1 Rückblick und Ausblick

Im ersten Teil wurden Permutationen und Breakpoint-Graphen behandelt. Es wurde gezeigt, dass eine kürzeste Folge von Reversionen, die den Breakpoint-Graphen sortiert, eine kürzeste Folge von Reversionen induziert, welche die Permutation sortiert, also in die Identität überführt. In diesem zweiten Teil wird ein Algorithmus beschrieben, der den Breakpoint-Graphen mit einer kürzesten Folge von Reversionen sortiert.

2 Kreise in Breakpoint-Graphen

Maßgeblich für die benötigte Zahl an Reversionen ist die topologische Struktur des Breakpoint-Graphen. Um diese hinreichend zu beschreiben, sind einige Begriffe und Definitionen notwendig.

Definition 1. *Eine Desire-Edge a ist unorientiert, wenn beim Ablaufen des korrespondierenden Reality-Desire-Kreises die zu a adjazenten Reality-Edges bezüglich des Reality-Obverse-Kreises in gleicher Richtung abgelaufen werden. Andernfalls ist a orientiert.*

Definition 2. *Zwei Desire-Edges $a = \{a_1, a_2\}$ und $b = \{b_1, b_2\}$ schneiden einander, wenn sich die inzidenten Knoten von a und b auf dem Reality-Obverse-Kreis abwechseln; d.h. wenn es nicht möglich ist, auf dem Reality-Obverse-Kreis von a_1 nach a_2 zu gelangen, ohne zuerst b_1 oder b_2 zu besuchen.*

In Abbildung 1 sind diese beiden Definitionen illustriert.

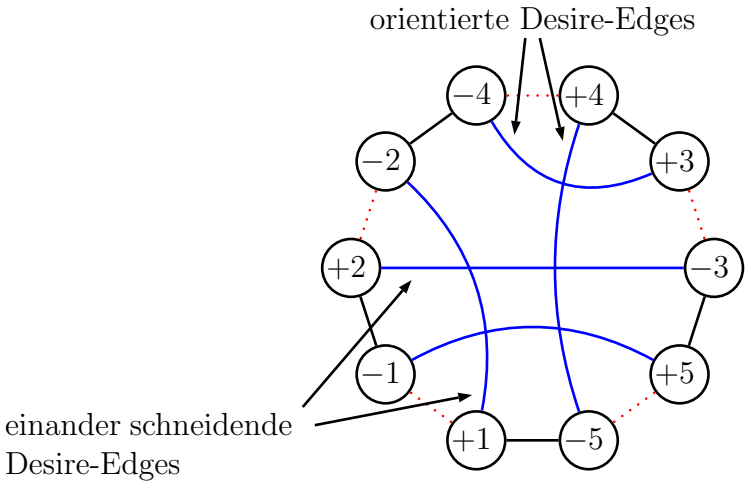

Abbildung 1: Der Breakpoint-Graph der Permutation $(+3, -4, +2, +1, +5)$. Die Desire-Edges $\{+3, -4\}$ und $\{+4, -5\}$ sind orientiert, die übrigen Desire-Edges sind unorientiert. Zu erkennen sind auch die einander schneiden bzw. nicht schneidenden Desire-Edges. Ein schneidendes Kantenpaar ist z.B. $a = \{+1, -2\}$ und $b = \{+2, -3\}$. Denn läuft man von $+1 \in a$ im Uhrzeigersinn den Reality-Obverse-Kreis entlang, trifft man zuerst auf $+2 \in b$ und erst danach auf $-2 \in a$. Wenn wir hingegen gegen den Uhrzeigersinn laufen, gelangen wir zuerst zu $-3 \in b$ und erst anschließend zu $-2 \in a$.

Definition 3. *Die Länge eines alternierenden Reality-Desire-Kreises ist die Zahl seiner Desire-Edges. Ein Reality-Desire-Kreis der Länge 1 wird als trivialer Kreis bezeichnet.*

Definition 4. *Eine Menge M von Reality-Desire-Kreisen wird als Komponente bezeichnet, wenn die folgenden beiden Bedingungen erfüllt sind:*

1. *Für je zwei Reality-Desire-Kreise $A, B \in M$ gibt es eine Folge $(K_1, \ldots, K_k)$ von Reality-Desire-Kreisen, sodass $A = K_1, B = K_k$ und für alle $i \in [k-1]$ gilt: K_i schneidet K_{i+1}.*

2. *Es gibt keine echte Obermenge von M, welche den ersten Punkt erfüllt.*

Definition 5. *Eine Komponente ist unorientiert, wenn sie nur unorientierte Desire-Edges enthält. Andernfalls ist sie orientiert. Eine unorientierte Komponente wird als trivial bezeichnet, wenn sie ein trivialer Kreis ist.*

In Abbildung 2 sind diese beiden Definitionen illustriert.

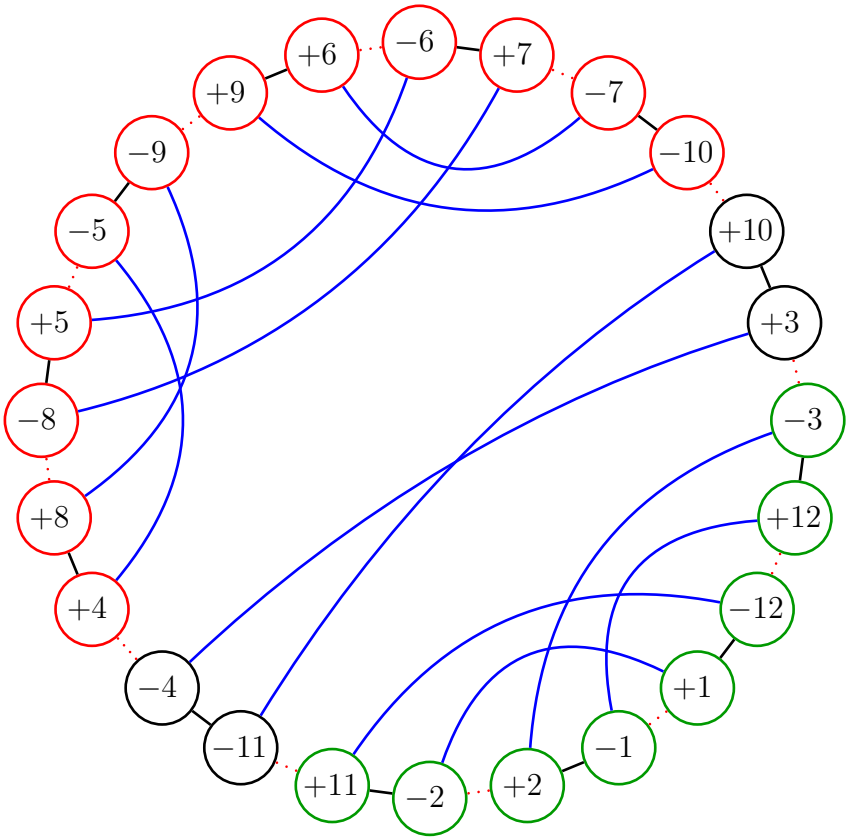

Abbildung 2: Der Breakpoint-Graph der Permutation $(+3, -10, +7, +6, -9, +5, +8, -4, +11, +2, +1, +12)$. Die zu den roten bzw. schwarzen bzw. grünen Knoten gehörenden Reality-Desire-Kreise bilden jeweils eine Komponente des Breakpoint-Graphen. Die rote und die schwarze Komponente sind beide orientiert, während die grüne (untere) Komponente unorientiert ist.

Definition 6. *Eine nichttriviale unorientierte Komponente H ist eine Hurdle, wenn sie keine zwei anderen nichttrivialen unorientierten Komponenten bezüglich des Reality-Obverse-Kreises voneinander trennt. Mit anderen Worten: H ist eine Hurdle, wenn es möglich ist, von jeder nichttrivialen unorientierten Komponente $A \neq H$ entlang des Reality-Obverse-Kreises zu jeder anderen nichttrivialen unorientierten Komponente $B \neq H$ zu gelangen, ohne eine Reality-Edge von H zu verwenden.*

Definition 7. *Eine Hurdle H ist eine Super-Hurdle, wenn eine unorientierte Komponente, welche keine Hurdle ist, durch die Löschung von H zu einer Hurdle wird. Andernfalls ist H eine einfache Hurdle.*

Definition 8. *Zwei Hurdles H_1 und H_2 mit $H_1 \neq H_2$ sind benachbart, wenn man auf dem Reality-Obverse-Kreis von einer Reality-Edge von H_1 zu einer Reality-Edge von H_2 gelangen kann, ohne eine Reality-Edge einer Hurdle H_3 mit $H_3 \neq H_1$ und $H_3 \neq H_2$ zu besuchen.*

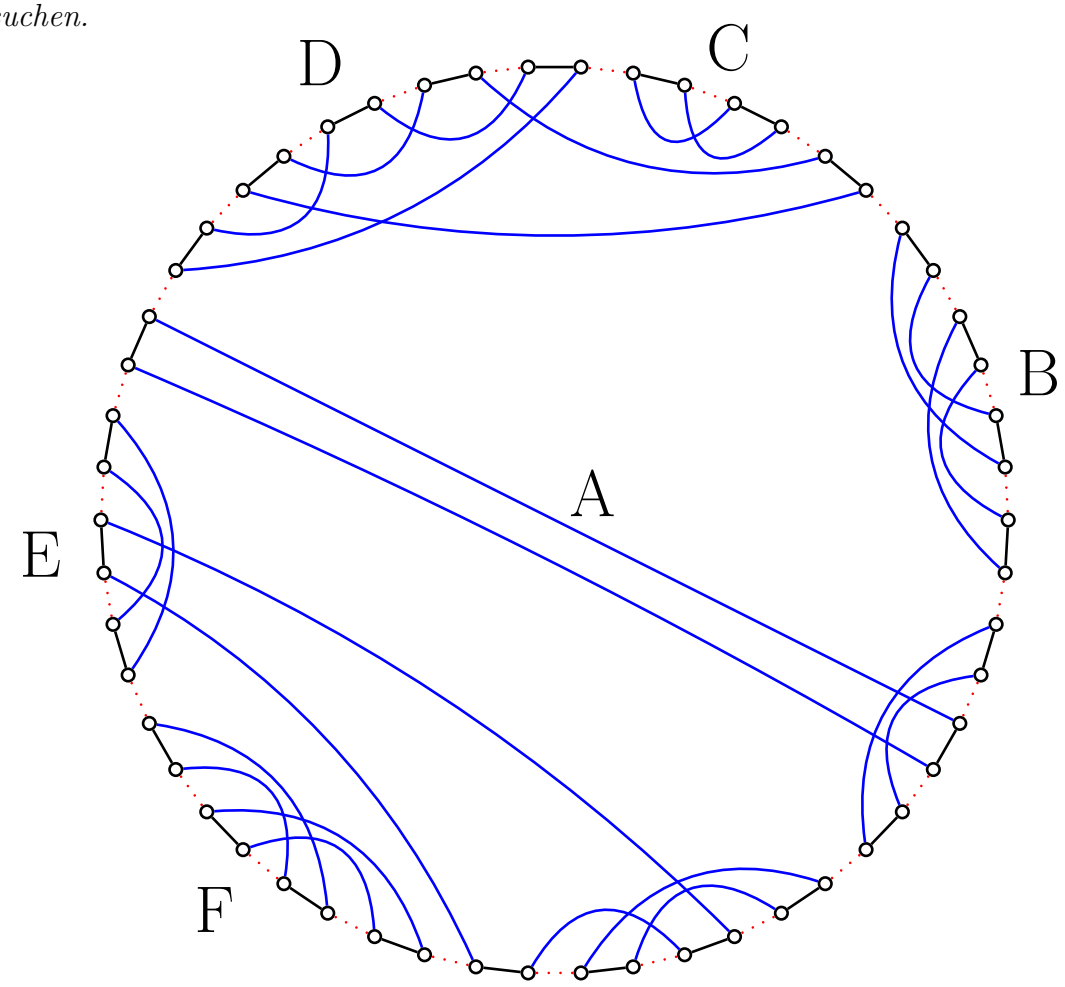

Abbildung 3: Zu sehen ist ein Breakpoint-Graph mit sieben Komponenten, wobei alle bis auf C unorientiert sind. Die Komponente C spielt also für die Hurdle-Eigenschaften der unorientierten Komponenten keine Rolle. F, B und D sind Hurdles. F ist sogar eine Super-Hurdle, da durch Entfernen von F die Komponente E zu einer Hurdle wird. Da es weniger als vier Hurdles gibt, sind automatisch alle Hurdles miteinander benachbart. Dieser Breakpoint-Graph ist keine Fortress.

Definition 9. *Ein Breakpoint-Graph ist eine Fortress, wenn die Zahl der Hurdles ungerade ist und jede Hurdle eine Super-Hurdle ist.*

Diese Definitionen sind in Abbildung 3 visualisiert.

Notation. *Sei G ein Breakpoint-Graph.*

- *Die Zahl der Reality-Edges von G schreiben wir als $n(G)$.*
- *Die Zahl der Reality-Desire-Kreise von G schreiben wir als $c(G)$.*
- *Die Zahl der Hurdles von G schreiben wir als $h(G)$.*
- *$f(G)$ ist eine Indikatorvariable, die genau dann 1 annimmt, wenn G eine Fortress ist, und andernfalls 0.*

Theorem 10 (Hannenhalli und Pevzner)**.** *Die Zahl der Reversionen, die benötigt werden, um einen Breakpoint-Graphen G einer Permutation zu sortieren, beträgt exakt*

$$d(G) := n(G) - c(G) + h(G) + f(G).$$

$d(G)$ wird als Reversal-Distanz von G bezeichnet. Der in Abbildung 3 dargestellte Breakpoint-Graph hat damit die Reversal-Distanz $27 - 12 + 3 + 0 = 18$. Um diesen Breakpoint-Graphen zu sortieren, werden also genau 18 Reversionen benötigt.

Definition 11. *Sei G der Breakpoint-Graph einer Permutation π. Eine Reversion ρ auf G heißt sicher, wenn $d(\rho(G)) = d(G) - 1$ ist, d.h. wenn sich die Reversal-Distanz durch ρ verkleinert.*

Prinzipiell könnte man eine optimale Reversionsfolge dadurch finden, dass man im aktuellen Breakpoint-Graphen so lange unterschiedliche Reversionen durchprobiert, bis man eine sichere Reversion gefunden hat. Da es maximal $\binom{n(G)}{2}$ mögliche Reversionen gibt, muss man im Worst-Case quadratisch viele Reversionen betrachten und analysieren. Wir wollen geschickter vorgehen.

Dazu benötigen wir zuerst die folgende Definition.

Definition 12. *Eine Reversion ρ auf einer Desire-Edge d eines nichttrivialen Kreises (bzw. eine zu d korrespondierende Reversion ρ) ist diejenige Reversion, bei der die zu der Desire-Edge adjazenten Reality-Edges ersetzt werden. Diese Reversion bezeichnen wir als ρ_d.*

Wir halten fest, dass jede Desire-Edge, die zu einem nichttrivialen Kreis gehört, stets zwei unterschiedliche Reality-Edges als Nachbarn hat. Weiterhin ist eine orientierte Desire-Edge nie Teil eines trivialen Kreises.

Die generelle Idee ist, orientierte und unorientierte Komponenten getrennt zu betrachten. Falls es eine orientierte Komponente und damit eine orientierte Desire-Edge gibt, wird eine geeignete Reversion auf einer der orientierten Desire-Edges durchgeführt. Falls es keine orientierte Komponente gibt und der Breakpoint-Graph noch nicht sortiert ist, werden geeignete Operationen auf den (zwangsläufig vorhanden) Hurdles durchgeführt, um neue orientierte Komponenten zu erhalten.

3 Verfahren mit orientierten Komponenten

In diesem Abschnitt wird beschrieben, wie wir in einem Breakpoint-Graph mit einer orientierten Kante eine Reversion finden können, mit der eine kürzeste Folge von Reversionen beginnt.

Lemma 13. *Sei G der Breakpoint-Graph einer Permutation, welche mindestens eine orientierte Komponente und damit mindestens eine orientierte Desire-Edges enthält. Dann gibt es eine orientierte Desire-Edge a von G, sodass die Reversion ρ_a sicher ist.*

Es genügt also, so lange zu orientierten Desire-Edges korrespondierende Reversionen durchzuprobieren, bis wir eine Reversion finden, die die Reversal-Distanz verkleinert.

4 Erzeugung von orientierten Komponenten

In diesem Abschnitt wird gezeigt, wie wir sichere Reversionen finden können, wenn keine orientierte Desire-Edge im Breakpoint-Graphen existiert. Ein Nebeneffekt der hier beschriebenen sicheren Reversionen ist, dass orientierte Desire-Edges und damit orientierte Komponenten entstehen.

Definition 14. *Sei G ein Breakpoint-Graph einer Permutation.*

- *Eine Reversion, die auf Reality-Edges von zwei verschiedenen Hurdles operiert, wird als Hurdle-Merging bezeichnet.*
- *Eine Reversion, die auf einer Desire-Edge einer Hurdle operiert, wird als Hurdle-Cutting bezeichnet.*

Das folgende Lemma beschreibt, in welchen Fällen ein Hurdle-Merging bzw. ein Hurdle-Cutting eine sichere Reversion darstellt.

Lemma 15. *Sei G ein Breakpoint-Graph einer Permutation ohne orientierte Komponenten.*

- *Wenn $h(G) \geq 4$, dann ist ein Hurdle-Merging auf zwei nicht benachbarten Hurdles sicher.*
- *Wenn $h(G) = 2$, dann ist ein Hurdle-Merging auf den beiden Hurdles sicher.*
- *Wenn $h(G) = 1$, dann ist ein Hurdle-Cutting auf der einzigen Hurdle sicher.*
- *Wenn $h(G) = 3$ und es gibt eine einfache Hurdle H, dann ist ein Hurdle-Cutting auf H sicher.*
- *Wenn $h(G) = 3$ und es gibt nur Super-Hurdles, dann ist ein Hurdle-Merging auf zwei benachbarten Hurdles sicher.*

5 Pseudocode und Implementation

Es folgt der Pseudo-Code des Algorithmus.

Listing 1: Sorting by oriented reversals

```
Input: Breakpoint-Graph G = (V, E) einer Permutation

begin
while c(G) < n(G) do
  if G enthält einen orientierten Reality-Desire-Kreis then
    führe eine sichere Reversion auf einer orientierten Desire-Edge aus
  else if h(G) ≥ 4 then
    Hurdle-Merging auf zwei nicht benachbarten Hurdles
  else if h(G) = 2 then Hurdle-Merging
  else if h(G) = 1 then Hurdle-Cutting
  else if h(G) = 3 and es gibt eine einfache Hurdle H then
    Hurdle-Cutting auf H
  else Hurdle-Merging auf zwei benachbarten Hurdles
end
```

Es stellt sich noch die Frage, wie der Breakpoint-Graph einigermaßen effizient analysiert werden kann. Die folgenden Punkte lassen sich mit einer Gesamtlaufzeit von $O(n(G)^2)$ berechnen.

- Welche Kanten bilden einen Reality-Desire-Kreis? Es genügt, an einer Reality-Edge zu starten und so lange alternierend Reality- und Desire-Edges abzulaufen, bis man eine bereits besuchte Kante erreicht. Alle abgelaufenen Kanten bilden dann einen Kreis. Die Laufzeit liegt in $O(n(G))$.

- Welche Kanten/Kreise bilden eine Komponente? Man betrachtet je zwei unterschiedliche Desire-Edges $\{a, b\}$, wobei a und b in unterschiedlichen Reality-Desire-Kreisen liegen. $a = \{a_1, a_2\}$ und $b = \{b_1, b_2\}$ schneiden einander, wenn sich die zu den Knoten a_1 und a_2 gehörenden Elemente der Permutation mit den zu den Knoten b_1 und b_2 gehörenden Elementen der Permutation bezüglich ihrer Anordnung in der Permutation abwechseln. Wenn ein Knoten a_i von a und ein Knoten b_j von b zum selben Permutationselement π_k gehören, müssen die Vorzeichen von a_i, b_j, π_k betrachtet werden: Wenn $\pi_k > 0$ ist, dann wird derjenige der beiden Knoten a_i und b_j mit dem negativem Vorzeichen als direkter Vorgänger des Knotens mit dem positivem Vorzeichen interpretiert. Wenn $\pi_k < 0$ ist, dann ist es genau umgekehrt.

  Um in $O(1)$ feststellen zu können, an welcher Position in der Permutation sich ein bestimmtes Element befindet, erstellt man das zu π inverse Array. Dieses beinhaltet an Position $i \in [n(G)]$ diejenige Position, an der sich das Element i bzw. $-i$ in π befindet. Wir können damit in $O(n(G)^2)$ alle Komponenten ermitteln.

- Welche unorientierten Komponenten sind Hurdles? Man erstellt einen Hilfsgraphen $H_1(G)$, der als Knoten alle Reality-Edges des Breakpoint-Graphen enthält.

Jeder Knoten von $H_1(G)$ wird mit der Komponentennummer gelabelt, zu der die korrespondierende Reality-Edge gehört. Zwischen zwei Knoten a, b von $H_1(G)$ existiert genau dann eine Kante, wenn die korrespondierenden Reality-Edges im Breakpoint-Graphen zu einer gemeinsamen Obverse-Edge adjazent sind.

In $H_1(G)$ werden nun alle Knoten entfernt, die im Breakpoint-Graphen zu orientierten und zu trivialen Komponenten gehören. Immer, wenn ein Knoten entfernt wird, werden die beiden Nachbarn des Knotens miteinander verbunden. Dann kontrahiert man iterativ je zwei benachbarte Knoten, die zur selben Komponente gehören. Den durch diese Prozedur aus $H_1(G)$ resultierenden Graphen bezeichnen wir als $H_2(G)$. Eine unorientierte Komponente k des Breakpoint-Graphen ist genau dann eine Hurdle, wenn im Hilfsgraphen $H_2(G)$ genau ein Knoten mit k gelabelt ist. Die Laufzeit beträgt $O(n(G))$.

- Welche Hurdles sind benachbart? Man betrachtet den im vorherigen Punkt beschriebenen Hilfsgraphen $H_2(G)$. Indem man aus $H_2(G)$ alle Knoten entfernt, die keine Hurdles sind, erhält man den Hilfsgraphen $H_3(G)$. Zwei Hurdles sind genau dann im Breakpoint-Graphen benachbart, wenn die beiden korrespondierenden Knoten in $H_3(G)$ benachbart sind. Die Laufzeit ist in $O(n(G))$.

- Welche Hurdles sind Super-Hurdles? Erneut betrachten wir den Hilfsgraphen $H_2(G)$. Eine Hurdle ist genau dann eine Super-Hurdle, wenn der korrespondiernde Knoten im Hilfsgraphen zwei verschiedene Nachbarknoten besitzt, diese beiden Nachbarknoten zur selben unorientierten Komponente gehören und diese unorientierte Komponente bezüglich $H_2(G)$ nur aus diesen beiden Knoten besteht (d.h. es gibt keinen dritten Knoten, der zur selben Komponente wie die beiden Nachbarn der Hurdle gehört). Die Laufzeit liegt in $O(n(G))$.

Die Reversal-Distanz eines Breakpoint-Graphen G mit $n(G)$ Reality-Edges ist stets weniger als $2n(G)$. Pro durchgeführter sicherer Reversion muss einmal der aktuelle Breakpoint-Graph analysiert werden. Um eine sichere Reversion zu finden, müssen maximal $n(G)$ Reversionen durchgeführt und die entstehenden Breakpoint-Graphen analysiert werden. Pro Versuch liegen die Kosten in $O(n(G)^2)$. Die Gesamtlaufzeit liegt also in $O(2n(G) \cdot n(G) \cdot n(G)^2) = O(n(G)^4)$. Wenn wir also eine Permutation mit n Elementen als Input erhalten, können wir eine optimale sortierende Reversionsfolge in $O(n^4)$ finden.

Diese Laufzeit ist asymptotisch ebenso gut wie die der ersten von Hannenhalli und Pevzner veröffentlichten Version des Algorithmus. Die besten bekannten Algorithmen besitzen durch ausgeklügelte Datenstrukturen, Darstellungen und Reversionsauswahlverfahren eine Laufzeit von $O(n^2)$. Die bloße Ermittlung der Reversal-Distanz ohne Ermittlung einer optimalen Reversionsfolge ist sogar in $O(n)$ möglich.