Rhythm Inference Helping Writing Music Scores

François Schwarzentruber

Univ Rennes, IRISA, CNRS

CNIA 2024
Editing rhythms in graphical user interface...

Tedious to enter the rhythm for each note.
Editing rhythms in Lilypond...

Tedious to enter the rhythm for each note.
EDITING RHYTHMS IN ABC...

X: 3
T: Amazing Grace
R: waltz
M: 3/4
L: 1/8
K: Dmaj
V:1 Ad|"D"d4 fe/d/|"D"f4 fe|"G"d4 B2|"D"A4 Ad|
V:2 d2|A4 dB/A/|d4 AB|B4 d2|f4 df|
V:1 Bm"d4 fe/d/|"E7" f4 ef|"Asus" a6|"A/G"a4 fa|
V:2 f4 dB/A/|B4 Bd|d6|c4 df|
V:1 "D"a4 fe/d/|"D"f4 fe|"G"d4 B2|"D"A4 Ad|
V:2 f4 dB/A/|d4 AB|B4 d2|f4 df|
V:1 "Bm"d4 fe/d/|"E7"f4 "G/A"e2|"D" d6|"D"D4|
V:2 f4 dB/A/|B4 G2|F6|F4|

Tedious to enter the rhythm for each note.
Our contribution

- A language called *abcd* closer to a real graphical score
  → Musical language like *Markdown*

- Easy to read
  → Rhythm is only *partially specified*

https://github.com/francoisschwarzentruber/abcd
demo
Outline

1. Rhythm inference
2. Our model: a mathematical program
3. Future work
Problem definition

Definition (Rhythm inference)

- input:
  - approximative durations \( \hat{\delta}_1, \ldots, \hat{\delta}_n \in \mathbb{R}^+ \),
  - finite domains \( \Delta_1, \ldots, \Delta_n \subseteq \mathbb{R} \),
    \[
    \Delta_1 = \{ 3/2, 3/4, 3/8, 3/16, 3/32, 3/64 \}
    \]
    \[
    \Delta_2 = \{ 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64 \}
    \]
  - duration \( T \) of a measure;
- output: inferred durations \( \delta_1, \ldots, \delta_n \) that form a solution of a the mathematical program we are going to define!
  \[
  \hat{\delta}_1 = 5, \hat{\delta}_2 = 1
  \]
  \[
  \Delta_1 = \{ 3/2, 3/4, 3/8, 3/16, 3/32, 3/64 \}
  \]
  \[
  \Delta_2 = \{ 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64 \}
  \]
  \[
  T = 1
  \]
  \[
  \delta_1 = 3/4, \delta_2 = 1/4
  \]
Outline

1. Rhythm inference
2. Our model: a mathematical program
3. Future work
Consistency

\[ \delta_i \in \Delta_i \text{ for all } i = 1..n \quad (1) \]

\[ \sum_{i=1}^{n} \delta_i = T \quad (2) \]
Taking the approximate durations into account

\[
\begin{align*}
\text{minimize} & \quad \sum_{i,j=1}^{n} \text{err}_{ij} \\
\delta_i & \in \Delta_i \text{ for all } i = 1..n \\
\sum_{i=1}^{n} \delta_i & = T \\
\delta_j - \delta_i & \leq \text{err}_{ij} \text{ if } \hat{\delta}_i \geq \hat{\delta}_j \\
\text{err}_{ij} & \geq 0
\end{align*}
\]

Technical trick

It is then transformed into a Mixed-Integer linear program by:
- introducing variables \( x_{id} = 1 \) for all \( i = 1..n \), for all \( d \in \Delta_i \)
- adding the constraint \( \sum_{d \in \Delta_i} x_{id} = 1: \delta_i \text{ takes a unique value} \)
- replacing each occurrence of \( \delta_i \) by \( \sum_{d \in \Delta_i} d \times x_{id} \).
Outline

1. Rhythm inference
2. Our model: a mathematical program
3. Future work
Future work: modeling

- Improving the inference
  \[\rightarrow\] Improving the model

- Take the existing score into account
  \[\rightarrow\] Automatically take a previous rhythm in the score
Future work: algorithmic questions

- Some input, e.g. $4/4 \ a \ b \ c \ |$ may have several solutions
  $\rightarrow$ Minimize the number of rhythm indication to add for having a unique solution

- Some input, e.g. $4/4 \ a4 \ b4 \ c2 \ |$ are over specified
  $\rightarrow$ Remove a maximum number of rhythm indication to still have this very unique solution

- Need for real-time, need for an efficient algorithm
  $\rightarrow$ Study theoretical complexity? Design efficient algorithms?
Future work: development

- Improve the tool
- Integrate this feature in Musescore and/or Lilypond and/or ABC?
Other ideas

- Link with type inference in programming language?
- Preferences? Default reasoning?
- Data mining to the most frequent rhythms?
Thanks to Charlotte Truchet

Thanks in advance to the potential group of students working on that project

and

Thank you!