

Algorithmen und Datenstrukturen für ein auf GUIDO Music Notation basierendes Notensatzsystem

Dr. Kai Renz
renz@noteserver.org

Abstract: Die vorliegende Dissertation behandelt die Fragestellung, inwieweit die kürzlich entwickelte text-basierte Musikrepräsentationssprache „GUIDO Music Notation“ automatisch in traditionellen Notensatz umgewandelt werden kann, bzw. welche Algorithmen und Datenstrukturen für eine vollständig automatische Umwandlung notwendig sind. Da GUIDO nicht von vorneherein auf den Notensatz beschränkt ist, sondern vielmehr einen generellen Musikrepräsentationsformalismus darstellt, ist die Umwandlung von beliebigen GUIDO-Beschreibungen in konventionellen Notensatz im Allgemeinen keine leichte Aufgabe.

Zunächst wird die dreigeteilte Struktur von GUIDO erläutert und andere Musikrepräsentationssprachen mit GUIDO verglichen. Anschließend wird gezeigt, wie GUIDO Beschreibungen in eine geeignete Datenstruktur umgewandelt werden können und wie Algorithmen für den Notensatz auf dieser Datenstruktur operieren.

Die Algorithmen für den automatischen Zeilenausgleich sowie den Zeilen- und Seitenumbruch bilden einen Schwerpunkt der vorliegenden Arbeit. Der bisher bekannte Zeilenausgleichsalgorithmus wurde so verbessert, dass auch komplexe Rhythmen optisch korrekt dargestellt werden. Zusätzlich wurde ein Algorithmus für einen automatischen *Seitenausgleich* neu entwickelt, der hier erstmals vorgestellt wird.

Die beschriebenen Algorithmen und Datenstrukturen bilden die Grundlage für verschiedene, im Rahmen dieser Dissertation entwickelte, Notensatzanwendungen. Dazu zählt auch ein kostenloser Internet-Service, der völlig plattformunabhängig GUIDO-Beschreibungen in konventionellen Notensatz umwandelt.

1 Einleitung

Die heutzutage allgemein verwendete konventionelle Notenschrift entwickelte sich über einen langen Zeitraum, dessen Anfänge noch deutlich vor der Erfindung von Druckverfahren liegen. Dabei spielte Guido d’Arezzo (★ ca. 992, † ca. 1050) eine äußerst wichtige Rolle: Er hatte als erster die Idee, Notenköpfe *auf* und *zwischen* Notenlinien zu platzieren, um die entsprechende Tonhöhe anzuzeigen. Dies war die Geburtsstunde der heutzutage üblichen zwei-dimensionalen Struktur der graphischen Notenschrift: Wie in Abbildung 1 zu sehen ist, wird die Tonhöhe durch die vertikale Platzierung auf und zwischen den Linien der Notenzeile angegeben. Die horizontale Position beschreibt die zeitliche Abfolge; gleichzeitig klingende Töne haben die gleiche horizontale Position.

Im Laufe ihrer Geschichte wurde die Notenschrift kontinuierlich an die wachsenden Anforderungen durch komplexere Kompositionen und neue Instrumente angepasst. Die um 1500 erfundenen Buchdrucktechniken wurden ebenfalls zunehmend für das Erstellen

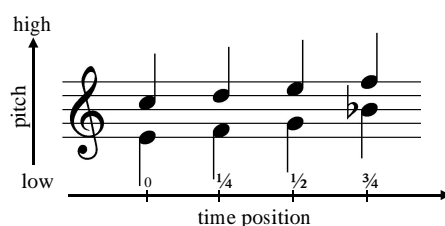


Abbildung 1: Zwei-dimensionale Struktur der konventionellen Notenschrift

von Noten benutzt. Die Drucktechnik, die von ca. 1600 bis zur Erfindung von computergesteuerten Druckern für die Erstellung von Noten verwendet wurde, heißt *Notenstich* (engl. engraving). Im Notenstich werden die Noten in eine weiche Kupferplatte graviert, wobei eine Vielzahl von speziellen Werkzeugen und Stempeln verwendet wird. Die Qualität und Lesbarkeit des so erzeugten Notenbildes hängt sehr stark von der Erfahrung des Notenstechers ab. Oftmals gibt es verschiedene Möglichkeiten, eine musikalische Idee in Noten abzubilden: „Notenschrift ist nicht in sich logisch konsistent. Ihre visuelle Grammatik hat keine definierte Grenze – viel stärker als in jedem numerischen System oder jeder formalen Sprache.“¹

Die Erfindung von Computern und hoch auflösenden Laserdruckern hat die Art und Weise, wie Noten produziert werden, radikal geändert. Fast alle Noten werden heutzutage mit der Hilfe von Computerprogrammen gesetzt; dennoch muss festgestellt werden, dass immer noch ein hoher Anteil an menschlicher Eingabe erforderlich ist, um Partituren einer hohen Qualität zu erzeugen.²

Die Fragestellungen und Probleme beim vollautomatischen Notensatz sind vielfältig. Don Byrd, der bereits 1984 eine Dissertation zu diesem Thema verfasst hat, definiert drei Hauptaufgaben, die jedes Notensatzsystem lösen muss: *Auswahl*, *Positionierung* und *Drucken* der musikalischen Symbole [By84]. Alle drei Aufgaben können nur dann befriedigend gelöst werden, wenn das Wissen und die Erfahrung von Notenstechern in eine computergeeignete Form überführt werden kann. Verschiedene Ansätze für diesen Wissenstransfer wurden bisher versucht; die meisten Versuche verwenden einen Beispiel- bzw. Regel-basierten Ansatz um automatisch die notwendigen musikalischen Symbol auszuwählen und zu platzieren [Gi01].

Obwohl eine Reihe von kommerziellen und nicht-kommerziellen Notensatzsystemen existieren, gibt es nach wie vor kein weit verbreitetes, standardisiertes Austauschformat für den Notensatz. Das einzige Format, das sich weltweit für den Austausch von Musik auf Notenebene durchsetzen konnte, ist MIDI, das allerdings nicht als Austauschformat für den Notensatz geeignet ist, da eine Reihe von notensatzrelevanten Informationen nicht mittels MIDI übertragen werden können.

Als Musikrepräsentationsformat für das im Rahmen dieser Arbeit entwickelten Notensatzsystems wird GUIDO Music Notation verwendet, ein vom Menschen direkt lesba-

¹Originalzitat: „Musical notation is not logically self-consistent. Its visual grammar is open-ended – far more so than any numerical system of formal grammar.“[SF97] (p. 15)

²Selbst heute gibt es Musikverlage, die (menschliche) Notenstecher beschäftigen.

res, adäquates Format, das nicht primär auf Aspekte des Notensatzes fixiert ist, sondern vielmehr die Repräsentation von musikalischen, strukturellen sowie notationellen Informationen zulässt [HHRK01]. Abbildung 2 zeigt ein intuitiv verständliches Beispiel einer GUIDO-Beschreibung zusammen mit der dazugehörigen konventionellen Notenschrift.³

```
[ \clef<"treble"> \key<"D"> \meter<"4/4">
a1*1/2 b a/4. g/8 f#/4 g a/2 b
a/4. g/8 f#/4 g a/2 a b c#2/4 d
c#/2 b1 a/1 ]
```




Abbildung 2: Ein einfaches GUIDO Beispiel

Die Umwandlung von GUIDO-Dateien in konventionelle Notenschrift ist keineswegs trivial, da eine *beliebige* GUIDO-Beschreibungen keinerlei Formatierungsanweisungen enthalten muss. Diese Information muss erst aus den vorhandenen Daten inferiert werden. Ein wesentlicher Punkt der Arbeit lag in der Erkennung, Klassifizierung und Implementierung von Musik-Notations-Algorithmen. Die generelle Vorgehensweise wird in Abbildung 3 dargestellt.

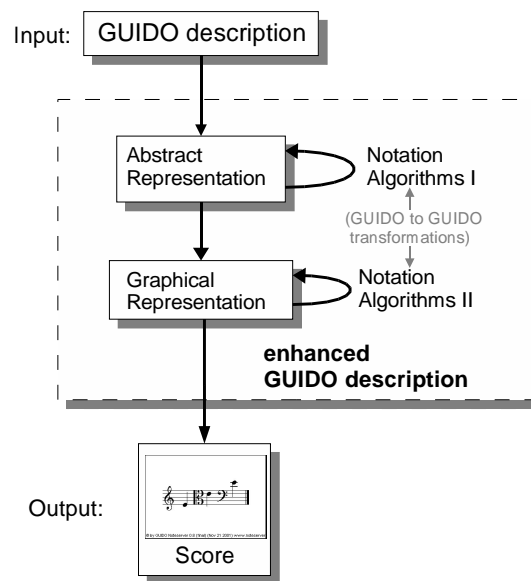


Abbildung 3: Allgemeiner Ansatz zur Konvertierung von GUIDO-Beschreibungen in konventionelle Notenschrift

Für die effiziente Bearbeitung von GUIDO-Beschreibungen im Rechner wurde eine objekt-orientierte Datenstruktur mit dem Namen „Abstract Representation“ spezifiziert

³Der englische Begriff [c]lef ist der Notenschlüssel, [key] ist die Tonart und [meter] ist die Taktart.

und implementiert. Um eine graphische Notenausgabe zu erzeugen wurde eine weitere objekt-orientierte Klassenbibliothek, die „Graphical Representation“ entwickelt. Alle Symbole einer Partitur haben eine Entsprechung in der „Graphical Representation“. Zwei Schritte sind notwendig, um eine beliebige GUIDO-Beschreibung in konventionelle Notenschrift umzuwandeln: zuerst muss die die GUIDO-Beschreibung in eine Instanz der „Abstract Representation“ umgewandelt werden; die Daten dieser Instanz werden mit den Musiknotations-Algorithmen manipuliert, um dann eine Instanz der „Graphical Representation“ zu erzeugen. Dabei werden die musikalischen Symbole der Partitur erzeugt. Die exakte Platzierung dieser Elemente erfordert ausgefeilte Algorithmen, vor allem im Bereich des Zeilenausgleichs, des Zeilen- und Seitenumbruchs sowie des Seitenausgleichs.

Das entstandene Musiknotationssystem ist frei verfügbar. Es existiert nicht nur eine statische Anwendung sondern zusätzlich ein frei verfügbarer Internet-Dienst, mit dem unter Verwendung von üblichen Web-Browsern aus GUIDO-Beschreibungen konventionelle Notenschrift erzeugt werden kann [RH01].⁴

2 Das GUIDO Music Notation Format

GUIDO Music Notation ist eine Musikbeschreibungssprache, die seit 1996 entwickelt wird, und mittlerweile in einigen Projekten der Musikinformatik als Datenformat eingesetzt wird. GUIDO wurde bewusst nicht nur als Beschreibungssprache für Notenschrift entwickelt, sondern stellt vielmehr einen allgemeinen Formalismus für die Repräsentation von musikalischem Material dar. GUIDO ist ein textbasiertes (d.h. im Unterschied zu binären Formaten direkt durch einen menschlichen Benutzer lesbares) Datenformat, das in drei aufeinander aufbauende Schichten (*Basic*, *Advanced*, und *Extended* GUIDO) aufgeteilt ist:

Basic GUIDO Mittels *Basic* GUIDO können alle relevanten Aspekte von „einfacher“ Musik, seien es primitive musikalische Objekte wie einzelne Töne oder Pausen, bis hin zu kompletten, mehrstimmigen Stücken zusammen mit dynamischen Markierungen oder anderen Artikulationszeichen, repräsentiert werden.

GUIDO besitzt zwei grundlegende syntaktische Elemente: Ereignisse (engl. *events*) und Markierungen (engl. *tags*). Ein *Event* beschreibt ein musikalisches Ereignis, das mit einer Dauer assoziiert werden kann (i.A. Noten und Pausen). *Tags* werden zur Definition von musikalischen Attributen (z.B. Taktart, Tonart, Vorzeichen, Bindebögen, etc.) verwendet. Noten sind durch ihren Namen, eventuelle Alterierungen, ihre Oktave und ihre Dauer eindeutig beschreibbar. Eine komplette Notenbeschreibung der Form „c#1 * 1 / 4.“ beschreibt die punktierte Viertel-Note „cis“ in der ersten Oktave (gerade über dem eingestrichenen C). Die Informationen von Oktave und Dauer werden von der vorhergehenden Note übernommen, falls sie nicht spezifiziert sind.

GUIDO besitzt zwei orthogonale Konstrukte, um musikalische Ereignisse in grössere Strukturen einzubetten: Sequenzen und Segmente. Sequenzen – begrenzt mittels eckiger Klammern '[' und ']' – bestehen aus einer Reihe von zeitlich hintereinanderfolgenden Ereignissen und Markierungen. Dahingegen werden Segmente – begrenzt mittels geschweif-

⁴Dieser Internet-Service ist unter der URL <http://www.noteserver.org> zu erreichen.

ten Klammern '{' und '}' – aus *gleichzeitig* ablaufenden Ereignissen aufgebaut; die einfachste Form eines Segments ist ein Akkord, in dem mehrere Töne gleicher Länge gleichzeitig erklingen. Die beiden Konstruktionen entsprechen genau der zwei-dimensionalen Struktur konventioneller Notenschrift: einzelne Stimmen einer Partitur werden durch Sequenzen repräsentiert, während Segmente die horizontale (d.h. gleichzeitige) Schichtung der Stimmen darstellen. Tags werden innerhalb von *Basic GUIDO* verwendet, um allgemein übliche musikalische Attribute zu repräsentieren.

Abbildung 4: Vergleich der Musikrepräsentationssprachen

Format	Fokus: Aufführung, Logisch und/oder Notation	Unter- stützt exakte Formatie- rung	Verwend- bar als Austausch- format	Intuitiv zu lesen	Einfach er- weiterbar	Nach GUIDO übersetz- bar
Midi ^a	A	nein	ja ^b	nein	partiell	ja ^c
DARMS	N	nein	ja	nein ^d	nein	ja
MuseData	L & N	ja	ja	nein ^d	nein	ja
SCORE	N	ja	nein	nein	nein	ja
cmn	N	ja	nein ^e	ja	ja	ja ^e
PaE-Code ^f	N	nein	ja	nein ^d	nein	ja
MusiX _{TeX}	N	ja	nein	nein	nein	nein ^g
LilyPond	N ^h	ja	nein ⁱ	ja	ja	partiell ⁱ
NIFF ^a	N	ja	ja	nein	nein	ja
SMDL	L & N	ja	ja	nein	ja	partiell ^k
XML	L & N ^l	ja ^l	ja	ja ^l	ja	ja ^l
GUIDO	L & N	ja	ja	ja	ja	ja

^a Binäres Format

^b MIDI ist das am meisten verwendete Format für den Austausch von Musik auf Strukturebene. Notensatzinformationen lassen sich nur sehr eingeschränkt übertragen.

^c Die Umwandlung kann zusätzliche Berechnungen benötigen, um eine lesbare Notation zu erhalten

^d Abhängig vom Kenntnisstand

^e cmn Beschreibungen sind LISP Ausdrücke und erfordern einen LISP Interpreter

^f Plaine and Easie Code

^g MusiX_{TeX} enthalten wenig Informationen über die musikalische Struktur

^h Falls notwendig kann die Eingabe so kodiert werden, dass logische Informationen extrahierbar sind

ⁱ LilyPond Dateien werden zunächst von einer funktionalen Programmiersprache (Scheme) interpretiert

^k SMDL kann mehr Informationen speichern als normalerweise in einer GUIDO Beschreibung enthalten sind

^l Hängt von der implementierten DTD (Document Type Definition) ab

Advanced GUIDO Die auf *Basic GUIDO* aufbauende Schicht heißt *Advanced GUIDO* und zeichnet sich vor allem durch die Möglichkeit, ein exaktes Layout der Notenseiten

sowie die Formatierungen aller musikalischer Symbole exakt anzugeben, aus [HHR99]. Dies wird zum einen durch neue Parameter von bereits in *Basic* GUIDO definierten Tags sowie durch Einführung neuer Tags realisiert.

Extended GUIDO Um auch Merkmale von nicht-konventioneller Musik in GUIDO repräsentieren zu können, wurde als vorerst letzte Schicht *Extended* GUIDO definiert. Mit *Extended* GUIDO lassen sich unter Anderem mikrotonale (Ver-)Stimmungen und Skalen einfach definieren. Desweiteren ist es möglich, die Dauer von Ereignissen exakt (und damit absolut) in Millisekunden anzugeben (eine Note 'c' von einer Sekunde Dauer lässt sich dann folgendermaßen kodieren: $c * 1000ms$). Zusätzlich werden hierarchische Partituren durch eine beliebige Schachtelung von Sequenzen und Segmenten ermöglicht. Alle diese Merkmale haben im konventionellen Notensatz noch keine standardisierte Entsprechung, weshalb sie in der vorliegenden Arbeit nicht weiter berücksichtigt werden.

In der vollständigen Dissertation wird GUIDO ausführlich mit elf anderen Musikrepräsentationssprachen verglichen und die wesentlichen Gemeinsamkeiten und Unterschiede werden detailliert anhand von Beispielen herausgearbeitet. Abbildung 4 fasst diese Ergebnisse zusammen. Einen guten Überblick über Musikrepräsentationssprachen gibt [SF97]. Insgesamt lässt sich feststellen, dass es durchaus andere leistungsfähige, für den Notensatz geeignete Musikrepräsentationssprachen gibt, dass GUIDO aber viele Vorteile anderer Formate in sich vereint und deshalb ein guter Kandidat für die Grundlage eines Musiknotationssystems ist.

3 Algorithmen für den Notensatz

Um GUIDO-Beschreibungen in konventionellen Notensatz umzuwandeln, sind eine Reihe von Schritten notwendig, die bereits in Abbildung 3 schematisch dargestellt wurden. Zunächst wird die (textbasierte) GUIDO-Beschreibung in die sogenannte „Abstract Representation“, übersetzt. Anschließend werden eine Reihe von Musiknotations-Algorithmen ausgeführt, um die Daten innerhalb der „Abstract Representation“ auf die Umwandlung in die sogenannte „Graphical Representation“ vorzubereiten. Die „Graphical Representation“ ist eine weitere objekt-orientierte Datenstruktur, deren Objekte eng mit den Elementen einer Partitur (Noten, Zeilen, Seiten, etc.) verbunden sind. Die Umwandlung der „Abstract Representation“ in die „Graphical Representation“ geschieht in zwei Schritten: Zunächst werden die graphischen Elemente (z.B. Noten und Pausen, Bindebögen, Artikulationszeichen, etc.) erzeugt. Anschließend werden die Zeilen- und Seitenumbrüche sowie der jeweils notwendige Zeilenausgleich berechnet.

3.1 Musiknotations-Algorithmen I

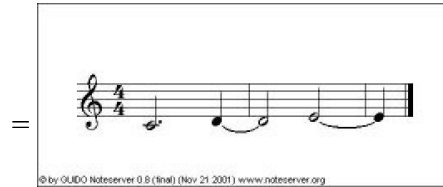
Eine Reihe von Musiknotations-Algorithmen manipulieren die Daten der „Abstract Representation“, um eine graphische Darstellung überhaupt erst zu ermöglichen. Dabei wird sichergestellt, dass jedes Objekt innerhalb der „Abstract Representation“ durch genau ein graphisches Symbol auf einer Notenseite dargestellt werden kann. Ein Beispiel für diese Vorgehensweise ist in Abbildung 5 zu sehen: In der ursprünglichen GUIDO-

Beschreibung ist die Taktart (`\meter<"4/4">`) angegeben, die es erforderlich macht, das in der graphischen Partitur Taktstriche (engl. bar lines) automatisch eingefügt werden. Dies erfordert wiederum, dass einige Noten aufgeteilt und durch Bindebögen (engl. tie) verbunden werden.

(a) [`\meter<"4/4"> c*3/4 d e]`

↓

(b) [`\clef<"treble"> \meter<"4/4">
c*3/4 \tie(d*1/4 \bar d*2/4)
\tie(e*2/4 \bar e*1/4)]`



(c)

Abbildung 5: Einfache GUIDO-nach-GUIDO-Transformation

Es wurden insgesamt acht Algorithmen identifiziert und implementiert, die dafür sorgen, dass im Anschluss an ihre Ausführung eine enge Koppelung zwischen den Daten der „Abstract Representation“ und den graphischen Elementen einer Notenseite sichergestellt ist.

3.2 Musiknotations-Algorithmen II

Wenn die graphischen Elemente der Partitur erzeugt worden sind muss als nächstes der *Positionierungs*-Schritt folgen. In diesem Zusammenhang müssen die Notationselemente der „Graphical Representation“ so auf Zeilen und Seiten verteilt werden, dass ein „ansprechendes“ und dem musikalischen so wie dem rhythmischen Verlauf entsprechendes Notenbild entsteht.

3.2.1 Zeilenausgleich (Spacing)

Der Zeilenausgleich beschreibt den Prozess, in dem der Zwischenraum zwischen den Noten, Pausen und anderen graphischen Elementen den Notendauern entsprechend verteilt wird. Traditionell wird den Noten (und Pausen) in Abhängigkeit von ihrer Dauer d ein bestimmtes *Hinterfleisch* $space(d)$ zugeteilt, das durch eine logarithmische Funktion bestimmt werden kann:

$$\psi(d) = 1 + \log_2 \left(\frac{d}{d_{\min}} \right) \quad \text{und} \quad space(d) = \psi(d) \cdot space(d_{\min}) \quad (1)$$

wobei d die Notendauer und d_{\min} die kleinste vorkommende Dauer und $space(d_{\min})$ das vordefinierte Hinterfleisch einer Note der kleinsten Dauer ist.

In den meisten computergestützten Notensatzsystemen wird für den Zeilenausgleich ein sogenanntes Feder-Stab-Modell verwendet, bei dem zwischen den Elementen einer Notenzeile (gedachte) Federn gespannt werden, die mittels einer berechneten Kraft auf

die gewünschte Zeilenlänge gezogen werden⁵ [Go87]. Um eine horizontale Überlappung auszuschließen, werden (gedachte) Stäbe eingesetzt, die den Federn eine Mindestlänge (und damit eine Grundspannung) vorgeben.

Die bisher veröffentlichten Formeln für die Berechnung der Federkonstanten führen beim Zusammenspiel von komplexeren Rhythmen in verschiedenen Stimmen zu sichtbaren Fehlern, die ein menschlicher Notenstecher vermeiden würde. Im Zuge der Dissertation wurde diese Berechnung verbessert [Re02]. Abbildung 6 zeigt die Ausgleichsberechnung mit dem ursprünglichen und dem verbesserten Algorithmus. Es ist deutlich zu sehen, dass die Abstände zwischen den Noten der ersten Stimme im Fall (b) absolut gleichmäßig sind, während sie im Fall (a) ungleichmäßig sind.

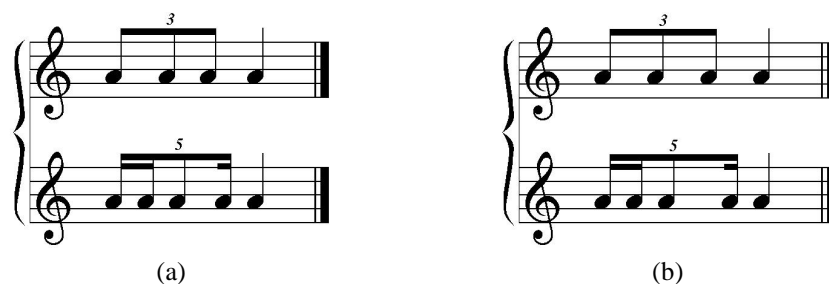


Abbildung 6: Gourlays Algorithmus (a) verglichen mit dem verbesserten Ausgleichs-Algorithmus (b)

3.2.2 Zeilenumbruch

Zeilenumbruch (engl. line breaking) beschreibt die Aufteilung eines Musikstückes in einzelne Notenzeilen. Dabei dürfen die entstehenden Notenzeilen nicht zu eng, aber auch nicht zu weit gesetzt sein; des weiteren muss auch die letzte Notenzeile ausgeglichen gesetzt werden (im Gegensatz zum Textsatz, bei dem die letzte Zeile eines Absatzes nicht vollständig gefüllt sein muss).

Interessanterweise lässt sich der von Knuth beschriebene optimale Zeilenumbruch im Textsatz relativ einfach auf den Notensatz übertragen [HG87]. Beim optimalen Zeilenumbruch wird für jede potentielle Zeile ein Strafwert (engl. penalty) berechnet. Ein Zeilenumbruch ist *optimal*, wenn die Summe der Strafwerte über alle Zeilen minimal ist. Bei der Übertragung von Knuths Algorithmus auf den Zeilenumbruch beim Notensatz muss zunächst eine Bewertungsfunktion für potentielle Notenzeilen bestimmt werden. Im Rahmen der Dissertation wurde dazu die sogenannte „Space-Force-Function“ (abgekürzt *sff*) entwickelt, die für eine potentielle Notenzeile diejenige Kraft berechnet, die notwendig ist, um die Zeile auf die gewünschte Zeilenlänge zu ziehen. Der Strafwert für eine Notenzeile wird dann als Differenz zwischen dem durch die *sff* berechneten Wert und einer vorgegebenen „optimalen“ Kraft berechnet. Abbildung 7 zeigt die „Space-Force-Function“ für eine Notenzeile, die mit unterschiedlichen Kräften gezogen wird.

⁵Es gilt das Hook'sche Federgesetz ($F = c \cdot x$ mit F : Kraft, c : Federkonstante, x : Ausdehnung)

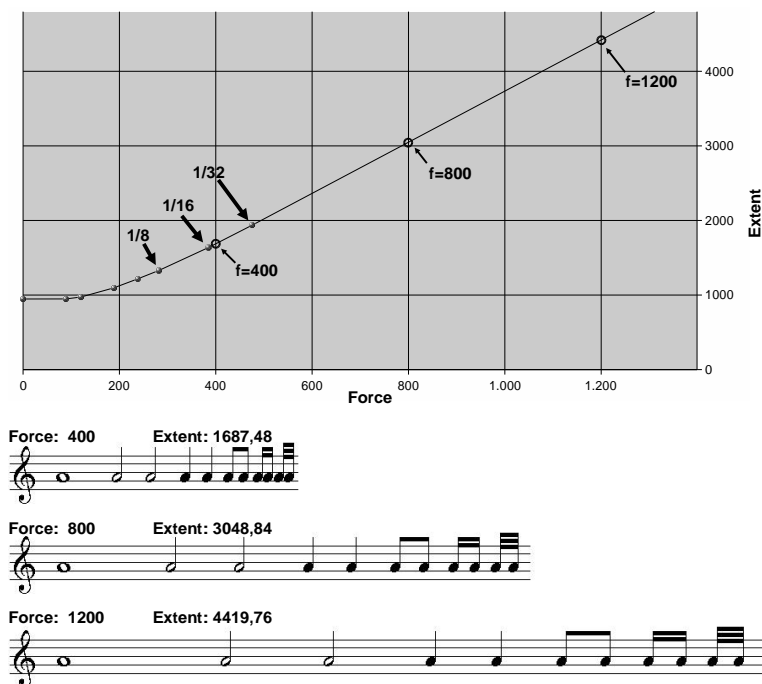


Abbildung 7: Die „Space-Force-Funktion“

3.2.3 Seitenausgleich

Der aus dem Textsatz weitestgehend übernommene Zeilenumbruchsalgorithmus hat den Nachteil, dass die Zeilen fortlaufend auf die Seiten verteilt werden. Im Notensatz endet ein Stück allerdings meistens *am Ende* der letzten Seite. Im Rahmen der vorliegenden Dissertation wurde der Zeilenumbruchsalgorithmus so erweitert, dass nunmehr auch ein optimaler Seitenausgleich berechnet werden kann. Dabei wird eine Notenseite in eine feste Anzahl von Bereichen eingeteilt und der Zeilenumbruchsalgorithmus jeweils separat für alle möglichen Bereiche berechnet. Diese Berechnung verlängert zwar die Laufzeit des Algorithmus um einen konstanten Faktor, ist aber frei skalierbar und liefert für übliche Musikstücke sehr gute Resultate.

Abbildung 8 zeigt drei verschiedene Möglichkeiten, wie die ersten acht Takte der Bach Fuge (BWV 846) auf einer Seite, die in 12 Bereiche eingeteilt wurde, umgebrochen werden können. In Abbildung 8 unten werden nur zwei Notenzeilen und fünf Bereiche benötigt, um die ersten acht Takte zu setzen. Die Gesamtstrafe hierfür ist der sehr hohe Wert 1735, was aber bei dem gezeigten sehr engen Notensatz nicht weiter verwundert. In der Mitte von Abbildung 8 endet der achte Takt im siebten Bereich; die Gesamtstrafe ist hier 797. Auf der oben abgebildeten Seite von Abbildung 8 endet der achte Takte im neunten Bereich mit einer Gesamtstrafe von 618. Welcher Umbruch letztendlich verwendet wird, hängt nur von der Minimierung der Gesamtstrafe am Ende des Stückes ab.



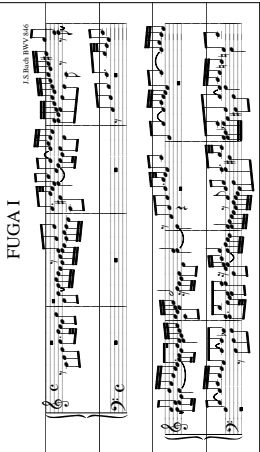


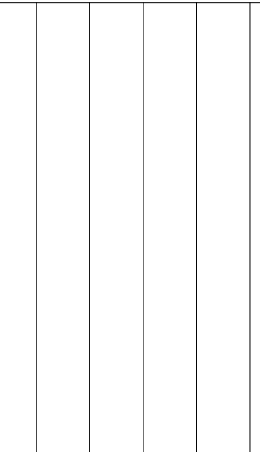
<p>FUGA I</p> 	<p>FUGA I</p> 	<p>FUGA I</p> 
		
<p>(c) Kai Renz, musical data taken from MusiData database and automatically converted to GUIDO</p>	<p>(c) Kai Renz, musical data taken from MusiData database and automatically converted to GUIDO</p>	<p>(c) Kai Renz, musical data taken from MusiData database and automatically converted to GUIDO</p>

Abbildung 8: Drei Möglichkeiten für den Umbruch der ersten acht Takte von BWV 846

4 Zusammenfassung

In der vorliegenden Dissertation wurden Algorithmen und Datenstrukturen für ein Musiknotationssystem vorgestellt, mit dem Daten aus dem textbasierten GUIDO Music Notation Format in konventionellen Notensatz umgewandelt werden können. Das vorliegende System unterscheidet sich von vorhandenen System darin, dass sämtliche Musiknotations-Algorithmen als GUIDO-nach-GUIDO-Transformationen beschrieben werden können, d.h. die Eingabe ist eine (in den meisten Fällen nicht vollständig formatierte) GUIDO-Beschreibung, die durch die beschriebenen Musiknotations-Algorithmen in eine erweiterte GUIDO-Beschreibung umgewandelt wird, die sämtliche für die Formatierung einer Notenseite wesentlichen Informationen enthält. Die im Zuge der Dissertation entwickelten Algorithmen für den Zeilenausgleich und den Seitenausgleich verbessern bisherige Algorithmen oder werden hier erstmals vorgestellt. Das resultierende Notensatzsystem ist frei verfügbar und kann sowohl als alleinstehende Anwendung sowie als Dienstleistung im Internet verwendet werden.

Literatur

- [By84] Byrd, D.: *Music Notation by Computer*. PhD thesis. Department of Computer Science, Indiana University. 1984.
- [Gi01] Giesecking, M.: *Code-basierte Generierung interaktiver Notengraphik*. PhD thesis. Universität Osnabrück. 2001.
- [Go87] Gourlay, J. S.: *Spacing a Line of Music*. Technical report. Ohio State. 1987.
- [HG87] Hegazy, W. A. und Gourlay, J. S.: *Optimal line breaking in music*. Technical Report OSU-CISRC-8/87-TR33. The Ohio State University. 1987.
- [HHR99] Hoos, H. H., Hamel, K. A., und Renz, K.: *Using Advanced GUIDO as a Notation Interchange Format*. In: *Proceedings of the 1999 International Computer Music Conference*. S. 395–398. Tsinghua University, Beijing, China. 1999. International Computer Music Association.
- [HHRK01] Hoos, H. H., Hamel, K., Renz, K., und Kilian, J.: *Representing Score-Level Music Using the GUIDO Music-Notation Format*. In: Hewlett, W. B. und Selfridge-Field, E. (Hrsg.), *The Virtual Score*. volume 12 of *Computing in Musicology*. chapter 5. The Center for Computer Assisted Research in the Humanities and the MIT Press. 2001.
- [Re02] Renz, K.: *An Improved Algorithm for Spacing a Line of Music*. In: *Proceedings of the ICMC 2002*. 2002.
- [RH01] Renz, K. und Hoos, H. H.: *WEB delivery of Music using the Guido NoteServer*. In: *Proceedings of the First International Conference on WEB Delivering of Music – WEDELMUSIC*. S. 193. IEEE Computer Society. 2001.
- [SF97] Selfridge-Field, E. (Hrsg.): *Beyond MIDI – The Handbook of Musical Codes*. The MIT Press, Cambridge, Massachusetts, London, England. 1997.