

Eine untere Schranke für die Approximierbarkeit des gewichteten Multiple-Sequence-Alignment- Problems

Diplomarbeit

im Rahmen des Diplomstudiengangs Informatik

vorgelegt von

Bodo Siebert

Betreuer:

Prof. Dr. math. Rüdiger Reischuk



Medizinische Universität zu Lübeck
Technisch-Naturwissenschaftliche Fakultät
Institut für Theoretische Informatik

September 2000

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Lübeck,

Inhaltsverzeichnis

1	Einleitung	2
1.1	Problemstellung	2
1.2	Stand der Forschung	3
1.3	Eigene Forschungsergebnisse	4
1.4	Aufbau der Arbeit	5
2	Definitionen und Notationen	6
2.1	Notationen	6
2.2	Alignment und SP-Kosten	6
3	Berechnung optimaler Alignments	10
3.1	Dynamische Programmierung	10
3.2	Beschleunigung des Verfahrens	12
3.3	Bemerkungen	13
4	Approximationsalgorithmen	15
4.1	Center-Star-Algorithmus	15
4.2	Verallgemeinerung des Center-Star-Algorithmus	16
4.3	Ausgewogene Mengen	17
5	Komplexität von Optimierungsproblemen	20
5.1	\mathcal{NP} -Optimierungsprobleme	20
5.2	Approximation von Optimierungsproblemen	21
5.3	MAX- \mathcal{SNP}	24
6	Eine untere Approximierbarkeitsschranke für MIN-BMSA	27
6.1	Eine Approximierbarkeitsschranke für MAX-E2-neg-Lin2	28
6.2	MAX-E2-neg-Lin2 ist MAX- \mathcal{SNP} -vollständig	31
6.3	MIN-BMSA ist MAX- \mathcal{SNP} -hart	32
7	Reduktion von MIN-WMSA auf MIN-BMSA	39
8	Zusammenfassung	42
	Literatur	43

1 Einleitung

1.1 Problemstellung

In der Biologie und speziell in der Molekularbiologie und Genetik ist es oft notwendig, verschiedene DNA- oder Aminosäuresequenzen miteinander zu vergleichen und Ähnlichkeiten zwischen ihnen zu finden. Dabei stehen zwei Anwendungen im Vordergrund: zum einen das Erkennen von Zusammenhängen zwischen Aminosäuresequenzen und der Funktion und Struktur der Proteine, die durch die entsprechenden Sequenzen kodiert werden; zum anderen die Rekonstruktion der Evolution anhand phylogenetischer Bäume. Hierbei werden anhand der DNA- oder Aminosäuresequenzen Verwandtschaften und Abstammungen ermittelt.

Dazu wird angenommen, dass evolutionäre Verwandtschaft von Sequenzen oder eine ähnliche Funktion oder Struktur bei Proteinen eine Ähnlichkeit der betreffenden Sequenzen impliziert. Dies wird damit begründet, dass in der Natur Strukturen, die sich als erfolgreich erwiesen haben, wie z. B. Proteine oder regulatorische DNA-Sequenzen, kopiert, wiederverwendet und modifiziert werden (Gusfield [21]). Umgekehrt wird angenommen, dass ähnliche Sequenzen eine gemeinsame Herkunft aufweisen.

Die Erkennung und Quantifizierung der Ähnlichkeiten von Sequenzen kann automatisiert werden. Hier zeigt sich aber, dass eine globale Suche, also ein Durchprobieren aller Möglichkeiten, aufgrund des großen Lösungsraums nicht praktikabel ist. Die Aufgabe der Informatik ist es daher, effiziente Algorithmen zu entwickeln, mit denen die Ähnlichkeiten von Sequenzen bewertet werden können, oder zu zeigen, dass diese Probleme schwer, also z.B. \mathcal{NP} -vollständig, sind. Dazu wurden verschiedene Lösungsmöglichkeiten entwickelt und diskutiert. Chan, Wong und Chiu geben in [9] einen Überblick über verschiedene Methoden.

Eine der meistverbreiteten Herangehensweisen zum Vergleich mehrerer Sequenzen ist das so genannte Multiple-Sequence-Alignment. Wir wollen zunächst eine Spezialisierung dieses Problems für zwei Sequenzen betrachten, nämlich die Bestimmung des Editierabstandes (siehe Gusfield [21]). Der Editierabstand zweier Sequenzen ist ein Maß für die Ähnlichkeit dieser Sequenzen. Die Annahme ist, dass die eine Sequenz durch Einfügen (Insertion), Löschen (Deletion) und Ändern (Substitution) einzelner Zeichen aus der anderen Sequenz hervorgeht. Diesen Operationen können Kosten in Abhängigkeit vom gelöschten, eingefügten oder veränderten Zeichen zugewiesen werden. Der Editierabstand der beiden Sequenzen entspricht den minimal notwendigen Kosten zur Transformation der einen Sequenz in die andere.

Wird eine Deletion mit D, eine Insertion mit I und eine Substitution mit S markiert, dann kann “alignment” wie folgt in “algorithm” transformiert werden:

$$\begin{array}{ccccccc} & D & S & S & S & D & I & I \\ a & l & i & g & n & m & e & n & t \\ a & l & g & o & r & i & t & h & m \end{array}$$

Der Editierabstand zweier Sequenzen führt auf das so genannte paarweise Alignment. Dazu werden in beide Sequenzen “Leerzeichen”, so genannte Gaps, eingefügt. Einer Deletion in einer Sequenz entspricht dann ein Gap an der entsprechenden Stelle. Eine Insertion führt zu einem Gap in der anderen Sequenz. Die Transformation von “alignment” in “algorithm” würde als paarweises Alignment wie folgt aussehen, wobei “-” das Gap sei:

$$\begin{array}{ccccccc} a & l & i & g & n & m & e & n & t & - & - \\ a & l & - & g & o & r & i & - & t & h & m \end{array}$$

Dieses Verfahren kann nun einfach auf mehrere Sequenzen verallgemeinert werden. Es muss allerdings definiert werden, was unter einem optimalen Alignment mehrerer Sequenzen verstanden werden soll. Hier sind die so genannten SP-Kosten (sum-of-pair score, siehe Carrillo und Lipman [8]) weit verbreitet, da sie eine einfache Verallgemeinerung der Kostenfunktionen für paarweise Alignments darstellen. SP-Kosten werden auch in dieser Arbeit verwendet.

Eine andere Möglichkeit, Alignments zu bewerten, ist, das Alignment zu suchen, welches am “wahrscheinlichsten” ist. Hierbei werden z.B. Hidden-Markov-Modelle (siehe Rabiner [35]) benutzt. Eine explizite Kostenfunktion ist meist nicht gegeben, folgt aber implizit aus den zugrundeliegenden Übergangswahrscheinlichkeiten. Diese Möglichkeit wird von Baldi und Brunak in [7] ausführlich behandelt.

Einen Überblick über die Problemstellung, mögliche Herangehensweisen und Lösungen wird von Baldi und Brunak in [7], Gusfield in [21] und Waterman in [43] gegeben.

1.2 Stand der Forschung

Ein optimales Alignment bezüglich SP-Kosten kann mit dynamischer Programmierung berechnet werden, es ergibt sich jedoch eine exponentielle Laufzeit (siehe z.B. Waterman [43]). Carrillo und Lipman beschleunigen in [8] diese Methode. Gupta, Kececioğlu und Schäffer verbessern sie in [19] bezüglich des Platzverbrauchs. Die Laufzeit ist aber trotz der Verbesserungen exponentiell.

Wang und Jiang zeigen in [41], dass die Berechnung eines Alignments mit minimalen SP-Kosten \mathcal{NP} -vollständig ist. Es ist also in der Praxis bislang nicht möglich, ein optimales Alignment zu berechnen. Man begnügt sich daher mit einem Alignment, dessen Kosten nicht allzu weit von den Kosten eines optimalen Alignments abweichen. Gusfield stellt dazu in [20] den Center-Star-Algorithmus vor, der ein Alignment berechnet, dessen Kosten höchstens zweimal so hoch sind wie die Kosten eines optimalen Alignments. Pevzner [34] sowie Bafna und andere [6] verbessern diese Methode (siehe Kapitel 4).

Eine andere Möglichkeit, ein Alignment zu berechnen, ist seine Konstruktion mit Hilfe eines phylogenetischen Baumes. Eine Herangehensweise ist es, einen Baum zu konstruieren, in den neben den gegebenen Sequenzen weitere Sequenzen eingefügt werden, die so gewählt werden, dass die Summe der Kosten entlang der Kanten minimal ist. Hierbei handelt es sich um ein Steiner-Baum-Problem (siehe Jungnickel [26]). In Wang und Jiang [41] und Wareham [42] wird gezeigt, dass dieses Problem $\text{MAX-}\mathcal{SNP}$ -hart ist.

Das Problem, Sequenzen den Knoten eines gegebenen Baums optimal zuzuordnen, ist ebenfalls \mathcal{NP} -vollständig, besitzt aber ein Polynomialzeit-Approximations-Schema (siehe Jiang et al. [25], Wang und Gusfield [40]).

Vingron und von Häseler stellen in [39] einen Algorithmus vor, der iterativ einen Baum konstruiert, indem Sequenzen eingefügt werden. Dabei wird sukzessiv ein Alignment berechnet. Dazu verwenden sie Algorithmen, mit denen ein Alignment von mehreren Gruppen von Sequenzen berechnet werden kann (siehe Gotoh [18]). Das berechnete Alignment ist allerdings im Allgemeinen nicht optimal und eine Fehlerschranke ist nicht bekannt.

Einen Überblick über verschiedene Methoden zur Berechnung eines Multiple-Sequence-Alignments geben Altschul und Lipman in [2].

Das Multiple-Sequence-Alignment-Problem kann durch die Einführung von Gewichten verallgemeinert werden (siehe Wu et al. [44]). Dazu wird jedem Paar von Sequenzen ein Gewicht zugeordnet. Die gewichteten Kosten eines Alignments sind dann die entsprechend gewichteten Kosten der Paare. Damit läßt sich bereits bekanntes Wissen über die Verwandtschaft der Sequenzen modellieren, indem nahe verwandte Paare von Sequenzen ein hohes Gewicht und weniger verwandte Paare ein kleineres Gewicht bekommen.

1.3 Eigene Forschungsergebnisse

Just zeigt in [27], dass das Multiple-Sequence-Alignment-Problem $\text{MAX-}\mathcal{SNP}$ -hart ist für den Fall, dass die Kostenfunktion nicht die Bedingungen einer Metrik erfüllt. Natürlicherweise ist aber zumindest die Dreiecksunglei-

chung erfüllt. Für metrische Kostenfunktionen ist ein solches Ergebnis bisher nicht bekannt. Laut Jiang et al. [24] ist die Approximierbarkeit des Multiple-Sequence-Alignment-Problems eines der großen offenen Probleme der molekularen Bioinformatik.

In dieser Arbeit werden wir das Problem der Approximierbarkeit des Multiple-Sequence-Alignment-Problems nicht lösen. Wir werden aber beweisen, dass die Verallgemeinerung des Multiple-Sequence-Alignment-Problems durch Gewichte bei metrischen Kostenfunktionen MAX- \mathcal{SNP} -hart ist. Dieses Resultat gilt bereits für den eingeschränkten Fall des binär gewichteten Multiple-Sequence-Alignment-Problems, bei dem die Gewichte auf 0 und 1 beschränkt sind. Des Weiteren zeigen wir eine explizite untere Schranke für die Approximierbarkeit dieses Problems.

1.4 Aufbau der Arbeit

Der Aufbau dieser Arbeit ist wie folgt. Zunächst werden in Kapitel 2 die notwendigen Begriffe eingeführt. In den Kapiteln 3 bis 5 werden bekannte Ergebnisse vorgestellt. Kapitel 3 behandelt Methoden zur Berechnung optimaler Alignments. Thema von Kapitel 4 sind schnelle Approximationsalgorithmen, bei denen das berechnete Alignment zwar im Allgemeinen nicht optimal ist, aber höchstens um einen konstanten Faktor größere Kosten hat als ein optimales Alignment. Kapitel 5 enthält eine Einführung in die Theorie der Komplexität von Optimierungsproblemen.

Schließlich werden in den Kapiteln 6 und 7 die eigenen Ergebnisse vorgestellt. In Kapitel 6 wird eine untere Schranke für die Approximierbarkeit des binär gewichteten Multiple-Sequence-Alignment-Problems bewiesen und es wird gezeigt, dass dieses Problem MAX- \mathcal{SNP} -hart ist. In Kapitel 7 werden wir das gewichtete Multiple-Sequence-Alignment-Problem auf das binär gewichtete Multiple-Sequence-Alignment-Problem reduzieren und damit zeigen, dass jede untere Schranke für die Approximierbarkeit des gewichteten Multiple-Sequence-Alignment-Problems auch eine untere Schranke für den eingeschränkten Fall darstellt. Die beiden Probleme sind also äquivalent bezüglich der Approximierbarkeit.

Zum Schluss werden in Kapitel 8 die vorgestellten Resultate zusammengefasst.

2 Definitionen und Notationen

2.1 Notationen

Sei Σ ein endliches Alphabet, dann ist Σ^* die Menge aller Sequenzen über Σ . Für $L \in \mathbb{N}$ ist Σ^L die Menge aller Sequenzen der Länge L . $|S|$ ist die Länge einer Sequenz S . $S[k]$ bezeichnet das k -te Zeichen einer Sequenz S .

Für die verwendeten Begriffe aus der Komplexitätstheorie wird auf Reichuk [36], für die verwendeten Begriffe aus der Graphentheorie auf Jungnickel [26] verwiesen.

2.2 Alignment und SP-Kosten

In den folgenden Definitionen ist $\mathcal{S} = \{S_1, \dots, S_n\}$ eine Multimenge von Sequenzen über Σ , d.h. es kann $S_j = S_k$ für $j \neq k$ gelten.

Um das Einfügen und Löschen von Zeichen in Sequenzen zu modellieren wird ein zusätzliches Zeichen “-” $\notin \Sigma$ benötigt, das so genannte Gap. Sei $\Sigma' := \Sigma \cup \{-}$.

Definition 2.1 (Alignment) Eine Multimenge $\mathcal{A} = \{A_1, \dots, A_n\}$ heißt Alignment von \mathcal{S} , falls es eine Konstante L gibt, so dass für alle $j = 1, \dots, n$ gilt:

- (i) $A_j \in \Sigma'^L$.
- (ii) Es gibt Positionen $1 \leq a_{j,1} < a_{j,2} < \dots < a_{j,|S_j|} \leq L$, so dass $S_j = A_j[a_{j,1}] \dots A_j[a_{j,|S_j|}]$ ist. Für alle $b \notin \{a_{j,i} | i = 1, \dots, |S_j|\}$ ist $A_j[b] = \text{“-”}$.

Anschaulich geht A_j aus S_j durch Einfügen von $L - |S_j|$ Gaps hervor. Aus der Definition ist ersichtlich, dass $L \geq \max_{1 \leq j \leq n} |S_j|$ ist. Ein Alignment $\mathcal{A} = \{A_1, \dots, A_n\}$ mit $|A_j| = L$, $j = 1, \dots, n$, kann als Matrix der Größe $n \times L$ mit Einträgen $A_j[i]$, $1 \leq j \leq n$ und $1 \leq i \leq L$, aufgefasst werden. Eine Spalte i_0 dieser Matrix, in der $A_j[i_0] = \text{“-”}$ für alle $j = 1, \dots, n$ gilt, heißt Gap-Spalte.

Stehen $S_j[i]$ und $S_{j'}[i']$ in der gleichen Spalte der Matrix, d.h. ist $a_{j,i} = a_{j',i'}$, dann sagen wir, dass $S_j[i]$ mit $S_{j'}[i']$ korrespondiert. Im Fall $A_j[a_{j,i}] = \text{“-”}$ sagen wir, dass $S_j[i]$ mit einem Gap in $S_{j'}$ korrespondiert.

Zur Bewertung der Alignments werden hier die so genannten SP-Kosten verwendet (Carrillo, Lipman [8]). Dazu werden zunächst Kostenfunktionen definiert.

Definition 2.2 (Kostenfunktion) Eine Funktion $d : \Sigma'^2 \rightarrow \mathbb{N}$ heißt Kostenfunktion, falls sie eine Metrik ist, d.h. falls für alle $x, y, z \in \Sigma'$ gilt:

- (i) $d(x, y) = 0 \Leftrightarrow x = y$
- (ii) $d(x, y) = d(y, x)$ (Symmetrie)
- (iii) $d(x, z) \leq d(x, y) + d(y, z)$ (Dreiecksungleichung)

Damit sind die SP-Kosten wie folgt definiert.

Definition 2.3 (SP-Kosten) Seien A und A' zwei Sequenzen mit $|A| = |A'| = L$. Dann sind die Kosten von A und A' bezüglich einer Kostenfunktion d gegeben durch

$$D(A, A') := \sum_{i=1}^L d(A[i], A'[i]).$$

Sei nun $\mathcal{A} = \{A_1, \dots, A_n\}$ ein Alignment, dann sind die SP-Kosten (sum-of-pair score) von \mathcal{A} gegeben durch

$$D(\mathcal{A}) := \sum_{1 \leq j < k \leq n} D(A_j, A_k).$$

\mathcal{A} heißt optimales Alignment von \mathcal{S} , falls \mathcal{A} ein Alignment mit minimalen SP-Kosten unter allen Alignments von \mathcal{S} ist. Ein optimales Alignment ist im Allgemeinen nicht eindeutig bestimmt. Mit $D(\mathcal{S})$ werden die Kosten eines optimalen Alignments von \mathcal{S} bezeichnet.

Ist \mathcal{A} ein Alignment und \mathcal{A}' das Alignment, das aus \mathcal{A} durch Weglassen aller Gap-Spalten entsteht, dann gilt $D(\mathcal{A}) = D(\mathcal{A}')$ wegen $d(-, -) = 0$.

Beispiel 2.4 Seien $S_1 = \text{ACTG}$, $S_2 = \text{ATCG}$ und $S_3 = \text{ATCGA}$. Weiterhin sei

$$d(x, y) = \begin{cases} 1 & \text{falls } x \neq y, \\ 0 & \text{falls } x = y. \end{cases}$$

Dann sind $\mathcal{A} = \{A_1, A_2, A_3\}$, $\mathcal{A}' = \{A'_1, A'_2, A'_3\}$ und $\mathcal{A}'' = \{A''_1, A''_2, A''_3\}$ optimale Alignments von $\mathcal{S} = \{S_1, S_2, S_3\}$, jeweils mit Kosten 6, wobei \mathcal{A} , \mathcal{A}' und \mathcal{A}'' wie folgt gegeben sind:

$$\begin{array}{lll} A_1 = \text{AC-TG--} & A'_1 = \text{ACTG-} & A''_1 = \text{ACT-G-} \\ A_2 = \text{AT-CG--} & A'_2 = \text{ATCG-} & A''_2 = \text{A-TCG-} \\ A_3 = \text{AT-CG-A} & A'_3 = \text{ATCGA} & A''_3 = \text{A-TCGA} \end{array}$$

Es ist zu sehen, dass \mathcal{A}' aus \mathcal{A} durch Eliminierung der Gap-Spalten hervorgeht.

Aus den genannten Definitionen ergibt sich das folgende Entscheidungsproblem:

$$\text{MSA} := \{(\mathcal{S}, k) \mid D(\mathcal{S}) \leq k\}$$

Wang und Jiang zeigen in [41], dass MSA \mathcal{NP} -vollständig ist. Dabei wird allerdings Bedingung (i) aus Definition 2.2 durch die abgeschwächte Bedingung $d(-, -) = 0$ ersetzt.

MIN-MSA ist das Optimierungsproblem, zu Sequenzen \mathcal{S} ein Alignment mit minimalen Kosten zu berechnen. In Kapitel 4 werden Approximationsalgorithmen für MIN-MSA vorgestellt.

Es wird nun noch der Begriff des induzierten Alignments (siehe Pevzner [34]) definiert, der in Kapitel 4 und Kapitel 6 benötigt wird.

Definition 2.5 (Induziertes Alignment) Sei $\mathcal{A} = \{A_1, \dots, A_n\}$ ein beliebiges Alignment von \mathcal{S} . Weiterhin sei $\Omega \subseteq \mathcal{S}$. Dann ist $\{A_j \mid S_j \in \Omega\}$ das durch \mathcal{A} induzierte Alignment von Ω . Die Kosten dieses Alignments sind gegeben durch

$$D(\Omega \mid \mathcal{A}) := \sum_{\substack{1 \leq j < k \leq n \\ S_j, S_k \in \Omega}} D(A_j, A_k).$$

Offensichtlich ist $D(\Omega \mid \mathcal{A}) \geq D(\Omega)$. Auch in dem Fall, dass \mathcal{A} ein optimales Alignment von \mathcal{S} ist, gilt im Allgemeinen nicht $D(\Omega \mid \mathcal{A}) = D(\Omega)$.

Eine Verallgemeinerung von MSA wird von Wu und anderen in [44] vorgestellt. Sie ordnen jedem Paar von Sequenzen Gewichte zu, mit denen die jeweiligen Kosten multipliziert werden. Mit diesen Gewichten lassen sich bekannte Informationen über die Verwandtschaften modellieren: Nahe verwandte Paare von Sequenzen werden ein hohes Gewicht erhalten, während Paare, die vermutlich kaum verwandt sind, ein kleineres Gewicht erhalten werden.

Definition 2.6 (Gewichtete Alignmentkosten) Sei $\mathcal{A} = \{A_1, \dots, A_n\}$ ein Alignment von \mathcal{S} und $W = (w_{S_j, S_k})_{S_j, S_k \in \mathcal{S}}$ eine symmetrische Gewichtsmatrix, wobei die Gewichte $w_{S_j, S_k} \in \mathbb{N}$ polynomiell in n beschränkt sind. Dann sind die gewichteten Alignmentkosten von \mathcal{A} definiert durch

$$D_W(\mathcal{A}) := \sum_{1 \leq j < k \leq n} w_{S_j, S_k} \cdot D(A_j, A_k).$$

\mathcal{A} heißt optimales gewichtetes Alignment von \mathcal{S} , falls \mathcal{A} ein Alignment mit minimalen gewichteten Kosten unter allen Alignments von \mathcal{S} ist. Auch ein optimales gewichtetes Alignment ist im Allgemeinen nicht eindeutig bestimmt. Mit $D_W(\mathcal{S})$ werden die Kosten eines optimalen gewichteten Alignments bezeichnet.

Die gewichteten Kosten eines durch ein Alignment \mathcal{A} induzierten Alignments von Ω ergeben sich analog zum ungewichteten Fall:

$$D_W(\Omega|\mathcal{A}) := \sum_{\substack{1 \leq j < k \leq n \\ S_j, S_k \in \Omega}} w_{S_j, S_k} \cdot D(A_j, A_k)$$

Fortsetzung von Beispiel 2.4 *Seien zusätzlich die Gewichte $w_{S_1, S_2} = 1$, $w_{S_1, S_3} = 2$ und $w_{S_2, S_3} = 0$ gegeben. Dann sind die gewichteten Kosten der genannten Alignments jeweils 8.*

WMSA (Weighted Multiple Sequence Alignment) und MIN-WMSA sind analog zu MSA und MIN-MSA definiert.

Eine eingeschränkte Version des gewichteten Multiple Sequence Alignment ist binär gewichtetes Multiple Sequence Alignment. Bei diesem Problem sind die Gewichte auf 0 und 1 beschränkt. Das entsprechende Entscheidungsproblem heißt BMSA (Binary Weighted Multiple Sequence Alignment), das Optimierungsproblem MIN-BMSA. In Kapitel 6 wird eine untere Schranke für die Approximierbarkeit dieses Problems bewiesen.

3 Berechnung optimaler Alignments

3.1 Dynamische Programmierung

Ein optimales Alignment kann mittels dynamischer Programmierung (siehe Cormen et al. [11]) berechnet werden.

Dazu definieren wir so genannte Alignment-Graphen (siehe Pevzner [34]).

Definition 3.1 (Alignment-Graph) *Der Alignment-Graph zu Sequenzen $\mathcal{S} = \{S_1, \dots, S_n\}$ ist ein gerichteter azyklischer Graph $G_{\mathcal{S}} = (V_{\mathcal{S}}, E_{\mathcal{S}})$ und ist gegeben durch:*

- (i) $V_{\mathcal{S}} := \{v = (v_1, \dots, v_n) \in \mathbb{N}^n \mid 0 \leq v_j \leq |S_j| \text{ für } 1 \leq j \leq n\}$ und
- (ii) $E_{\mathcal{S}} := \{(u, v) \in V_{\mathcal{S}}^2 \mid v_j = u_j \vee v_j = u_j + 1 \text{ für } 1 \leq j \leq n \text{ und } u \neq v\}$.

Ein Pfad $W = (e_1, \dots, e_L)$ von der Quelle $(0, \dots, 0)$ des Alignment-Graphen zu seiner Senke $(|S_1|, \dots, |S_n|)$ kann als Alignment interpretiert werden. Dazu wird eine Kante $e = (u, v)$ des Alignment-Graphen als Vektor $f(e) = (f_1(e), \dots, f_n(e)) \in \Sigma^n$ aufgefaßt:

$$f_j(e) := \begin{cases} S_j[v_j] & \text{falls } u_j < v_j \\ - & \text{falls } u_j = v_j \end{cases} \quad (3.1)$$

Damit repräsentiert der Pfad W das Alignment $\mathcal{A}_W = \{A_1, \dots, A_n\}$ mit $A_j = f_j(e_1) \dots f_j(e_L)$.

Beispiel 3.2 *Sei $\mathcal{S} = \{S_1, S_2\}$ mit $S_1 = \text{AT}$ und $S_2 = \text{TG}$. Dann repräsentiert der Pfad $((0, 0), (1, 0), ((1, 0), (2, 1)), ((2, 1), (2, 2)))$ das Alignment*

$$\begin{aligned} A_1 &= \text{AT-} \\ A_2 &= \text{-TG.} \end{aligned}$$

Man beachte, dass ein durch einen Pfad repräsentiertes Alignment keine Gap-Spalten besitzt, da der Graph azyklisch ist. Da $d(-, -) = 0$ für die Kostenfunktionen gefordert wurde, stellt dies keine Einschränkung dar.

Nun können die Kanten des Alignment-Graphen mit Kosten s belegt werden:

$$s(e) = \sum_{1 \leq j < k \leq n} d(f_j(e), f_k(e)) \quad (3.2)$$

Die Kosten einer Kante e aus dem Pfad W sind gerade die Kosten der entsprechenden Spalte eines Alignments \mathcal{A}_W . Damit sind die Kosten eines durch einen Pfad $W = (e_1, \dots, e_L)$ repräsentierten Alignments \mathcal{A}_W durch

$$D(\mathcal{A}_W) = \sum_{i=1}^L s(e_i)$$

gegeben. Ein Pfad, der ein optimales Alignment repräsentiert, heißt optimaler Pfad.

Wir stellen jetzt dar, wie $D(\mathcal{S})$ berechnet werden kann. Jeder Knoten $v = (v_1, \dots, v_n)$ im Graphen repräsentiert Präfixe der Sequenzen S_1 bis S_n , wobei die Längen der Präfixe durch v_1 bis v_n gegeben sind. Jedem Knoten ordnen wir die Kosten eines optimalen Alignments dieser Sequenzpräfixe zu; diese Kosten bezeichnen wir mit $D(v)$. Formal: Sei $S_j^{v_j}$ das Präfix der Länge v_j von S_j , dann bezeichnet $D(v)$ die Kosten eines optimalen Alignments von $\{S_1^{v_1}, \dots, S_n^{v_n}\}$. Da der Knoten $(|S_1|, \dots, |S_n|)$ gerade die vollständigen Sequenzen S_1, \dots, S_n repräsentiert, ist $D(|S_1|, \dots, |S_n|) = D(\mathcal{S})$. Es ergibt sich die Rekursionsgleichung

$$D(v) = \min_{(u,v) \in E_{\mathcal{S}}} D(u) + s(u, v).$$

Bei rekursiver Berechnung der $D(v)$ würden viele Werte mehrfach berechnet. Dies kann man mit dynamischer Programmierung vermeiden. Aus dieser Idee resultiert der folgende Algorithmus.

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

Konstruiere $G_{\mathcal{S}} = (V_{\mathcal{S}}, E_{\mathcal{S}})$

$D(0, \dots, 0) = 0$

for $v = (0, \dots, 0, 1)$ **to** $(|S_1|, \dots, |S_n|)$ **do**

$D(v) = \min_{(u,v) \in E_{\mathcal{S}}} D(u) + s(u, v)$

end for

Ausgabe: Kosten eines optimalen Alignments $D(|S_1|, \dots, |S_n|)$

Dieser Algorithmus kann gleichzeitig zur Berechnung eines optimalen Alignments verwendet werden, indem man sich für jeden Eintrag $D(v)$ merkt, für welchen Knoten u gerade $D(u) + s(u, v) = D(v)$ gilt. Aus dem so konstruierten Pfad kann gemäß (3.1) ein Alignment berechnet werden. Eine ausführliche Darstellung dieses Verfahrens ist in Waterman [43] zu finden.

Bei diesem Vorgehen ergibt sich eine Laufzeit von $O\left(2^n \cdot \prod_{j=1}^n |S_j|\right)$, wobei $\prod_{j=1}^n |S_j|$ die Anzahl der Schleifendurchläufe und 2^n die Anzahl der in

jedem Schleifendurchlauf betrachteten Kanten ist. Besitzen alle Sequenzen höchstens die Länge l , so ist die Laufzeit durch $O(2^n \cdot l^n)$ beschränkt. Damit ist dieses Verfahren nur für wenige Sequenzen anwendbar.

Mit dem oben beschriebenen Algorithmus können auch optimale gewichtete Alignments berechnet werden. Hierzu werden die Kantengewichte aus (3.2) ersetzt durch

$$s(e) = \sum_{1 \leq j < k \leq n} w_{S_j, S_k} \cdot d(f_j(e), f_k(e)).$$

3.2 Beschleunigung des Verfahrens

Ein Nachteil des Algorithmus ist, dass für alle Knoten $v \in V_S$ die Kosten $D(v)$ berechnet werden. Viele Knoten werden aber an keinem optimalen Alignment beteiligt sein. In [8] stellen Carrillo und Lipman eine Möglichkeit vor, “unnötige” Knoten rechtzeitig zu erkennen.

Sei \mathcal{A}^* ein optimales und \mathcal{A} ein beliebiges Alignment von \mathcal{S} . Dann gilt für $1 \leq k_1 < k_2 \leq n$ wegen $D(\{S_{k_1}, S_{k_2}\} | \mathcal{A}^*) \geq D(\{S_{k_1}, S_{k_2}\})$

$$\begin{aligned} D(\mathcal{A}) &\geq D(\mathcal{A}^*) = \sum_{1 \leq j_1 < j_2 \leq n} D(\{S_{j_1}, S_{j_2}\} | \mathcal{A}^*) \\ &\geq D(\{S_{k_1}, S_{k_2}\} | \mathcal{A}^*) + \sum_{\substack{1 \leq j_1 < j_2 \leq n \\ (j_1, j_2) \neq (k_1, k_2)}} D(\{S_{j_1}, S_{j_2}\}). \end{aligned}$$

Damit lassen sich die Kosten eines durch ein optimales Alignment induzierten Alignments von S_{k_1} und S_{k_2} wie folgt abschätzen:

$$D(\{S_{k_1}, S_{k_2}\} | \mathcal{A}^*) \leq D(\mathcal{A}) - \overbrace{\sum_{\substack{1 \leq j_1 < j_2 \leq n \\ (j_1, j_2) \neq (k_1, k_2)}}}^{U_{k_1, k_2} :=} D(\{S_{j_1}, S_{j_2}\})$$

Dies ist also eine notwendige Bedingung für ein optimales Alignment. Da wir nur an optimalen Alignments interessiert sind, können wir den Graphen unter Ausnutzung dieser Bedingung einschränken. Dazu sei $\mathcal{W}_{j,k}$ die Menge aller Pfade W von der Quelle zur Senke des Alignment-Graphen, die $D(\{S_j, S_k\} | \mathcal{A}_W) \leq U_{j,k}$ erfüllen. Dann liegen alle optimalen Pfade in der Menge

$$\mathcal{W} := \bigcap_{1 \leq j < k \leq n} \mathcal{W}_{j,k}.$$

$\mathcal{Q}_{j,k}$ bezeichnet die Menge der Knoten des Alignment-Graphen, die an mindestens einem Pfad aus $\mathcal{W}_{j,k}$ beteiligt sind. Dann ist

$$\mathcal{Q} := \bigcap_{1 \leq j < k \leq n} \mathcal{Q}_{j,k}$$

die Menge aller zu betrachtenden Knoten, d.h. alle Pfade aus \mathcal{W} verlaufen vollständig in \mathcal{Q} . Für die Berechnung eines optimalen Alignments genügt es also die Knoten aus \mathcal{Q} zu betrachten.

Die Mengen $\mathcal{Q}_{j,k}$ lassen sich dadurch berechnen, dass in dem Alignment-Graphen $G_{\{S_j, S_k\}}$ von S_j und S_k für alle Knoten $v \in V_{\{S_j, S_k\}}$ mittels dynamischer Programmierung die Kosten $D(v)$ von der Quelle zu v und $\overline{D}(v)$ von v zur Senke berechnet werden. Ein Knoten v ist genau dann in $\mathcal{Q}_{j,k}$, wenn er Teil eines Pfades W ist, der die Bedingung $D(\{S_j, S_k\} | \mathcal{A}_W) \leq U_{j,k}$ erfüllt. Dies wiederum ist genau dann der Fall, wenn $D(v) + \overline{D}(v) \leq U_{j,k}$ ist. Man kann so alle $\mathcal{Q}_{j,k}$ in Zeit $O\left(\sum_{1 \leq j < k \leq n} |S_j| \cdot |S_k|\right)$ berechnen. Eine ausführliche Beschreibung dieser Methode geben Altschul und Erickson in [1] und Zuker in [45].

Bisher ist nichts darüber gesagt worden, wie man das frei wählbare Alignment \mathcal{A} , das für die Berechnung der $U_{j,k}$ benötigt wird, am günstigsten wählt. Um die Menge \mathcal{Q} möglichst klein zu halten, sollten die $U_{j,k}$ klein sein, das Alignment \mathcal{A} sollte also möglichst geringe Kosten besitzen. Für \mathcal{A} kann beispielsweise eine Näherungslösung verwendet werden, die mit einem Algorithmus aus Kapitel 4 berechnet wird.

Man beachte, dass sich diese Möglichkeit zur Beschleunigung zwar in der Praxis als erfolgreich erweist, die asymptotische Laufzeit aber nicht verbessert wird.

Gupta, Kececioğlu und Schäffer verbessern dieses Vorgehen in [19] bezüglich seines Platzverbrauchs. Sie nutzen aus, dass die Berechnung eines optimalen Alignments der Berechnung eines kürzesten Pfades (siehe Cormen et al. [11]) von der Quelle zur Senke des Alignment-Graphen entspricht. Die Kosten eines optimalen Alignments sind gerade die Kosten eines solchen kürzesten Pfades.

3.3 Bemerkungen

Bei allen Ansätzen zur Berechnung von Alignments stellt sich heraus, dass nicht die Länge der Sequenzen, sondern ihre Anzahl die Komplexität des Problems ausmacht. Dies äußert sich zum einen darin, dass der hier besprochene Algorithmus eine Laufzeit von $O(l^n 2^n)$, also polynomiell in der Länge der

Sequenzen, hat. Insbesondere kann für jede konstante Anzahl von Sequenzen ein optimales Alignment in polynomieller Zeit berechnet werden.

Zum anderen ist die Güte der Alignments, die mit den Algorithmen aus Kapitel 4 berechnet werden können, nur von der Anzahl der Sequenzen abhängig. Ferner genügen in Kapitel 6 Sequenzen konstanter Länge, um eine untere Schranke für die Approximierbarkeit von MIN-BMSA zu zeigen.

4 Approximationsalgorithmen

In diesem Kapitel werden Algorithmen zur Approximation von MIN-MSA vorgestellt, mit denen ein Alignment berechnet werden kann, dessen Kosten höchstens um einen konstanten Faktor größer sind als die Kosten eines optimalen Alignments. Eine ausführliche Beschreibung der hier vorgestellten Algorithmen ist in [37] zu finden.

4.1 Center-Star-Algorithmus

In [20] und [21] beschreibt Gusfield den so genannten Center-Star-Algorithmus zur Berechnung eines Alignments von n Sequenzen, dessen Kosten höchstens um einen Faktor $2 - \frac{2}{n}$ größer sind als die Kosten eines optimalen Alignments.

Die Grundlage für diesen Algorithmus bildet das folgende Theorem.

Theorem 4.1 (Feng, Doolittle [14], Gusfield [21]) *Sei $G = (\mathcal{S}, E)$ ein ungerichteter Baum mit Knoten $\mathcal{S} = \{S_1, \dots, S_n\}$. Dann gibt es ein Alignment $\mathcal{A} = \{A_1, \dots, A_n\}$ von \mathcal{S} , so dass für alle $\{S_j, S_k\} \in E$ gilt*

$$D(\{S_j, S_k\}) = D(A_j, A_k).$$

Das Alignment \mathcal{A} ist im Allgemeinen nicht optimal, sondern minimiert die Kosten entlang der Kanten des Baumes. Ein Alignment \mathcal{A} , das diese Bedingung erfüllt, heißt kompatibel zu G . Ein zu einem Baum kompatibles Alignment kann in Zeit $O(l^2 \cdot n)$ berechnet werden, falls die Längen der Sequenzen durch l beschränkt sind (siehe Gusfield [21]).

Ein Stern ist ein ungerichteter Graph $G_c = (\mathcal{S}, E_c)$ mit Kanten $E_c = \{\{S_j, S_c\} | 1 \leq j \leq n \text{ und } j \neq c\}$, also ein Graph, bei dem alle Knoten (Sequenzen) nur mit S_c verbunden sind. Offensichtlich ist G_c ein Baum, es gibt also ein zu G_c kompatibles Alignment, das effizient berechnet werden kann.

Das folgende Theorem beruht auf Theorem 4.1 und liefert die Grundlage des Center-Star-Algorithmus.

Theorem 4.2 (Gusfield [20]) *Für $c = 1, \dots, n$ sei $G_c = (\mathcal{S}, E_c)$ ein Stern und $\mathcal{A}_c = \{A_1^c, \dots, A_n^c\}$ ein zu G_c kompatibles Alignment. Des Weiteren sei $D_c = \sum_{j=1}^n D(\{S_j, S_c\})$. Wähle c^* so, dass $D_{c^*} = \min_{1 \leq c \leq n} D_c$ ist. Dann gilt die Abschätzung*

$$D(\mathcal{A}_{c^*}) \leq \left(2 - \frac{2}{n}\right) \cdot D(\mathcal{S}).$$

Im Beweis des Theorems wird die Tatsache verwendet, dass sich die Kosten eines optimalen Alignments nach unten durch die Summe der Kosten der optimalen paarweisen Alignments abschätzen lassen:

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

Des Weiteren wird ausgenutzt, dass die Dreiecksungleichung auch für die Kosten der optimalen Alignments gilt:

$$D(\{S_j, S_k\}) \leq D(\{S_j, S_c\}) + D(\{S_c, S_k\})$$

Um ein Alignment zu berechnen, das höchstens um den Faktor $2 - \frac{2}{n}$ höhere Kosten als ein optimales Alignment hat, muß zunächst c^* bestimmt werden und anschließend ein zu G_{c^*} kompatibles Alignment berechnet werden. Zur Bestimmung von c^* müssen $O(n^2)$ Alignments von 2 Sequenzen berechnet werden. Ein Alignment von zwei Sequenzen der Länge l kann in Zeit $O(l^2)$ berechnet werden, c^* kann folglich in Zeit $O(l^2 \cdot n^2)$ berechnet werden. Weiterhin kann das zu G_{c^*} kompatible Alignment in Zeit $O(l^2 \cdot n)$ berechnet werden. Es ergibt sich also insgesamt eine Laufzeit von $O(l^2 \cdot n^2)$.

4.2 Verallgemeinerung des Center-Star-Algorithmus

Pevzner verallgemeinert in [34] den Center-Star-Algorithmus von Gusfield. Dazu definiert er Graphen mit bestimmten topologischen Eigenschaften, die er Konfigurationen nennt. Hier werden wir nur die so genannten p -Sterne und p -Sterne mit q -Clique, definieren, da diese die für Approximationsalgorithmen interessanten Konfigurationen sind. p -Sterne und p -Sterne mit q -Clique sind Verallgemeinerungen von Sternen.

Definition 4.3 (p -Stern, Pevzner [34]) Sei $G = (V, E)$ ein ungerichteter Graph. G heißt p -Stern, falls sich V in paarweise disjunkte Mengen C, V_1, \dots, V_m zerlegen läßt, so dass gilt:

- (i) $|C| = 1$
- (ii) $|V_j| = p - 1$ für alle $1 \leq j \leq m$
- (iii) $E = E_1 \cup \dots \cup E_m$, wobei $E_j = \{\{v, w\} | v \neq w \text{ und } v, w \in V_j \cup C\}$

G heißt p -Stern mit q -Clique für $2 \leq q \leq p - 1$, wenn (i) und (iii) gelten und $|V_m| = q - 1$ und $|V_j| = p - 1$ für $j = 1, \dots, m - 1$ ist.

Ein p -Stern hat ein Zentrum C , bestehend aus einem Knoten, welcher in jeder der m p -Cliques $(V_j \cup C, E_j)$ enthalten ist. Man beachte, dass p -Sterne immer n Knoten mit $n \equiv 1 \pmod{p-1}$ besitzen. p -Sterne mit q -Clique besitzen n Knoten mit $n \equiv q \pmod{p-1}$. Im Folgenden werden p -Sterne mit q -Clique (p, q) -Sterne genannt.

Wie bei Sternen können auch zu p -Sternen und (p, q) -Sternen kompatible Alignments definiert werden. Ein Alignment \mathcal{A} von \mathcal{S} heißt kompatibel zu einem p -Stern bzw. (p, q) -Stern $G = (\mathcal{S}, E)$, falls für alle maximalen Cliques Ω von G gilt

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

Ein zu G kompatibles Alignment wird mit \mathcal{A}^G bezeichnet. Auch für p -Sterne und (p, q) -Sterne kann die Existenz kompatibler Alignments bewiesen werden. Sie können in Zeit $O(l^p \cdot n)$ berechnet werden.

Über p -Sterne läßt sich das folgende Theorem zeigen.

Theorem 4.4 (Pevzner [34]) *Sei $G = (V, E)$ ein p -Stern mit $|V| = n$ und \mathcal{G}^G die Menge aller zu G isomorphen Graphen mit Knoten \mathcal{S} . Dann gilt*

$$\min_{H \in \mathcal{G}^G} D(\mathcal{A}^H) \leq \left(2 - \frac{p}{n}\right) \cdot D(\mathcal{S}).$$

Für $p = 2$ ergibt sich die Abschätzung aus Theorem 4.2.

Das Alignment \mathcal{A}^H , $H \in \mathcal{G}^G$, mit minimalen Kosten heißt optimales kompatibles Alignment zu G .

Für (p, q) -Sterne wird hier nur der Fall $p = 3$ und $q = 2$ betrachtet. Sei also $G = (V, E)$ ein 3-Stern mit 2-Clique. Dann ergibt sich mit den Bezeichnungen aus Theorem 4.4 die Abschätzung (siehe [34])

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

$(3, 2)$ -Sterne gibt es für jede gerade Anzahl Knoten (d.h. Sequenzen), 3-Sterne für jede ungerade Anzahl.

4.3 Ausgewogene Mengen

Die Menge der Graphen, die zu einem p -Stern mit $p \geq 3$ isomorph sind, ist sehr groß. Deswegen kann das optimale kompatible Alignment zu einem p -Stern nicht durch einfaches Ausprobieren aller Möglichkeiten gefunden werden, wie dies für $p = 2$ effizient möglich ist. Speziell für 3-Sterne (bzw.

(3, 2)-Sterne, falls eine gerade Anzahl Sequenzen vorliegt) kann das optimale kompatible Alignment durch perfektes minimales Matching (siehe Gabow [15], Galil [16] und Pevzner [34]) gefunden werden. Es ergibt sich eine Laufzeit von $O(l^3 \cdot n^3 + n^4)$ sowohl für eine gerade als auch für eine ungerade Anzahl von Sequenzen.

Für $p \geq 4$ sind keine effizienten Algorithmen zur Bestimmung eines optimalen zu einem p -Stern kompatiblen Alignments bekannt.

Um eine Fehlerabschätzung durchführen zu können ist es nicht unbedingt erforderlich, das optimale kompatible Alignment zu finden. Im Beweis von Theorem 4.4 genügt es nämlich, ein kompatibles Alignment zu berechnen, dessen Kosten höchstens so groß sind wie die durchschnittlichen Kosten aller zu isomorphen Graphen kompatiblen Alignments.

Bafna, Lawler und Pevzner zeigen in [6], dass man dazu nur bestimmte Teilmengen der isomorphen Graphen betrachten muss, die sie ausgewogene Mengen nennen.

Die Idee folgt aus der Beobachtung, dass in p - und (p, q) -Sternen der kürzeste Weg zwischen zwei Knoten eindeutig bestimmt ist. Sei $G = (\mathcal{S}, E)$ ein p - oder (p, q) -Stern. $\Gamma_{k,k'}$ sei die Menge der Kanten des kürzesten Weges zwischen zwei Knoten S_k und $S_{k'}$ aus \mathcal{S} . Dann bezeichnet $w_{S_j, S_{j'}}^G$ die Anzahl der kürzesten Wege, die die Kante $\{S_j, S_{j'}\}$ enthalten, d.h.

$$w_{S_j, S_{j'}}^G := |\{\Gamma_{k,k'} \mid \{S_j, S_{j'}\} \in \Gamma_{k,k'} \text{ und } k < k'\}|.$$

Die $w_{S_j, S_{j'}}^G$ können konkret angegeben werden. Wir betrachten hier nur den Fall, dass G ein (p, q) -Stern ist; die Werte für reine p -Sterne können daraus hergeleitet werden. Sei S_c das Zentrum von G . Ω bezeichnet die Knoten der q -Clique ohne S_c . Dann gilt

$$w_{S_j, S_{j'}}^G = \begin{cases} 1 & \text{falls } S_j, S_{j'} \text{ in der gleichen Clique sind, } j \neq j' \\ & \text{und } j, j' \neq c, \\ n - (p - 1) & \text{falls } j = c, j \neq j' \text{ und } S_{j'} \notin \Omega \text{ oder umgekehrt,} \\ n - (q - 1) & \text{falls } j = c \text{ und } S_{j'} \in \Omega \text{ oder umgekehrt,} \\ 0 & \text{sonst (d.h. } \{S_j, S_{j'}\} \notin E \text{).} \end{cases}$$

Wir konstruieren eine Gewichtsmatrix $W^G = (w_{S_j, S_k}^G)_{S_j, S_k \in \mathcal{S}}$ (siehe Definition 2.6). Damit gilt für beliebige Alignments die Abschätzung

$$D(\mathcal{A}) \leq D_{WG}(\mathcal{A}). \quad (4.1)$$

Die Ungleichung folgt aus der Tatsache, dass sich mit Hilfe der Dreiecksungleichung die Kosten eines Paares von Sequenzen durch die Summe der

Kosten der Paare auf dem Weg zwischen diesen Sequenzen abschätzen lassen.

Unter Nutzung der Gewichtsmatrix W^G lassen sich nun ausgewogene Mengen definieren.

Definition 4.5 (Ausgewogene Menge, Bafna et al. [6]) *Sei ein (p, q) -Stern G und eine Teilmenge \mathcal{G} aller zu G isomorphen Graphen mit Knoten \mathcal{S} gegeben. Wenn eine Konstante $\kappa(\mathcal{G}) \in \mathbb{N}$ existiert, so dass für alle $j, k = 1, \dots, n$ gilt*

$$\sum_{H \in \mathcal{G}} w_{S_j, S_k}^H = \begin{cases} \kappa(\mathcal{G}) & \text{falls } j \neq k, \\ 0 & \text{falls } j = k, \end{cases}$$

dann heißt \mathcal{G} ausgewogen mit Parameter $\kappa(\mathcal{G})$.

Unter anderem ist die Menge aller zu G isomorphen Graphen mit Knoten \mathcal{S} aufgrund der Symmetrie ausgewogen.

Für ausgewogene Mengen \mathcal{G} wird in [6] gezeigt, dass gilt

$$\min_{H \in \mathcal{G}} D_{W^H}(\mathcal{A}^H) \leq \frac{\kappa(\mathcal{G})}{|\mathcal{G}|} \cdot D(\mathcal{S}). \quad (4.2)$$

Mit den Ungleichungen (4.1) und (4.2) ergibt sich die Abschätzung

$$\min_{H \in \mathcal{G}} D(\mathcal{A}^H) \leq \min_{H \in \mathcal{G}} D_{W^H}(\mathcal{A}^H) \leq \frac{\kappa(\mathcal{G})}{|\mathcal{G}|} \cdot D(\mathcal{S}).$$

Daher ist es nun die Aufgabe, möglichst kleine ausgewogene Mengen von p -Sternen zu finden. Die Konstruktion kleiner ausgewogener Mengen ist allerdings nicht einfach, außer für bestimmte p und n .

Bafna et al. zeigen in [6], dass es möglich ist, exponentiell große ausgewogene Mengen zu konstruieren und mittels Matching-Algorithmen den optimalen p -Stern dennoch effizient zu ermitteln. Damit konnten sie beweisen, dass es für jede Konstante p einen polynomiellen Algorithmus gibt, der ein Alignment mit garantierter Fehlerschranke $2 - \frac{p}{n}$ berechnet. Man beachte, dass es sich dabei nicht um ein Polynomialzeit-Approximations-Schema (PTAS, siehe Ausiello et al. [5] oder Kapitel 5) handelt, da sich asymptotisch für jede Konstante p die Schranke 2 ergibt.

In [37] wird die Laufzeit für $p = 3$ mittels ausgewogener Mengen verbessert, und zwar auf $O(n^2 \cdot l^3)$ für eine gerade Anzahl Sequenzen bei Fehlerschranke $2 - \frac{3}{n}$ bzw. auf $O(n^3 \cdot l^3)$ für eine ungerade Anzahl Sequenzen bei Fehlerschranke $2 - \frac{3}{n} + \frac{2}{n \cdot (n-1)}$.

5 Komplexität von Optimierungsproblemen

5.1 \mathcal{NP} -Optimierungsprobleme

Bei einem Optimierungsproblem Π geht es darum, für eine Instanz X eine Lösung L zu finden, die, je nach Problemstellung, eine Kostenfunktion μ minimiert oder maximiert. Viele bekannte Optimierungsprobleme, wie z. B. MSA, sind \mathcal{NP} -vollständig. Da \mathcal{NP} eine Klasse von Entscheidungsproblemen ist, muss ein Optimierungsproblem dafür zunächst in ein Entscheidungsproblem umformuliert werden. Dazu führt man eine Schranke für die Kostenfunktion ein und definiert eine dem Optimierungsproblem zugehörige Sprache. Die zu einem Minimierungsproblem gehörende Sprache ist

$$\{(X, k) \mid X \text{ hat eine Lösung } L \text{ mit } \mu(X, L) \leq k\}.$$

Analog erhält man die zu einem Maximierungsproblem gehörende Sprache.

Ein Optimierungsproblem wird durch die Betrachtung als Sprache auf ein Entscheidungsproblem eingeschränkt. Es gibt keine Näherungslösungen, da ein Wort entweder in der Sprache ist oder nicht. Eigenschaften wie Approximierbarkeit gehen bei dieser Betrachtungsweise also verloren. Für Untersuchungen über die Approximierbarkeit von Optimierungsproblemen müssen diese in ihrer Struktur erhalten bleiben, weshalb man für Optimierungsprobleme eigene Klassen einführt. Hier werden wir uns auf so genannte \mathcal{NP} -Optimierungsprobleme beschränken.

Definition 5.1 (\mathcal{NP} -Optimierungsproblem, Crescenzi et al. [12])

Ein \mathcal{NP} -Optimierungsproblem ist ein 4-Tupel $\Pi = (I, \text{sol}, \mu, \text{goal})$ mit folgenden Eigenschaften:

- (i) *I ist eine in polynomieller Zeit entscheidbare Menge von Instanzen.*
- (ii) *Für jede Instanz $X \in I$ bezeichnet $\text{sol}(X)$ die Menge aller möglichen Lösungen für X . Die Lösungen in $\text{sol}(X)$ sind polynomiell in $|X|$ beschränkt und $\text{sol}(X)$ ist in polynomieller Zeit entscheidbar.*
- (iii) *Sei X eine Instanz und $L \in \text{sol}(X)$, dann bezeichnet $\mu(X, L) \in \mathbb{N}$ das Maß von L . μ ist in polynomieller Zeit berechenbar.*
- (iv) *$\text{goal} \in \{\min, \max\}$.*

Eine optimale Lösung zu einer Instanz $X \in I$ ist eine Lösung $L_{\text{opt}} \in \text{sol}(X)$ mit $\mu(X, L_{\text{opt}}) = \text{goal}_{L \in \text{sol}(X)} \mu(X, L)$. Im Folgenden sei $\text{sol}(I) := \bigcup_{X \in I} \text{sol}(X)$ die Menge aller möglichen Lösungen für Instanzen von Π .

Beispiel 5.2 Das Traveling-Salesman-Problem (TSP) ist ein \mathcal{NP} -Optimierungsproblem $(I, \text{sol}, \mu, \text{goal})$. Dabei ist

- (i) eine Instanz $V \in I$ des TSP ein vollständiger Graph mit Knoten $\{v_1, \dots, v_n\}$ und Kantengewichten $d(v_j, v_k) \in \mathbb{N}$,
- (ii) $\pi \in \text{sol}(V)$ ein Rundweg durch den Graphen, d.h. eine Permutation π der Knoten $(v_{\pi(1)}, \dots, v_{\pi(n)})$,
- (iii) $\mu(V, \pi) = \sum_{i=1}^{n-1} d(v_{\pi(i)}, v_{\pi(i+1)}) + d(v_{\pi(n)}, v_{\pi(1)})$ die Länge des Rundweges und
- (iv) $\text{goal} = \min$, d.h. gesucht ist der kürzeste Rundweg durch den Graphen.

MIN-MSA, MIN-WMSA und MIN-BMSA sind ebenfalls \mathcal{NP} -Optimierungsprobleme.

Aus der Definition folgt, dass die zu einem \mathcal{NP} -Optimierungsproblem gehörende Sprache in \mathcal{NP} liegt. \mathcal{NPO} ist die Klasse aller \mathcal{NP} -Optimierungsprobleme.

\mathcal{PO} ist die Klasse aller \mathcal{P} -Optimierungsprobleme. Ein Optimierungsproblem $\Pi \in \mathcal{PO}$ kann deterministisch in polynomieller Zeit exakt gelöst werden und die zu Π gehörende Sprache liegt in \mathcal{P} .

5.2 Approximation von Optimierungsproblemen

Ist die zu einem Optimierungsproblem gehörende Sprache \mathcal{NP} -vollständig, dann sagt dies nur wenig über die wirkliche Schwierigkeit des Problems aus. Man könnte sich fragen, ob es nicht einen effizienten Algorithmus gibt, der immer eine Lösung berechnet, deren Kosten nicht allzuweit von den Kosten der optimalen Lösung abweichen. Dazu die folgende Definition.

Definition 5.3 (Güte, Ausiello et. al. [5]) Sei Π ein Optimierungsproblem. Für eine Instanz X seien $\text{opt}(X)$ die Kosten einer optimalen Lösung für X . Sei $L \in \text{sol}(X)$, dann bezeichnet

$$R(X, L) := \max \left\{ \frac{\text{opt}(X)}{\mu(X, L)}, \frac{\mu(X, L)}{\text{opt}(X)} \right\}$$

die Güte von L bezüglich X .

Damit läßt sich der Begriff eines Approximationsalgorithmus für Optimierungsprobleme definieren.

Definition 5.4 (δ -Approximation, Ausiello et al. [5]) Sei Π ein Optimierungsproblem und $\delta \geq 1$ eine Konstante. Eine Funktion $A : I \rightarrow \text{sol}(I)$ heißt δ -Approximation für Π , falls A für alle Instanzen $X \in I$ eine Lösung $A(X) \in \text{sol}(X)$ berechnet, so dass

$$R(X, A(X)) \leq \delta$$

gilt.

Ist Π ein \mathcal{NP} -Optimierungsproblem, dann heißt Π δ -approximierbar, falls es eine in polynomieller Zeit berechenbare δ -Approximation für Π gibt.

Allgemeiner können auch r -Approximationen mit Funktionen $r : I \rightarrow [1, \infty[$ definiert werden. Eine Funktion A heißt r -Approximation für Π , falls sie für alle Instanzen $X \in I$ eine Lösung $A(X) \in \text{sol}(X)$ mit $R(X, A(X)) \leq r(X)$ berechnet. Üblicherweise hängt der Funktionswert von r von der Länge der Eingabe X ab. Ein \mathcal{NP} -Optimierungsproblem Π heißt r -approximierbar, falls es eine in polynomieller Zeit berechenbare r -Approximation für Π gibt. Z.B. besitzt MIN-MSA eine 2-Approximation (siehe Gusfield [20] und Kapitel 4) und MIN-WMSA eine r -Approximation, wobei r durch $r(\mathcal{S}) := \log^2(\#\text{Sequenzen in } \mathcal{S})$ gegeben ist (siehe Wu et al. [44]).

$\mathcal{APX} \subseteq \mathcal{NPO}$ ist die Klasse der \mathcal{NP} -Optimierungsprobleme, die eine in polynomieller Zeit berechenbare δ -Approximation für eine Konstante $\delta \geq 1$ besitzen. Man kann zeigen, dass die Inklusion $\mathcal{APX} \subseteq \mathcal{NPO}$ genau dann echt ist, wenn $\mathcal{NP} \neq \mathcal{P}$ ist. Es gibt also unter der Annahme $\mathcal{NP} \neq \mathcal{P}$ \mathcal{NP} -Optimierungsprobleme, die nicht mit konstanter Güte approximiert werden können.

Andererseits gibt es Optimierungsprobleme, die für jede Konstante $\delta > 1$ eine δ -Approximation zulassen. Das bedeutet, dass das Problem beliebig gut approximiert werden kann. Allerdings hängt die Laufzeit des Approximationsalgorithmus meist stark von der gewählten Güte δ ab – für ein konstantes δ muss sie aber polynomiell in der Eingabelänge sein. Eine solche Menge von Algorithmen, die für jedes $\delta > 1$ eine δ -Approximation für Π enthält, heißt *Polynomialzeit-Approximations-Schema* (PTAS, polynomial time approximation scheme, siehe Ausiello et al. [5]). Die Klasse der Probleme, die ein Polynomialzeit-Approximations-Schema besitzen, heißt \mathcal{PTAS} .

\mathcal{APX} und \mathcal{NPO} besitzen vollständige Probleme. Die für Entscheidungsprobleme verwendeten Reduktionen können aber für die Reduktion von Optimierungsproblemen nicht verwendet werden, da es bei Entscheidungsproblemen den Begriff der Approximation nicht gibt. Deswegen definiert man für Optimierungsprobleme eigene Reduktionsbegriffe.

Definition 5.5 (PTAS-Reduktion, Crescenzi et al. [12]) Seien Π und Π' \mathcal{NP} -Optimierungsprobleme. Π heißt PTAS-reduzierbar auf Π' ($\Pi \leq_{\text{PTAS}} \Pi'$)

Π'), falls es drei Funktionen $f_1 : I \times]1, \infty[\rightarrow I'$, $f_2 : I \times \text{sol}'(I') \times]1, \infty[\rightarrow \text{sol}(I)$ und $\varphi :]1, \infty[\rightarrow]1, \infty[$ gibt, so dass für alle Instanzen $X \in I$ von Π Folgendes gilt:

- (i) Für jedes $\delta > 1$ berechnet f_1 in polynomieller Zeit eine Instanz $X' = f_1(X, \delta) \in I'$ von Π' .
- (ii) Für alle $L' \in \text{sol}'(X')$ kann $f_2(X, L', \delta) = L \in \text{sol}(X)$ in polynomieller Zeit berechnet werden.
- (iii) φ ist berechenbar und invertierbar.
- (iv) Für jedes $L' \in \text{sol}'(X')$ und für jedes $\delta > 1$ gilt:

$$R(X', L') \leq \varphi(\delta) \Rightarrow R(X, L) \leq \delta$$

Bei PTAS-Reduktionen überträgt sich die Qualität der Lösung für X' auf die mittels f_2 errechnete Lösung für X . Ein Approximationsalgorithmus für Π' liefert somit einen Approximationsalgorithmus für Π .

Ein Optimierungsproblem Π heißt hart für \mathcal{APX} bzw. \mathcal{NPO} , falls sich alle Probleme aus \mathcal{APX} bzw. \mathcal{NPO} auf Π PTAS-reduzieren lassen. Π heißt vollständig für die entsprechende Klasse, wenn Π selbst in der Klasse enthalten ist.

Die Vielfalt des Traveling-Salesman-Problems bzgl. der Approximierbarkeit zeigt das folgende Beispiel.

Beispiel 5.6 TSP ist \mathcal{NPO} -vollständig (siehe Orponen, Mannila [30]).

Metric-TSP sei das Problem, einen kürzesten Rundweg zu finden, wobei die Distanzfunktion d die Dreiecksungleichung erfüllt. Papadimitriou und Yannakakis zeigen in [33], dass Metric-TSP \mathcal{APX} -vollständig ist. Diese Version des Traveling-Salesman-Problems besitzt eine $\frac{3}{2}$ -Approximation (siehe Christofides [10]).

Ein Spezialfall von Metric-TSP ist Geometric-TSP. Hierbei sind die Knoten in einer Ebene angeordnet und der Abstand zwischen zwei Punkten ist durch den euklidischen Abstand gegeben. In [3] zeigt Arora, dass Geometric-TSP ein Polynomialzeit-Approximations-Schema besitzt.

Die zu TSP, Metric-TSP und Geometric-TSP gehörenden Sprachen sind \mathcal{NP} -vollständig (siehe Garey und Johnson [17], Papadimitriou [31]).

Ein weiteres Beispiel für ein \mathcal{APX} -vollständiges Problem ist MAX-3SAT (siehe Ausiello et al. [5] und Papadimitriou et al. [32]).

Definition 5.7 (MAX-3SAT) Sei $\mathcal{C} = \{C_1, C_2, \dots, C_t\}$ eine Menge von disjunktiven Klauseln. Jede Klausel bestehe aus genau drei Literalen. Dann ist MAX-3SAT das Optimierungsproblem, die maximale Anzahl gleichzeitig erfüllbarer Klauseln aus \mathcal{C} zu ermitteln.

Für jedes \mathcal{APX} -vollständige Problem Π gilt: $\Pi \in \mathcal{PTAS}$ genau dann, wenn $\mathcal{NP} = \mathcal{P}$. Analog sind \mathcal{NPO} -harte Probleme unter der Annahme $\mathcal{NP} \neq \mathcal{P}$ nicht in \mathcal{APX} enthalten. Es ergibt sich die folgende Hierarchie, wobei die drei Inklusionen genau dann echt sind, wenn $\mathcal{NP} \neq \mathcal{P}$ ist (siehe [5]):

$$\mathcal{PO} \subseteq \mathcal{PTAS} \subseteq \mathcal{APX} \subseteq \mathcal{NPO} \quad (5.1)$$

5.3 MAX- \mathcal{SNP}

Papadimitriou und Yannakakis führen in [32] zwei Klassen von Maximierungsproblemen mit den Namen MAX- \mathcal{SNP} und MAX- \mathcal{NP} ein. Diese Klassen sind aus Fagins syntaktischer Definition von \mathcal{NP} (siehe Fagin [13] und Immerman [23]) abgeleitet.

Fagin zeigt in [13], dass sich jedes Problem in \mathcal{NP} als Prädikat auf Strukturen G in der Form

$$\exists S \forall x \exists y \psi(x, y, G, S)$$

schreiben läßt. Dabei ist S eine Struktur und damit $\exists S$ eine Quantor zweiter Stufe. $\forall x$ und $\exists y$ sind Quantoren erster Stufe, ψ ist quantorenfrei.

Beispiel 5.8 (Papadimitriou et al. [32]) 3SAT kann wie folgt geschrieben werden:

$$\exists T \forall C \exists x ((P(C, x) \wedge T(x)) \vee (N(C, x) \wedge \neg T(x)))$$

Dabei bezeichnet T die Menge der auf true gesetzten Variablen, $P(C, x)$ heißt, dass x in der Klausel C positiv auftritt, und $N(C, x)$ heißt, dass x in C negiert auftritt.

Bei einigen Problemen, kann auf eine Alternation der Quantoren verzichtet werden. Diese können in der Form

$$\exists S \forall x \psi(x, G, S)$$

geschrieben werden. Die Probleme, die sich in dieser Form beschreiben lassen, bilden eine Teilmenge \mathcal{SNP} (strict \mathcal{NP}) von \mathcal{NP} . 3SAT ist nicht nur

in \mathcal{NP} , sondern auch in \mathcal{SNP} enthalten.

Fortsetzung von Beispiel 5.8 Seien C_j ($j = 0, 1, 2, 3$) die Mengen der Klauseln mit j negierten Literalen. Dabei sollen in jeder Klausel $C \in C_j$ die j negierten Literale vorne stehen. Dann läßt sich 3SAT wie folgt beschreiben:

$$\begin{aligned} \exists T \forall (x_1, x_2, x_3) \quad & (C_0(x_1, x_2, x_3) \Rightarrow (T(x_1) \vee T(x_2) \vee T(x_3))) \\ & \wedge (C_1(x_1, x_2, x_3) \Rightarrow (\neg T(x_1) \vee T(x_2) \vee T(x_3))) \\ & \wedge (C_2(x_1, x_2, x_3) \Rightarrow (\neg T(x_1) \vee \neg T(x_2) \vee T(x_3))) \\ & \wedge (C_3(x_1, x_2, x_3) \Rightarrow (\neg T(x_1) \vee \neg T(x_2) \vee \neg T(x_3))) \end{aligned}$$

Zu jedem Prädikat P der Form $\exists S \forall x \psi(x, G, S)$ aus \mathcal{SNP} kann ein Maximierungsproblem $\max P$ formuliert werden:

$$\max_S |\{x \mid \psi(x, G, S)\}|$$

MAX- \mathcal{SNP} ist die Menge aller Maximierungsprobleme $\max P$, die sich in dieser Form beschreiben lassen (siehe [32]). Analog ist MAX- \mathcal{NP} die Klasse der Maximierungsprobleme, die in der Form

$$\max_S |\{x \mid \exists y \psi(x, y, G, S)\}|$$

beschrieben werden können. Die Maximierungsprobleme in MAX- \mathcal{SNP} und MAX- \mathcal{NP} sind \mathcal{NP} -Optimierungsprobleme. Für ein Problem Π in MAX- \mathcal{SNP} der Form $\max_S |\{x \mid \psi(x, G, S)\}|$ entspricht die Menge der G den Instanzen des Optimierungsproblems. Die Menge der S entspricht den möglichen Lösungen. Das Maß einer Lösung S einer Instanz G ist die Kardinalität der Menge $\{x \mid \exists y \psi(x, y, G, S)\}$, d.h.

$$\mu(G, S) = |\{x \mid \exists y \psi(x, y, G, S)\}|$$

Näheres zur Klassifizierung von Optimierungsproblemen anhand ihrer logischen Darstellung ist in Neuber [29] zu finden.

Papadimitriou und Yannakakis zeigen, dass es Probleme gibt, die vollständig für MAX- \mathcal{SNP} bezüglich so genannter L-Reduktionen sind.

Definition 5.9 (L-Reduktion, Papadimitriou et al. [32]) Seien Π und Π' zwei Optimierungsprobleme. Dann heißt Π L-reduzierbar auf Π' ($\Pi \leq_L \Pi'$), wenn es zwei in polynomieller Zeit berechenbare Funktionen $f_1 : I \rightarrow I'$, $f_2 : \text{sol}(I') \times I \rightarrow \text{sol}(I)$ und Konstanten $\gamma_1, \gamma_2 > 0$ gibt, so dass für jede Instanz $X \in I$ von Π gilt:

- (i) f_1 erzeugt eine Instanz $f_1(X) = X'$ von Π' mit $\text{opt}(X') \leq \gamma_1 \cdot \text{opt}(X)$.
- (ii) Für eine Lösung L' von X' berechnet f_2 eine Lösung $L = f_2(L', X)$ von X , so dass $|\mu(X, L) - \text{opt}(X)| \leq \gamma_2 \cdot |\mu'(X', L') - \text{opt}(X')|$ gilt.

MAX-3SAT ist nicht nur \mathcal{APX} -vollständig bezüglich PTAS-Reduktion, sondern auch MAX- \mathcal{SNP} -vollständig bezüglich L-Reduktion (siehe [32]). Gilt $\Pi \leq_L \Pi'$, so gilt auch $\Pi \leq_{\text{PTAS}} \Pi'$. Die Umkehrung gilt jedoch nicht.

Bei L-Reduktionen – genau wie bei PTAS-Reduktionen – liefert ein Approximationsalgorithmus für Π' einen Approximationsalgorithmus für Π . Aus $\Pi \leq_L \Pi'$ und $\Pi' \in \mathcal{APX}$ (bzw. $\Pi' \in \mathcal{PTAS}$) folgt also, dass auch Π in \mathcal{APX} (bzw. in \mathcal{PTAS}) enthalten ist.

Für jedes MAX- \mathcal{SNP} -vollständige Problem Π gibt es zwei Konstanten $\delta_\Pi > \epsilon_\Pi > 1$, so dass es eine δ_Π -Approximation für Π gibt und aus einer ϵ_Π -Approximation für Π sofort $\mathcal{NP} = \mathcal{P}$ folgen würde (Arora [4]). Unter der Annahme $\mathcal{NP} \neq \mathcal{P}$ gilt daher MAX- $\mathcal{SNP} \not\subseteq \mathcal{PTAS}$.

Da jedes Problem in MAX- \mathcal{SNP} eine δ -Approximation besitzt, gilt folglich MAX- $\mathcal{SNP} \subseteq \mathcal{APX}$. Weil MAX- \mathcal{SNP} aber keine Minimierungsprobleme enthält, gilt $\mathcal{PO} \not\subseteq \text{MAX-}\mathcal{SNP}$. Daher kann MAX- \mathcal{SNP} nicht in die Hierarchie (5.1) eingeordnet werden.

Um dennoch MAX- \mathcal{SNP} und \mathcal{APX} vergleichen zu können, betrachten wir den Abschluss von MAX- \mathcal{SNP} unter PTAS-Reduktionen. Den Abschluss einer Komplexitätsklasse \mathcal{C} unter einer Reduzierbarkeit R bezeichnen wir mit \mathcal{C}_R . Er ist wie folgt definiert:

$$\mathcal{C}_R := \{\Pi \mid \exists \Pi' \in \mathcal{C} : \Pi \leq_R \Pi'\}$$

Khanna und andere zeigen in [28], dass

$$\text{MAX-}\mathcal{SNP}_{\text{PTAS}} = \text{MAX-}\mathcal{NP}_{\text{PTAS}} = \mathcal{APX}$$

ist.

6 Eine untere Approximierbarkeitsschranke für MIN-BMSA

Inhalt dieses Kapitels ist ein Beweis für die MAX- \mathcal{NP} -Härte von MIN-BMSA sowie für eine untere Schranke für die Approximierbarkeit dieses Problems.

Dazu betrachten wir zunächst das Problem MAX-Ek-Lin2.

Definition 6.1 (MAX-Ek-Lin2) Sei $(\{1, -1\}, \cdot)$ die zweielementige multiplikative Gruppe $\{1, -1\}$ und $k \geq 2$. Sei weiterhin $\mathcal{G} = \{G_1, \dots, G_t\}$ eine Multimenge linearer Gleichungen mit jeweils genau k Variablen über den Variablen x_1, \dots, x_r , $G_i \hat{=} x_{j_{i,1}} \cdot x_{j_{i,2}} \cdot \dots \cdot x_{j_{i,k}} = a_i$, $j_{i,q} \in \{1, \dots, r\}$ für $1 \leq i \leq t$ und $1 \leq q \leq k$. Dabei ist die rechte Seite der Gleichung $a_i \in \{1, -1\}$ eine Konstante. MAX-Ek-Lin2 ist das Optimierungsproblem, die maximale Anzahl gleichzeitig erfüllbarer Gleichungen zu bestimmen.

Håstad zeigt in [22], dass MAX-Ek-Lin2 für $k \geq 3$ nicht mit Güte $2 - \epsilon$ (für beliebige $\epsilon > 0$) approximiert werden kann, es sei denn $\mathcal{NP} = \mathcal{P}$.

Eine eingeschränkte Version dieser Optimierungsprobleme ist MAX-Ek-neg-Lin2.

Definition 6.2 (MAX-Ek-neg-Lin2) MAX-Ek-neg-Lin2 ist das Optimierungsproblem, zu einer Instanz von MAX-Ek-Lin2, bei der nur rechte Seiten $a_i = -1$ vorkommen, die maximale Anzahl gleichzeitig erfüllbarer Gleichungen zu bestimmen.

Eine Instanz von MAX-Ek-neg-Lin2 bzw. MAX-Ek-Lin2 mit t Gleichungen heißt η -erfüllbar für eine reelle Zahl η mit $0 \leq \eta \leq 1$, falls $\eta \cdot t$ die maximale Anzahl gleichzeitig erfüllbarer Gleichungen ist. Håstad zeigt in [22], dass es für jedes $\epsilon > 0$ \mathcal{NP} -hart ist, $(\frac{1}{2} + \epsilon)$ - und $(1 - \epsilon)$ -erfüllbare Instanzen von MAX-E3-Lin2 zu unterscheiden.

In Abschnitt 6.1 werden wir MAX-E3-Lin2 auf MAX-E2-neg-Lin2 reduzieren um eine untere Schranke für die Approximierbarkeit dieses Problems zu zeigen.

Um die MAX- \mathcal{NP} -Härte von MIN-BMSA zu zeigen, reduzieren wir in Abschnitt 6.2 zunächst MAX-Cut auf MAX-E2-neg-Lin2. Papadimitriou und Yannakakis zeigen in [32], dass MAX-Cut MAX- \mathcal{NP} -vollständig ist.

Definition 6.3 (MAX-Cut) Eine Instanz von MAX-Cut ist ein ungerichteter Graph $G = (V, E)$. Eine Lösung dieses Problems ist eine Partition von V in zwei Mengen V_1 und V_2 (d.h. $V_1 \cap V_2 = \emptyset$ und $V_1 \cup V_2 = V$). Ziel ist es, die Anzahl der Kanten zwischen V_1 und V_2 zu maximieren.

Anschließend werden wir in Abschnitt 6.3 MAX-E2-neg-Lin2 auf MIN-BMSA reduzieren und damit zeigen, dass dieses Problem MAX- \mathcal{SNP} -hart ist. Des Weiteren folgt aus dieser Reduktion eine untere Schranke für die Approximierbarkeit des Problems.

Da MIN-WMSA eine Verallgemeinerung von MIN-BMSA ist, gelten diese Resultate auch für dieses Problem.

MIN-WMSA ist approximierbar mit Güte $O(\log^2(n))$, wobei n die Anzahl der Sequenzen ist (siehe Wu et al. [44]). Da MIN-BMSA eine Einschränkung von MIN-WMSA ist, gilt diese obere Schranke auch für MIN-BMSA.

In Kapitel 7 werden wir zeigen, dass MIN-BMSA und MIN-WMSA mit gleicher Güte approximiert werden können.

6.1 Eine Approximierbarkeitsschranke für MAX-E2-neg-Lin2

Wir werden nun MAX-E3-Lin2 auf MAX-E2-neg-Lin2 reduzieren. Sei dazu $\mathcal{G} = \{G_1, \dots, G_t\}$ eine Multimenge von Gleichungen über Variablen $\mathcal{V} = \{x_1, \dots, x_r\}$. Sei $G_i \hat{=} x_{j_{i,1}} \cdot x_{j_{i,2}} \cdot x_{j_{i,3}} = a_i$ mit $a_i \in \{-1, 1\}$.

Wir konstruieren nun eine Instanz \mathcal{G}' von MAX-E2-neg-Lin2 mit $22 \cdot t$ Gleichungen und $4 \cdot t + 2 \cdot r + 2$ Variablen. Die Reduktion ähnelt Håstads Reduktion von MAX-E3-Lin2 auf MAX-E2-Lin2 in [22]; Håstad verwendet dabei Techniken von Trevisan und anderen aus [38] zur Konstruktion solcher Reduktionen.

Die Menge der Variablen von \mathcal{G}' sei \mathcal{V}' und gegeben durch

$$\mathcal{V}' = \{x_j^+, x_j^- | 1 \leq j \leq r\} \cup \{z^+, z^-\} \cup \{p_{i,1}, p_{i,2}, p_{i,3}, p_{i,z} | 1 \leq i \leq t\}.$$

Man beachte: Falls eine Gleichung $y \cdot y' = -1$ durch eine Belegung erfüllt wird, dann wird sie auch durch die negierte Belegung erfüllt. Daher nehmen wir im weiteren ohne Einschränkung an, dass z^+ der Wert 1 zugewiesen wird.

Wir interpretieren $x_j^+ = x_j$. Ist $x_j^+ \neq x_j^-$, also $x_j^+ = x_j = (-x_j^-)$, dann heißt die Belegung *variablen-konsistent* für x_j . Ist eine Belegung variablen-konsistent für alle $j = 1, \dots, r$ und gilt zusätzlich $z^+ \neq z^-$, so heißt die Belegung *variablen-konsistent*.

Für eine Gleichung $G_i \hat{=} x_{j_{i,1}} \cdot x_{j_{i,2}} \cdot x_{j_{i,3}} = a_i$ werden zunächst folgende 12 Gleichungen erstellt:

$$\begin{aligned} x_{j_{i,q}}^+ \cdot p_{i,q'} &= -1 && \text{für } q, q' = 1, 2, 3 \text{ und } q \neq q' \\ x_{j_{i,q}}^+ \cdot p_{i,z} &= -1 && \text{für } q = 1, 2, 3 \\ x_{j_{i,q}}^- \cdot p_{i,q} &= -1 && \text{für } q = 1, 2, 3 \end{aligned}$$

Hinzu kommen im Fall $a_i = 1$ die vier Gleichungen

$$\begin{aligned} z^+ \cdot p_{i,q} &= -1 \quad \text{für } q = 1, 2, 3 \text{ und} \\ z^- \cdot p_{i,z} &= -1 \end{aligned}$$

und im Fall $a_i = -1$ die vier Gleichungen

$$\begin{aligned} z^- \cdot p_{i,q} &= -1 \quad \text{für } q = 1, 2, 3 \text{ und} \\ z^+ \cdot p_{i,z} &= -1. \end{aligned}$$

Es folgen die drei Gleichungen

$$\begin{aligned} x_{j,i,1}^+ \cdot x_{j,i,1}^- &= -1 \\ x_{j,i,2}^+ \cdot x_{j,i,2}^- &= -1 \\ x_{j,i,3}^+ \cdot x_{j,i,3}^- &= -1 \end{aligned}$$

und schließlich drei Gleichungen der Form

$$z^+ \cdot z^- = -1.$$

Die für eine Gleichung $G_i \in \mathcal{G}$ erstellten 22 Gleichungen heißen Repräsentation dieser Gleichung.

Durch die letzten sechs Gleichungen wird sichergestellt, dass entweder $x_j^+ \neq x_j^-$ und $z^+ \neq z^-$ gilt, oder eine grosse Anzahl von Gleichungen nicht erfüllt ist. \mathcal{G}' enthält $3 \cdot t$ Gleichungen der Form $z^+ \cdot z^- = -1$. Sei n_j die Anzahl der Vorkommen von x_j in \mathcal{G} , dann enthält \mathcal{G}' n_j -mal die Gleichung $x_j^+ \cdot x_j^- = -1$.

Zunächst zwei Lemmata über die Anzahl der gleichzeitig erfüllbaren Gleichungen in \mathcal{G}' .

Lemma 6.4 *Sei eine Belegung der Variablen \mathcal{V} gegeben. Wir setzen $z^- = -1$ und $x_j^+ = (-x_j^-) = x_j$. Dann kann man es durch Belegen von $p_{i,1}$, $p_{i,2}$, $p_{i,3}$ und $p_{i,z}$ erreichen, dass 18 bzw. 16 Gleichungen der Repräsentation von G_i gleichzeitig erfüllt sind, falls G_i durch die Belegung der x_1, \dots, x_r erfüllt bzw. nicht erfüllt wird. Es ist in beiden Fällen nicht möglich, mehr Gleichungen der Repräsentation von G_i zu erfüllen.*

Beweis: Das Lemma kann durch Ausprobieren aller 16 möglichen Belegungen von $p_{i,1}$, $p_{i,2}$, $p_{i,3}$ und $p_{i,z}$ bewiesen werden.

Exemplarisch behandeln wir an dieser Stelle den Fall, dass eine Gleichung $x_{j,i,1} \cdot x_{j,i,2} \cdot x_{j,i,3} = 1$ durch die Belegung $x_{j,i,1} = x_{j,i,2} = x_{j,i,3} = 1$ erfüllt wird.

Wir setzen $p_{i,1} = p_{i,2} = p_{i,3} = p_{i,z} = -1$. Dann sind die sechs Gleichungen $x_{j,i,q}^+ \cdot p_{i,q'} = -1$ ($q \neq q'$ und $q, q' = 1, 2, 3$) erfüllt. Des Weiteren sind

die drei Gleichungen $x_{j_{i,q}}^+ \cdot p_{i,z} = -1$ ($q = 1, 2, 3$) erfüllt. Hinzu kommen die drei Gleichungen $z^+ \cdot p_{i,q}$ ($q = 1, 2, 3$). Schließlich sind sowohl die drei zu der Repräsentation gehörenden Gleichungen $z^+ \cdot z^- = -1$, als auch für $q = 1, 2, 3$ die Gleichung $x_{j_{i,q}}^+ \cdot x_{j_{i,q}}^- = -1$ ebenfalls erfüllt. Insgesamt werden durch die gewählte Belegung von $p_{i,1}, p_{i,2}, p_{i,3}, p_{i,z}$ 18 Gleichungen erfüllt. ■

Erfüllt eine Belegung von \mathcal{V} g der t Gleichungen aus \mathcal{G} , dann erfüllt die entsprechende variablen-konsistente Belegung von \mathcal{V}' bei geeigneter Belegung der $p_{i,1}, p_{i,2}, p_{i,3}$ und $p_{i,z}$ ($i = 1, \dots, t$) $16 \cdot t + 2 \cdot g$ Gleichungen von \mathcal{G}' . Werden umgekehrt durch eine variablen-konsistente Belegung von \mathcal{V}' $16 \cdot t + 2 \cdot g$ Gleichungen aus \mathcal{G}' erfüllt, so erfüllt die korrespondierende Belegung für \mathcal{V} g Gleichungen aus \mathcal{G} .

Lemma 6.5 *Aus einer beliebigen Belegung der Variablen \mathcal{V}' , die g' Gleichungen aus \mathcal{G} erfüllt, kann in polynomieller Zeit eine variablen-konsistente Belegung konstruiert werden, die mindestens g' Gleichungen erfüllt.*

Beweis: Angenommen, es gilt $z^+ = z^-$. Dann werden $3 \cdot t$ Gleichungen der Form $z^+ \cdot z^- = -1$ nicht erfüllt. Wir setzen $z^- = (-z^+)$. Dann werden alle diese $3 \cdot t$ Gleichungen zusätzlich erfüllt. Andererseits kommt z_j^- in höchstens $3 \cdot t$ weiteren Gleichungen vor. Es gibt also maximal $3 \cdot t$ Gleichungen, die nun nicht mehr erfüllt sind. Insgesamt verringert sich deshalb die Anzahl der erfüllten Gleichungen durch die Änderung nicht.

Nehmen wir nun an, für ein j gilt $x_j^+ = x_j^-$. Dann sind die n_j Gleichungen $x_j^+ \cdot x_j^- = -1$ nicht erfüllt. Wir setzen $x_j^- = (-x_j^+)$. Durch diese Veränderung der Belegung werden nun alle oben genannten n_j Gleichungen erfüllt. Andererseits kommt x_j^- in höchstens n_j weiteren Gleichungen vor. Es gibt also maximal n_j Gleichungen, die nun nicht mehr erfüllt sind, obwohl sie von der ursprünglichen Belegung erfüllt wurden. Insgesamt wird deshalb die Anzahl der erfüllten Gleichungen durch die Änderung nicht kleiner.

Auf diese Weise kann sukzessiv ein variablen-konsistentes Alignment konstruiert werden. Die Änderungen können offensichtlich in polynomieller Zeit durchgeführt werden. ■

Mit diesen Lemmata kann das folgende Theorem bewiesen werden.

Theorem 6.6 *Es ist für jedes $\epsilon > 0$ \mathcal{NP} -hart, $(\frac{18}{22} - \epsilon)$ - und $(\frac{17}{22} + \epsilon)$ -erfüllbare Instanzen von MAX-E2-neg-Lin2 zu unterscheiden.*

Beweis: Eine Instanz von MAX-E3-Lin2 mit t Gleichungen ist η -erfüllbar genau dann, wenn die zugehörige Instanz von MAX-E2-neg-Lin2 $\frac{(16+2\cdot\eta)\cdot t}{22\cdot t}$ -erfüllbar ist. Da es für jedes $\xi > 0$ \mathcal{NP} -hart ist, zwischen $(1 - \xi)$ - und

$(\frac{1}{2} + \xi)$ -erfüllbaren Instanzen von MAX-E3-Lin2 zu unterscheiden und MAX-E3-Lin2 in polynomieller Zeit auf MAX-E2-neg-Lin2 reduziert werden kann, ist es ebenfalls \mathcal{NP} -hart, $\frac{(16+2\cdot(1-\xi))\cdot t}{22\cdot t}$ - und $\frac{(16+2\cdot(\frac{1}{2}+\xi))\cdot t}{22\cdot t}$ -erfüllbare Instanzen von MAX-E2-neg-Lin2 zu unterscheiden. Wir setzen $\xi = 11 \cdot \epsilon$ und erhalten die Behauptung. ■

Aus Theorem 6.6 folgt sofort eine untere Schranke für die Approximierbarkeit von MAX-E2-neg-Lin2.

Korollar 6.7 *Unter der Annahme $\mathcal{NP} \neq \mathcal{P}$ gibt es keine $(\frac{18}{17} - \epsilon)$ -Approximation für MAX-E2-neg-Lin2.* ■

6.2 MAX-E2-neg-Lin2 ist MAX- \mathcal{SNP} -vollständig

Mittels Reduktion von MAX-Cut werden wir nun beweisen, dass MAX-E2-neg-Lin2 MAX- \mathcal{SNP} -vollständig ist.

Sei $G = (V, E)$ eine Instanz von MAX-Cut. Die Variablen $\mathcal{V}' = \{\check{v} | v \in V\}$ der zu konstruierenden Instanz von MAX-E2-neg-Lin2 ergeben sich aus V . Für jede Kante $\{u, v\} \in E$ erstellen wir eine Gleichung $\check{u} \cdot \check{v} = -1$, die Gleichungen sind also gegeben durch $\mathcal{G}' = \{\check{u} \cdot \check{v} = -1 | \{u, v\} \in E\}$.

Sei nun eine beliebige Belegung der Variablen gegeben. Wir interpretieren $\check{v} = 1$ als $v \in V_1$ und $\check{v} = -1$ als $v \in V_2$. Man erkennt, dass die Anzahl der erfüllten Gleichungen aus \mathcal{G}' gleich der Anzahl der Kanten zwischen V_1 und V_2 ist. Damit kann das folgende Theorem bewiesen werden.

Theorem 6.8 *MAX-E2-neg-Lin2 ist MAX- \mathcal{SNP} -vollständig.*

Beweis: In [32] wird gezeigt, dass MAX-Cut MAX- \mathcal{SNP} -vollständig ist. Wir zeigen $\text{MAX-Cut} \leq_L \text{MAX-E2-neg-Lin2}$.

Algorithmus f_1 (siehe Definition 5.9) ist durch die Konstruktion von \mathcal{G}' gegeben. Wähle $\gamma_1 = 1$, dann gilt nach Konstruktion die Ungleichung

$$\text{opt}(\mathcal{G}') \leq \gamma_1 \cdot \text{opt}(G).$$

Algorithmus f_2 erhält als Eingabe eine Belegung der Variablen \mathcal{V}' , die g der $|E|$ Gleichungen erfüllt, und erzeugt eine Partition V_0, V_1 , wie dies oben beschrieben wurde. Dann ist die Anzahl der Kanten zwischen V_0 und V_1 gerade g . Für $\gamma_2 = 1$ gilt also die Ungleichung

$$|g - \text{opt}(G)| \leq \gamma_2 \cdot |g - \text{opt}(\mathcal{G}')|$$

wegen $\text{opt}(G) = \text{opt}(G')$. Damit ist gezeigt, dass MAX-E2-neg-Lin2 MAX- \mathcal{SNP} -hart ist.

Es muss nun noch gezeigt werden, dass das Problem auch in MAX- \mathcal{SNP} enthalten ist. Dies ist der Fall, weil MAX-E2-neg-Lin2 sich in der Form

$$\max_P \{(j, k, i) | G(j, k, i) \wedge ((P(j) \wedge \neg P(k)) \vee (\neg P(j) \wedge P(k)))\}$$

beschreiben lässt. Dabei ist $G(j, k, i)$ genau dann wahr, wenn die Gleichung G_i die Form $x_j \cdot x_k = -1$ hat. $P(j)$ ist genau dann wahr, wenn $x_j = 1$ gesetzt wird. ■

6.3 MIN-BMSA ist MAX- \mathcal{SNP} -hart

In diesem Abschnitt werden wir MAX-E2-neg-Lin2 auf MIN-BMSA reduzieren, um zu zeigen, dass MIN-BMSA MAX- \mathcal{SNP} -hart ist. Sei dazu $\mathcal{G} = \{G_1, \dots, G_t\}$ eine Multimenge linearer Gleichungen über den Variablen $\mathcal{V} = \{x_1, \dots, x_r\}$, wobei die Gleichungen die Form $G_i \hat{=} x_{j_{i,1}} \cdot x_{j_{i,2}} = -1$ haben.

Aus \mathcal{G} konstruieren wir Sequenzen $\mathcal{S} = \{Z\} \cup \{X_j | j = 1, \dots, r\} \cup \{Y_{i,1}, Y_{i,2} | i = 1, \dots, t\}$ über dem Alphabet $\Sigma = \{\bullet, \circ, \times\}$. Sei

$$Z := \circ \circ \circ \circ \circ \circ \circ \circ.$$

Z hat die Länge 8 und wird als Kontrollsequenz verwendet. Seien

$$X_j := \bullet \circ \circ \circ \circ \circ \circ \circ \bullet$$

für $j = 1, \dots, r$. X_j hat die Länge 9 und repräsentiert den Wert der Variablen $x_j \in \mathcal{V}$. Wir konstruieren jeweils Sequenzen $Y_{i,1}$ und $Y_{i,2}$ für $i = 1, \dots, t$, wiederum mit der Länge 9.

$$\begin{aligned} Y_{i,1} &:= \bullet \circ \circ \times \circ \times \circ \circ \bullet \\ Y_{i,2} &:= \bullet \circ \circ \circ \times \circ \circ \circ \bullet \end{aligned}$$

$Y_{i,1}$ repräsentiert den Wert der ersten, $Y_{i,2}$ den Wert der zweiten Variable der Gleichung $G_i \in \mathcal{G}$.

Die Kostenfunktion ist wie folgt gegeben:

	-	•	◦	×
-	0	1	2	5
•	1	0	1	4
◦	2	1	0	3
×	5	4	3	0

Ferner sei die Gewichtsmatrix $W = (w_{I,J})_{I,J \in \mathcal{S}}$ gegeben durch

$$w_{I,J} := \begin{cases} 1 & \text{falls } I = Y_{i,1} \text{ und } J = Y_{i,2} \text{ oder umgekehrt,} \\ 1 & \text{falls } I = Z \text{ und } J = Y_{i,q} \text{ oder umgekehrt } (q = 1, 2), \\ 1 & \text{falls } I = Y_{i,q} \text{ und } J = X_{j_i,q} \text{ oder umgekehrt } (q = 1, 2), \\ 0 & \text{sonst.} \end{cases}$$

Die Menge $\mathcal{S}_i := \{Y_{i,1}, Y_{i,2}, X_{j_{i,1}}, X_{j_{i,2}}\}$ heißt Repräsentation der Gleichung G_i . Man beachte, dass eine Sequenz X_j im Allgemeinen in mehreren Repräsentationen enthalten ist. Ist x_j die q -te Variable in der Gleichung G_i , so sagen wir, dass $Y_{i,q}$ zu x_j gehört.

Für ein Alignment $\mathcal{A} = \{\tilde{S} | S \in \mathcal{S}\}$ von \mathcal{S} bezeichnet $D_i(\mathcal{A})$ die Kosten der Gleichung G_i , d.h. die gewichteten Kosten des durch \mathcal{A} induzierten Alignments von $\mathcal{S}_i \cup \{Z\}$. $D_i(\mathcal{A})$ ist gegeben durch

$$D_i(\mathcal{A}) = D(\tilde{Y}_{i,1}, \tilde{X}_{j_{i,1}}) + D(\tilde{Y}_{i,2}, \tilde{X}_{j_{i,2}}) + D(\tilde{Y}_{i,1}, \tilde{Y}_{i,2}) + D(\tilde{Y}_{i,1}, \tilde{Z}) + D(\tilde{Y}_{i,2}, \tilde{Z}).$$

Wie man erkennt, gilt aufgrund der Struktur der Gewichtsmatrix

$$D_W(\mathcal{A}) = \sum_{i=1}^t D_W(\mathcal{S}_i \cup \{Z\} | \mathcal{A}) = \sum_{i=1}^t D_i(\mathcal{A}).$$

Die Sequenzen \mathcal{S} und die Gewichtsmatrix W können in polynomieller Zeit erstellt werden.

Definition 6.9 (variablen-konsistentes Alignment) *Ein Alignment $\mathcal{A} = \{\tilde{S} | S \in \mathcal{S}\}$ von \mathcal{S} heißt variablen-konsistent bezüglich einer Belegung der Variablen \mathcal{V} , falls nach Eliminierung aller Gap-Spalten gilt:*

- (i) $\tilde{Z} = -Z-$
- (ii) $\tilde{X}_j = \begin{cases} X_j- & \text{falls } x_j = -1 \\ -X_j & \text{falls } x_j = 1 \end{cases}$
- (iii) $\tilde{Y}_{i,q} = \begin{cases} Y_{i,q}- & \text{falls } x_{j_{i,q}} = -1 \\ -Y_{i,q} & \text{falls } x_{j_{i,q}} = 1 \end{cases}$

Ein Alignment heißt variablen-konsistent, falls eine Belegung existiert, so dass \mathcal{A} variablen-konsistent bezüglich dieser Belegung ist.

Variablen-konsistente Alignments können wie folgt einfacher charakterisiert werden.

Fakt 6.10 Ein Alignment ist genau dann variablen-konsistent, wenn für alle $i = 1, \dots, t$ und $q = 1, 2$ Folgendes gilt:

(E1) Es korrespondiert entweder nur $Y_{i,q}[1]$ oder nur $Y_{i,q}[9]$ mit einem Gap in Z und

(E2) es korrespondiert kein Zeichen von $Y_{i,q}$ mit einem Gap in $X_{j_{i,q}}$.

Beispiel 6.11 Sei $\mathcal{G} = \{G_1\}$ und $G_1 \hat{=} x_1 \cdot x_2 = -1$. G_1 ist für $x_1 = -1$ und $x_2 = 1$ erfüllt. Das zugehörige variablen-konsistente Alignment sieht folgendermaßen aus:

$$\begin{array}{rcccccccccccc}
 \tilde{Y}_{1,1} & = & \bullet & \circ & \circ & \times & \circ & \times & \circ & \circ & \bullet & - \\
 \tilde{Y}_{1,2} & = & - & \bullet & \circ & \circ & \times & \circ & \circ & \circ & \bullet & \\
 \hline
 \tilde{X}_1 & = & \bullet & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \bullet & - \\
 \tilde{X}_2 & = & - & \bullet & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \bullet \\
 \hline
 \tilde{Z} & = & - & \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ & -
 \end{array}$$

Zu beachten ist die funktionale Region des Paares $\{Y_{i,1}, Y_{i,2}\}$, die in dem obigen Diagramm grau unterlegt ist. Repräsentieren $Y_{i,1}$ und $Y_{i,2}$ den gleichen Wert, dann ist $x_{j_{i,1}} \cdot x_{j_{i,2}} = 1$, d.h. die Gleichung G_i wird durch die Belegung nicht erfüllt. In diesem Fall liefert die funktionale Region Kosten 9:

$$\begin{array}{c}
 \times \circ \times \\
 \circ \times \circ
 \end{array}$$

Repräsentieren $Y_{i,1}$ und $Y_{i,2}$ verschiedene Werte, wird also G_i durch die Variablenbelegung erfüllt, so liefert diese Region Kosten 3:

$$\begin{array}{c}
 \times \circ \times \\
 \circ \times \circ
 \end{array}$$

Insgesamt liefert ein Paar $\{Y_{i,1}, Y_{i,2}\}$ Kosten 9, falls $Y_{i,1}$ und $Y_{i,2}$ den gleichen Wert repräsentieren und Kosten 7, falls sie unterschiedliche Werte repräsentieren.

$Y_{i,1}$ enthält zwei “ \times ”, während $Y_{i,2}$ nur ein “ \times ” enthält. Daher liefert ein variablen-konsistentes Alignment Kosten 8 für jedes Paar $\{Y_{i,1}, Z\}$ und 5 für jedes Paar $\{Y_{i,2}, Z\}$. Ein Paar $\{Y_{i,1}, X_{j_{i,1}}\}$ liefert Kosten 6, während ein Paar $\{Y_{i,2}, X_{j_{i,2}}\}$ nur Kosten 3 liefert.

Sei eine Belegung gegeben, die g der t Gleichungen erfüllt. Ein Alignment \mathcal{A} , das variablen-konsistent bezüglich dieser Belegung ist, liefert Kosten in

Höhe von

$$\begin{aligned}
& \underbrace{8 \cdot t}_{t \text{ Paare } \{Y_{i,1}, Z\}} + \underbrace{5 \cdot t}_{t \text{ Paare } \{Y_{i,2}, Z\}} + \underbrace{6 \cdot t}_{t \text{ Paare } \{Y_{i,1}, X_{j_{i,1}}\}} + \underbrace{3 \cdot t}_{t \text{ Paare } \{Y_{i,2}, X_{j_{i,2}}\}} + \\
& \underbrace{9 \cdot (t - g)}_{\text{nicht erfüllte Gleichungen}} + \underbrace{7 \cdot g}_{\text{erfüllte Gleichungen}} \\
= & 31 \cdot t - 2 \cdot g .
\end{aligned}$$

Es ist also $D_i(\mathcal{A}) = 29$, falls G_i durch die Belegung erfüllt wird, und $D_i(\mathcal{A}) = 31$, falls die Gleichung nicht erfüllt wird. Umgekehrt gilt, dass die durch ein variablen-konsistentes Alignment mit Kosten $31 \cdot t - 2 \cdot g$ repräsentierte Belegung g Gleichungen erfüllt.

Die folgenden beiden Lemmata werden für den Beweis von Lemma 6.14 benötigt, in dem gezeigt wird, dass aus einem Alignment mit Kosten $31 \cdot t - 2 \cdot g$ effizient ein variablen-konsistentes Alignment mit höchstens diesen Kosten berechnet werden kann.

Lemma 6.12 *Ein Paar $\{Y_{i,1}, Z\}$ bzw. $\{Y_{i,2}, Z\}$ liefert gewichtete Kosten 8 bzw. 5, falls (E1) erfüllt ist. Jedes Alignment von $\{Y_{i,1}, Z\}$ bzw. $\{Y_{i,2}, Z\}$, das (E1) nicht erfüllt, liefert mindestens Kosten 10 bzw. 7.*

Beweis: Ein Alignment des Paares $\{Y_{i,1}, Z\}$ bzw. $\{Y_{i,2}, Z\}$, das (E1) genügt, liefert offensichtlich Kosten 8 bzw. 5.

Wir betrachten nun ein Alignment von $\{Y_{i,1}, Z\}$, das (E1) nicht erfüllt. Dann gibt es zwei Fälle. Im ersten Fall korrespondiert weder $Y_{i,1}[1]$ noch $Y_{i,1}[9]$ mit einem Gap in Z . Dann muss mindestens eines der übrigen Zeichen $Y_{i,1}[2], \dots, Y_{i,1}[8]$ von $Y_{i,1}$ mit einem Gap in Z korrespondieren. Im zweiten Fall korrespondiert mindestens eines der Zeichen $Y_{i,1}[1]$ und $Y_{i,1}[9]$ mit einem Gap in Z . Da (E1) nicht erfüllt ist, gibt es mindestens ein Zeichen in Z , das mit einem Gap in $Y_{i,1}$ korrespondiert. Insgesamt korrespondiert also mindestens eines der Zeichen $Y_{i,1}[2], \dots, Y_{i,1}[8], Z[1], \dots, Z[8]$ mit einem Gap in der jeweils anderen Sequenz.

Wir unterscheiden wiederum zwei Fälle. Korrespondiert ein “×” aus $Y_{i,1}$ mit einem Gap in Z , so hat das Alignment mindestens Kosten 5 für dieses “×” plus 3 für das andere “×” plus jeweils 1 für die beiden “•”, also insgesamt mindestens Kosten 10. Anderenfalls korrespondiert ein “o” mit einem Gap in der anderen Sequenz. Das Alignment hat dann mindestens Kosten 2 für dieses “o” plus jeweils 3 für die beiden “×” plus jeweils 1 für die beiden “•”, also insgesamt auch mindestens Kosten 10.

Die Argumentation für ein Alignment von $\{Y_{i,2}, Z\}$ ist ähnlich; der einzige Unterschied ist, dass nur ein “×” in $Y_{i,2}$ zu beachten ist. ■

Lemma 6.13 *Ein Alignment des Paares $\{Y_{i,1}, X_{j_{i,1}}\}$ bzw. $\{Y_{i,2}, X_{j_{i,2}}\}$ liefert Kosten 6 bzw. 3, falls (E2) erfüllt ist.*

Jedes Alignment von $\{Y_{i,1}, X_{j_{i,1}}\}$ bzw. $\{Y_{i,2}, X_{j_{i,2}}\}$, das (E2) nicht erfüllt, liefert mindestens Kosten 8 bzw. 5.

Beweis: Ein Alignment eines Paares $\{Y_{i,1}, X_{j_{i,1}}\}$ bzw. $\{Y_{i,2}, X_{j_{i,2}}\}$, das (E2) genügt, liefert offensichtlich Kosten 6 bzw. 3.

Wir betrachten nun zunächst den Fall, dass ein Alignment von $\{Y_{i,1}, X_{j_{i,1}}\}$ (E2) nicht erfüllt. Dann korrespondiert mindestens ein Zeichen aus $Y_{i,1}$ mit einem Gap in $X_{j_{i,1}}$ und mindestens ein Zeichen aus $X_{j_{i,1}}$ mit einem Gap aus $Y_{i,1}$.

Wir unterscheiden zwei Fälle. Korrespondiert ein “×” aus $Y_{i,1}$ mit einem Gap in $X_{j_{i,1}}$, so liefert das Alignment mindestens Kosten 5 für dieses “×” und mindestens Kosten 3 für das andere “×”. Insgesamt ergeben sich also mindestens Kosten 8.

Korrespondiert kein “×” aus $Y_{i,1}$ mit einem Gap in $X_{j_{i,1}}$, so liefert zum einen das Zeichen aus $Y_{i,1}$, das mit einem Gap in $X_{j_{i,1}}$ korrespondiert, mindestens Kosten 1, zum anderen liefert das Zeichen aus $X_{j_{i,1}}$, das mit einem Gap in $Y_{i,1}$ korrespondiert, ebenfalls mindestens Kosten 1. Hinzu kommen jeweils Kosten 3 für die beiden “×”, also insgesamt wieder mindestens Kosten 8.

Die Argumentation für ein Alignment von $\{Y_{i,2}, X_{j_{i,2}}\}$ verläuft analog, es ist jedoch nur ein “×” in $Y_{i,2}$ zu berücksichtigen. ■

Man kann leicht nachrechnen, dass ein optimales Alignment von $Y_{i,1}$ und $Y_{i,2}$ Kosten 7 hat.

Das folgende Lemma sagt aus, dass aus einem beliebigen Alignment effizient ein variablen-konsistentes Alignment mit kleineren oder gleichen Kosten berechnet werden kann. Dazu wird die Tatsache verwendet, dass Alignments, bei denen Paare von Sequenzen gegen (E1) oder (E2) verstoßen, wegen Lemma 6.12 und Lemma 6.13 deutlich höhere Kosten liefern.

Lemma 6.14 *Aus einem beliebigen Alignment \mathcal{A} kann in polynomieller Zeit ein variablen-konsistentes Alignment \mathcal{A}' mit $D_W(\mathcal{A}') \leq D_W(\mathcal{A})$ konstruiert werden.*

Beweis: Sei \mathcal{A} ein beliebiges Alignment von \mathcal{S} .

Sei I die Menge aller der Indizes i , für die alle Sequenzen aus \mathcal{S}_i (E1) bzw. (E2) genügen. Dies impliziert eine Belegung der Variablen $\mathcal{V}_I := \{x_j | \exists i \in I : X_j \in \mathcal{S}_i\} \subseteq \mathcal{V}$. Sei $\bar{I} = \{1, \dots, n\} \setminus I$. Da es für alle $i \in \bar{I}$ in \mathcal{S}_i eine Sequenz $Y_{i,q}$ gibt, die (E1) oder (E2) nicht genügt, gilt für alle $i \in \bar{I}$ wegen Lemma 6.12 und Lemma 6.13 $D_i(\mathcal{A}) \geq 31$.

Wir richten für alle $i \in \bar{I}$ alle $Y_{i,q}$, die zu einer Variable $x_j \in \mathcal{V}_I$ gehören, entsprechend der Belegung von x_j aus.

Nun weisen wir allen Variablen aus $\bar{\mathcal{V}}_I = \mathcal{V} \setminus \mathcal{V}_I$ einen beliebigen Wert zu und richten die entsprechenden X_j und $Y_{i,q}$ danach aus.

Sei \mathcal{A}' das durch diese Transformationen erhaltene Alignment. Dann gilt für $i \in I$ $D_i(\mathcal{A}') = D_i(\mathcal{A})$ und für $i \in \bar{I}$ $D_i(\mathcal{A}') \leq 31 \leq D_i(\mathcal{A})$. Damit gilt $D_W(\mathcal{A}') \leq D_W(\mathcal{A})$.

\mathcal{A}' ist nach Konstruktion variablen-konsistent und kann in polynomieller Zeit aus \mathcal{A} berechnet werden. \blacksquare

Damit kann nun die MAX- \mathcal{SNP} -Härte von MIN-BMSA bewiesen werden.

Theorem 6.15 MIN-BMSA ist MAX- \mathcal{SNP} -hart.

Beweis: Wir zeigen $\text{MAX-E2-neg-Lin2} \leq_L \text{MIN-BMSA}$.

Algorithmus f_1 (siehe Definition 5.9) ist gegeben durch die Konstruktion der Sequenzen \mathcal{S} aus den Gleichungen \mathcal{G} .

Eine Gleichung aus \mathcal{G} wird von 2 der 4 möglichen Variablenbelegungen erfüllt. Eine zufälle Belegung der Variablen \mathcal{V} erfüllt daher im Mittel die Hälfte der Gleichungen aus \mathcal{G} . Deshalb gibt es für jede Multimenge \mathcal{G} von t Gleichungen eine Belegung, die $\frac{t}{2}$ Gleichungen erfüllt. Es ist also $\text{opt}(\mathcal{G}) \geq \frac{t}{2}$. Andererseits kann immer ein Alignment von \mathcal{S} mit gewichteten Kosten $31 \cdot t$ gefunden werden, daher ist $\text{opt}(\mathcal{S}) \leq 31 \cdot t$. Wir setzen $\gamma_1 = 62$, dann gilt die Ungleichung

$$\text{opt}(\mathcal{S}) \leq 31 \cdot t = \gamma_1 \cdot \frac{t}{2} \leq \gamma_1 \cdot \text{opt}(\mathcal{G}).$$

Algorithmus f_2 ist durch die Transformation aus Lemma 6.14 gegeben. Er konstruiert aus einem Alignment \mathcal{A} von \mathcal{S} mit Kosten $31 \cdot t - 2 \cdot \tilde{g}$ in polynomieller Zeit eine Belegung für \mathcal{V} , indem er aus \mathcal{A} ein Alignment \mathcal{A}' mit $D_W(\mathcal{A}') = 31 \cdot t - 2 \cdot g \leq 31 \cdot t - 2 \cdot \tilde{g} = D_W(\mathcal{A})$ konstruiert, das variablen-konsistent bezüglich dieser Belegung ist. Aufgrund der Eigenschaften variablen-konsistenter Alignments erfüllt die Belegung $g \geq \tilde{g}$ Gleichungen. Wir setzen $\gamma_2 = \frac{1}{2}$, dann gilt

$$\begin{aligned} |g - \text{opt}(\mathcal{G})| &= \gamma_2 \cdot |2 \cdot g - 31 \cdot t - (2 \cdot \text{opt}(\mathcal{G}) - 31 \cdot t)| \\ &= \gamma_2 \cdot |31 \cdot t - 2 \cdot g - \text{opt}(\mathcal{S})| \\ &= \gamma_2 \cdot |D_W(\mathcal{A}') - \text{opt}(\mathcal{S})| \\ &\leq \gamma_2 \cdot |D_W(\mathcal{A}) - \text{opt}(\mathcal{S})|. \end{aligned}$$

Damit ist $\text{MAX-E2-neg-Lin2} \leq_L \text{MIN-BMSA}$ bewiesen. Mit Theorem 6.8 ergibt sich die Behauptung. \blacksquare

Mit der beschriebenen Konstruktion kann auch eine untere Schranke für die Approximierbarkeit von MIN-BMSA bewiesen werden.

Theorem 6.16 *Unter der Annahme $\mathcal{NP} \neq \mathcal{P}$ gibt es für beliebige $\epsilon > 0$ keine $(\frac{324}{323} - \epsilon)$ -Approximation für MIN-BMSA.*

Beweis: Eine Instanz von MAX-E2-neg-Lin2, die aus t Klauseln besteht, ist η -erfüllbar genau dann, wenn die zugehörige Instanz von MIN-BMSA ein Alignment mit Kosten $(31 - 2 \cdot \eta) \cdot t$ besitzt.

Angenommen, MIN-BMSA besitzt eine $(\frac{324}{323} - \epsilon)$ -Approximation für ein $\epsilon > 0$. Sei eine $(\frac{18}{22} - \xi)$ -erfüllbare Instanz von MAX-E2-neg-Lin2 gegeben. Dann hat das optimale Alignment der zugehörigen Instanz von MIN-BMSA Kosten $(31 - 2 \cdot (\frac{18}{22} - \xi)) \cdot t = \frac{323 + \xi}{11} \cdot t$ und es kann mittels der $(\frac{324}{323} - \epsilon)$ -Approximation immer ein Alignment berechnet werden, das höchstens Kosten $(\frac{324}{323} - \epsilon) \cdot \frac{323 + \xi}{11} \cdot t =: K_1$ besitzt. Dabei ist $\tilde{\xi} = 22 \cdot \xi$.

Wir betrachten andererseits eine $\frac{17}{22} + \xi$ -erfüllbare Instanz von MAX-E2-neg-Lin2. Jedes Alignment der zugehörigen Instanz von MIN-BMSA hat mindestens Kosten $(31 - 2 \cdot (\frac{17}{22} + \xi)) \cdot t = \frac{324 - \tilde{\xi}}{11} \cdot t =: K_2$.

Es gilt nun

$$\begin{aligned} K_1 &< K_2 \\ \Leftrightarrow \left(\frac{324}{323} - \epsilon\right) \cdot \frac{323 + \xi}{11} \cdot t &< \frac{324 - \tilde{\xi}}{11} \cdot t \\ \Leftrightarrow \xi = \frac{1}{22} \cdot \tilde{\xi} &< \frac{1}{22} \cdot \frac{323^2 \cdot \epsilon}{647 - 323 \cdot \epsilon}. \end{aligned}$$

Wir wählen ξ beliebig mit $0 < \xi < \frac{1}{22} \cdot \frac{323^2 \cdot \epsilon}{647 - 323 \cdot \epsilon}$. Dann können mit Hilfe der $(\frac{324}{323} - \epsilon)$ -Approximation für MIN-BMSA $(\frac{18}{22} - \xi)$ - und $(\frac{17}{22} + \xi)$ -erfüllbare Instanzen von MAX-E2-neg-Lin2 unterschieden werden. Hieraus folgt nach Theorem 6.6 $\mathcal{NP} = \mathcal{P}$. ■

Da MIN-WMSA eine Verallgemeinerung von MIN-BMSA ist, ergeben sich sofort die folgenden beiden Korollare.

Korollar 6.17 *MIN-WMSA ist MAX- \mathcal{SNP} -hart.* ■

Korollar 6.18 *Unter der Annahme $\mathcal{NP} \neq \mathcal{P}$ gibt es für beliebige $\epsilon > 0$ keine $(\frac{324}{323} - \epsilon)$ -Approximation für MIN-WMSA.* ■

7 Reduktion von MIN-WMSA auf MIN-BMSA

In diesem Abschnitt werden wir zeigen, dass MIN-WMSA und MIN-BMSA gleiche Approximierbarkeitseigenschaften besitzen, d.h. eine λ -Approximation für MIN-BMSA liefert eine λ -Approximation für MIN-WMSA und eine untere Schranke für die Approximierbarkeit von MIN-WMSA ist auch eine untere Schranke für die Approximierbarkeit von MIN-BMSA. Die Umkehrungen dieser Eigenschaften gelten auch, da MIN-WMSA eine Verallgemeinerung von MIN-BMSA ist.

Wir zeigen nun, wie zu einer Instanz von MIN-WMSA eine Instanz von MIN-BMSA konstruiert werden kann. Seien dazu $\mathcal{S} = \{S_1, \dots, S_n\}$ Sequenzen über Σ und $W = (w_{S_j, S_k})_{j,k=1}^n = (w_{j,k})_{j,k=1}^n$ eine Gewichtsmatrix. Sei l das Maximum der Sequenzlängen und d_{\max} das Maximum der Kostenfunktion d .

Wir konstruieren eine Menge von Sequenzen \mathcal{S}' wie folgt. Sei $K := 2 \cdot d_{\max} \cdot l$. Für jede Sequenz $S_j \in \mathcal{S}$ erstellen wir K Kopien T_j^ν ($\nu = 1, \dots, K$) dieser Sequenz. Erstelle weiterhin für $k = 1, \dots, n$ je $w_{j,k}$ Kopien $S_j^{k,\theta}$ ($\theta = 1, \dots, w_{j,k}$).

Die Gewichtsmatrix $W' = (w'_{I,J})_{I,J \in \mathcal{S}'}$ ist gegeben durch

$$w'_{IJ} := \begin{cases} 1 & \text{falls } I = S_j^{k,\theta} \text{ und } J = S_k^{j,\theta}, \\ 1 & \text{falls } I = S_j^{k,\theta} \text{ und } J = T_j^\nu \text{ oder umgekehrt,} \\ 0 & \text{sonst.} \end{cases}$$

Die Eingabelänge N der Instanz von MIN-WMSA erfüllt die Ungleichung

$$N \geq \Omega \left(n \cdot l + \sum_{j,k=1}^n w_{j,k} + \sum_{x,y \in \Sigma} d(x,y) \right).$$

Andererseits kann die Eingabelänge N' der konstruierten Instanz von MIN-BMSA durch

$$N' \leq O \left(\underbrace{2 \cdot d_{\max} \cdot n \cdot l^2}_{T'} + \underbrace{l \cdot \sum_{j,k=1}^n w_{j,k}}_{S'} + \underbrace{\left(2 \cdot d_{\max} \cdot n \cdot l + \sum_{j,k=1}^n w_{j,k} \right)^2}_{W'} \right)$$

abgeschätzt werden. N' ist also nur polynomiell größer als N .

Für die Beweise der nächsten beiden Lemmata benötigen wir die folgende Definition.

Definition 7.1 (Konsistentes Alignment, Block) Sei $\mathcal{A}' = \{\tilde{I} | I \in \mathcal{S}'\}$ ein beliebiges Alignment von \mathcal{S}' . Dann heißt ein induziertes Alignment $\mathcal{A}'_j = \{\tilde{I} | I = S_j^{k,\theta} \text{ oder } I = T_j^\nu\}$ von \mathcal{A}' konsistent mit Block B_j , falls für alle $\tilde{I} \in \mathcal{A}'_j$ $\tilde{I} = B_j$ gilt.

Sind alle \mathcal{A}'_j , $j = 1, \dots, n$, konsistent mit Block B_j , dann heißt \mathcal{A}' konsistent. Ein konsistentes Alignment mit Blöcken B_1, \dots, B_n induziert ein Alignment $\mathcal{B} = \{B_1, \dots, B_n\}$ von \mathcal{S} .

Lemma 7.2 Mit den genannten Bezeichnungen gilt

$$D_{W'}(\mathcal{S}') \leq D_W(\mathcal{S}).$$

Beweis: Wir zeigen, dass es zu jedem Alignment \mathcal{A} von \mathcal{S} ein Alignment \mathcal{A}' von \mathcal{S}' gibt, so dass $D_{W'}(\mathcal{A}') = D_W(\mathcal{A})$.

Sei $\mathcal{A} = \{A_1, \dots, A_n\}$ ein beliebiges Alignment von \mathcal{S} mit Kosten $D_W(\mathcal{A})$. Dann sei $\mathcal{A}' = \{\tilde{I} | I \in \mathcal{S}'\}$ ein konsistentes Alignment von \mathcal{S}' mit Blöcken A_1, \dots, A_n . Für die Kosten von \mathcal{A}' gilt

$$\begin{aligned} D_{W'}(\mathcal{A}') &= \sum_{j=1}^n \sum_{k=1}^n \sum_{\theta=1}^{w_{j,k}} \sum_{\nu=1}^K \underbrace{D(\tilde{S}_j^{k,\theta}, \tilde{T}_j^\nu)}_{=0} + \sum_{1 \leq j < k \leq n} \overbrace{\sum_{\theta=1}^{w_{j,k}} D(\tilde{S}_j^{k,\theta}, \tilde{S}_k^{j,\theta})}^{w_{j,k} \cdot D(A_j, A_k)} \\ &= \sum_{1 \leq j < k \leq n} w_{j,k} \cdot D(A_j, A_k) = D_W(\mathcal{A}). \end{aligned}$$

■

Lemma 7.3 Sei \mathcal{A}' ein Alignment von \mathcal{S}' . Dann kann in polynomieller Zeit ein Alignment \mathcal{B} von \mathcal{S} mit $D_W(\mathcal{B}) \leq D_{W'}(\mathcal{A}')$ berechnet werden.

Beweis: Sei $\mathcal{A}' = \{\tilde{I} | I \in \mathcal{S}'\}$ ein beliebiges Alignment von \mathcal{S}' mit Kosten $D_{W'}(\mathcal{A}')$.

Wir betrachten den Fall, dass für ein \hat{j} $\mathcal{A}'_{\hat{j}}$ nicht konsistent ist. Wir unterscheiden zwei Fälle. Sind nicht alle $\tilde{T}_{\hat{j}}^\nu$ gleich, so sei für $\nu = 1, \dots, K$

$$D_\nu = \sum_{k=1}^n \sum_{\theta=1}^{w_{\hat{j},k}} D(\tilde{S}_{\hat{j}}^{k,\theta}, \tilde{T}_{\hat{j}}^\nu).$$

Wir wählen $\hat{\nu}$ so, dass $D_{\hat{\nu}}$ minimal unter allen D_ν ist, und setzen $\tilde{T}_{\hat{j}}^{\hat{\nu}} = \tilde{T}_{\hat{j}}^{\hat{\nu}}$ für alle $\nu \neq \hat{\nu}$. Das so konstruierte Alignment hat gleiche oder geringere Kosten.

Wir betrachten nun den Fall, dass $\tilde{T}_j^\nu = B_j$ für alle $\nu = 1, \dots, K$ ist. Dann gibt es eine Sequenz $\tilde{S}_j^{k,\theta} \neq B_j$. Diese Sequenz liefert mindestens Kosten 1 mit jeder Sequenz \tilde{T}_j^ν aufgrund der Definition der Kostenfunktion, also insgesamt mindestens Kosten K . Wir setzen $\tilde{S}_j^{k,\theta} = B_j$. Dann liefert $\tilde{S}_j^{k,\theta}$ Kosten 0 mit den Sequenzen \tilde{T}_j^ν und maximal Kosten K mit der Sequenz $\tilde{S}_k^{j,\theta}$. Das so konstruierte Alignment hat also gleiche oder geringere Kosten.

Auf diese Weise erhalten wir ein Alignment, das konsistent mit Blöcken B_1, \dots, B_n ist. Das induzierte Alignment $\mathcal{B} = \{B_1, \dots, B_n\}$ von \mathcal{S} hat Kosten

$$D(\mathcal{B}) = \sum_{1 \leq j < k \leq n} w_{j,k} \cdot D(B_j, B_k) = \sum_{1 \leq j < k \leq n} \sum_{\theta=1}^{w_{j,k}} D(\tilde{S}_j^{k,\theta}, \tilde{S}_k^{j,\theta}) \leq D_{W'}(\mathcal{A}).$$

■

Mit dieser Konstruktion kann nun das folgende Theorem bewiesen werden.

Theorem 7.4 *Ist MIN-BMSA λ -approximierbar für eine Konstante $\lambda > 1$, dann ist auch MIN-WMSA λ -approximierbar.*

Beweis: Sei (\mathcal{S}, W) eine Instanz für MIN-WMSA. Wir erstellen (\mathcal{S}', W') wie beschrieben. Dann gilt nach Lemma 7.2 $D_{W'}(\mathcal{S}') \leq D_W(\mathcal{S})$. Da MIN-BMSA λ -approximiert werden kann, kann in polynomieller Zeit ein Alignment \mathcal{A}' von \mathcal{S}' mit $D_{W'}(\mathcal{A}') \leq \lambda \cdot D_{W'}(\mathcal{S}')$ berechnet werden. Nach Lemma 7.3 kann aus \mathcal{A}' ein Alignment \mathcal{A} berechnet werden, so dass gilt

$$D_W(\mathcal{A}) \leq D_{W'}(\mathcal{A}') \leq \lambda \cdot D_{W'}(\mathcal{S}') \leq \lambda \cdot D_W(\mathcal{S}).$$

Daraus folgt die Behauptung. ■

Aus Theorem 7.4 folgt umgekehrt, dass eine untere Schranke für die Approximierbarkeit von MIN-WMSA bereits eine untere Schranke für die Approximierbarkeit von MIN-BMSA darstellt.

8 Zusammenfassung

In dieser Arbeit wurde eine untere Schranke für die Approximierbarkeit des gewichteten Multiple-Sequence-Alignment-Problems (MIN-WMSA) bewiesen, einer Verallgemeinerung des Multiple-Sequence-Alignment-Problems. Des Weiteren wurde gezeigt, dass dieses Problem MAX- \mathcal{SNP} -hart ist. Diese Resultate gelten sogar für das binär gewichtete Multiple-Sequence-Alignment-Problem (MIN-BMSA), bei dem die Gewichte auf 0 und 1 beschränkt sind.

Es wurde auch gezeigt, dass jede untere Schranke für die Approximierbarkeit von MIN-WMSA bereits eine untere Schranke für die Approximierbarkeit von MIN-BMSA ist.

Eine offene Frage ist aber, ob diese beiden Probleme in \mathcal{APX} liegen, also eine Approximation mit konstanter Güte besitzen. Mittels der Reduktion von MIN-WMSA auf MIN-BMSA wurde gezeigt, dass entweder beide Probleme in \mathcal{APX} liegen oder keines der beiden. Die hier bewiesene untere Schranke für die Approximierbarkeit von MIN-WMSA und MIN-BMSA ist allerdings sehr weit von der besten bisher bekannten oberen Schranke für die Approximierbarkeit von MIN-WMSA von $O(\log^2 n)$ entfernt. Ein offensichtliches Ziel ist es, diese Lücke zu verkleinern.

Interessant ist jedoch auch die Approximierbarkeit des ungewichteten Multiple-Sequence-Alignment-Problems. Für dieses Problem ist bislang nur eine obere Schranke bekannt (siehe Kapitel 4). Die Approximierbarkeit des Multiple-Sequence-Alignment-Problems bezeichnen Jiang und andere in [24] als eines der großen offenen Probleme der Bioinformatik.

Literatur

- [1] ALTSCHUL, S. F. und B. W. ERICKSON: *Locally Optimal Subalignments Using Nonlinear Similarity Functions*. Bulletin of Mathematical Biology, 48(5/6):633–660, 1986.
- [2] ALTSCHUL, S. F. und D. J. LIPMAN: *Trees, Stars, and Multiple Biological Sequence Alignment*. SIAM Journal on Applied Mathematics, 49:197–209, 1989.
- [3] ARORA, S.: *Polynomial Time Approximation Schemes for Euclidean TSP and other Geometric Problems*. In: *37th IEEE Annual Symposium on Foundations of Computer Science*, Seiten 2–11, 1996.
- [4] ARORA, S., C. LUND, R. MOTWANI, M. SUDAN und M. SZEGEDY: *Proof Verification and the Hardness of Approximation Problems*. Journal of the ACM, 45(3):501–555, 1998.
- [5] AUSIELLO, G., P. CRESCENZI, G. GAMBOSI, V. KANN, A. MARCHETTI-SPACCAMELA und M. PROTASI: *Complexity and Approximation*. Springer, 1999.
- [6] BAFNA, V., E. L. LAWLER und P. A. PEVZNER: *Approximation Algorithms for Multiple Sequence Alignment*. Theoretical Computer Science, 182(1–2):233–244, 1997.
- [7] BALDI, P. und S. BRUNAK: *Bioinformatics: The Machine Learning Approach*. MIT Press, 1998.
- [8] CARRILLO, H. und D. J. LIPMAN: *The Multiple Sequence Alignment Problem in Biology*. SIAM Journal on Applied Mathematics, 48:1073–1082, 1988.
- [9] CHAN, S. C., A. K. C. WONG und D. K. Y. CHIU: *A Survey of Multiple Sequence Comparison Methods*. Bulletin of Mathematical Biology, 54(4):563–598, 1992.
- [10] CHRISTOFIDES, N.: *Worst-Case Analysis of a New Heuristic for the Traveling Salesman Problem*. Technischer Bericht, Graduate School of Industrial Administration, Carnegie–Mellon University, 1976.
- [11] CORMAN, T. H., C. E. LEISERSON und RONALD L. RIVEST: *Introduction to Algorithms*. MIT Press, 1990.

- [12] CRESCENZI, P. und L. TREVISAN: *On Approximation Scheme Preserving Reducibility and its Applications*. In: *Foundations of Software Technology and Theoretical Computer Science*, Band 880 der Reihe *Lecture Notes in Computer Science*, Seiten 330–341. Springer, 1994.
- [13] FAGIN, R.: *Generalized First-Order Spectra and Polynomial-Time Recognizable Sets*. *Complexity and Computation*, 7:43–73, 1974.
- [14] FENG, D. und R. F. DOOLITTLE: *Progressive Alignment as a Prerequisite to Correct Phylogenetic Trees*. *Journal of Molecular Evolution*, 25:351–360, 1987.
- [15] GABOW, H. H.: *An Efficient Implementation of Edmonds' Algorithm for Maximum Matching on Graphs*. *Journal of the ACM*, 23(2):221–234, 1976.
- [16] GALIL, Z.: *Sequential and Parallel Algorithms for Finding Maximum Matchings in Graphs*. *Annual Review of Computer Science*, 1:197–224, 1986.
- [17] GAREY, M. R. und D. S. JOHNSON: *Computers and Intractability: A Guide to NP-Completeness*. W. H. Freeman and Company, 1979.
- [18] GOTOH, O.: *Optimal Alignment between Groups of Sequences and its Application to Multiple Sequence Alignment*. *Computer Applications in the Biosciences*, 9(3):361–370, 1993.
- [19] GUPTA, S. K., J. D. KECECIOGLU und A. A. SCHÄFFER: *Making the Shortest-Paths Approach to Sum-of-Pairs Multiple Sequence Alignment More Space Efficient in Practice*. In: *Proceedings of the 6th Annual Symposium on Combinatorial Pattern Matching*, Band 937 der Reihe *Lecture Notes in Computer Science*, Seiten 128–143. Springer, 1995.
- [20] GUSFIELD, D.: *Efficient Methods for Multiple Sequence Alignment with Guaranteed Error Bounds*. *Bulletin of Mathematical Biology*, 55(1):141–154, 1993.
- [21] GUSFIELD, D.: *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, 1997.
- [22] HÅSTAD, J.: *Some Optimal Inapproximability Results*. In: *Proceedings of the 29th Annual Symposium on Theory of Computing*, Seiten 1–10. ACM, 1997. Vollständige Version: *Electronic Colloquium on Computational Complexity*, Technischer Bericht 97-037, <http://www.eccc.uni-trier.de/eccc/>.

- [23] IMMERMANN, N.: *Descriptive and Computational Complexity*. In: *Computational Complexity Theory, Proceedings of Symposia in Applied Mathematics*, Band 38, Seiten 75–91. AMS, 1989.
- [24] JIANG, T., P. E. KEARNEY und M. LI: *Some Open Problems in Computational Molecular Biology*. ACM SIGACT News, 30(3):43–49, 1999.
- [25] JIANG, T., E. L. LAWLER und L. WANG: *Aligning Sequences via an Evolutionary Tree: Complexity and Approximation*. In: *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, Seiten 760–769, 1994.
- [26] JUNGnickel, D.: *Graphen, Netzwerke und Algorithmen*. BI-Wissenschaftsverlag, 3. Auflage, 1994.
- [27] JUST, W.: *Computational Complexity of Multiple Sequence Alignment with SP-score*. Technischer Bericht, Ohio University, 1999.
- [28] KHANNA, S., R. MOTWANI, M. SUDAN und U. VAZIRANI: *On Syntactic versus Computational Views of Approximability*. SIAM Journal on Computing, 28(1):164–191, 1998.
- [29] NEUBER, E. H.: *Klassifizierung von Optimierungsproblemen mittels Logik zweiter Ordnung*. Diplomarbeit, Technische Hochschule Darmstadt, 1994.
- [30] ORPONEN, P. und H. MANNILA: *On Approximation Preserving Reductions: Complete Problems and Robust Measures*. Technischer Bericht C-1987-28, University of Helsinki, Department of Computer Science, 1990.
- [31] PAPADIMITRIOU, C. H.: *The Euclidean Traveling Salesman Problem is NP-complete*. Theoretical Computer Science, 4(3):237–244, 1977.
- [32] PAPADIMITRIOU, C. H. und M. YANNAKAKIS: *Optimization, Approximation, and Complexity Classes*. Journal of Computer and System Sciences, 43(3):425–440, 1991.
- [33] PAPADIMITRIOU, C. H. und M. YANNAKAKIS: *The Traveling Salesman Problem With Distances One and Two*. Mathematics of Operations Research, 18:1–11, 1993.
- [34] PEVZNER, P. A.: *Multiple Alignment, Communication Cost, and Graph Matching*. SIAM Journal on Applied Mathematics, 52(6):1763–1779, 1992.

- [35] RABINER, L. R.: *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*. Proceedings of the IEEE, 77(2):257–285, 1989.
- [36] REISCHUK, K. R.: *Komplexitätstheorie*, Band 1. B. G. Teubner, 1999.
- [37] SIEBERT, B.: *Die Komplexität des Multiple Sequence Alignment Problems bei nichtmetrischen Kostenfunktionen*. Technischer Bericht A-00-05, Medizinische Universität zu Lübeck, Institut für Theoretische Informatik, 2000. Studienarbeit.
- [38] TREVISAN, L., G. B. SORKIN, M. SUDAN und D. P. WILLIAMSON: *Gadgets, Approximation, and Linear Programming*. SIAM Journal on Computing, 29(6):2074–2097, 2000.
- [39] VINGRON, M. und A. V. HAESLER: *Towards Integration of Multiple Alignment and Phylogenetic Tree Construction*. Journal of Computational Biology, 4(1):23–34, 1997.
- [40] WANG, L. und D. GUSFIELD: *Improved Approximation Algorithms for Tree Alignment*. Journal of Algorithms, 25(2):255–273, 1997.
- [41] WANG, L. und T. JIANG: *On the Complexity of Multiple Sequence Alignment*. Journal of Computational Biology, 1(4):337–348, 1994.
- [42] WAREHAM, H. T.: *A Simplified Proof of the NP- and MAX-SNP-Hardness of Multiple Sequence Tree Alignment*. Journal of Computational Biology, 2(4):509–514, 1995.
- [43] WATERMAN, M. S.: *Introduction to Computational Biology*. Chapman & Hall, 1995.
- [44] WU, B. Y., G. LANCIA, V. BAFNA, K. CHAO, R. RAVI und C. Y. TANG: *A Polynomial-Time Approximation Scheme for Minimum Routing Cost Spanning Trees*. SIAM Journal on Computing, 29(3):761–778, 1999.
- [45] ZUKER, M.: *Suboptimal Sequence Alignment in Molecular Biology: Alignment with Error Analysis*. Journal of Molecular Biology, 221(1):403–420.