}_n eine ausgewogene Menge von 3-Sternen mit 2-Clique und es gilt:

$$\min_{G \in \mathcal{G}_n} D(\mathcal{A}^G) \leq \left(2 - \frac{3}{n} + \frac{1}{n(n-1)} \right) D(\mathcal{A}(\mathcal{S}))$$

Beweis: Zu zeigen ist, daß \mathcal{G}_n eine ausgewogene Menge mit Parameter $2n^2 - 5n + 4$ ist. Dann gilt mit Lemma 5.11 und 5.13 und $|\mathcal{G}_n| = n(n - 1)$:

$$\begin{aligned} \min_{G \in \mathcal{G}_n} D(\mathcal{A}^G) &\stackrel{5.11}{\leq} \min_{G \in \mathcal{G}_n} D(\mathcal{A}^G, C_G) \stackrel{5.13}{\leq} \frac{2n^2 - 5n + 4}{|\mathcal{G}_n|} D(\mathcal{A}(\mathcal{S})) \\ &= \left(2 - \frac{3}{n} + \frac{1}{n(n-1)} \right) D(\mathcal{A}(\mathcal{S})) \end{aligned}$$

Sei $C = \sum_{G \in \mathcal{G}_n} C_G$, $C = (c_{ij})_{S_i, S_j \in \mathcal{S}}$. Es ist $c_{ii} = 0$. Für beliebige $i \neq j$ gilt:

$$\begin{aligned}
c_{ij} &= \underbrace{n-2}_{S_i, S_j \text{ in gleicher Clique (5.16)}} + \underbrace{(n-2)(n-2)}_{S_i \text{ Zentrum, } S_j \text{ in 3-Clique}} + \underbrace{n-1}_{S_i \text{ Zentrum, } S_j \text{ in 2-Clique}} \\
&+ \underbrace{(n-2)(n-2)}_{S_j \text{ Zentrum, } S_i \text{ in 3-Clique}} + \underbrace{n-1}_{S_j \text{ Zentrum, } S_i \text{ in 2-Clique}} \\
&= 2n^2 - 5n + 4
\end{aligned}$$

\mathcal{G}_n ist also ausgewogen mit Parameter $2n^2 - 5n + 4$. ■

5.3.2 Die Algorithmen

Mit dem Theorem 5.15 ergibt sich der folgende Algorithmus für eine ungerade Anzahl von Sequenzen:

Algorithmus 2 (3-Stern-Algorithmus für ungerade Anzahl Sequenzen)

Input: Sequenzen $\mathcal{S} = \{S_0, \dots, S_{n-1}\}$, n ungerade

Output: Alignment \mathcal{A}^* mit Kosten D^*

- 1: **for** $c = 0$ to $n - 1$ **do**
- 2: $D_c = 0$
- 3: **for** $i, j \in [0 : n - 1], i + j \equiv c \pmod{n}$ **do**
- 4: $D_c = D_c + D(S_i, S_j, S_c, 1, n - 2, n - 2)$
- 5: **end for**
- 6: **end for**
- 7: $c^* \in [0 : n - 1]$ mit $D_{c^*} \leq D_c \forall c \in [0 : n - 1]$
- 8: Konstruiere G_{c^*} aus \mathcal{G}_n
- 9: $\mathcal{A}^*, D^* = \text{compatible_alignment}(G_{c^*})$

Theorem 5.18

Algorithmus 2 liefert für eine ungerade Anzahl n von Sequenzen ein Alignment mit garantierter Fehlerschranke $2 - \frac{3}{n}$ und hat eine Laufzeit von $O(n^2 l^3)$.

Beweis: Die Fehlerschranke folgt direkt aus Theorem 5.15. Zu zeigen ist, daß Algorithmus 2 eine Laufzeit von $O(n^2 l^3)$ hat.

Die Schleifen von Zeile 1-6 werden $O(n^2)$ mal durchlaufen. Die Kosten eines optimalen gewichteten Alignments von 3 Sequenzen zu berechnen ist in Zeit $O(l^3)$ möglich. Die Zeilen 1-6 benötigen Zeit $O(n^2 l^3)$.

Das Finden des Minimums in Zeile 7 gelingt in $O(n)$. Der Graph (Zeile 7) kann in Zeit $O(nl)$ konstruiert werden.

Um ein mit einem 3-Stern kompatibles Alignment zu konstruieren sind $O(n)$ gewichtete Alignments von 3 Sequenzen zu berechnen. Ein gewichtetes Alignment von 3 Sequenzen zu berechnen ist in Zeit $O(l^3)$ möglich. Alle Alignments können also in Zeit $O(nl^3)$ konstruiert werden.

Nach dem Berechnen der $O(n)$ Alignments müssen für ein Alignment von 3 Sequenzen in $n-3$ Sequenzen Gaps eingefügt werden, um ein kompatibles Alignment zu erzeugen. Es müssen maximal $O(l)$ Gaps eingefügt werden. Das Einfügen der Gaps ist also in Zeit $O(n^2l)$ möglich.

Aus diesen Überlegungen folgt, daß Algorithmus 2 eine Laufzeit von $O(n^2l^3)$ hat. ■

Desweiteren ergibt sich aus Theorem 5.17 ein Algorithmus für eine gerade Anzahl von Sequenzen:

Algorithmus 3 (3-Stern-Algorithmus für gerade Anzahl Sequenzen)

Input: Sequenzen $\mathcal{S} = \{S_0, \dots, S_{n-1}\}$, n gerade

Output: Alignment \mathcal{A}^* mit Kosten D^*

- 1: **for** $c = 0$ to $n - 1$ **do**
- 2: **for** $e \in [0 : n - 1] \setminus \{c\}$ **do**
- 3: $D_{ce} = 0$
- 4: **for** $i, j \in [0 : n - 1], \varphi_e(i) + \varphi_e(j) \equiv \varphi_e(c) \pmod{n}$ **do**
- 5: $D_{ce} = D_{ce} + D(S_i, S_j, S_c, 1, n - 2, n - 2)$
- 6: **end for**
- 7: $D_{ce} = D_{ce} + (n - 1)D(S_c, S_e)$
- 8: **end for**
- 9: **end for**
- 10: $c^*, e^* \in [0 : n - 1]$ mit $c^* \neq e^*$ und $D_{c^*e^*} \leq D_{ce} \forall c, e \in [0 : n - 1], c \neq e$
- 11: Konstruiere $G_{c^*e^*}$ aus \mathcal{G}_n
- 12: $\mathcal{A}^*, D^* = \text{compatible_alignment}(G_{c^*e^*})$

Theorem 5.19

Algorithmus 3 liefert für eine gerade Anzahl n von Sequenzen ein Alignment mit garantierter Fehlerschranke $2 - \frac{3}{n} + \frac{1}{n(n-1)}$ und hat eine Laufzeit von $O(n^3l^3)$.

Beweis: Die Fehlerschranke folgt direkt aus Theorem 5.17. Zu zeigen ist, daß Algorithmus 3 eine Laufzeit von $O(n^3l^3)$ hat.

Die Laufzeitabschätzung ist analog zum Beweis von Theorem 5.19. Lediglich werden, statt $O(n)$ 3-Sterne wie in Algorithmus 2, $O(n^2)$ 3-Sterne mit 2-Clique konstruiert. Damit wird für die Zeilen 1-9 Zeit $O(n^3l^3)$ benötigt.

Die Konstruktion des kompatiblen Alignments ist in der gleichen Zeit möglich wie bei Algorithmus 2. Die zusätzliche 2-Clique fällt mit einem Aufwand von $O(l^2)$ nicht ins Gewicht.

Es ergibt sich also eine Laufzeit von $O(n^3l^3)$. ■

5.4 Verallgemeinerung der Fehlerabschätzung

Die in den vorherigen Abschnitten vorgestellten Algorithmen setzen für die Fehlerabschätzung die Dreiecksungleichung voraus. Gilt diese nicht, so wird keine Aussage über den maximal möglichen Fehler gemacht. Ich werde hier die Fehlerabschätzung auf den Fall verallgemeinern, daß die Dreiecksungleichung um einen konstanten Faktor s verletzt ist.

Eine Kostenfunktion d erfüllt die s -Dreiecksungleichung, wenn $d(x, z) \leq s \cdot (d(x, y) + d(y, z))$ für alle $x, y, z \in \Sigma_\Delta$.

Keine Kostenfunktion d kann die s -Dreiecksungleichung für $s < 1$ erfüllen, da in diesem Fall $d(x, y) \leq s \cdot (d(x, x) + d(x, y)) = s \cdot d(x, y) < d(x, y)$ gelten würde. Für $s = 1$ ergibt sich die "normale" Dreiecksungleichung.

Theorem 5.20 (Verallgemeinerung von Theorem 5.10)

Sei $G = (V, E)$ eine Konfiguration mit $|V| = |\mathcal{S}|$ und d eine Kostenfunktion, die die s -Dreiecksungleichung erfüllt. Dann gilt:

$$\min_{G' \in \mathcal{G}^G} \frac{D(\mathcal{A}^{G'})}{D(\mathcal{S})} \leq s \cdot b(G)$$

Beweis: Der Beweis ist ähnlich zu dem Beweis von Theorem 5.10 in Pevzner [18].

Sei $G = (V, E)$ eine Konfiguration. Für $v, w \in V$ ist $\gamma_G(v, w)$ die Menge der Kanten eines kürzesten Weges in G von v nach w . Beachte, daß in einer Konfiguration der kürzeste Weg zwischen zwei Knoten eindeutig bestimmt ist. Für $e \in E$ sei $\Gamma_G(e) := |\{\gamma_G(v, w) | e \in \gamma_G(v, w)\}|$ die Anzahl der kürzesten Wege, die die Kante e enthalten. Es gilt damit für die Verbindungskosten:

$$c(G) = 2 \cdot \sum_{e \in E} \Gamma_G(e) \tag{5.1}$$

Ist \mathcal{A} ein Alignment von \mathcal{S} und $G = (\mathcal{S}, E)$ eine Konfiguration, dann ist

$$F_G(\mathcal{A}) := \sum_{\{S_i, S_j\} \in E} \Gamma_G(\{S_i, S_j\}) \cdot D(\{S_i, S_j\} | \mathcal{A}).$$

Behauptung: Für ein beliebiges Alignment \mathcal{A} von \mathcal{S} , eine Konfiguration $G = (\mathcal{S}, E)$ und eine Kostenfunktion d , die die s -Dreiecksungleichung erfüllt, gilt

$$D(\mathcal{A}) \leq s \cdot F_G(\mathcal{A}).$$

Beweis: Aufgrund der Definitionen und durch Abschätzen mit Hilfe der s -Dreiecksungleichung ergibt sich:

$$\begin{aligned}
D(\mathcal{A}) &= \sum_{1 \leq i < j \leq n} D(\{S_i, S_j\} | \mathcal{A}) \\
&\leq \sum_{1 \leq i < j \leq n} \left(\sum_{\{S_r, S_t\} \in \gamma_G(S_i, S_j)} s \cdot D(\{S_r, S_t\} | \mathcal{A}) \right) \\
&= s \cdot \sum_{\{S_r, S_t\} \in E} \left(\sum_{\substack{1 \leq i < j \leq n \\ \{S_r, S_t\} \in \gamma_G(S_i, S_j)}} D(\{S_r, S_t\} | \mathcal{A}) \right) \\
&= s \cdot \sum_{\{S_r, S_t\} \in E} \Gamma_G(\{S_r, S_t\}) \cdot D(\{S_r, S_t\} | \mathcal{A}) = s \cdot F_G(\mathcal{A})
\end{aligned}$$

Für eine Konfiguration $G = (V, E)$ mit $|V| = n$ ist Π_G die Menge aller bijektiven Abbildungen $\pi : V \rightarrow \mathcal{S}$. Sei $G_\pi = (\mathcal{S}, E_\pi)$ der Graph mit $E_\pi = \{\{\pi(v), \pi(w)\} | \{v, w\} \in E\}$. Beachte, daß damit $\mathcal{G}^G = \{G_\pi | \pi \in \Pi_G\}$.

Unabhängig von der gewählten Kostenfunktion zeigt Pevzner in [18]: Sei \mathcal{A} ein beliebiges Alignment und $G = (V, E)$ eine Konfiguration. Dann gilt für eine beliebige Kante $\{v, w\} \in E$

$$\sum_{\pi \in \Pi_G} D(\{\pi(v), \pi(w)\} | \mathcal{A}) = \frac{2 \cdot |\mathcal{G}^G|}{n(n-1)} D(\mathcal{A}). \quad (5.2)$$

Sei für $G' \in \mathcal{G}^G$ $\mathcal{A}(G')$ ein Alignment mit $F_{G'}(\mathcal{A}(G')) \leq F_{G'}(\mathcal{A})$ für alle Alignments \mathcal{A} von \mathcal{S} . Desweiteren ist $G^* \in \mathcal{G}^G$ der Graph mit $F_{G^*}(\mathcal{A}(G^*)) \leq F_{G'}(\mathcal{A}(G'))$ für alle $G' \in \mathcal{G}^G$.

Sei M die Anzahl der Graphen in \mathcal{G}^G , die eine beliebige aber feste Kante $e \in E$ enthalten. Beachte, daß aufgrund der Symmetrie

$$\frac{|\mathcal{G}^G|}{M} = \frac{n(n-1)}{2|E|} \quad (5.3)$$

gilt. Definiere $W = \frac{1}{M} \sum_{G' \in \mathcal{G}^G} F_{G'}(\mathcal{A}(G'))$. Dann gilt

$$W \geq \frac{1}{M} \sum_{G' \in \mathcal{G}^G} F_{G'}(\mathcal{A}(G')) \geq \frac{|\mathcal{G}^G|}{M} F_{G^*}(\mathcal{A}(G^*)).$$

Mit Gleichung (5.3) und der Behauptung ergibt sich

$$W \geq \frac{n(n-1)}{2 \cdot s \cdot |E|} D(\mathcal{A}(G^*)). \quad (5.4)$$

Andererseits gilt

$$\begin{aligned} W &= \frac{1}{M} \sum_{G'=(\mathcal{S}, E') \in \mathcal{G}^G} \left(\sum_{\{S_i, S_j\} \in E'} \Gamma_{G'}(\{S_i, S_j\}) \cdot D(\{S_i, S_j\} | \mathcal{A}(\mathcal{S})) \right) \\ &= \frac{1}{M} \sum_{\pi \in \Pi_G} \left(\sum_{\{v, w\} \in E} \Gamma_{G_\pi}(\{\pi(v), \pi(w)\}) \cdot D(\{\pi(v), \pi(w)\} | \mathcal{A}(\mathcal{S})) \right). \end{aligned}$$

Es gilt $\Gamma_G(\{v, w\}) = \Gamma_{G_\pi}(\{\pi(v), \pi(w)\})$ für $\{v, w\} \in E$. Mit dieser Eigenschaft, Gleichung (5.2) und (5.1) gilt:

$$\begin{aligned} W &= \frac{1}{M} \sum_{\{v, w\} \in E} \Gamma_G(\{v, w\}) \left(\sum_{\pi \in \Pi_G} D(\{\pi(v), \pi(w)\} | \mathcal{A}(\mathcal{S})) \right) \\ &= \frac{1}{M} \frac{2|\mathcal{G}^G|}{n(n-1)} D(\mathcal{A}(\mathcal{S})) \cdot \sum_{\{v, w\} \in E} \Gamma_G(\{v, w\}) \\ &= \frac{1}{M} \frac{2|\mathcal{G}^G|}{n(n-1)} D(\mathcal{A}(\mathcal{S})) \frac{c(G)}{2} \end{aligned}$$

Mit Gleichung (5.3):

$$W = \frac{1}{|E|} D(\mathcal{A}(\mathcal{S})) \frac{c(G)}{2}$$

Aufgrund der Ungleichung (5.4) gilt:

$$\frac{D(\mathcal{A}(G^*))}{D(\mathcal{A}(\mathcal{S}))} \leq s \frac{c(G)}{n(n-1)} = s \cdot b(G)$$

■

Lemma 5.21 (Verallgemeinerung von Lemma 5.11)

Für jedes Alignment \mathcal{A} von \mathcal{S} , k -Stern oder k -Stern mit q -Clique $G = (\mathcal{S}, E)$ und Kostenfunktion d , die die s -Dreiecksungleichung erfüllt, gilt:

$$D(\mathcal{A}) \leq s \cdot D(\mathcal{A}, C_G)$$

Beweis: Der Beweis ist ähnlich dem Beweis von Lemma 5.11 in Bafna et al. [4].

Sei S_c das Zentrum von G und $g(i)$ die Größe der Clique, in der sich S_i , $i \neq c$, befindet. Da d die s -Dreiecksungleichung erfüllt, gilt für beliebige i, j, p $d(a_{ip}, a_{jp}) \leq$

$s \cdot (d(a_{ip}, a_{cp}) + d(a_{cp}, a_{jp}))$. Damit ergibt sich für eine beliebige Spalte $p \in [1 : L]$ die Abschätzung:

$$\begin{aligned}
& \sum_{1 \leq i < j \leq n} d(a_{ip}, a_{jp}) \\
\leq & \underbrace{\sum_{\substack{1 \leq i < j \leq n \\ \{S_i, S_j\} \notin E}} s \cdot (d(a_{ip}, a_{cp}) + d(a_{cp}, a_{jp}))}_{S_i, S_j \text{ in verschiedenen Cliques}} + \underbrace{\sum_{\{S_i, S_j\} \in E} s \cdot d(a_{ip}, a_{jp})}_{S_i, S_j \text{ in der gleichen Clique}} \\
= & \sum_{1 \leq i \leq n, i \neq c} (n - g(i) + 1) \cdot s \cdot d(a_{ip}, a_{cp}) + \sum_{\substack{\{S_i, S_j\} \in E \\ i, j \neq c}} s \cdot d(a_{ip}, a_{jp}) \\
= & \sum_{1 \leq i < j \leq n} (C_G)_{ij} \cdot s \cdot d(a_{ip}, a_{jp})
\end{aligned}$$

Daraus folgt:

$$D(\mathcal{A}) = \sum_{p=1}^L \sum_{1 \leq i < j \leq n} d(a_{ip}, a_{jp}) \leq \sum_{1 \leq i < j \leq n} (C_G)_{ij} \cdot s \cdot d(a_{ip}, a_{jp}) = s \cdot D(\mathcal{A}, C_G)$$

■

Lemma 5.13 gilt unabhängig von der gewählten Kostenfunktion (siehe Beweis in [4]). Damit ergibt sich:

Theorem 5.22

Ist \mathcal{G} eine ausgewogene Menge von k -Sternen oder k -Sternen mit q -Clique mit Parameter p und d eine Kostenfunktion, die die s -Dreiecksungleichung erfüllt. Dann gilt:

$$\min_{G \in \mathcal{G}} D(\mathcal{A}^G) \leq s \frac{p}{|\mathcal{G}|} D(\mathcal{S})$$

Beweis: Das Theorem folgt direkt aus Lemma 5.21 und 5.13. ■

Literatur

- [1] S. F. Altschul and D. J. Lipman. Trees, stars and multiple biological sequence alignment. *SIAM Journal of Applied Mathematics*, 49:197–209, 1989.
- [2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, May 1998.
- [3] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation. Combinatorial Optimization Problems and their Approximability*. Springer-Verlag, 1999.
- [4] V. Bafna, E. L. Lawler, and P. A. Pevzner. Approximation algorithms for multiple sequence alignment. *Theoretical Computer Science*, 182(1–2):233–244, 1997.
- [5] H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM Journal of Applied Mathematics*, 48:1073–1082, 1988.
- [6] D. Feng and R. Doolittle. Progressive alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25:351–360, 1987.
- [7] H. H. Gabow. An efficient implementation of edmonds’ algorithm for maximum matching on graphs. *Journal of the ACM*, 23(2):221–234, 1976.
- [8] Z.Z. Galil. Sequential and parallel algorithms for finding maximum matchings in graphs. *Ann. Rev. Comput. Sci.*, 1:197–224, 1986.
- [9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to NP-Completeness*. W. H. Freeman and Company, 1979.
- [10] O. Gotoh. Optimal alignment between groups of sequences and its application to multiple sequence alignment. *Computational Applications in the Biosciences*, 9(3):361–370, 1993.
- [11] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bulletin of Mathematical Biology*, 55(1):141–154, 1993.
- [12] D. Gusfield. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, 1995.
- [13] Johan Hastad. Some optimal inapproximability results. In *Proc. 29th Ann. ACM Symp. on Theory of Comp.*, pages 1–10. ACM, 1997.
- [14] D. S. Hochbaum. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, 1997.
- [15] D. Maier. The complexity of some problems on subsequences and supersequences. *Journal of the ACM*, 25(2):322–336, 1978.

- [16] T. Ottmann and P. Widmayer. *Algorithmen und Datenstrukturen*. Spektrum Verlag, 1996.
- [17] C. H. Papadimitriou and M. Yannakaki. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.
- [18] P. A. Pevzner. Multiple alignment, communication cost and graph matching. *SIAM Journal of Applied Mathematics*, 52(6):1763–1779, 1992.
- [19] M. Vingron and A. v. Haeseler. Towards integration of multiple alignment and phylogenetic tree construction. *Journal of Computational Biology*, 4(1):23–34, 1997.
- [20] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348, 1994.
- [21] H. T. Wareham. A simplified proof of the NP- and MAX-SNP-hardness of multiple sequence tree alignment. *Journal of Computational Biology*, 2(4):509–514, 1995.
- [22] M. S. Waterman. *Introduction to Computational Biology*. Chapman & Hall, 1995.