Betrachtungen zum Cutting sticks-Problem

Alexander Büchel
Ulrich Gilleßen
Kurt-Ulrich Witt

Publisher: Dean Prof. Dr. Wolfgang Heiden

University of Applied Sciences Bonn-Rhein-Sieg,
Department of Computer Science

Sankt Augustin, Germany

June 2016

Technical Report 01-2016

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

ISSN 1869-5272

Digital Object Identifier doi:10.18418/978-3-96043-031-5
DOI-Resolver http://dx.doi.org/

Betrachtungen zum Cutting sticks-Problem

Alexander Büchel[2], Ulrich Gilleßen[2], Kurt-Ulrich Witt[1,2]

Hochschule Bonn-Rhein-Sieg
Fachbereich Informatik
[1]b-it Applied Science Institute
[2]Arbeitsgruppe Diskrete Mathematik und Optimierung (ADIMO)
alexander.buechel@smail.inf.h-brs.de, ulrich.gillessen@smail.inf.h-brs.de kurt-ulrich.witt@h-brs.de

Zusammenfassung Das Cutting sticks-Problem ist ein NP-vollständiges Problem mit Anwendungspotenzialen im Bereich der Logistik. Es werden grundlegende Definitionen für die Behandlung sowie bisherige Ansätze zur Lösung des Problems aufgearbeitet und durch einige neue Aussagen ergänzt. Insbesondere stehen Ideen für eine algorithmische Lösung des Problems bzw. von Varianten des Problems im Fokus.
Schlüsselwörter: Cutting sticks-Problem, Mengenpartionierungsproblem, Teilsummenaufteilung

1 Problemstellung

Es sei $\mathbb{N} = \{1, 2, 3, \ldots\}$ die Menge der natürlichen Zahlen. Für eine endliche Menge $M \subset \mathbb{N}$ setzen wir

$$\sum M = \sum_{x \in M} x$$

und für $n \in \mathbb{N}$ setzen wir $[\, n \,] = \{1, \ldots, n\}$ sowie

$$\Delta_n = \sum [\, n \,] = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Δ_n ist die Folge der *Dreieckszahlen*, die sich als Summen der Zahlen von 1 bis n ergeben.

Sei $n, k \in \mathbb{N}$ sowie $\left\{T_j\right\}_{1 \le j \le k}$ eine Folge von Teilmengen $T_j \subseteq [\, n \,]$ und $\left\langle\, t_j \,\right\rangle_{1 \le j \le k}$ eine Folge von natürlichen Zahlen mit

(i) $\sum T_j = t_j \ge n, 1 \le j \le k$,

(ii) $T_i \cap T_j = \emptyset$ für $1 \le i, j \le k$ und $i \ne j$,

(iii) $\bigcup_{j=1}^{k} T_j = [\, n \,]$,

Es folgt unmittelbar, dass

$$\Delta_n = \sum_{j=1}^{k} t_j \tag{1.1}$$

gelten muss. Für den speziellen Fall $t_j = t = \text{const}$, $1 \leq j \leq k$, gilt

$$\Delta_n = k \cdot t \tag{1.2}$$

Wir nennen die Folge $\left\langle t_j \right\rangle_{1 \leq j \leq k}$ eine $(k; t_1, \ldots, t_k)$-*Partition* von Δ_n und $\left\{T_j\right\}_{1 \leq j \leq k}$ eine $(k; t_1, \ldots, t_k)$-*Partitionierung* von $[\,n\,]$. Analog sprechen wir im spezeillen Fall $t_j = t = \text{const}$, $1 \leq j \leq k$, von einer (k, t)-*Partition* von Δ_n bzw. von einer (k, t)-*Partitionierung* von $[\,n\,]$.

Für die weiteren Betrchtungen benötigen wir noch die Teilermengen von natürlichen Zahlen. Für $m \in \mathbb{N}$ sei

$$D^{(*)}(m) = \{x \in \mathbb{N} \mid x \,|\, m\}$$

die Menge aller *(positiven) Teiler* von m und

$$D^{(+)}(m) = D^{(*)}(m) - \{1, m\}$$

die Menge der *echten (positiven) Teiler* von m. Die Zerlegung

$$m = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_s^{\alpha_s}$$

mit $p_r \in \mathbb{P}$ und $\alpha_r \geq 1$ für $1 \leq r \leq s$ und $p_r < p_{r+1}$ für $1 \leq r \leq s-1$, heißt (*kanonische*) *Faktorisierung* von m. Es gilt

$$D^{(*)}(m) = \left\{ p_1^{\beta_1} \cdot p_2^{\beta_2} \cdot \ldots \cdot p_s^{\beta_s} \,\middle|\, 0 \leq \beta_r \leq \alpha_r,\, 1 \leq r \leq s \right\}$$

1.1 Das allgemeine Problem

Das *Cutting sticks-Problem* ist gegeben durch die Längen $t_1, \ldots, t_k \in \mathbb{N}$ von k Stäben (engl. sticks), die eine Mindestlänge von n haben, d.h. es ist, $t_j \geq n$, $1 \leq j \leq k$, und für die

$$\sum_{j=1}^{k} t_j = \Delta_n$$

gilt. Die Frage ist, ob die gegebenen Stäbe in n Teile geschnitten werden können, die die Längen von 1 bis n haben.

Wenn die Mengen $T_j = \left\{t_{j1}, \ldots, t_{jk_j}\right\}$, $1 \leq j \leq k$, die Längen der Teilstäbe von t_j enthalten, dann gilt $T_j \subseteq [\,n\,]$ sowie $\sum T_j = t_j$. Ist zudem $T_i \cap T_j = \emptyset$ für $1 \leq i, j \leq k$ und $i \neq j$, dann erfüllt die Teilmengenfolge $\left\{T_j\right\}_{1 \leq j \leq k}$ die obigen Eigenschaften (i) – (iii). Damit ist das Cutting sticks-Problem äquivalent zu folgendem Mengen-Partitionierungssproblem: Existiert eine $(k; t_1, \ldots, t_k)$-Partitionierung von $[\,n\,]$?

Dies ist ein NP-hartes Entscheidungsproblem. In Fu und Hu (1992) wird gezeigt, dass es für $k, l, t \in \mathbb{N}$ mit $0 < l \leq \Delta_n$ und $(k-1)t + l + \Delta_{k-2} = \Delta_n$ eine $(k; t, t+1, \ldots, t+k-2, l)$-Partition von Δ_n und eine entsprechende Partitionierung von $[\,n\,]$ gibt. Chen et al. (2005) beweisen, dass es eine $(k, t_1, \ldots, t_k)$-Partitionierung von $[\,n\,]$ gibt, falls $\sum_{j=1}^{k} t_j = \Delta_n$ und

$t_j \geq t_{j+1}$ für $1 \leq j \leq k-1$ sowie $t_{k-1} \geq n$ ist. In Kapitel 3 wird ein 0/1- Programm für die Lösung allgemeiner Partitionierungsprobleme angegeben.

Neben der Entscheidungsvariante kann man sich auch für folgende Optimierungsvariante interessieren: Gegeben seien die Längen $t_1, \ldots, t_k$, bestimme das kleinste t, so dass die Menge $[\,t\,]$ in k Teilmengen $T_1, \ldots, T_k$ so partitioniert werden kann, dass $\sum T_j \geq t_j$ für $1 \leq j \leq k$ ist. In Jagadish (2015) werden polynomielle Approximationen mit konstanter Güte für dieses Minimierungsproblem angegeben.

1.2 Ein spezielles Problem

In dieser Arbeit betrachten wir nur die Entscheidungsvariante und zwar zunächst für das spezielle Cutting sticks-Problem. Bei diesem haben alle Stäbe dieselben Längen:

$$t_j = t = \text{const},\ 1 \leq j \leq k$$

Äquivalent dazu ist das folgende Mengen-Partition(ierung)sproblem: Existiert eine (k,t)-Partition von Δ_n, und existiert eine (k,t)-Partitionierung von $[\,n\,]$ mit $\Delta_n = k \cdot t$. Wir schließen im Folgenden den trivialen Fall $k = 1$ aus.

In Kapitel 2 zeigen wir, dass für alle $n \in \mathbb{N}$ (k,t)-Partitionen von Δ_n existieren. Des Weiteren geben wir einen Algorithmus an, der zu jedem n alle diese k und t berechnet; und wir geben einen Algorithmus an, der zu einer (k,t)-Partition von Δ_n eine (k,t)-Partitionierung von $[\,n\,]$ berechnet. Im Allgemeinen haben (k,t)-Partitionen von Δ_n mehrere (k,t)-Partionierungen von $[\,n\,]$. Sei z.B. $n = 7$, dann gibt es für die $(4,7)$-Partition von $\Delta_7 = 28$ nur die Partitionierung $T_1 = \{7\}$, $T_2 = \{1,6\}$, $T_3 = \{2,5\}$, $T_4 = \{3,4\}$, während es für die $(2,14)$-Partition unter anderem die folgenden beiden Partitionierungen von $[\,7\,]$ gibt:

$$T_1 = \{1,6,7\} \qquad T'_1 = \{1,2,4,7\}$$
$$T_2 = \{2,3,4,5\} \qquad T'_2 = \{3,5,6\}$$

Wir werden Fälle charakterisieren, für die es zu einer (k,t)-Partition von Δ_n genau eine (k,t)-Partitionierung von $[\,n\,]$ gibt.

Im Kapitel 3 geben wir, wie bereits oben erwähnt, ein 0/1-Programm zur Lösung aller Cutting sticks-Probleme an. Kapitel 4 fasst die Ergebnisse dieser Arbeit zusammen und gibt Hinweise auf weitere Ansätze zur effizienten Lösung von Partion(ierung)sproblemen, d.h. insbesondere zur Berechnung von Partitionierungen von $[\,n\,]$.

2 Lösungen für das spezielle Problem

In Straight und Schillo (1979) wird der folgende Satz gezeigt.

Satz 2.1 *Existieren zu $n \in \mathbb{N}$ zwei Zahlen $k, t \in \mathbb{N}$ mit $t \geq n$ und $\Delta_n = k \cdot t$, dann existiert eine (k,t)-Partition von Δ_n.* □

In Ando et al. (1990) wird von der Anforderung $\Delta_n = k \cdot t$ abgewichen und gezeigt, dass die Menge $[\,n\,]$ genau dann in k disjunkte Teilmengen T_j mit $\sum T_j = t$ für $1 \leq j \leq k$ zerlegt werden kann, wenn $k(2k-1) \leq k \cdot t \leq \Delta_n$ ist.

Wir zeigen zunächst, dass es für alle $n \in \mathbb{N}$ (k,t)-Partitionen von Δ_n gibt, und wir geben alle möglichen Paare (k,t) dazu an. Wir untersuchen dann, für welche (k,t)-Partitionen von Δ_n genau eine Partitionierung von $[\,n\,]$ existiert. Des Weiteren stellen wir einen Algorithmus vor, der zu jeder (k,t)-Partition von Δ_n eine Partitionierung $\left\{T_j\right\}_{1\le j\le k}$ von $[\,n\,]$ bestimmt.

2.1 Allgemeine Existenz von Partitionen und Partitionierungen

Lemma 2.1 *Zu jedem $n \in \mathbb{N}$ existieren $k_n \in \mathbb{N}$ und $t_n \in \mathbb{U}_+$ mit $t_n \ge n$ und $\Delta_n = k_n \cdot t_n$.*

Beweis *Wir betrachten für $n \in \mathbb{N}$ die beiden möglichen Fälle (1) $n \in \mathbb{G}_+$, $n \ge 2$, sowie (2) $n \in \mathbb{U}_+$.*

Zu (1): Es existiert $k_n \in \mathbb{N}$ mit $n = 2k_n$. Damit gilt

$$\Delta_n = \frac{2k_n(n+1)}{2} = k_n \cdot (n+1) = k_n \cdot t_n$$

mit $t_n = n+1 > n$.

Zu (2): Es ist $n+1 \in \mathbb{G}_+$ mit $n \ge 1$. So existiert $k_n \in \mathbb{N}$ mit $n+1 = 2k_n$. Damit gilt

$$\Delta_n = \frac{n \cdot 2k_n}{2} = k_n \cdot n = k_n \cdot t_n$$

mit $t_n = n \ge n$.

Damit ist für beide Fälle die Behauptung gezeigt. □

Es folgt unmittelbar

Korollar 2.1 *Für jedes $n \in \mathbb{N}$ existiert eine (k,t)-Partition von Δ_n.* □

Das folgende Lemma gibt für jedes n eine unmittelbar bestimmbare Partition von Δ_n an.

Lemma 2.2 *Für jedes $n \in \mathbb{N}$ ist $\left(\left\lceil \frac{n}{2} \right\rceil, n + \frac{1}{2} - \left(-\frac{1}{2}\right)^{n+1}\right)$ eine Partition von Δ_n.*

Beweis *Es gilt*

$$\left\lceil \frac{n}{2} \right\rceil = \begin{cases} \frac{n}{2}, & n \in \mathbb{G}_+ \\ \\ \frac{n+1}{2}, & n \in \mathbb{U}_+ \end{cases}$$

$$n + \frac{1}{2} - \left(-\frac{1}{2}\right)^{n+1} = \begin{cases} n+1, & n \in \mathbb{G}_+ \\ n, & n \in \mathbb{U}_+ \end{cases}$$

Es folgt

$$\left\lceil \frac{n}{2} \right\rceil \cdot \left(n + \frac{1}{2} - \left(-\frac{1}{2}\right)^{n+1}\right) = \left\{ \begin{array}{ll} \frac{n}{2} \cdot (n+1), & n \in \mathbb{G}_+ \\ \\ \frac{n+1}{2} \cdot n, & n \in \mathbb{U}_+ \end{array} \right\} = \Delta_n,\ n \in \mathbb{N}$$

Das heißt: Wenn wir $k_n = \lceil \frac{n}{2} \rceil$ *und* $t_n = n + \frac{1}{2} - \left(-\frac{1}{2}\right)^{n+1}$ *wählen, gilt* $\Delta_n = k_n \cdot t_n$ *mit* $t_n \geq n$*, womit die* (k_n, t_n)*-Partition zu jedem* Δ_n *existiert.* □

Das folgende Korollar gibt für jedes $n \in \mathbb{N}$ die einzig mögliche $\left(\lceil \frac{n}{2} \rceil, n + \frac{1}{2} - \left(-\frac{1}{2}\right)^{n+1}\right)$-Partitionierung $\left\{T_j\right\}_{1 \leq j \leq \lceil \frac{n}{2} \rceil}$ von $[\, n \,]$ an.

Korollar 2.2 **a)** *Für* $n \in \mathbb{U}_+$ *ist*

$$T_1 = \{n\}, T_i = \{n - (i-1), i-1\},\ 2 \leq i \leq \frac{n+1}{2}$$

die einzige $\left(\frac{n+1}{2}, n\right)$*-Partitionierung von* $[\, n \,]$.

b) *Für* $n \in \mathbb{G}_+$ *ist*

$$T_i = \{n - (i-1), i\},\ 1 \leq i \leq \frac{n}{2}$$

die einzige $\left(\frac{n}{2}, n+1\right)$*-Partitionierung von* $[\, n \,]$. □

Mit obigen Lemmata und Korollaren kann der Satz 2.1 von Straight und Schillo (1979) verallgemeinert werden.

Satz 2.2 *Für jede natürliche Zahl* $n \in \mathbb{N}$ *existiert eine* (k,t)*-Partition von* Δ_n *und eine* (k,t)*-Partitionierung von* $[\, n \,]$. □

Bemerkung 2.1 *Wir nennen die eindeutigen Partitionierungen von Korollar 2.2 Gauß-Partitionierung, denn hier werden die Elemente von* $[\, n \,]$ *hälftig aufsteigend und hälftig absteigend übereinandergelegt, was an die vom jungen Gauß überlieferte Geschichte über die Berechnung der Summe der Zahlen von* 1 *bis* n *erinnert:*

Für $n \in \mathbb{U}_+$ *lässt sich die Partitionierung nämlich schematisch durch*

$$\begin{array}{ccccccc} T_1 & T_2 & \ldots & T_i & \ldots & T_{\frac{n+1}{2}} \\ n & n-1 & \ldots & n-i+1 & \ldots & \frac{n+1}{2} \\ 0 & 1 & \ldots & i-1 & \ldots & \frac{n-1}{2} \end{array}$$

und für $n \in \mathbb{G}_+$ *durch*

$$\begin{array}{ccccccc} T_1 & T_2 & \ldots & T_i & \ldots & T_{\frac{n}{2}} \\ n & n-1 & \ldots & n-i+1 & \ldots & \frac{n+2}{2} \\ 1 & 2 & \ldots & i & \ldots & \frac{n}{2} \end{array}$$

darstellen.

Im Folgenden geben wir eine rekursive Vorschrift zur Konstruktion der Gauß-Partitionierungen von $[\,n\,]$, $n \geq 1$, *an. Dabei bezeichnen wir mit* $T_j^{[\,n\,]}$ *die* j*-te Teilmenge der Partitionierung von* $[\,n\,]$. *Offensichtlich gilt*

$$T_1^{[\,1\,]} = \{1\}$$
$$T_1^{[\,2\,]} = \{1, 2\}$$

Mit diesem Rekursionsanfang ergibt sich für

- $n \in \mathbb{U}_+$:

$$T_1^{[\,n+1\,]} = \{1, n+1\}$$

$$T_j^{[\,n+1\,]} = \left\{\max T_{j-1}^{[\,n\,]}, \min T_{j+1}^{[\,n\,]}\right\} \quad \textit{für} \quad 2 \leq j \leq \frac{n-1}{2}$$

$$T_{\frac{n+1}{2}}^{[\,n+1\,]} = \left\{\max T_{\frac{n-1}{2}}^{[\,n\,]}, \max T_{\frac{n+1}{2}}^{[\,n\,]}\right\}$$

- $n \in \mathbb{G}_+$:

$$T_1^{[\,n+1\,]} = \{n+1\}$$

$$T_j^{[\,n+1\,]} = T_{j-1}^{[\,n\,]} \quad \textit{für} \quad 2 \leq j \leq \frac{n}{2}$$

□

2.2 Berechnung aller Partitionen

Im Allgemeinen gibt es für jedes n mehrere Paare $k, t \in \mathbb{N}$ mit $t \geq n$ und $\Delta_n = k \cdot t$. Dazu verwenden wir die Faktorisierung von Δ_n. So gilt z.B.

$$\Delta_{20} = 210 = 2 \cdot 3 \cdot 5 \cdot 7$$

Aus dieser Faktorisierung folgt, dass genau die Partitionen

$$(2, 105), (3, 70), (5, 42), (6, 35), (7, 30), (10, 21) \tag{2.1}$$

von Δ_{20} existieren; und für $n = 15$ gilt

$$\Delta_{15} = 120 = 2^3 \cdot 3 \cdot 5$$

womit genau die Partitionen

$$(2, 60), (3, 40), (4, 30), (5, 24), (6, 20), (8, 15) \tag{2.2}$$

von Δ_{15} existieren.

Lemma 2.3 *Für $n \in \mathbb{N}$ sind die Partitionen von Δ_n gegeben durch $\left(k, \frac{\Delta_n}{k}\right)$ mit*

$$k \in \left\{ x \in D^{(+)}(\Delta_n) \;\middle|\; \frac{\Delta_n}{x} \geq n \right\}$$

□

Aus Lemma 2.2 und Korollar 2.2 wissen wir, dass für $n \in \mathbb{U}_+$ die Partition $\left(\frac{n+1}{2}, n\right)$ bzw. für $n \in \mathbb{G}_+$ die Partition $\left(\frac{n}{2}, n+1\right)$ existieren. n bzw. $n+1$ sind die kleinsten t, so dass $\left(\frac{\Delta_n}{t}, t\right)$ eine Partition von Δ_n ist. Als weitere mögliche Werte für t kommen nur die echten Teiler von Δ_n größer n infrage. Damit ergibt sich das folgende Lemma.

Lemma 2.4 **a)** *Für $n \in \mathbb{U}_+$ sind die Partitionen von Δ_n gegeben durch $\left(\frac{\Delta_n}{t}, t\right)$ mit*

$$t \in \left\{ x \in D^{(+)}(\Delta_n) \mid n \leq x < \Delta_n \right\}$$

b) *Für $n \in \mathbb{G}_+$ sind die Partitionen von Δ_n gegeben durch $\left(\frac{\Delta_n}{t}, t\right)$ mit*

$$t \in \left\{ x \in D^{(+)}(\Delta_n) \mid n+1 \leq x < \Delta_n \right\}$$

□

Für zwei Partitionen (k,t) und (k',t') von Δ_n gilt $k \cdot t = \Delta_n$ und $k' \cdot t' = \Delta_n$. Je zwei Partitionen (k,t) und (k',t') von Δ_n stehen also in der Beziehung $k \cdot t = k' \cdot t'$. Daraus folgt unmittelbar

Lemma 2.5 **a)** *Sei (k,t) eine Partition von Δ_n und k' ein Teiler von t sowie $t' = k \cdot \frac{t}{k'}$, dann ist auch (k',t') eine Partition von Δ_n.*

b) *Sei (k,t) eine Partition von Δ_n und k' ein Teiler von k und $t' = \frac{k}{k'} \cdot t$, dann ist auch (k',t') eine Partition von Δ_n.* □

Dieses Lemma ist die Grundlage für den Algorithmus 1, der aus den aus Lemma 2.2 und Korollar 2.2 bekannten Partitionen $\left(\frac{n+1}{2}, n\right)$ für $n \in \mathbb{U}_+$ bzw. $\left(\frac{n}{2}, n+1\right)$ für $n \in \mathbb{G}_+$ alle weiteren Partitionen für Δ_n berechnet.

2.3 Berechnung von Partitionierungen

In diesem Abschnitt stellen wir den Algorithmus 2 vor, der für jede Partition (k,t) von Δ_n eine (k,t)-Partitonierung $\left\{T_j\right\}_{1 \leq j \leq k}$ von $[\,n\,]$ berechnet. Er basiert auf der folgenden Idee:

1. Sei N die Teilmenge von $[\,n\,]$, die noch nicht zugeteilt ist, d.h. am Anfang ist $N = [\,n\,]$.
2. T_j wird in der Reihenfolge $j = 1, 2, \ldots, k$ wie folgt befüllt: Nehme aus N die noch jeweils größte Zahl, so dass die Summe der Zahlen in T_j kleiner gleich t bleibt.

procedure CUTTINGSTICKS1(**int** n)
  if $n+1 \in \mathbb{P}$ **then**
    $k = \frac{n}{2}; t = n+1$
    write (k,t)
    for all $k' \in D^{(+)}(k)$ **do**
      $t' = \frac{k}{k'} \cdot t$
      write (k',t')
    end for
  else
    if odd(n) **then** $k = \frac{n+1}{2}; t = n$
    else $k = \frac{n}{2}; t = n+1$
    end if
    write (k,t)
    $k' =$ größter echter Teiler von t echt kleiner k
    while $k' \neq 1$ **do**
      $t' = k \cdot \frac{t}{k'}$
      write (k',t')
      $k = k'$
      $t = t'$
      $k' =$ größter echter Teiler von t echt kleiner k
    end while
  end if
end procedure

Algorithmus 1: Berechnet alle Partitionen (k,t) von Δ_n.

Beispiel 2.1 *Die Anwendung des Algorithmus auf die Eingaben* $n = 20$*,* $k = 6$ *und* $t = 35$ *ergibt die folgende Partitionierung der Menge* $[\,20\,]$*:*

$$\begin{aligned} T_1 &= \{20, 15\} \\ T_2 &= \{19, 16\} \\ T_3 &= \{18, 17\} \\ T_4 &= \{14, 13, 8\} \\ T_5 &= \{12, 11, 10, 2\} \\ T_6 &= \{9, 7, 6, 5, 4, 3, 1\} \end{aligned}$$

□

Wir betrachten im Folgenden die Mengen T_j als Folgen, in denen die Elemente absteigend aufgeführt sind, d.h. jede Menge

$$T_j = \left\{ x_{j1}, x_{j2}, \ldots, x_{jk_j} \right\} \tag{2.3}$$

soll so aufgezählt werden, dass $x_{jr} > x_{jr+1}$, $1 \leq r < k_j$, gilt (im obigen Beispiel sind die Mengen so aufgezählt).

```
procedure CUTTINGSTICKS2(int n, k, t)
    if k · t = Δ_n then
        N = [n]
        for (j = 1; j ≤ k; j++) do
            T_j = ∅
            while Σ T_j < t do
                x = max{z ∈ N | Σ T_j + z ≤ t}
                T_j = T_j ∪ {x}
                N = N − {x}
            end while
            write(T_j)
        end for
    else write(„Falsche Eingabe“)
    end if
end procedure
```

Algorithmus 2: Berechnet für eine (k,t)-Partition von Δ_n eine (k,t)-Partitionierung $\left\{T_j\right\}_{1\le j\le k}$ von $[n]$.

Folgendes Lemma hält wesentliche Eigenschaften der von Algorithmus 2 erzeugten Partitionierungen fest.

Lemma 2.6 *Die zur Eingabe n,k,t mit $k\cdot t=\Delta_n$ von Algorithmus 2 berechnete (k,t)-Partitionierung $\left\{T_j\right\}_{1\le j\le k}$ besitzt die folgenden Eigenschaften:*

(1) $x_{j1}=\max T_j$ *und* $x_{11}=n$*;*

(2) $x_{i1}>x_{j1}$ *für* $1\le i<j\le k$*;*

(3) Ist für $i\neq j$, $1\le i<j\le k$, $T_i'\subseteq T_i$ *und* $T_j'\subseteq T_j$ *sowie* $x=\max\left(T_i'\cup T_j'\right)$*, dann gilt: Ist* $\sum T_i'=\sum T_j'$*, dann muss* $x\in T_i'$ *sein.*

Beweis *(1.1) Als erste wird die Menge T_1 gefüllt. Da $t\ge n$ ist und aus N noch kein Element entnommen wurde, also $n=\max N$ zu diesem Zeitpunkt ist, kommt das Element n in die Menge T_1; es ist also $x_{11}=n$.*

(1.2) Sei $N_j=N-\bigcup_{i=1}^{j}T_i$, $1\le j\le k$, die Menge der Elemente, die nach dem j-ten Durchlauf der Schleife noch in N enthalten sind. Offensichtlich gilt $\max N_j<n\le t$ für $1\le j\le k-1$. Damit ist $x_{j+1,1}=\max N_j$. T_{j+1} erhält also als erstes Element das aktuell größte Element aus der verbliebenen Menge N; alle weiteren Elemente in T_{j+1} sind kleiner.

Mit (1.1) und (1.2) ist die Eigenschaft (1) gezeigt.

(2) Aus (1.2) folgt unmittelbar, dass $\max N_j>\max N_{j+1}$, $1\le j\le k-1$, ist, womit auch die Eigenschaft (2) gezeigt ist.

procedure CUTTINGSTICKS3($\{T_j\}_{1 \le j \le k}$)
  $fertig =$ **false**
  while not $fertig$ **do**
    if $\exists i, j \in \{1, \ldots, k\} : i < j,\ T' \subseteq T_i,\ T'' \subseteq T_j,\ \max(T' \cup T'') \in T''$ **then**
      $T_i = (T_i - T') \cup T''$
      $T_j = (T_j - T'') \cup T'$
    else $fertig =$ **true**
    end if
  end while
end procedure

Algorithmus 3: Transformation einer Partitionierung, so dass sie Eigenschaft (3) von Lemma 2.6 erfüllt.

(3) Wenn bei zwei Mengen T_i und T_j mit $i < j$ die Teilfolgen $T'_i \subseteq T_i$ und $T'_j \subseteq T_j$ dieselbe Summe ergeben, also $\sum T'_i = \sum T'_j$ ist, so können diese untereinander ausgetauscht werden. Dadurch, dass der Algorithmus Elemente in absteigender Reihenfolge aus N und somit immer die größtmögliche Zahl $x \in N$ auswählt und in T_i legt, ist $x = \max\left(T'_i \cup T'_j\right)$. Der Algorithmus wählt grundsätzlich die größte vorhandene Zahl x aus, die in die aktuelle Teilmenge T_i passt. Wenn x passt, dann würde auch eine Teilmenge aus N passen, welche dieselbe Summe besitzt wie x. Es folgt, dass der Algorithmus die Zahl x in T_i legt, anstatt eine summengleiche Teilmenge zu nehmen. Genau so verhält es sich, wenn eine Teilmenge T'_i in der Summe einer Teilmenge T'_j gleicht: $\sum T'_i = \sum T'_j$ mit $x = \max\left(T'_i \cup T'_j\right)$. Beim Durchlaufen des Algorithmus wird dieses x vor allen anderen der sich in der Vereinigungsmenge befindlichen Elementen genommen und somit T_i zugeordnet. Daraus folgt Eigenschaft (3). □

Das nächste Lemma besagt, dass zu einer (k, t)-Partition von Δ_n immer eine (k, t)-Partitionierung von $[\,n\,]$ existiert, welche die Eigenschaften (1) – (3) aus Lemma 2.6 besitzt.

Lemma 2.7 *Sei (k, t) eine Partition von Δ_n, dann existiert eine Partitionierung $\{T_j\}_{1 \le j \le k}$ von $[\,n\,]$, welche die Eigenschaften (1) – (3) aus Lemma 2.6 besitzt.*

Beweis *Gemäß des Satzes von Straight und Schillo (1979) bzw. gemäß Satz 2.2 existiert mindestens eine Partitionierung $\{T_j\}_{1 \le j \le k}$ von $[\,n\,]$. Die Mengen T_j, $1 \le j \le k$, können als absteigend sortierte Folgen betrachtet werden. Daraus ergibt sich offensichtlich die erste Eigenschaft von Aussage (1) in Lemma 2.6. Des Weiteren kann die Indexierung der Teilmengen T_j absteigend gemäß ihrer maximalen Elemente vorgenommen werden, womit die zweite Eigenschaft von Aussage (1) und Aussage (2) von Lemma 2.6 erfüllt sind.*

Wenn die Partitionierung Eigenschaft (3) nicht erfüllt, dann wenden wir auf diese Algorithmus 3 an. Solange Eigenschaft (3) nicht für alle Mengen T_j, $1 \le j \le k$, erfüllt ist, werden die dafür verantwortlichen Teilmengen dieser Mengen ausgetauscht. □

Aus den beiden Lemmata 2.6 und 2.7 folgt:

Satz 2.3 *Der Algorithmus 2 ist korrekt.* □

3 Lösen des allgemeinen Problems durch ein 0/1-Programm

In diesem Kapitel betrachten wir das allgemeine Cutting sticks-Problem als Optimierungsproblem, und wir geben ein 0/1-Programm (LP) an, mit dem es gelöst werden kann. Das heißt, wir wollen für gegebene $n, k, t_1, \ldots, t_k \in \mathbb{N}$ mit $t_j \geq n$ für $1 \leq j \leq k$ mithilfe eines LP eine $(k; t_1, \ldots, t_k)$-Partitionierung von $[\,n\,]$ berechnen. Dabei müssen natürlich die Nebenbedingungen, wie z.B. $\Delta_n = \sum_{j=1}^{k} t_j$ eingehalten werden.

Wir führen dazu binäre Variablen $x_{ij} \in \{0, 1\}$ ein mit:

$$x_{ij} = \begin{cases} 1, & i \in T_j \\ 0, & i \notin T_j \end{cases}$$

Die Lösung des folgenden LPs ist eine Matrix $\mathbf{X} = \left(x_{ij}\right)_{\substack{1 \leq i \leq n \\ 1 \leq j \leq k}}$, wobei x_{ij} angibt, ob $i \in [\,n\,]$ Element der Teilmenge T_j ist.

Die zu maximierende Zielfunktion ist definiert durch:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{k} i \cdot x_{ij}$$

Sie maximiert die Summe der genommenen Elemente aus $[\,n\,]$. Da jede Zahl aus $[\,n\,]$ laut Definition des Problems exakt einmal genommen werden muss und somit einer Teilmenge zugeordnet wird, ist das Maximum dieser Zielfunktion grundsätzlich die Dreieckszahl Δ_n von n.

Dabei müssen folgende Nebenbedingungen eingehalten werden:

(i) Für alle $j \in \{1, \ldots, k\}$ muss gelten: $\sum_{i=1}^{n} i \cdot x_{ij} = t_j$, denn die Summe der Elemente i in der Teilmenge T_j muss gleich t_j sein;

(ii) Für alle $i \in \{1, \ldots, n\}$ muss gelten: $\sum_{j=1}^{k} x_{ij} = 1$, denn das Element i muss genau einer Teilmenge T_j zugeordnet werden.

(iii) Des Weiteren muss die Gesamtbilanz stimmen: $\sum_{i=1}^{n} \sum_{j=1}^{k} x_{ij} = n$

Wir haben also insgesamt $n + k + 1$ Nebenbedingungen.

Wir zeigen an einem – aus darstellungstechnischen Gründen – sehr kleinen Beispiel die Berechnung einer Partitionierung mithilfe des obigen LP.

Beispiel 3.1 *Wir wählen $n = 5$, $k = 2$, sowie die Teilmengensummen $t_1 = 9$ und $t_2 = 6$. Die zu maximierende Zielfunktion ist*

$$\max x_{11} + x_{12} + 2 \cdot x_{21} + 2 \cdot x_{22} + 3 \cdot x_{31} + 3 \cdot x_{32} + 4 \cdot x_{41} + 4 \cdot x_{42} + 5 \cdot x_{51} + 5 \cdot x_{52}$$

mit den Nebenbedingungen:

$$\begin{aligned} 1 \cdot x_{11} + 2 \cdot x_{21} + 3 \cdot x_{31} + 4 \cdot x_{41} + 5 \cdot x_{51} &= 9 \\ 1 \cdot x_{12} + 2 \cdot x_{22} + 3 \cdot x_{32} + 4 \cdot x_{42} + 5 \cdot x_{52} &= 6 \end{aligned}$$

$$\begin{aligned} x_{11} + x_{12} &= 1 \\ x_{21} + x_{22} &= 1 \\ x_{31} + x_{32} &= 1 \\ x_{41} + x_{42} &= 1 \\ x_{51} + x_{52} &= 1 \end{aligned}$$

$$x_{11} + x_{12} + x_{21} + x_{22} + x_{31} + x_{32} + x_{41} + x_{42} + x_{51} + x_{52} = 5$$

Durch die Anwendung eines ganzzahligen LP-Solvers erhalten wir die folgende binäre Matrix

$$\mathbf{X} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Die Partitionierung der Menge $[\,5\,]$ kann daraus unmittelbar abgelesen werden:

$$\begin{aligned} T_1 &= \{4, 5\} \\ T_2 &= \{1, 2, 3\} \end{aligned}$$

Die Berechnung weiterer, größerer Partitionierungen mithilfe des obigen LP erfolgte korrekt und effizient. □

4 Zusammenfassung und Ausblick

Wir haben in diesem Papier eine Einführung in das Cutting sticks-Problem, welches wir mathematisch als Mengen-Partitionierungsproblem formulieren, gegeben. Dabei wird gezeigt, dass es für das spezielle Problem mit konstanten Summen t in den k Teilmengen zu jedem $n \in \mathbb{N}$ eine (k, t)-Partition von Δ_n sowie eine (k, t)-Partitionierung von $[\,n\,]$ gibt. Für eine Reihe von Spezialfällen haben wir unmittelbar Partitionierungen angeben können. Des Weiteren haben wir Algorithmen angegeben für die Berechnung aller (k, t)-Partitionen von Δ_n bzw. für die Berechnung einer (k, t)-Partitionierung von $[\,n\,]$ und deren Korrektheit nachgewiesen. Außerdem haben wir ein 0/1-Programm zur Berechnung von $(k; t_1, \ldots, t_k)$-Partitionierungen von $[\,n\,]$ angegeben.

Bei all diesen Betrachtungen und Algorithmen haben wir deren Komplexität vernachlässigt. Da das betrachtete Problem im Allgemeinen NP-hart ist, sind die Algorithmen nicht effizient. Es stellt sich die Frage, ob – neben den in diesem Papier betrachteten – weitere Spezialfälle oder Teilklassen von Problemen charakterisiert werden können, die effizient lösbar

sind. Eine Idee ist z.B. den (einfachen) Fall der Gauß-Partitionierung (siehe Bemerkung 2.1) auch bei anderen Fällen auszunutzen. Betrachten wir als Beispiel $n = 20$ mit der $(5, 42)$-Partition von $\Delta_{20} = 210$. Wir können die Teilmengensumme 42 aufteilen in $42 = 20 + 22$, und wir erzeugen dann zunächst anstelle der $(5, 42)$-Partitionierung von $[\,20\,]$ eine $(9; 20, 20, 20, 20, 22, 22, 22, 22, 42)$-Partitionierung, weil wir für die Teilmengensummen 20 bzw. 22 leicht Gauß-Partitionierungen finden können:

20	20	20	20	22	22	22	22	42
20	19	18	17	16	15	14	13	12
0	1	2	3	6	7	8	9	11
								10
								5
								4

Es ist klar, dass, wenn für alle Teilmengen mit der Summe 20 und allen mit der Summe 22 eine Gauß-Aufteilung gefunden wird, die restlichen Zahlen als Summe die ursprüngliche Teilmengensumme 42 ergeben müssen. Aus der gefundenen Partitionierung kann jetzt leicht eine Partitionierung für die ursprüngliche $(5, 42)$-Partition bestimmt werden, nämlich durch Vereinigung jeweils einer der Mengen mit der Summe 20 und einer mit der Summe 22:

$$\begin{aligned} T_1 &= \{6, 16, 20\} \\ T_2 &= \{1, 7, 15, 19\} \\ T_3 &= \{2, 8, 14, 18\} \\ T_4 &= \{3, 9, 13, 17\} \\ T_5 &= \{4, 5, 10, 11, 12\} \end{aligned}$$

Allgemein ist die Idee, für den Fall, dass $t \geq 2n$ ist, t aufzusplitten in $t = n \cdot y + z$ mit $n \leq z < 2n$ und damit eine

$$\big(\,(y+1)\cdot(k-1)+1; t_{1,1}, \ldots, t_{1,y}, t_{1,y+1}, \ldots, t_{k-1,1}, \ldots, t_{k-1,y}, t_{k-1,1y+1}, t_k\,\big) \qquad (4.1)$$

-Partitionierung zu bestimmen. Dabei ist:

$$t_{i,j} = \begin{cases} n, & 1 \leq j \leq y \\ z, & j = y+1 \end{cases} \text{ für jeweils } 1 \leq i \leq k-1$$

$$t_k = t$$

Wir wissen, dass es für die $t_{ij} = n$, $1 \leq i \leq k-1$, $1 \leq j \leq y$, eine Gauß-Partitionierung gibt. Falls diese auch für die $t_{iy+1} = z$, $1 \leq i \leq k-1$, existieren, haben wir insgesamt einfach eine Lösung für (4.1) gefunden, die für die Bestimmung einer (k, t)-Partitionierung geeignet vereinigt werden kann.

Es bleibt zu untersuchen, für welche Fälle dieser Ansatz für eine effiziente Berechnung einer (k, t)-Partitionierung von $[\,n\,]$ genutzt werden kann. Des Weiteren kann man versuchen, die

Probleminstanzen in Klassen einzuteilen, so dass möglicherweise Klassen entstehen, deren Instanzen effizient gelöst werden können. Wenn man eindeutige Charakterisierungen findet, kann man versuchen, die schwierigen Instanzen mit Ansätzen wie Randomisierung oder mit Heuristiken zu lösen.

Literatur

Ando, K., Gervacio, S., Kano, M.: Disjoint Subsets of Integers having a constant Sum, *Discrete Mathematics 82,* 1990, 7 - 11

Chen, F.-L., Fu, H.-L., Wang, Y., Zhou, J.: Partition of a Set of Integers into Subsets with prescribed Sums, *Taiwanese Journal of Mathematics 9,* 2005, 629 - 638

Fu, H.-L., Hu, W.-H.: A Special Partition of the Set I_n, *Bulletin of the Institute of Combinatorics and its Applications 6,* 1992, 57 - 60

Jagadish, M.: An Approximation Algorithm for the Cutting-Sticks Problem, *Information Processing Letters,* 2015, 170 - 174

Straight, H. J., Schillo, P.: On the Problem of Partitioning $\{1, \ldots, n\}$ into Subsets having equal Sums, *Proceedings of the American Mathematical Society 74(2),* 1979, 229 - 231