# MIDI to SP-MIDI Transcoding Using Phrase Stealing

Simon Lui, Andrew Horner, and Lydia Ayers
*Hong Kong University of Science and Technology*

**The mobile phone industry has a problem with conversion from MIDI to SP-MIDI, which handles specific polyphonic limits. Simple specific polyphonic reduction algorithms, such as note-stealing, may lose or interrupt important musical information. Therefore, we present a phrase-stealing algorithm that drops the perceptually least important notes and preserves the most important phrases. Tests show that listeners reacted positively to this solution.**

Music on mobile phones has become a hot topic in recent years. Sending customized ring tones is especially popular. Unlike most music synthesizers and sound cards, only a few mobile phones support the musical instrument digital interface (MIDI) standard,[1] which represents music in an efficient note-level format. Instead, most mobile phones support specific polyphonic MIDI (SP-MIDI),[2] a constrained version of MIDI.

This is a serious limitation, since relatively few of the vast quantity of MIDI files are in SP-MIDI format. MIDI phones sending ring tones to SP-MIDI phones often have compatibility problems. Conversion from MIDI to SP-MIDI (called *transcoding*) is problematic because it frequently requires discarding notes and even whole channels.[3] For more background information on this problem, see the sidebar called "Problems Transcoding between MIDI and SP-MIDI."

In this article, we consider how to perform such transcoding while minimizing the impact on the music. As a solution, we introduce a phrase-stealing algorithm for MIDI to SP-MIDI transcoding. It automatically reduces the polyphony of a MIDI file to a specified *maximum instantaneous polyphony* (MIP) limit. The phrase-stealing algorithm takes musical context into account when deleting notes.

## Phrase-stealing algorithm

We can compare music to a book, where notes correspond to words, phrases correspond to sentences, and musical form corresponds to chapters. If we want to limit the number of words in an article, randomly deleting a word in a sentence (the literary equivalent to note-stealing) will result in a loss of information. Probably the best way to limit the number of words is to delete less important complete sentences. This is the principle of phrase stealing.

Although we call our algorithm phrase stealing, it's really a phrase-preserving algorithm because it identifies and keeps the most important phrases and deletes those less important. In particular, phrase stealing uses the following principles:

- Phrase stealing maintains smooth bass lines. Bass lines are easy for listeners to follow and underpin musical progressions.[4]
- Phrase stealing maintains smooth phrases. A phrase is a continuous line of notes. It can be a melody, countermelody, or an important accompanying line.

## Reprocessing

Two types of less-important notes are easy to identify and omit, even without phrase stealing: unison notes and nonessential percussion notes. Dropping unison notes belonging to the same channel (and therefore the same instrument) will certainly make a negligible impact on the music. Combining the percussion channels into a single channel simplifies processing. Once combined, we prioritize them as shown in Table 1.

For pop music, the bass and snare drums naturally have a high priority. The tambourine and ride cymbal can be important depending on when they occur. Tom-toms are normally less important, although a rolling tom-tom usually indicates the end of a phrase, and therefore it receives the second highest priority in Table 1.

Because most percussion sounds are short and impulsive, when we reduced the percussion channel to MIP = 1, the examples we tested showed only relatively small perceptual changes.

A cymbal on the first beat of a bar usually



1070-986X/06/$20.00 © 2006 IEEE

indicates the start of a phrase, so we gave it the highest priority. However, a cymbal on other beats and the wind chime are less important and require many MIDI events that can saturate the bandwidth.[5] This often causes truncation of other notes even when the MIP limit isn't reached. Therefore, we gave them the lowest priority. Figure 1 (next page) shows an example of MIP = 1 percussion reduction using the priority list in Table 1. Even after deleting many of the constant background events, the moving line retains its essence.

### Feature extraction

We can use various musical features to help us decide which phrases to preserve and which to delete, such as key signature, chord type, and bass notes.

#### Key signature

Finding the key signature can help identify other details such as chords and cadences. We apply the Krumhansl-Schmuckler key-finding algorithm.[6] The algorithm judges the key of a piece by correlating its key profile with a set of standard key profile models. The model yielding the highest correlation gives the preferred key.

#### Chord type

We define a chord as a set of three or more simultaneous notes with a duration of at least an eighth note. We classify chords into different types for the purpose of identifying cadences and dividing phrases. To classify the chord type, we find the best matching triad. We ignore extra notes including passing tones because they don't contribute to cadence recognition.

#### Top notes and bass notes

We use top notes and bass notes to help identify the bass lines. In our definition, a top note isn't necessarily a melody note, but it can never be a bass note. For example, in a chord with only one note, if that note is identified as a top note, it won't be a bass note even though it has the lowest pitch in the chord.

A note that satisfies all of the following is regarded as a top note:

- it's the highest pitch in the chord;
- its pitch is above the average pitch of the piece minus 10 percent of the standard deviation; and

## Problems Transcoding between MIDI and SP-MIDI

SP-MIDI was conceived as a solution for third-generation mobile applications and systems. It shares many similarities with MIDI, but prioritizes channels. The mobile phone plays as many SP-MIDI channels as possible without exceeding the number of hardware voices, which is called *maximum instantaneous polyphony* (MIP). [1]

Similarly, a piece's MIP is the maximum number of notes played at any time. If a piece's MIP is larger than the hardware MIP limit, the algorithm discards lower priority channels. The number of hardware voices limits the number of simultaneous voices, but not the number of channels. For example, with a hardware limit of two voices, it's possible to have more than two channels as long as there are no more than two simultaneous notes.

Most MIDI files[2] contain music that isn't composed with concern for channel priority or MIP limits. To use these files, ring tone providers can manually write different versions of the same piece (a monophonic version, a MIP = 4 version, and so on). However, this is tedious and time consuming. Automating the process requires an effective MIDI to SP-MIDI transcoding algorithm. Most hardware music synthesizers solve the MIP limit problem using note stealing—a first-in, first-out (FIFO) strategy where the oldest note is turned off when the number of simultaneous notes exceeds the number of hardware voices.

Note stealing takes advantage of the perceptual importance of note attacks over sustains and releases. However, note stealing may truncate important notes in the melody, making it hard to recognize the piece. Even worse, a piece with many simultaneous voices may degrade to thrashing on a phone containing only a few hardware voices,[3] with notes constantly turning on and off in a flurry of activity, leaving no resemblance to the original piece of music.

### References


1. MIDI Manufacturers Assoc., *Scalable Polyphony MIDI Specification, Version 1.0*, 2002.
2. MIDI Manufacturers Assoc., *Standard MIDI Giles 1.0,* tech report RP-001, 1996.
3. R. Maher, "Wavetable Synthesis Strategies for Mobile Devices," *J. Audio Eng. Soc.* (JAES), vol. 53, no. 3, 2005, pp. 205-212.


*Table 1. Priority list for percussion instruments.*

| Priority | Instrument |
|---|---|
| 1st | Cymbal on the first beat |
| 2nd | Rolling tom-tom |
| 3rd | Bass drum |
| 4th | Snare drum |
| 5th | Tambourine/ride cymbal on weak beat |
| 6th | Hi-hat |
| 7th | Tambourine/ride cymbal on strong beat |
| 8th | Tom-tom |
| 9th | Others |
| 10th | Cymbal not on first beat/windchime |

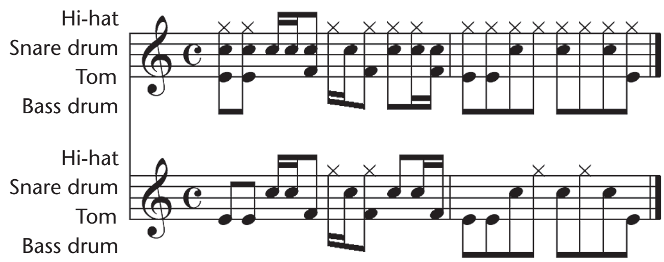



*Figure 1. Comparison of the original percussion track (top) and MIP – 1 reduced version (bottom).*

*Figure 2. Notes grouped into two phrases, indicated by slurs.*

```
public static final int[ ][ ] voiceLeadingFromV = {
// To\From  0    1    2     3    4    5    6     7    8    9    10   11
            { 3,  99,  2,  99,  2, 11, 99, 10, 99, 12,  99,   1 },   // 0
            { 1,  99,  2,  99, 12, 11, 99, 10, 99, 12,  99,  10 },   // 1
            { 2,  99,  1,  99,  3, 11, 99, 10, 99, 12,  99,  10 },   // 2
            { 12, 99,  2,  99,  1, 11, 99, 10, 99, 12,  99,  10 },   // 3
            { 2,  99,  4,  99,  1,  4, 99,  3, 99, 11,  99,  12 },   // 4
            { 12, 99, 10,  99,  2,  1, 99,  3, 99, 12,  99,  10 },   // 5
            { 12, 99, 10,  99, 12,  2, 99,  1, 99, 12,  99,  10 },   // 6
            { 2,  99, 10,  99,  3, 11, 99,  1, 99, 12,  99,  10 },   // 7
            { 12, 99, 10,  99, 12, 11, 99,  1, 99,  2,  99,  10 },   // 8
            { 12, 99, 10,  99, 12, 11, 99,  2, 99,  1,  99,  10 },   // 9
            { 12, 99, 10,  99, 12, 11, 99, 10, 99,  2,  99,   1 },   // 10
            { 2,  99, 10,  99, 12, 11, 99,  3, 99, 12,  99,   1 },   // 11
  };
```

**(a)**

```
public static final int[ ][ ] voiceLeadingToI = {
// To\From  0    1    2     3    4    5    6     7    8    9    10   11
            { 1,   1, 12,  12,  4, 12, 12,  2, 12, 12,  12,  10 },   // 0
            { 99, 99, 99,  99, 99, 99, 99, 10, 99, 99,  99,  99 },   // 1
            { 3,   2,  1,   2,  3, 12, 12,  4, 12, 12,  12,   2 },   // 2
            { 99, 99, 99,  99, 99, 99, 99, 99, 99, 99,  99,  99 },   // 3
            { 5,  10, 12,   1,  1, 12, 12,  4, 12, 12,  12,  12 },   // 4
            { 11, 12, 12,  12,  2,  1,  2,  3, 12, 12,  12,   3 },   // 5
            { 99, 99, 99,  99, 99, 99, 99, 99, 99, 99,  99,  99 },   // 6
            { 4,  10, 12,  12,  4, 12,  1,  1,  1, 12,  12,   2 },   // 7
            { 99, 99, 99,  99, 99, 99, 99, 99, 99, 99,  99,  99 },   // 8
            { 12, 12, 12,  12,  5, 12, 12,  2,  2,  1,   2,  10 },   // 9
            { 99, 99, 99,  99, 99, 99, 99, 12, 99, 99,  99,  99 },   // 10
            { 2,  12, 12,  12, 12, 12, 12, 12, 12, 12,   1,   1 },   // 11
  };
```

**(b)**

*Figure 3. Voice-leading table: (a) fromV chord and (b) toI chord.*

- it's closer to the top note than the bass note of the previous chord.

Next, notes that satisfy all of the following are regarded as bass notes:

- it's the lowest pitch in the chord;
- its pitch is below the average pitch of the piece plus 10 percent of the standard deviation; and
- it's not a top note.

Finally, the bass notes form the bass line of the piece, which we'll use in our subsequent analysis.

## Phrase-stealing algorithm

Phrase stealing takes musical context into account when deleting notes. It preserves important phrases and deletes less important phrases. The phrase-stealing algorithm consists of the following steps:

- Phrase identification
- Parallel phrase reduction
- Phrase stealing

### Phrase identification

Before we decide which phrases to drop, we must identify the phrases. After grouping notes from the same channel, we construct phrases by grouping notes that preserve the best voice leading (that is, the most natural melodic continuity between notes).

We group notes in the following manner: for each consecutive pair of chords, let F be the chord with fewer notes and M be the chord with more notes. Resolve each tendency tone, and then group each note of F with its nearest neighbor in M. Each note can be grouped to only one other note, based on the following:

- For common chords such as I, V, and vii, use voice-leading tables to resolve tendency tones.
- For the other chords, group each note of the preceding chord with its nearest neighbor in the succeeding chord.

Figure 2 shows the grouping for a short musical example.

Figure 3 shows two voice-leading tables. The indices in all the tables are relative to the tonic (for example, the eleventh row/column represents the leading tone). The entry (Row, Column) indicates the voice leading priority from note

Row to note Column where smaller values indicate a higher priority.

For example, for a V–I chord progression, if the V chord has fewer notes, then the *fromV* table in Figure 3a shows that (11, 0) = 2 is smaller than (11, 4) = 12 and (11, 7) = 3, so the leading tone (row 11) will be grouped with the succeeding tonic (column 0). If the I chord has fewer notes, then the *toI* table in Figure 3b shows that (11, 0) = 2 is smaller than (2, 0) = 3 and (5, 0) = 11, so the tonic (column 0) will be grouped with the preceding leading tone (row 11).

The algorithm just described group consecutive notes. However, unquantized MIDI files may contain small gaps between notes that are really meant to be consecutive. To solve this problem, we temporarily extend notes that have gap durations of less than a 64th note, using a variable virtual end time. We use virtual end times during the reduction algorithm, but use actual end times when decoding the notes back to a MIDI file.

We define phrases to end at cadences[7] and use the chord type and key to identify six simple cadences such as V–I or vii–I for closing phrases. We chose to use only these cadences as using more complex cadential formulas gives too many false phrases, which produce a result similar to note stealing.

#### Parallel phrase reduction

If a second phrase largely parallels the melody, but is transposed down an octave, fifth, or third (see Figure 4),[8] it is a good candidate for elimination. Accompanying chords often follow the same rhythm, and we can use these to reduce such parallel phrases.[9] The top phrase is usually the most important, so we eliminate the others.[10] We thus reduce parallel phrases as a preprocessing step before phrase stealing to reduce the number of rhythmically similar phrases.

*Figure 4. Musical example where phrase A (top) largely parallels phrase C (bottom).*

For each pair of overlapping phrases, we build a probability table for the overlapped part. Each table has an event entry, which represents the rhythm (that is, the start and end times) and pitch trajectory (whether the pitch moves up or down) for each note. Table 2 shows the event tables for the example in Figure 4.

Then, we compare the similarity of tables $x$ and $y$ and calculate

$N_x$ = Number of entries only in table $x$
$N_y$ = Number of entries only in table $y$
$N_{xy}$ = Number of entries in both tables

The similarity of two tables $x$ and $y$ relative to table $x$ is

*Table 2. Event tables for the example in Figure 4.*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Phrase A** | | | | | | | | | | | | | |
| Start time | 0:000 | 0:120 | 0:240 | 0:360 | 1:000 | 1:060 | 1:120 | 1:180 | 1:240 | 1:300 | 1:360 | 1:420 | 2:000 |
| End time | 0:120 | 0:240 | 0:360 | 1:000 | 1:060 | 1:120 | 1:180 | 1:240 | 1:300 | 1:360 | 1:420 | 2:000 | 2:240 |
| Pitch trajectory | / | – | – | + | + | – | – | + | + | – | – | + | + |
| **Phrase B** | | | | | | | | | | | | | |
| Start time | 0:000 | 0:120 | 0:240 | | 1:000 | | | | | | | | 2:000 |
| End time | 0:120 | 0:240 | 1:000 | | 2:000 | | | | | | | | 2:240 |
| Pitch trajectory | / | – | – | | + | | | | | | | | – |
| **Phrase C** | | | | | | | | | | | | | |
| Start time | 0:000 | 0:120 | 0:240 | 0:360 | 1:000 | 1:060 | 1:120 | 1:180 | 1:240 | 1:300 | 1:360 | | 2:000 |
| End time | 0:120 | 0:240 | 0:360 | 1:000 | 1:060 | 1:120 | 1:180 | 1:240 | 1:300 | 1:360 | 2:000 | | 2:240 |
| Pitch trajectory | / | – | – | – | + | + | – | – | + | + | – | – | + |

*Table 3. Phrase- and note-stealing results, including the listening test results. (PS = phrase stealing; NS = note stealing.)*

| MIDI File (Genre) | Max MIP | Phrases Formed | Average Phrase Length (in Notes) | Percentage of Notes Dropped *(PS)* | Percentage of Notes Dropped/ Truncated *(NS)* | Effectiveness at Preserving Smooth Phrases | | Effectiveness at Preserving Perceptually Important Notes | | Overall Similarity | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | PS | NS | PS | NS | PS | NS |
| 1. Pop | 14 | 286 | 6 | 36 | 35 | 6.54 | 5.58 | 7.33 | 5.27 | 7.54 | 6.40 |
| 2. Pop | 15 | 53 | 13 | 39 | 39 | 8.92 | 7.96 | 8.38 | 6.08 | 9.56 | 6.77 |
| 3. Baroque | 16 | 487 | 30 | 60 | 58 | 8.58 | 5.04 | 7.88 | 4.10 | 8.35 | 4.71 |
| 4. Pop | 26 | 92 | 4 | 63 | 61 | 6.29 | 4.25 | 6.83 | 5.79 | 5.27 | 4.38 |
| 5. Classical | 15 | 1,317 | 10 | 28 | 28 | 6.17 | 4.65 | 7.88 | 5.58 | 7.42 | 4.58 |
| 6. Jazz | 34 | 1,083 | 4 | 50 | 46 | 6.58 | 5.15 | 6.79 | 5.48 | 7.23 | 4.65 |
| 7. Jazz | 48 | 898 | 3 | 64 | 60 | 6.08 | 4.42 | 6.90 | 5.17 | 5.63 | 4.21 |
| 8. Jazz | 18 | 455 | 8 | 43 | 40 | 6.54 | 4.13 | 5.88 | 4.40 | 7.19 | 4.65 |
| 9. Jazz | 10 | 68 | 14 | 10 | 10 | 6.33 | 4.63 | 6.00 | 5.25 | 6.50 | 4.75 |
| Average: | | 527 | 10 | 43.67 | 41.89 | 6.89 | 5.09 | 7.09 | 5.24 | 7.19 | 5.01 |

$$S_{xy} = \frac{100N_{xy}}{N_{xy} + N_x}\% \quad (1)$$

The similarity of two tables $x$ and $y$ relative to table $y$ is

$$S_{yx} = \frac{100N_{xy}}{N_{xy} + N_y}\% \quad (2)$$

Let $z$ be the table with the larger probability among $x$ and $y$. If the probability of $z$ is larger than 75 percent, we delete the overlapped area of $x$ and $y$ from phrase $z$.

For the example in Figure 4, first we compare tables $A$ and $B$: $N_A = 11$, $N_B = 3$, $N_{AB} = 2$; therefore, $S_{AB}$ = 15.384 percent, and $S_{BA}$ = 40 percent. Because the similarity values are less than the threshold, we preserve both phrases. Next, we compare tables $B$ and $C$: $N_B = 10$, $N_C = 3$, $N_{BC} = 2$; therefore $S_{BC}$ = 16.667 percent, and $S_{CB}$ = 40 percent. Again, we preserve both phrases. Finally, we compare tables $A$ and $C$: $N_A = 2$, $N_C = 1$, $N_{AC} = 11$; therefore, $S_{AC}$ = 84.615 percent, and $S_{CA}$ = 91.667 percent. Since 91.667 percent > 84.615 percent > 75 percent, we delete phrase $C$, and preserve phrases $A$ and $B$.

### Phrase stealing

We delete phrases in a last-in, first-out (LIFO) manner. Each phrase is deleted or preserved as a whole. We found it useful to handle different cases depending on the MIP value. For MIP = 1, we preserve the melody line at a higher priority than the bass line. For MIP = 2, we preserve the melody and percussion lines at a higher priority than the bass line. However, if the percussion channel is empty, we preserve the bass line. For MIP = 3 or more, we can afford to preserve a smooth bass line, so we combine all the bass notes together into a phrase. When there are too many notes at the same time, we attempt to preserve the melody and countermelody by deleting some of the accompaniment and eliminating the following phrases:

- the phrase with the smallest standard deviation of pitch (for example, repeated notes and those with small deviations) and
- the phrase with the shortest duration (for example, isolated downbeats and short figures).

## Results

We conducted statistical tests to confirm the perceptual effectiveness of the phrase-stealing algorithm. Our analysis focused on MIP = 4 reduction, since it's common in current mobile phones, and traditional note stealing usually gives poor results for MIP = 4 reduction.

Table 3 shows that, of the identified phrases, many had average phrase lengths of 8 or more notes. The overall average phrase length was 10, suggesting that obvious phrase continuities were identified and preserved. Phrase and note stealing both dropped a large percentage of notes (more than 40 percent). As we previously noted, phrase stealing does this by deleting entire phrases, while note stealing drops and truncates individual notes.

A listening test found that the phrase stealing versions sounded reasonably similar to the original files, and much better than the note-stealing versions. The listening test involved 48 listeners. All listeners performed the test together in a room with loudspeakers. Table 4 lists the excerpts tested. We included a wide variety of different genres to show the full functionality of the algorithm. In the test, listeners heard the original, the reduced file, and the original again. They were then asked to rate the similarity of the reduced file to the original on a scale from 0–10, where 0 represents entirely unrelated and 10 represents entirely identical (see Table 5).

The listening test results in Table 3 show that phrase stealing better preserves smooth phrases (a score of 6.89 versus 5.09), keeping perceptually important notes (7.09 versus 5.24), and maintaining overall similarity to the original (7.19 versus 5.01). Paired sample *t*-tests[11] revealed that the average overall similarity for phrase stealing is significantly higher than that for note stealing. We compared both excerpts ($t = 7.13$, $p < 0.001$) and participants ($t = 30.61$, $p < 0.001$), respectively. Effectiveness at preserving smooth phrases was also significantly higher ($t = 7.56$, $p < 0.001$), as was effectiveness at preserving perceptually important notes ($t = 6.93$, $p < 0.001$).

In particular, phrase stealing showed an obvious improvement over note stealing for Bach and pop excerpts, probably because those pieces have contrapuntal lines that phrase stealing preserves well. Phrase stealing had some difficulty identifying angular melodic lines in some jazz excerpts, but the method still performed better than note stealing.

***Table 4. Excerpts used in the listening test.***

| MIDI File | Genre |
|---|---|
| 1. Andy Rains: *26 Bottles of Beer* | Pop |
| 2. Theme song of the TV series *Reincarnated* | Pop |
| 3. J.S. Bach: *Brandenburg Concerto No.1*, 1st movement | Classical |
| 4. Anonymous : *Canyon* | Pop |
| 5. W.A. Mozart: *Piano Concerto No.17 in G*, 1st movement | Classical |
| 6. Anonymous: *Cooker* | Jazz |
| 7. Anonymous: *It Don't Mean A Thing* | Jazz |
| 8. Anonymous: *Jazzy Cat* | Jazz |
| 9. Anonymous: *Pinky* | Jazz |

***Table 5. Listening test rating scale.***

| Rating | Values Interpretation |
|---|---|
| 10 | Entirely identical |
| 9 | Almost identical |
| 8 | Very similar |
| 7 | Very similar, a bit unrelated |
| 6 | Quite similar |
| 5 | Somewhat similar but somewhat unrelated |
| 4 | Quite unrelated |
| 3 | Very unrelated, a few similar |
| 2 | Very unrelated |
| 1 | Almost unrelated |
| 0 | Entirely unrelated |

## Examples

As an example, Figure 5a shows an excerpt from a baroque composition—Bach's *Brandenburg Concerto No.1*—and Figure 5b shows the reduced output. The phrase-stealing algorithm makes a smooth switch between the instruments in the

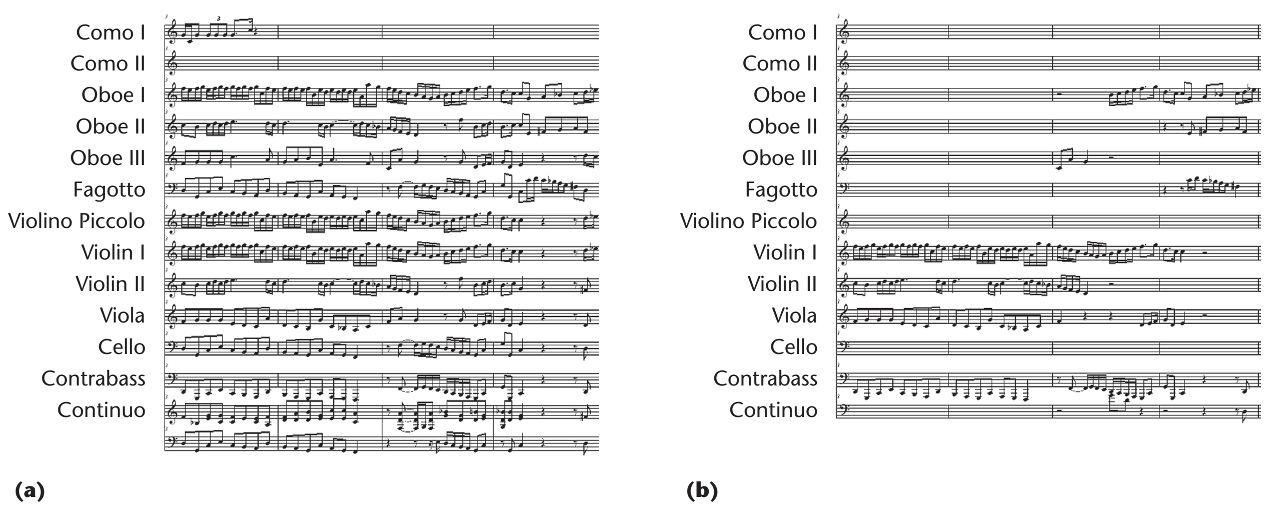



***Figure 5. Bach's Brandenburg Concerto No. 1, 1st movement: (a) the original and (b) the MIP = 4 reduction. A smooth melody/countermelody is retained in the score.***

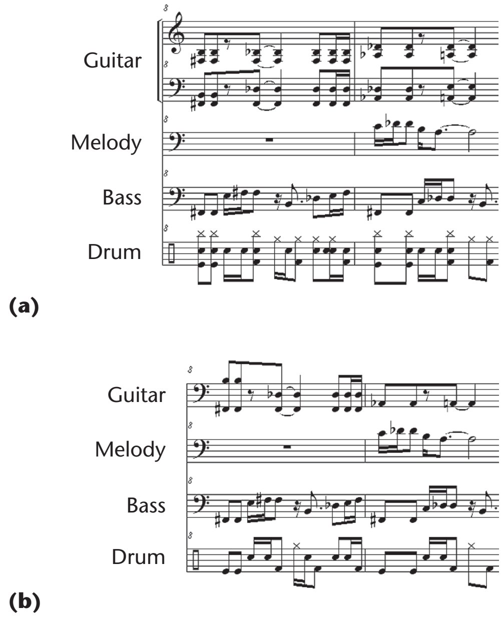



*Figure 6. Andy Rains'* **26 Bottles of Beer***: (a) the original and (b) the MIP = 4 reduction. The fourth drumset staff was reduced. The first staff with guitar chords was reduced from MIP = 3 to MIP = 1 or 2.*

last two bars. Figure 6a shows a pop excerpt—Andy Rains' *26 Bottles of Beer*—and Figure 6b shows the reduction. Although the number of percussion notes is reduced by 50 percent, the reduction is still perceptually similar to the original, because little other than the hi-hat has been lost. Although the chord accompaniment has also been reduced, its rhythmic structure has been retained, and there is no obvious perceptual impact.

## Discussion

Phrase stealing tries to preserve events that a listener expects to hear.[12] It usually preserves the melody and phrase lines to which people listen most closely. It tries to preserve smooth bass lines when there are enough hardware voices to do it.

However, the algorithm still has trouble identifying short discontinuous phrases. The excerpt from the anonymous pop piece *Canyon* in Figure 7a is an example. As this excerpt has very few continuous phrases, the phrase-stealing reduction in Figure 7b isn't much better than that of note stealing.

## Future work

Although MIDI ring tones might become obsolete when more powerful devices supporting MP3 or pulse code modulation (PCM) ring tones become popular, mobile phones that can only support SP-MIDI (or even monophonic SP-MIDI) currently have a huge market. It's worth noting here that it's also much easier and more convenient for users to write their own ring tones in MIDI than MP3 or PCM.

But aside from this, users often prefer to keep using the same format, even when a more powerful format becomes available. For example, General MIDI (GM) continues to be widely used even though GM2 is superior.[13] For these reasons, MIDI ring tones will probably continue to be popular for some years to come.

Regarding our algorithm, because phrase stealing's LIFO phrase-deletion strategy doesn't explicitly try to preserve melody lines, a possible solution is to select phrases using a dynamic programming approach.[14] The idea is to start with an empty SP-MIDI file, and then continuously add the highest-priority phrase to the file. Phrases with the largest pitch deviation are most likely melody or bass lines, and can be given higher priority.

*Figure 7. Excerpt from* **Canyon** *with short discontinuous phrases: (a) the original and (b) the MIP = 4 reduction. Only some smooth melody lines are well-preserved, while the other irregular phrases are lost.*

Also, unexpected results can occur for unusual chords that aren't included in the voice-leading tables. We plan to build a more extensive set of voice-leading tables to include as many chords as possible.

We'd also like to apply the phrase-stealing algorithm to other applications, such as measuring melodic similarity.[15] **MM**

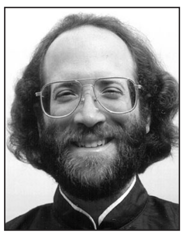
## Acknowledgments

This work was supported in part by the Hong Kong Research Grant Council's Project HKUST6167/03E and the Innovation and Technology Fund Project ITS/122/03. We thank Eno Tsin and Chris Wong of Sensestream for their discussion and valuable comments, all the participants of the listening test for their time and effort, and the three anonymous reviewers for their careful and invaluable comments, especially the suggestion that phrase stealing can be applied to measuring melodic similarity.

## References

1. MIDI Manufacturers Assoc., *MIDI 1.0 Detailed Specification, Version 4.2,* 1996.
2. MIDI Manufacturers Assoc., *Scalable Polyphony MIDI Specification, Version 1.0*, 2002.
3. Nokia, "Polyphonic Ringing Tones," http://www.nokia.com/nokia/0,8764,5343,00.html.
4. D. Temperley, *The Cognition of Basic Musical Structures*, MIT Press, 2001.
5. Nokia, *MIDI and True Tones in Nokia Phones*, *Version 1.2*, 2003.
6. C. Krumhansl and M. Schmuckler, *Cognitive Foundations of Musical Pitch*, Oxford Univ. Press, 1990.
7. R.L. Jacobs, *Understanding Harmony*, Greenwood Press, 1986.
8. E. Cambouropoulos, "Musical Parallelism and Melodic Segmentation," *Proc. XII Colloquium of Musical Informatics*, 1998.
9. E. Cambouropoulos, "A Computational Theory for Discovery of Parallel Melodic Passages," *Proc. XI Colloquium of Musical Informatics*, 1995.
10. A.S. Bregman, *Auditory Scene Analysis: The Perceptual Organization of Sound*, MIT Press, 1990.
11. R.V. Hogg and J. Ledolter, *Applied Statistics for Engineers and Physical Scientists,* Maxwell Macmillan, 1992.
12. E. Cambouropoulos, "Melodic Cue Abstraction, Similarity, and Category Formation: A Formal Model," *Music Perception*, vol. 18, no. 3, 2001, p. 347ff.
13. MIDI Manufacturers Assoc., *General MIDI 2 Specification*, *Version 1.1*, 2003.
14. R. Bellman, *Dynamic Programming*, Princeton Univ. Press, 1957.
15. R. Typke et al., "Using Transportation Distances for Measuring Melodic Similarity," *Proc. Int'l Conf. Music Information Retrieval* (ISMIR), 2003, pp. 107-114.

**Simon Lui** is a doctoral candidate in the Department of Computer Science of the Hong Kong University of Science and Technology. His research interests include music perception, models for music expressiveness, and Internet music performance systems.



**Andrew Horner** is a professor in the Department of Computer Science at the Hong Kong University of Science and Technology. His research interests include music synthesis, musical acoustics of Asian instruments, peak factor reduction in musical signals, music in mobile phones, and spectral discrimination. He received his PhD in computer science from the University of Illinois at Urbana-Champaign.

**Lydia Ayers** is a freelance composer and coauthor of the book *Cooking with Csound: Woodwind and Brass Recipes* (A-R Editions, 2002). Her primary research interests are microtonal tunings, extended flute techniques, and synthesizing musical expression for Asian musical instruments. She received her DMA in musical arts in Music Composition from the University of Illinois at Urbana-Champaign.

Readers may contact Simon Lui at virtuoso@cs.ust.hk.