

Verbände getwisteter Involutionen in Coxetergruppen

Christian Hoffmeister

11. Juli 2012

Inhaltsverzeichnis

1	Zweizykel in der getwisteten schwachen Ordnung	3
2	Algorithmus zur Berechnung der getwisteten schwachen Ordnung	4

1 Zweizykel in der getwisteten schwachen Ordnung

Definition 1.1 (Ein- und beidseitige Wirkung). Seien (W, S) ein Coxetersystem, $w \in W$ und $s \in S$. Falls $w\underline{s} = \theta(s)ws$ ist, so sagen wir, dass s beidseitig auf w wirkt. Andernfalls sagen wir s wirkt einseitig auf w .

Definition 1.2 (Ein- und beidseitige endende Gesamtwirkung). Seien (W, S) ein Coxetersystem, $w \in W$ und $s_1, \dots, s_n \in S$. Falls $ws_1 \cdots s_n = \theta(s_n)(ws_1 \cdots s_{n-1})s_n$ ist, so sagen wir, dass $s_1 \cdots s_n$ eine beidseitig endende Gesamtwirkung auf w hat. Andernfalls sagen wir $s_1 \cdots s_n$ hat eine einseitig endende Gesamtwirkung auf w .

Definition 1.3. Seien (W, S) ein Coxetersystem und $s, t \in S$ zwei verschiedene Erzeuger. Wir definieren:

$$[st]^n := \begin{cases} (st)^{\frac{n}{2}}, & n \text{ gerade} \\ (st)^{\frac{n-1}{2}}s, & n \text{ ungerade} \end{cases}$$

Definition 1.4 (Zweizykel). Seien (W, S) ein Coxetersystem und $s, t \in S$ zwei verschiedene Erzeuger. Dann nennen wir $wC_{\{s,t\}}$ den von s und t erzeugten Zweizykel bezüglich w .

Vermutung 1.5. Seien (W, S) ein Coxetersystem und $s, t \in S$ zwei verschiedene Erzeuger von W . Dann gilt:

1. Sei $m = \text{ord}(st) < \infty$. Falls $w[st]^n \neq w$ ist für alle $n \in \mathbb{N}, n < 2m$, dann gilt $w(st)^{2m} = w$.
2. In $wC_{\{s,t\}}$ existieren keine drei Element mit derselben getwisteten Länge.
3. Falls s einseitig auf w wirkt, dann gilt $wst < w\underline{s}$ oder $w\underline{t} > w$.
4. Sei $w[st]^n = w$ für ein $n \in \mathbb{N}$. Dann ist n gerade und es gilt eine der beiden folgenden Eigenschaften:
 - a) Für jedes $m \in \mathbb{N}$ hat das Element $[st]^m$ genau dann eine beidseitig endende Gesamtwirkung auf w , wenn $[st]^{n/2+m}$ eine beidseitig endende Gesamtwirkung auf w hat.
 - b) Für jedes $m \in \mathbb{N}$ hat das Element $[st]^m$ genau dann eine beidseitig endende Gesamtwirkung auf w , wenn $[st]^{n-m+1}$ eine beidseitig endende Gesamtwirkung auf w hat.

Anmerkung 1.6. Vermutung 1.5.2 bedeutet, dass Zweizykel in einem gewissen Sinne konkav sind. Vermutung 1.5.3 bedeutet, dass innerhalb eines Zweizykels einseitige Wirkungen ausschließlich am bzgl. der getwisteten Länge oberen oder unteren Ende auftreten können. Vermutung 1.5.4 bedeutet, dass in einem Zweizykel die ein- und beidseitigen Wirkungen achsen- oder punktsymmetrisch verteilt sind.

2 Algorithmus zur Berechnung der getwisteten schwachen Ordnung

Wir wollen nun einen Algorithmus zur Berechnung der getwisteten schwachen Ordnung $Wk(\theta)$ einer beliebigen Coxetergruppe W erarbeiten. Also Ausgangspunkt werden wir den Algorithmus aus [Haas and Helmnick, 2012, Algorithm 3.1.1] verwenden, der im wesentlichen benutzt, dass für jede getwistete Involution $w \in \mathcal{I}_\theta$ entweder $w_{\underline{s}} < w$ oder aber $w_{\underline{s}} > w$ gilt.

Algorithm 2.1 (Algorithmus 1).

```

1: procedure TWISTEDWEAKORDERINGALGORITHM1( $W$ )                                ▷  $W$  sei die Coxetergruppe
2:    $V \leftarrow \{(e, 0)\}$ 
3:    $E \leftarrow \{\}$ 
4:   for  $k \leftarrow 0$  to  $k_{\max}$  do
5:     for all  $(w, k_w) \in V$  with  $k_w = k$  do
6:       for all  $s \in S$  with  $\nexists(\cdot, w, s) \in E$  do                                ▷ Nur die  $s$ , die nicht schon nach  $w$  führen
7:          $y \leftarrow ws$ 
8:          $z \leftarrow \theta(s)y$ 
9:         if  $z = w$  then
10:           $x \leftarrow y$                                                          ▷  $s$  operiert ungetwistet auf  $w$ 
11:           $t \leftarrow s$ 
12:         else
13:           $x \leftarrow z$                                                          ▷  $s$  operiert getwistet auf  $w$ 
14:           $t \leftarrow \underline{s}$ 
15:         end if
16:          $isNew \leftarrow \mathbf{true}$ 
17:         for all  $(w', k_{w'}) \in V$  with  $k_{w'} = k + 1$  do                    ▷ Prüfen, ob  $x$  nicht schon in  $V$  liegt
18:           if  $x = w'$  then
19:              $isNew \leftarrow \mathbf{false}$ 
20:           end if
21:         end for
22:         if  $isNew = \mathbf{true}$  then
23:            $V \leftarrow V \cup \{(x, k + 1)\}$ 
24:            $E \leftarrow E \cup \{(w, x, t)\}$ 
25:         else
26:            $E \leftarrow E \cup \{(w, x, t)\}$ 
27:         end if
28:       end for
29:     end for
30:      $k \leftarrow k + 1$ 
31:   end for
32:   return  $(V, E)$                                                          ▷ The poset graph
33: end procedure

```

Dieser Algorithmus berechnet alle getwisteten Involutionen und deren getwistete Länge (w, k_w)

und deren Relationen (w', w, s) bzw. (w', w, \underline{s}) . Zu bemerken ist, dass zur Berechnung der getwisteten Involutionen der Länge k nur die Knoten aus V benötigt werden, mit der getwisteten Länge $k - 1$ und k sowie die Kanten aus E , die Knoten der Länge $k - 2$ und $k - 1$ bzw. $k - 1$ und k verbinden. Alle vorherigen Ergebnisse können schon persistiert werden, so dass nie das komplette Ergebnis im Speicher gehalten werden muss.

Eine Operation, die hier als elementar angenommen wurde ist der Vergleich von Elementen in W . Für bestimmte Gruppen wie z.B. die A_n , welche je isomorph zu $Sym(n + 1)$ sind, lässt sich der Vergleich von Element effizient implementieren. Will man jedoch mit Coxetergruppen im Allgemeinen arbeiten, so liegt W als frei präsentierte Gruppe vor und der Vergleich von Element ist eine sehr aufwendige Operation. Bei Algorithmus 2.1 muss jedes potentiell neue Element x mit allen schon bekannten w' von gleicher getwisteter Länge verglichen werden um zu bestimmen, ob x wirklich ein noch nicht bekanntes Element aus \mathcal{I}_θ ist.

Algorithm 2.2 (Algorithmus 2).

```

1: procedure TWISTEDWEAKORDERINGALGORITHM2( $W$ )           ▷  $W$  sei die Coxetergruppe
2:    $V \leftarrow \{(e, 0)\}$ 
3:    $E \leftarrow \{\}$ 
4:   for  $k \leftarrow 0$  to  $k_{\max}$  do
5:     TODO
6:   end for
7:   return  $(V, E)$                                        ▷ The poset graph
8: end procedure

```

Im Anhang findet sich eine Implementierung der Algorithmen 2.1 und 2.2 in GAP 4.5.4. Tabelle 2 zeigt ein Benchmark anhand von fünf ausgewählten Coxetergruppen.

W	A_9	A_{10}	A_{11}	E_6	E_7	E_8
$ Wk(\text{id}, W) $	9496	35696	140152	892	10208	199952
$\max \rho(w)$	25	30	36	20	35	64
TWOA1	00:02.180	00:31.442	11:04.241	00:03.044	06:11.728	–
TWOA2	00:01.372	00:06.276	00:29.830	00:00.268	00:02.840	11:03.278
TWOA1 element compares	13.531.414	185.791.174	2.778.111.763	85.857	7.785.186	–
TWOA2 element compares	42.156	173.356	737.313	2.347	29.687	682.227

Tabelle 2.1: Benchmark

Dabei sind die A_n als symmetrische Gruppen implementiert und die E_n als frei präsentierte Gruppen. Ausgeführt wurden die Messungen auf einem Intel Core i5-3570k mit vier Kernen zu je 3,40 GHz. Der Algorithmus ist dabei aber nur single threaded und kann so nur auf einem Kern laufen. Um die Messergebnisse nicht durch Limitierungen des Datenspeichers zu beeinflussen, wurden die Daten in diesem Benchmark nicht stückweise persistiert sondern ausschließlich berechnet. Wie zu erwarten ist der Geschwindigkeitsgewinn bei den Coxetergruppen vom Typ E_n deutlich größer, da in diesem Fall die Elementvergleiche deutlich aufwendiger sind als bei Gruppen vom Typ A_n .

Literatur

R. Haas and A. G. Helmnick. Algorithms for twisted involutions in weyl groups. *Algebra Colloquium* 19, 2012.

J. E. Humphreys. *Reflection Groups and Coxeter Groups*. Cambridge University Press, 1992.