

4. Das kürzeste gemeinsame Superstringproblem (KGS)

gegeben: Strings $s_1, s_2, \dots, s_m \in \Sigma^+$ für ein endliches Alphabet Σ .

gesucht: ein kürzester String s , so dass jedes $s_i, 1 \leq i \leq m$ ein Teilstring von s ist.

KGS ist NP-vollständig

(siehe: J.A. Storer: Data Compression: Methods and Theory, Computer Science Press 1988, p. 211 ff.)

⇒

Approximationsalgorithmen sind sinnvoll.

Literatur:

J. Tarhio, E. Ukkonen: A greedy approximation algorithm for constructing shortest common superstrings, TCS 57 (1988), 131 - 145.

J.S. Turner: Approximation algorithms for the shortest common superstring problem, Information and Computation 83 (1989), 1 - 20.

A. Blum, T. Jiang, M. Li, J. Tromp, M. Yannakakis: Linear approximation of shortest superstrings, JACM 41 (1994), 630 - 647.

D. Breslauer, T. Jiang, Z. Jiang: Rotations of periodic strings and short superstrings, J. Algorithms 24 (1997), 340 - 353.

Z. Sweedyk: A $2\frac{1}{2}$ -approximation algorithm for shortest superstring, SIAM J. Comput. 23(1994), 954-986.

Berechnungen:

Seien $p, q \in \Sigma^*$. v heißt Überlappung von p und q , falls $p = p'v$ und $q = vq'$ für $p', q' \in \Sigma^*$. v heißt maximale Überlappung von p und q , falls v die Überlappung maximaler Länge von p und q ist.

Wir schreiben dann

$$p \circ q := p'vq'$$

Wir sagen, dass eine Menge von Strings reduziert ist, falls kein String dieser Menge Teilstring eines anderen Strings dieser Menge ist. Wir können o.B.d.A. annehmen, dass die Menge S der Strings, für die wir einen Superstring konstruieren möchten, reduziert ist.

Übung:

Gegeben sei eine Menge S von Strings. Entwickeln Sie einen möglichst effizienten Algorithmus, der aus S mittels Entfernen von Strings, die Teilstrings von w in der Menge verbleibenden Strings sind, eine reduzierte Menge S' von Strings konstruiert. Welche Laufzeit hat Ihr Algorithmus?

Idee:

Bestimme ein Paar p, q von Strings mit größter maximaler Überlappung. Bilde $p \circ q$ und ersetze beide

Strings durch poq.

~>

Algorithmus GREEDY

Eingabe: $S = \{s_1, s_2, \dots, s_m\} \subseteq \Sigma^*$, $|\Sigma|$ endlich.

Ausgabe: Superstring $s \in \Sigma^+$ bzgl. S . D.h., s_i ist Teilstring von s für $1 \leq i \leq m$.

Methode:

(1) $R := S$;

(2) while $|R| > 1$

do

Wähle $p, q \in R$, ~~$p \neq q, q \neq p$~~ mit größter
maximale Überlappung;

$r := p \circ q$;

$R := (R \cup \{r\}) \setminus \{p, q\}$

od;

(3) Gib R aus.

Die Korrektheit folgt direkt aus der Konstruktion und ist leicht zu beweisen. Noch offen sind

- 1) effiziente Implementierung
- 2) Analyse der Güte der Ausgabe.

Implementierung:

Zunächst berechnen wir für alle Paare $(s_i, s_j) \in S^2$ die maximale Überlappung. Falls S nicht reduziert ist, dann können wir gleichzeitig aus S alle Strings

entfernen, die Teilstrings eines anderen in S sind. (195)

Berechnung der Überlappungen:

Betrachten wir $(s_i, s_j) \in S^2$.

Ziel:

- Berechnung der maximalen Überlappung von s_i und s_j .
- Entscheidung, ob $s_j \subseteq s_i$ oder nicht.

Beobachtung:

Obige Zielsetzung ist ähnlich zum üblichen String = matchingproblem. Da wir auch entscheiden möchten, ob $s_j \subseteq s_i$, ist es auch nicht einfacher.

⇒

Zur Lösung des Problems können wir übliche String = matchingalgorithmen verwenden.

z.B. Knuth-Morris-Pratt

benötigte Zeit $O(|s_i| + |s_j|)$

insgesamt \forall Paare $(s_i, s_j) \in S^2$:

$$O\left(\sum_{1 \leq i \neq j \leq m} (|s_i| + |s_j|)\right)$$

$$= O(m \cdot n),$$

$$\text{wobei } n := \sum_{1 \leq i \leq m} |s_i|.$$

(196)

Sortieren der Paare nach Größe der Überlappungen:

benötigte Zeit:

Verwendung von Bucketsort: $O(m^2)$

Verwendung von Mergesort: $O(m^2 \cdot \log m)$

Modifikation $R := (R \cup \{p, q\}) \setminus \{p, q\}$

Ersetze in der sortierten Liste

- jedes Paar (q, s') durch (p, q, s')
- jedes Paar (s', p) durch (s', p, q)

und entferne jedes Paar (p, s') bzw. (s', q) .

Beobachtung:

Da R reduziert ist, bleiben die Überlappungswerte in den neuen Paaren erhalten. Es ist daher auch korrekt, dass wir die Paare (p, s') und (s', q) aus der Liste entfernen. Da p und q längste maximale Überlappung haben, ist die verbleibende Liste auch reduziert.

benötigte Zeit: $O(m)$

insgesamt: $O(m^2)$.

Wir haben folgenden Satz bewiesen:

Satz 4.1

Der Algorithmus GREEDY benötigt $O(m \cdot n)$ Zeit, wobei m die Anzahl der Strings in der Eingabe und n die Summe der Stringlängen sind.

Es verbleibt noch die Analyse der Approximationsgüte. Wir werden zunächst zeigen, dass die Approximationsgüte nicht besser als zwei sein kann.

Betrachten wir

$$S := \{ ab^h, b^h b, b^h a \}.$$

⇒

Der kürzeste gemeinsame Superstring ist dann

$$s := ab^h b a.$$

GREEDY kann den Superstring

$$s' := ab^h a b^h b$$

konstruieren. Es gilt

$$|s| = h + 3 \quad \text{und} \quad |s'| = 2h + 3.$$

und

$$\frac{|ab^h a b^h b|}{|ab^h b a|} \xrightarrow{h \rightarrow \infty} 2.$$

⇒

∃ Konstante $c < 2$ mit der Eigenschaft, dass GREEDY die Approximationsgüte c hat.

Übung: Beispiel

Für obiges produziert GREEDY möglicherweise eine Ausgabe, deren Länge das Doppelte der minimalen Länge ist. Modifizieren Sie das Beispiel derart, dass GREEDY immer eine derartige Ausgabe produziert.

Vielorts wird folgende Vermutung, die noch wie vor offen ist, geäußert:

GREEDY hat Approximationsgüte zwei.

Bis 1991 ist es nicht gelungen, für irgendeinen Approximationsalgorithmus für KGS eine konstante Approximationsgüte zu beweisen. Dies gelang 1991 erstmals Avrim Blum et al. Im Jahr 1995 hat Elisabeth Sweedyk in ihrer Dissertation für einen Approximationsalgorithmus die Approximationsgüte 2,5 bewiesen, was noch wie vor den Rekord darstellt.

Ziel:

Entwicklung der Methode von Avrim Blum et al.

Berechnungen:

Seien $s, t \in \Sigma^+$ (nicht notwendigerweise verschieden).
Sei $v \in \Sigma^*$ der längste String, so dass $s = uv$ und $t = vw$ für nichtleere Strings u und w (d.h., $u, w \in \Sigma^+$).

- $o(s, t) := |v|$ Überlappung zwischen s und t .
- $\text{pref}(s, t) := u$ Präfix von s bzgl. t
- $d(s, t) := |\text{pref}(s, t)|$ Distanz von s nach t
- $\text{pref}(s, t) \cdot t = uvw$ Merge von s und t .

Bemerkung:

- $\text{pref}(s, t) \cdot t$ ist der kürzeste Superstring von s und t , in dem die Einbettung von s strikt vor der Ein =

bettung von t beginnt. Es gilt:

$$|\text{pref}(s,t) \cdot t| = d(s,t) + |t| = |s| + |t| - \text{ov}(s,t).$$

- Falls $\text{ov}(s,s) > 0$, dann gibt es eine selbstüberlappung von s .

Beispiel:

$s := \text{underground}$

Es gilt:

$$\text{ov}(s,s) = 5, \quad \text{pref}(s,s) = \text{undergro}, \quad d(s,s) = 8.$$

Seien $s_{i_1}, s_{i_2}, \dots, s_{i_r} \in \Sigma^+$. Dann ist der Superstring $s := \langle s_{i_1}, s_{i_2}, \dots, s_{i_r} \rangle$ definiert durch

$$s := \text{pref}(s_{i_1}, s_{i_2}) \cdot \text{pref}(s_{i_2}, s_{i_3}) \cdot \dots \cdot \text{pref}(s_{i_{r-1}}, s_{i_r}) \cdot s_{i_r}.$$

D.h., s ist der kürzeste String mit der Eigenschaft, dass $s_{i_1}, s_{i_2}, \dots, s_{i_r}$ in dieser Ordnung in dem String erscheinen.

Übung:

Zeigen Sie, dass für eine reduzierte Menge von Strings für jede Ordnung obiger String wohldefiniert ist.

Berechne für den Superstring s $\text{first}(s)$ den als ersten und $\text{last}(s)$ den als letzten eingebetteten String. D.h., $\text{first}(s) := s_{i_1}$ und $\text{last}(s) := s_{i_r}$.

Lemma 4.1

Seien $S = \{s_1, s_2, \dots, s_m\}$ eine reduzierte Menge von Strings und s ein kürzester Superstring von S .
Dann existiert eine Permutation π von $\{1, 2, \dots, m\}$ mit $s = \langle s_{\pi(1)}, s_{\pi(2)}, \dots, s_{\pi(m)} \rangle$.

Beweis:

Da S reduziert ist, gilt für alle $s_i, s_j \in S, i \neq j, s_i \not\subseteq s_j$. Wenn wir s von links nach rechts betrachten, dann erscheinen die Strings aus S in einer festen Ordnung. Wegen $s_i \not\subseteq s_j \forall i, j$ mit $i \neq j$ sind die Anfangspositionen dieser Strings verschieden. Also definiert diese Ordnung eine Permutation π .

z.z.: $s_{\pi(i)}$ und $s_{\pi(i+1)}$, $1 \leq i < m$ haben in s maximale Überlappung.

Dies folgt direkt aus der Minimalität von $|s|$. Andernfalls könnten wir s an der Stelle $s_{\pi(i)}, s_{\pi(i+1)}$ noch zusammenschieben. ■

Gegeben $S = \{s_1, s_2, \dots, s_m\}$ definieren wir einen kantengewichteten, gerichteten Graphen $G_S = (V, E, d)$ durch

$$V := \{1, 2, \dots, m\},$$
$$E := \{(i, j) \mid 1 \leq i, j \leq m, i \neq j\} \text{ und}$$
$$d: E \rightarrow \mathbb{N}_0, \text{ wobei } d(i, j) := d(s_i, s_j).$$

G_S heißt Distanzgraph bzgl. S .

Erinnerung:

Eine Rundreise durch einen Graphen $G_S = (V, E, d)$ ist ein einfacher Kreis in G_S , auf dem jeder Knoten in V liegt. Das Handlungsreisendenproblem (HRP) ist das Finden einer bzgl. den Kantengewichten kürzesten Rundreise.

Beweisne $HRP(G_S)$ die Länge einer kürzesten Rundreise in G_S und $OPT(S)$ die Länge eines kürzesten Superstrings von S . Dann gilt:

$$HRP(G_S) \leq OPT(S) - ov(\text{last}(s), \text{first}(s)) \leq OPT(S),$$

wobei $\text{last}(s)$ bzw. $\text{first}(s)$ den letzten bzw. ersten String in einem kürzesten Superstring s bezeichnet.

⇒

Die Länge einer kürzesten Rundreise in G_S bildet eine untere Schranke für die Länge eines kürzesten Superstrings bzgl. S .

Idee:

Berechne eine kürzeste Rundreise in G_S und schneide diese optimal auf.

Aber: HRP ist selbst NP-vollständig.

⇒

Betrachten wir folgendes ganzzahlige lineare Programm LP, in dem wir für jede Kante $(i, j) \in E$

eine Variable x_{ij} verwenden. In einer zulässigen Lösung für LP hat x_{ij} folgenden Wert:

$$x_{ij} = \begin{cases} 1 & \text{falls } (i,j) \text{ in Lösung} \\ 0 & \text{sonst} \end{cases}$$

$$\text{LP: } \min z = \sum_{(i,j) \in E} d(i,j) \cdot x_{ij}$$

$$\sum_{i=1}^m x_{ij} = 1 \quad j = 1, 2, \dots, m$$

$$\sum_{j=1}^m x_{ij} = 1 \quad i = 1, 2, \dots, m$$

$$x_{ij} \geq 0 \quad \text{ganzzahlig} \quad 1 \leq i, j \leq m$$

Eigenschaften von LP:

- 1) Jede Rundreise R in G_S bildet eine zulässige Lösung für LP.
- 2) Jede zulässige Lösung für LP besteht aus einer Menge $C = \{c_1, c_2, \dots, c_p\}$ von einfachen Kreisen, so dass jeder Knoten in V in exakt einem Kreis enthalten ist.
- 3) Das durch LP spezifizierte Problem, das so genannte Assignmentproblem ist in P .
- 4) Berechne $CYC(G_S)$ das Gewicht eines minimalen Assignments für G_S . Dann gilt:

$$CYC(G_S) \leq HRP(G_S) \leq OPT(S).$$

⇒

Folgender Approximationsalgorithmus bietet sich an:

Algorithmus CONCAT-CYCLES

Eingabe: $S = \{s_1, s_2, \dots, s_m\}$, S reduziert

Ausgabe: Superstring s für S .

Methode:

(1) Kreiere den Distanzgraphen $G_S = (V, E, d)$ und konstruiere ein Assignment C minimalen Gewichts auf G_S .

Sei $C = \{c_1, c_2, \dots, c_p\}$ die berechnete Kollektion von Kreisen.

(2) Für jeden Kreis $c_i = i_0 \rightarrow i_1 \rightarrow i_2 \rightarrow \dots \rightarrow i_{r-1} \rightarrow i_0$ sei $\tilde{s}_i := \langle s_{i_0}, s_{i_1}, \dots, s_{i_{r-1}} \rangle$ oberjenige String, den wir durch Öffnen des Kreises zwischen i_0 und i_{r-1} erhalten. Dabei ist i_0 beliebig gewählt.

(3) Bilde

$$s := \tilde{s}_1 \cdot \tilde{s}_2 \cdot \dots \cdot \tilde{s}_p.$$

Es ist leicht zu zeigen, dass der Algorithmus CONCAT-CYCLES einen Superstring für S ausgibt.

Ziel:

Analyse der Approximationsgüte des Algorithmus CONCAT-CYCLES.

Um dieses Ziel zu erreichen, entwickeln wir zu = nächst eine Theorie bzgl. der berechneten Kreise.

Berechnungen:

- Zwei Strings s und t heißen äquivalent ($s \equiv t$), falls der eine String ein zyklisches Shift des anderen Strings ist. D.h., $\exists u, v \in \Sigma^*$ mit $s = uv$ und $t = vu$.

\equiv ist eine Äquivalenzrelation. (überzeugen Sie sich.)

- Sei c ein gerichteter einfacher Kreis in G_S mit Knoten i_0, i_1, \dots, i_{r-1} in dieser zyklischen Ordnung auf c . Sei $\text{pref}(i, j) := \text{pref}(s_i, s_j)$ der mit der Kante (i, j) assoziierte String. Dann bezeichnet Strings(c) die Äquivalenzklasse

$$[\text{pref}(i_0, i_1) \cdot \text{pref}(i_1, i_2) \cdot \dots \cdot \text{pref}(i_{r-1}, i_0)]$$

Strings(c, i_k) bezeichnet denjenigen Repräsentanten von Strings(c), der mit $\text{pref}(i_k, i_{k+1})$ startet. D.h., den String

$$\text{pref}(i_k, i_{k+1}) \cdot \text{pref}(i_{k+1}, i_{k+2}) \cdot \dots \cdot \text{pref}(i_{k-1}, i_k),$$

wobei die Indizes modulo r gerechnet werden.

- Für $k > 0$ sagen wir, dass eine Äquivalenzklasse $[s]$ die Periodizität k besitzt, wenn s invariant unter einer Rotation von k

Buchstaben ist. D.h.,

$$S = u \cdot v = v \cdot u, \quad \text{wobei } |u| = \ell.$$

Folgendes Lemma fasst einige Eigenschaften von Periodizitäten zusammen.

Lemma 4.2

- $[s]$ hat die Periodizität $|s|$.
- Wenn $[s]$ die Periodizitäten a und b besitzt, dann hat $[s]$ auch die Periodizität $\text{ggT}(a, b)$.
- Die minimale Periodizität von $[s]$ ist gleich der Anzahl der verschiedenen Rotationen von s und somit gleich der Größe $\text{card}(s)$ der Äquivalenzklasse $[s]$.
- Alle Periodizitäten von $[s]$ sind Vielfache der minimalen Periodizität von $[s]$.

Beweis:

Übung

Bezeichnungen:

- Sei $i_0 \rightarrow i_1 \rightarrow i_2 \rightarrow \dots \rightarrow i_{r-1} \rightarrow i_0$ ein gerichteter Kreis c mit Knoten i_0, i_1, \dots, i_{r-1} . Dann ist $w(c) := |s|$, $s \in \text{Strings}(c)$ das Gewicht des Kreises c .

Interpretation: $w(c)$ ist die Länge des Kreises c .

- Wenn j ein Knoten auf dem Kreis c ist, dann sagen wir " s_j ist in c " oder auch " $s_j \in c$ ".

Seien im folgenden $c = i_0 \rightarrow i_1 \rightarrow i_2 \rightarrow \dots \rightarrow i_{r-1} \rightarrow i_0$ und c' Kreise in G_S . Zunächst werden wir einige Eigenschaften von Kreisen in G_S herleiten.

Lemma 4.3

Seien $s \in \text{Strings}(c)$ und k groß genug. Dann ist jeder String $s_{ij} \in c$ ein Teilstring von s^k .

Beweis:

Nachfolgend werden wir Knoten und korrespondierende Strings identifizieren. Was gemeint ist ergibt sich aus dem Kontext.

Konstruktion \Rightarrow

s_{ij} ist Präfix von

$$\text{pref}(i_j, i_{j+1}) \text{pref}(i_{j+1}, i_{j+2}) \dots \text{pref}(i_{j+k}, i_{j+k+1}) \cdot s_{ij+k}$$

$$\forall k \geq 0 \text{ (bzgl. Addition modulo } r \text{)}.$$

Seien

$$k := \left\lceil \frac{|s_{ij}|}{w(c)} \right\rceil \quad \text{und} \quad l := k \cdot r$$

\Rightarrow

s_{ij} ist Präfix von

$$\text{pref}(i_j, i_{j+1}) \dots \text{pref}(i_{j+k}, i_{j+k+1}) = \text{Strings}(c, i_j)$$

Also folgt die Behauptung aus der Eigenschaft, dass $\text{Strings}(c, i_j)^k$ ein Teilstring von s^{k+1} für beliebiges $s \in \text{Strings}(c)$ ist. ■

Lemma 4.4

Sei jeder String in einer reduzierten Menge $\{s_1, s_2, \dots, s_c\}$ von Strings Teilstring von s^k für einen String $s \in \text{Strings}(c)$ und genügend großes k . Dann existiert ein Kreis vom Gewicht $|s|$, der alle Strings in $\{s_1, \dots, s_c\}$ enthält.

Beweis:

Da $\{s_1, s_2, \dots, s_c\}$ reduziert ist, gilt für $i \neq j$

- s_i und s_j starten in verschiedenen Positionen in s .

Wir betrachten denjenigen Kreis, den wir aus s erhalten, indem wir Anfang und Ende von s konkateneren.

Die Startpositionen der Strings s_1, s_2, \dots, s_c auf dem Kreis definieren eine zirkuläre Ordnung dieser Strings. Die Distanzen aufeinanderfolgenden Strings addieren sich zu $|s|$, was das Gewicht dieses Kreises ist.



Lemma 4.5

Der Superstring $\langle s_{i_0}, s_{i_1}, \dots, s_{i_{r-1}} \rangle$ ist ein Teilstring von $\text{Strings}(c, i_0) s_{i_0}$.

Beweis:

$\langle s_{i_0}, s_{i_1}, \dots, s_{i_{r-1}} \rangle$ ist ein Teilstring von

$$\langle s_{i_0}, s_{i_1}, \dots, s_{i_{r-1}}, s_{i_0} \rangle = \text{pref}(i_0, i_1) \cdot \dots \cdot \text{pref}(i_{r-1}, i_0) s_{i_0}$$

$$= \text{Strings}(c, i_0) s_{i_0}.$$



Lemma 4.6

Falls $\text{Strings}(c') = \text{Strings}(c)$, dann existiert ein Kreis \tilde{c} mit

- i) $w(\tilde{c}) = w(c)$ und
- ii) \tilde{c} enthält alle Knoten in c und alle Knoten in c' .

Beweis:

Folgt direkt aus den Lemmata 4.3 und 4.4.



Übung:

Arbeiten Sie den Beweis von Lemma 4.6 aus.

Lemma 4.7

Es existiert ein Kreis \tilde{c} vom Gewicht $\text{card}(\text{Strings}(c))$, der alle Knoten in c enthält.

Beweis:

Sei $s \in \text{Strings}(c)$ beliebig aber fest. Sei u der kleinst nichtleere Präfix von s , so dass

$$s = uv = vu \quad \text{für ein } v \in \Sigma^*.$$

Folgendes kann leicht mittels Induktion bewiesen werden:

$$k := |u| \text{ teilt } w(c) \text{ und } s = u^j \text{ f\u00fcr } j := \frac{w(c)}{k}.$$

Da k minimal gew\u00e4hlt wurde, gilt:

$$\text{card}(\text{Strings}(c)) = k.$$

Die Anwendung des Lemmas 4.4. bzgl. u ergibt dann die Behauptung. ■

Zum Beweis der Approximationsg\u00fcte des Algorithmus CONCAT-CYCLES ben\u00f6tigen wir noch ein Lemma, das eine obere Schranke f\u00fcr die m\u00f6gliche \u00dcberlappung zweier Strings aus verschiedenen Kreisen angibt.

Lemma 4.8

Seien c und c' zwei Kreise in einem Assignment C minimalen Gewichtes. Seien $s \in c$ und $s' \in c'$. Dann gilt $ov(s, s') < w(c) + w(c')$.

Beweis:

Annahme:

s und s' \u00fcberlappen in einem String u mit $|u| \geq w(c) + w(c')$.

Bereiche

$$u_{i,j} := u_i u_{i+1} \dots u_j.$$

Beobachtung:

- $\text{Strings}(c) \neq \text{Strings}(c')$. Andernfalls könnte man gemäß Lemma 4.6 ein Assignment geringeren Gewichts mittels Kombination der Kreise c und c' konstruieren, was ein Widerspruch zur Minimalität von C wäre.
- Lemma 4.7 $\Rightarrow w(c) \leq \text{card}(\text{Strings}(c))$.
- Wegen $|u| > w(c)$ bzw. $|u| > w(c')$ können wir $x \in \text{Strings}(c)$ bzw. $x' \in \text{Strings}(c')$ wie folgt wählen:

$$x := u_{1, w(c)} \quad \text{und} \quad x' := u_{1, w(c')}.$$

Wegen $\text{Strings}(c) \neq \text{Strings}(c')$ gilt auch $w(c) \neq w(c')$.
O.B.d.A. sei $w(c) > w(c')$.

Dann kann wegen $|u| \geq w(c) + w(c')$ $u_{1, w(c)}$ wie folgt geschrieben werden:

$$\begin{aligned} u_{1, w(c)} &= u_{1+w(c'), w(c) + w(c')} \\ &= u_{1+w(c'), w(c)} u_{w(c)+1, w(c) + w(c')} \\ &= u_{1+w(c'), w(c')} u_{1, w(c')} \end{aligned}$$

\Rightarrow

$\text{Strings}(c)$ hat Periodizität

$$w(c') < w(c) \leq \text{card}(\text{Strings}(c)).$$

Dies ist ein Widerspruch dazu, dass $\text{card}(\text{Strings}(c))$ die minimale Periodizität von $\text{Strings}(c)$ ist.

$$\Rightarrow |u| < w(c) + w(c').$$

Satz 4.2

Der Algorithmus **CONCAT-CYCLES** berechnet einen Superstring s der Länge $\leq 4 \cdot \text{OPT}(S)$.

Beweis:

Da das berechnete Assignment $C := \{c_1, c_2, \dots, c_p\}$ ein optimales Assignment ist, gilt:

$$\text{CYC}(G_S) = \sum_{i=1}^p w(c_i) \leq \text{OPT}(S).$$

Eine weitere untere Schranke für $\text{OPT}(S)$ ergibt sich wie folgt:

Für jeden Kreis c_i , $1 \leq i \leq p$ bezeichne

$$w_i := w(c_i) \text{ und}$$

l_i die Länge eines längsten Strings in c_i .

Lemma 4.8 \Rightarrow

Falls wir aus jedem Kreis einen längsten String nehmen und alle p Strings optimal durch einen Superstring repräsentieren, dann ist die Gesamtüberlappung

$$\leq 2 \cdot \sum_{i=1}^P w_i.$$

\Rightarrow

Der Superstring hat mindestens die Länge

$$\sum_{i=1}^P (l_i - 2w_i).$$

Also gilt:

$$\text{OPT}(S) \geq \max \left\{ \sum_{i=1}^P w_i, \sum_{i=1}^P (l_i - 2w_i) \right\}.$$

Lemma 4.5 \Rightarrow

$$|S| \leq \sum_{i=1}^P (l_i + w_i).$$

Also gilt:

$$\begin{aligned} |S| &\leq \sum_{i=1}^P (l_i + w_i) \\ &= \sum_{i=1}^P (l_i - 2w_i) + \sum_{i=1}^P 3 \cdot w_i \\ &\leq \text{OPT}(S) + 3 \text{OPT}(S) \\ &= 4 \text{OPT}(S). \end{aligned}$$

Im Algorithmus CONCAT-CYCLES können wir zur Berechnung des optimalen Assignments C auf Standardverfahren zurückgreifen. Jedoch besitzt der aus der Eingabe $S = \{s_1, s_2, \dots, s_m\}$ konstruierte Graph

Eigenschaften, die die Entwicklung eines einfachen Greedyalgorithmus zur Berechnung eines optimalen Assignments zulassen.

Hierzu betrachten wir anstelle des originalen Problems "Berechnung eines Assignments minimalen Gewichtes im Distanzgraphen" das hierzu duale Problem "Berechnung eines Assignments maximalen Gewichtes im Überlappungsgraphen".

Gegeben $S = \{s_1, s_2, \dots, s_m\}$ definieren wir einen kantengewichteten Graphen $G'_S := (V, E, ov)$ durch

$$V := \{1, 2, \dots, m\},$$

$$E := \{(i, j) \mid 1 \leq i, j \leq m, i \neq j\} \text{ und}$$

$$ov: E \rightarrow \mathbb{N}_0, \text{ wobei } ov(i, j) = ov(s_i, s_j).$$

G'_S heißt Überlappungsgraph bzgl. S .

Wir präsentieren nun den Greedyalgorithmus und zeigen dann, dass dieser nichts anderes tut, als den Algorithmus CONCAT-CYCLES nachzuahmen, wovon sich unmittelbar die Approximationsgüte des Greedyalgorithmus ergibt.

Algorithmus MGREEDY

Eingabe: $S = \{s_1, s_2, \dots, s_m\}$, S reduziert.

Ausgabe: Superstring s für S .

Methode:(1) $R := S ; T := \emptyset ;$ (2) while $R \neq \emptyset$ doWähle $s, t \in R$ (nicht notwendigerweise verschieden), so dass $ov(s, t)$ maximal;if $s \neq t$ then $R := (R \cup \{sot\}) \setminus \{s, t\}$ else $R := R \setminus \{s\};$ $T := T \cup \{s\}$ fiod ;(3) $s :=$ Konkatination der Strings in T .Interpretation:

- MGREEDY wählt Strings s und t mit maximaler Überlappung, wobei durchaus $s = t$ sein kann und wählt dann in G'_S die gerichtete Kante von $last(s)$ nach $first(t)$.

 \Rightarrow

MGREEDY konstruiert und verbindet Pfade in G'_S und schließt diese dann zu Kreise. Am Schluss enthält T eine Kollektion M von disjunkten Kreisen, die die Knoten in G'_S überdecken.

Sei M das durch M_{GREEDY} konstruierte Assignment.

- Wir können die Verfahrensweise von M_{GREEDY} auch wie folgt interpretieren:

M_{GREEDY} erhält eine Liste der Kanten in E , sortiert in monoton fallender Ordnung bzgl. ihrer Überlappung und artseitigt diese von links nach rechts ab.

Wir sagen, dass eine Kante e eine andere Kante e' dominiert, falls e links von e' in der Liste steht und e und e' denselben Start- oder denselben Endknoten haben.

Es gilt:

Wenn M_{GREEDY} eine Kante e' in der Liste betrachtet, dann wählt M_{GREEDY} genau dann diese Kante e' aus, wenn M_{GREEDY} nicht bereits vorher eine Kante e , die e' dominiert, ausgewählt hat.

Satz 4.3

Das durch den Algorithmus M_{GREEDY} konstruierte Assignment M ist ein optimales Assignment für G_s .

Beweis:

Die Distanzgewichte und die Überlappungsgewichte eines jeden Assignments addieren sich zur Gesamt

länge aller Strings.

⇒

Ein Assignment ist genau dann optimal in G_S , wenn es maximale Gesamtüberlappung hat

Unter allen Assignments mit maximaler Gesamtüberlappung sei N eines, welches eine Maximale Anzahl von Kanten mit M gemeinsam hat.

Ziel: Beweis, dass $M = N$.

Folgendes Lemma liefert den Schlüssel für den Beweis:

Lemma 4.9

Seien u, u^+, v^- und v nicht notwendigerweise verschiedene Strings, so dass

$$ov(u, v) \geq \max \{ ov(u, u^+), ov(v^-, v) \}.$$

Dann gilt:

$$ov(u, v) + ov(v^-, u^+) \geq ov(u, u^+) + ov(v^-, v) \text{ und} \\ d(u, v) + d(v^-, u^+) \leq d(u, u^+) + d(v^-, v).$$

Interpretation:

Wenn wir die Wahl haben, u und u^+ sowie v^- und v anstatt u und v sowie v^- und u^+ zu überlappen, dann ist die beste Wahl solche, die ein Paar mit größter Überlappung enthält.

Bevor wir Lemma 4.9 beweisen, führen wir den Beweis von Satz 4.3 zu Ende.

Annahme: $M \neq N$

Sei e diejenige Kante mit maximaler Überlappung in $N \setminus M \cup M \setminus N$, die als erste in der Liste von MGREEDY erscheint. Wir unterscheiden zwei Fälle:

1. Fall: $e \in N \setminus M$

MGREEDY wählt nicht die Kante $e \Rightarrow$

\exists Kante e' , die e dominiert und gewählt wird.

N Assignment $\Rightarrow e' \notin N$

Also gilt $e' \in M \setminus N$

Widerspruch zur Wahl der Kante e .

2. Fall: $e := (k, j) \in M \setminus N$.

Betrachten wir die Kanten (i, j) und (k, e) in N , die Start- bzw. Endknoten mit e gemeinsam haben.

$(k, j) \in M \Rightarrow (i, j), (k, e) \notin M$.

\Rightarrow

(k, j) dominiert (i, j) und auch (k, e) .

Also folgt hieraus und der Wahl von $e = (k, j)$

$ov(k, j) \geq \max \{ ov(i, j), ov(k, e) \}$.

Lemma 4.9 =>

$$ov(i,j) + ov(k,e) \leq ov(k,j) + ov(i,e)$$

=>

Der Austausch der Kanten $(i,j), (k,e)$ in N durch die Kanten (k,j) und (i,e) ergibt ein Assignment N' mit

- N' hat mindestens eine Kante mehr als N mit M gemeinsam und
- N' hat keine geringere Überlappung als N .

Widerspruch zur Wahl von N .



Beweis von Lemma 4.9:

Es genügt die erste Ungleichung zu beweisen. Wegen

$$d(s,t) + |t| = |s| + |t| - ov(s,t) \quad \forall s,t$$

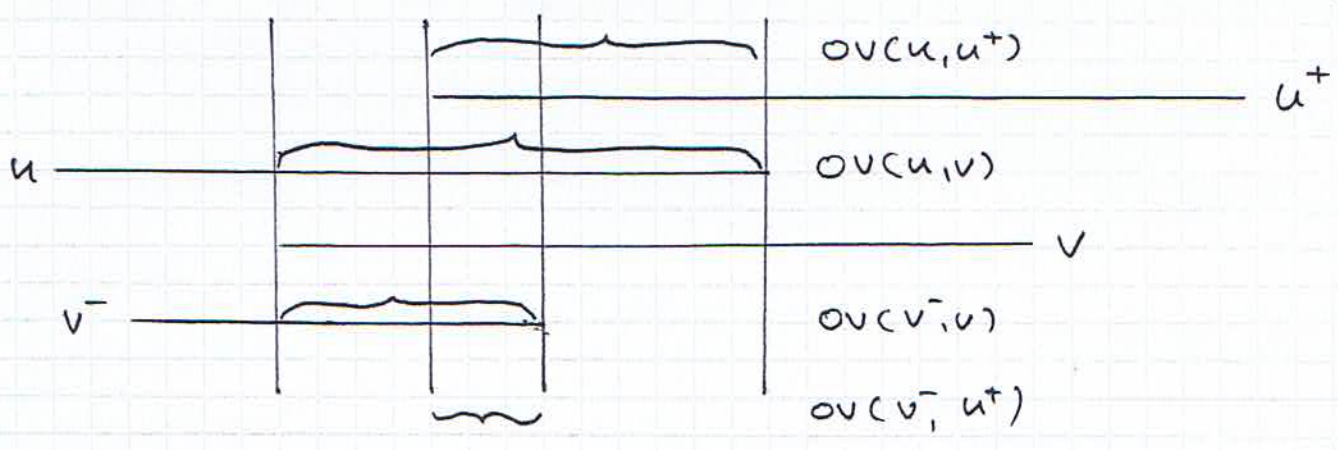
folgt dann aus dieser

$$d(u,v) + d(v^-, u^+) \leq d(u, u^+) + d(v^-, v).$$

zu zeigen

$$ov(u,v) + ov(v^-, u^+) \geq ov(u, u^+) + ov(v^-, v).$$

Folgendes Bild verdeutlicht die Situation:



Es gilt:

$$OV(u, v) \geq OV(u, u^+) + OV(v^-, v) - OV(v^-, u^+)$$

$$\Rightarrow OV(u, v) + OV(v^-, u^+) \geq OV(u, u^+) + OV(v^-, v)$$



Da MGREEDY ein optimales Assignment findet und die Kreise alle in der optimalen Position aufbricht, gilt:

Der durch MGREEDY konstruierte Superstring ist nicht länger als der durch CONCAT-CYCLES konstruierte Superstring.

Also folgt direkt aus Satz 4.2 der folgende Satz:

Satz 4.4

MGREEDY berechnet einen Superstring s der Länge $\leq 4 \cdot OPT(S)$.

Beobachtung:

MGREEDY konkateniert einfach alle Strings in T ohne weitere Kompression.

Diese Beobachtung wirft direkt folgende Frage auf:

Frage:

Falls wir versuchen, die Strings in T geschickt zu überlappen, können wir dann eine bessere Approximationsgüte als 4 erreichen?

Berechne T_{GREEDY} denjenigen Approximationsalgorithmus, den wir erhalten, indem wir in M_{GREEDY}

(3) $s :=$ Konkatenation der Strings in T
durch

(3') $s :=$ Ausgabe von $GREEDY$ bei Eingabe T
ersetzen.

Satz 4.5

T_{GREEDY} berechnet einen Superstring s der Länge $\leq 3 \cdot OPT(S)$.

Beweis:

Seien $S = \{s_1, s_2, \dots, s_m\}$ die Eingabe von T_{GREEDY} und s der von T_{GREEDY} berechnete Superstring. Sei $n := OPT(S)$.

Ziel: Beweis, dass $|s| \leq 3 \cdot n$.

Seien

T die von T_{GREEDY} in Schritt (2) berechnete Menge von "selbstüberlappenden" Strings.

C das korrespondierende Assignment.

$c_x \in C$ der zum String $x \in T$ korrespondierende Kreis

w_x Gewicht $w(c_x)$ des Kreises $c_x \in C$

$\|R\| := \sum_{x \in R} |x|$, wobei R eine Menge von Strings ist.

$w := \sum_{x \in T} w_x$.

Wegen $CYC(G_S) \leq HRP(G_S) \leq OPT(S)$ gilt:

$w \leq n$.

Lemma 4.8 \Rightarrow

(*) Die Kompression, die in einem kürzesten Superstring für T erreicht wird, ist $< 2 \cdot w$

Folgendes Lemma gibt eine untere Schranke für die Kompression, die GREEDY bei T erreicht, an.

Lemma 4.10

Sei $S := \{s_1, s_2, \dots, s_m\}$ eine Menge von Strings.

Berechne $OV_{opt}(S)$ die Überlappung in einem kürzesten Superstring für S und $OV_{GREEDY}(S)$ die vom Algorithmus GREEDY erzielte Überlappung.

Dann gilt:

$OV_{GREEDY}(S) \geq \frac{1}{2} OV_{opt}(S)$.

Bevor wir Lemma 4.10 beweisen, führen wir den Beweis von Satz 4.5 zu Ende.

Berechne n_T die Länge eines kürzesten Superstrings für T . D.h.,

$$OV_{OPT}(T) = \|T\| - n_T.$$

Ferner gilt:

$$OV_{GREEDY}(T) = \|T\| - |S|,$$

wobei s die Ausgabe von T_{GREEDY} ist.

Lemma 4.10 \Rightarrow

$$OV_{GREEDY}(T) \geq \frac{1}{2} OV_{OPT}(T)$$

$$\begin{aligned} (\Rightarrow) \quad \|T\| - |S| &\geq \frac{\|T\| - n_T}{2} \\ &= \|T\| - n_T - \frac{\|T\| - n_T}{2} \\ &\stackrel{(*)}{\geq} \|T\| - n_T - w \end{aligned}$$

$$\Leftrightarrow |S| \leq n_T + w$$

Ziel:

Ermittlung einer möglichst kleinen oberen Schranke für n_T .

Berechnungen:

Für $x \in T$ sei s_x derjenige String in $c_x \cap S$, der Präfix von x ist.

Seien

$$S' := \{s_x \mid x \in T\},$$

$$n' := \text{OPT}(S'),$$

$$S'' := \{ \text{Strings}(c_x, l_x) s_{i_x} \mid x \in T \} \text{ und}$$

$$n'' := \text{OPT}(S'').$$

Lemma 4.5 \Rightarrow

Ein Superstring für S'' ist auch ein Superstring für T .

$$\Rightarrow n_T \leq n''.$$

Für eine Permutation π der Strings in T bezeichne S'_π bzw. S''_π denjenigen Superstring, den wir mittels Überlappung der Strings in S' bzw. S'' in der durch π definierten Anordnung erhalten.

Beobachtung:

$\forall x \in T$ gilt: Der String $\text{Strings}(c_x, l_x) s_{i_x}$ fängt mit s_{i_x} an und hört mit s_{i_x} auf.

\Rightarrow

\forall Permutationen π der Strings in T gilt:

$$\begin{aligned} |S''_\pi| &\leq |S'_\pi| + \sum_{x \in T} w_x \\ &= |S'_\pi| + w. \end{aligned}$$

Betrachten wir eine Permutation π_0 mit $|S'_{\pi_0}| = n'$.
Wegen Lemma 4.1 existiert π_0 .

Dann gilt:

$$\begin{aligned}
 n' + w &= |S'_{\pi_0}| + w \\
 &\geq |S''_{\pi_0}| \\
 &\geq n''
 \end{aligned}$$

Ferner gilt: $S' \subseteq S \Rightarrow n' \leq n$.

Insgesamt erhalten wir

$$n_T \leq n'' \leq n' + w \leq n + w.$$

Die Kombination mit $|S| \leq n_T + w$ ergibt dann

$$|S| \leq n_T + w \leq n + 2w \leq 3n.$$

Beweis des Lemmas 4.10:

GREEDY wählt beim j -ten Durchlauf des Blockes der while-Schleife aus der aktuellen Menge R_{j-1} , der noch zu verarbeitenden Strings p_j und q_j mit maximaler Überlappung aus, bildet $p_j \circ q_j$ und führt die Modifikation $R_j := (R_{j-1} \cup \{p_j \circ q_j\}) \setminus \{p_j, q_j\}$ durch.

Bezeichne $OV_j(S)$, $0 \leq j < m$ die in einem Superstring maximaler Überlappung, der alle Überlappungen $p_i \circ q_i$, $1 \leq i \leq j$ enthält, erzielte Überlappung.

Dann gilt:

$$OV_0(S) = OV_{OPT}(S).$$

Idee: Wir beweisen für $0 < j < m$

$$OV_{j-1}(S) \leq OV_j(S) + 2|p_j \cap q_j|$$

Dann erhalten wir

$$\begin{aligned} OV_{\text{OPT}}(S) = OV_0(S) &\leq \sum_{j=1}^{m-1} 2|p_j \cap q_j| \\ &= 2 \cdot \sum_{j=1}^{m-1} |p_j \cap q_j| \\ &= 2 OV_{\text{GREEDY}}(S). \end{aligned}$$

Durchführung:

Hierzu fixieren wir einen Superstring \bar{S} maximaler Überlappung unter all denjenigen Superstrings, die die Überlappungen $p_i \cap q_i$, $1 \leq i < j$ enthalten.

Sei $\bar{U}(\bar{S})$ die Menge der Überlappungen in \bar{S} .

Wahl von $p_j, q_j \Rightarrow$

$$|p_j \cap q_j| = \max \{ |p \cap q| \mid p, q \in R_{j-1}, p \neq q \}.$$

Die Wahl der Überlappung $p_j \cap q_j$ durch GREEDY macht maximal drei Überlappungen in $\bar{U}(\bar{S})$ unzulässig. Wir unterscheiden zwei Fälle:

1. Fall: Maximal zwei Überlappungen werden unzulässig.

Dann folgt direkt aus der Maximalität von $|p_j \cap q_j|$, dass

$$OV_{j-1}(S) \leq OV_j(S) + 2|p_j \cap q_j|.$$

2. Fall : Drei Überlappungen werden unzulässig.

Dann sind diese Überlappungen von folgenden Arten:

- eine der Form $p_j \cap q_i, i \neq j,$
- eine der Form $p \cap q_j, p \neq p_j.$
- Die dritte Überlappung würde einen Kreis schließen. D.h., irgendeine Überlappung $\tilde{p} \cap \tilde{q}$ auf dem Kreis wird unzulässig.

Beobachtung:

$$(p_j \cap q_i, p \cap q_j)$$

Nach Wahl der Überlappung $p_j \cap q_i$ und der Entfernung der Überlappung $\tilde{p} \cap \tilde{q}$ ist die Überlappung $p \cap q$ zulässig. D.h. $p \cap q$ schließt keinen Kreis. Wäre dies der Fall, dann hätte es bereits vorher einen Kreis gegeben.

Also gilt:

$$(*) \quad OV_{j-1}(S) \leq OV_j(S) + |p_j \cap q_i| + |p \cap q_j| + |\tilde{p} \cap \tilde{q}| - |p \cap q|$$

Lemma 4.8 \Rightarrow

$$|p_j \cap q_j| + |p \cap q| \geq |p_j \cap q_i| + |p \cap q_j|$$

$$(**) \Leftrightarrow |p_j \cap q_j| \geq |p_j \cap q_i| + |p \cap q_j| - |p \cap q|$$

(*) und (**) \Rightarrow

$$OV_{j-1}(S) \leq OV_j(S) + 2 |p_j \cap q_j|,$$

womit das Lemma bewiesen ist.

Avrim Blum et al. haben in ihrer Arbeit bewiesen, dass GREEDY ein 4-Approximationsalgorithmus für das Superstringproblem ist. Hierin genügt es nicht, die für MGREEDY entwickelte Beweis- methode auf GREEDY zu übertragen. Sie erläutern in ihrer Arbeit ausführlich, warum dies nicht funktioniert und geben dann eine komplizier- tere Analyse, aus der sich die Approximations- güte 4 ergibt, an. In

H. Kaplan, N. Shafrit: The greedy algorithm for shortest superstrings, IPL 93 (2005), 13-17.

wird im Beweis von Avrim Blum et al. mit Hilfe eines von Breslauer, Jiang und Jiang ent- wickelten Lemmas eine obere Schranke verbessert, wodurch sich der Wert 4 für die Approximations- güte auf den Wert 3,5 reduziert.