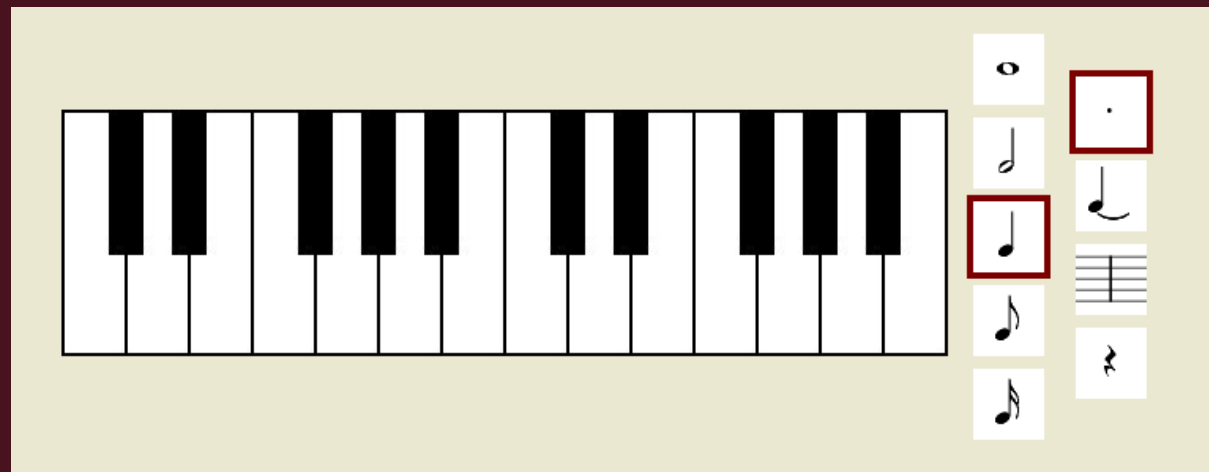


# Tools and Technologies for Music Search



Nathaniel Burns

# What is Hymnary.org?

- Website organizing information on hymns, hymnals, and hymn authors and composers.
- Run by the Christian Classics Ethereal Library under the direction of Dr. Harry Plantinga.
- Contains entries for 1,026,747 texts and 17,931 tunes.
- Making this data easy to browse and search is key!

# Why Music Search?

- Currently, hymn tunes can only be found via text-based searches.
- This means finding a tune can be difficult if the tune name or composer is not known.
- Music search escapes this problem by searching the actual melody of the tune.
- Great possibilities for hymn research – the ability to find tunes with similar melodies, with direct quotes from one another, and so on.

# Goals for Music Search

Music search at Hymnary.org should be:

- Melodically based
- Transposition insensitive
- Fault tolerant

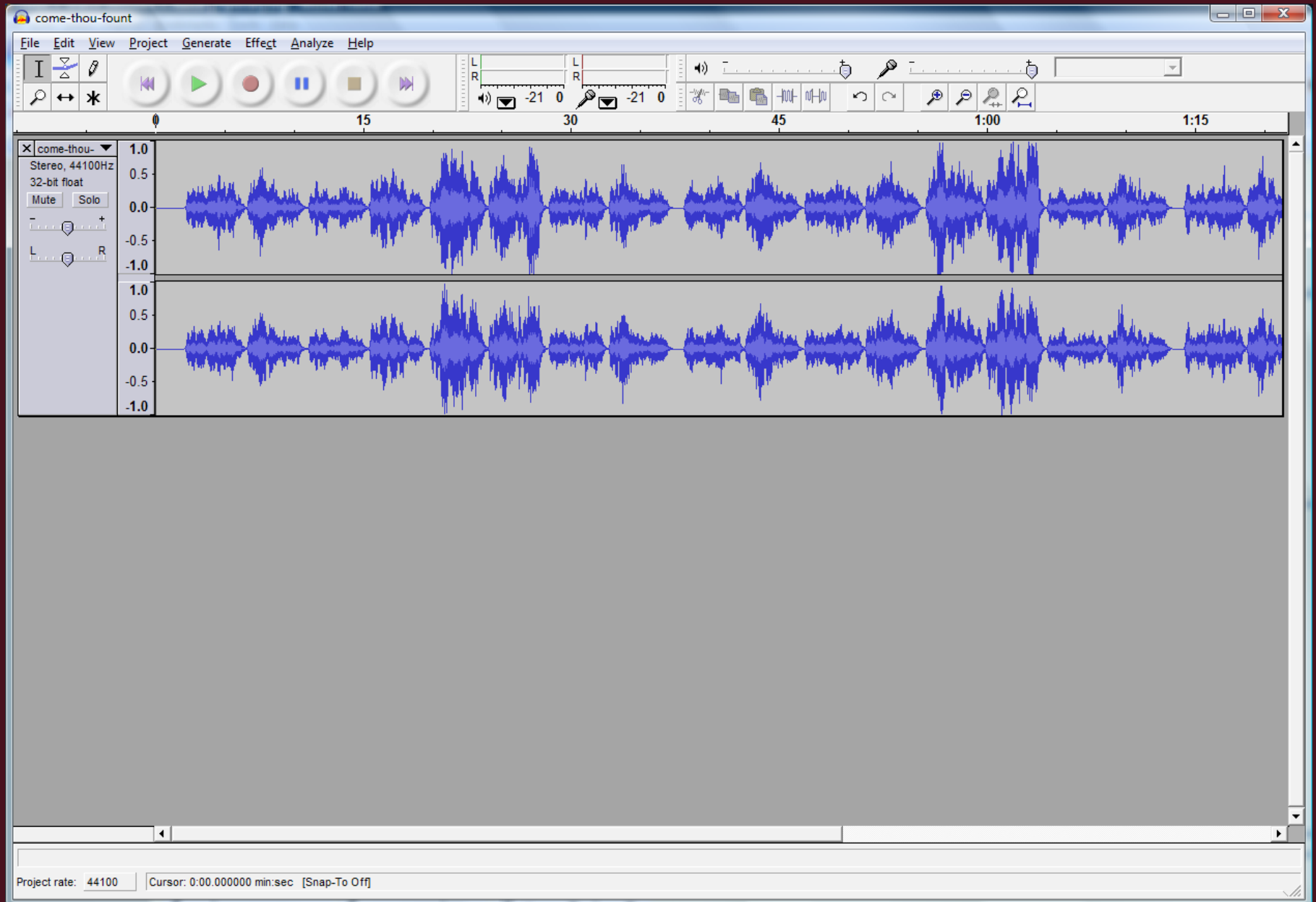
# Major Questions in Music Search

- How can we represent music for a computer to understand?
- How can we measure melodic similarity in a meaningful way?
- How can we apply these techniques to search efficiently?

# Representing Music

# Audio Formats

- Wav, mp3, wma, etc.
- Most widespread of music formats
- Represents musical *performances*
- Not exclusive to music
- Carries no meta-information (key signature, time signature, instrumentation)
- Even things notes and dynamics are troublesome to obtain



# Symbolic Formats

- MusicXML, Finale, Sibelius, Noteworthy
- Most common among musicians
- Represents a musical score
- Proprietary formats make extraction/conversion tricky

```
<score-partwise version="2.0">
  <movement-title>Elite Syncopations</movement-title>
  <identification>
    <creator type="composer">Scott Joplin</creator>
  </identification>
  <page-layout> ... </page-layout>
  <part-list>
    <score-part id="P1">
      <score-instrument>
        <instrument-name>Grand Piano</instrument-name>
      </score-instrument>
    </part-list>
    ...
  </score-partwise>
```

```
<score-partwise version="2.0">
```

```
...
```

```
<part id="P1">  
  <measure number="2">  
    <attributes>  
      <divisions>4</divisions>  
      <key> ... </key>  
      <time> ... </key>  
      <clef> ... </clef>  
    </attributes>  
    <note>  
      <step>C</step>  
      <octave>6</octave>  
      <duration>1</duration>  
    </note>  
    <note> ... </note>  
  </measure>  
</part>  
</score-partwise>
```

# MIDI Format

- A cross between audio and symbolic representation
- Small file sizes are ideal for websites
- Commonly seen on [bad websites](#) from the 1990s
- Can carry key signature, time signature, instrument info
- Specifies exact performance timing and dynamics

MFile 1 3 120

MTrk

0 TimeSig 3/4 24 8

0 KeySig 253 major

0 Tempo 600000

0 Meta TrkEnd

TrkEnd

MTrk

0 PrCh ch=2 p=0

0 On ch=2 n=67 v=100

59 On ch=2 n=65 v=100

60 Off ch=2 n=67 v=0

119 Off ch=2 n=65 v=0

119 On ch=2 n=63 v=100

239 Off ch=2 n=63 v=0

239 On ch=2 n=63 v=100

359 Off ch=2 n=63 v=0

360 On ch=2 n=67 v=100

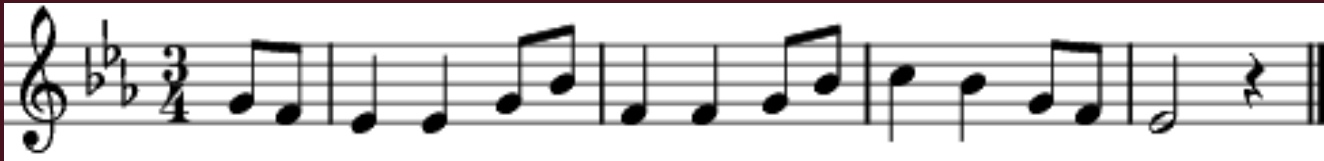
419 Off ch=2 n=67 v=0

TrkEnd



# Melodic Representation

- We can simply represent a melody as a set of notes with each note as an ordered pair of pitch and onset.
- A tune like this:



- Would be represented like this:

{ (67, 0), (65, 4), (63, 8), (63, 16), (67, 24), (70, 28), (65, 32), (65, 40),  
(67, 48), (70, 52), (72, 56), (70, 64), (67, 72), (65, 76), (63, 80) }

# Extracting Melodies

- Only requiring pitch and timing data means lots of potential data sources
- We'll standardize the formats in our database
- Symbolic formats are definitely compatible
- Audio formats are definitely not (for now)
- MIDI files may or may not be, depending

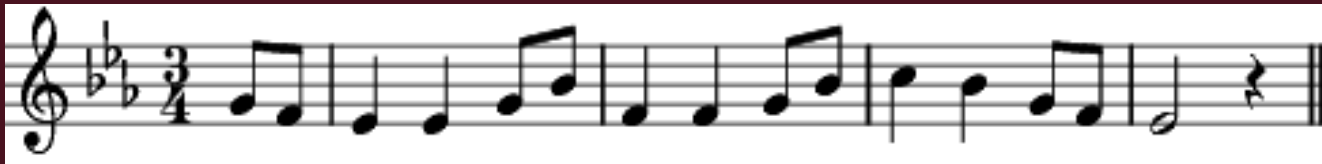
# Measuring Melodic Similarity

# Melodic Similarity

- Searching for exact matches is easiest – an extension of a plaintext search
- Melodies may have slight changes that throw off computers, but not humans
- A variety of techniques exist to counter this:
  - Contour Matching
  - Edit Distances
  - Feature Indexing
  - Geometric Overlap
  - Transportation Distances

# Contour Matching

- Melodies represented according to their “contour”
- Notes are designated “U”, “D”, or “S” depending on their relation to the previous note
- A melody like this:



- Would be represented:

DDSUUDSUUDDDD

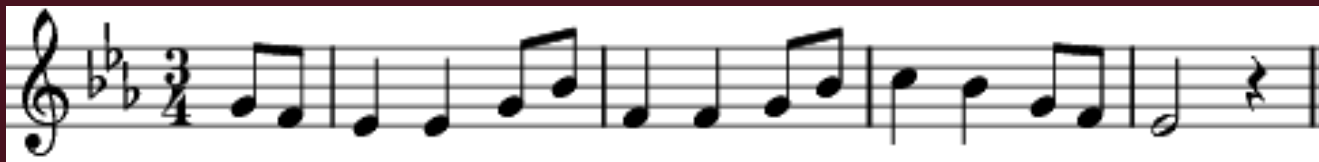


# Contour Matching

- Advantages:
  - Simple to implement
  - Not affected by transpositions
  - Fast
- Disadvantages:
  - Doesn't account for differences in rhythm
  - Potential for false positives

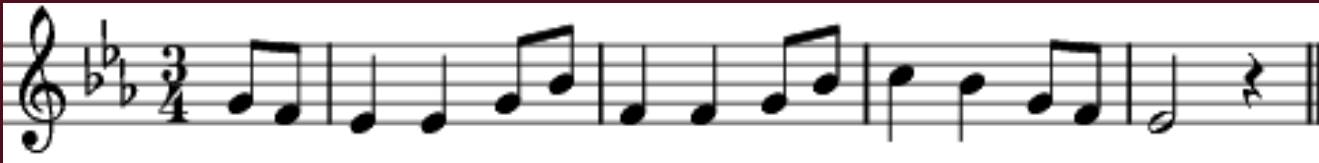
# Edit Distances

- Comes from the idea of “Levenshtein Distances” for text
- Matches are ranked based on the number of “edits” required to convert one melody to another
- Edits are additions, substitutions, or removals of notes
- Again, each melody gets a string representation:



(67 0) (65 4) (63 8) (63 16) (67 24) (70 28) (65 32) (65 40)  
(67 48) (70 52) (72 56) (70 64) (67 72) (65 76) (63 80)

# Edit Distances



(67 0) (65 4) (63 8) (63 16) (67 24) (70 28) (65 32) (65 40) (67 48) (70 52) (72 56) (70 64) (67 72) (65 76) (63 80)

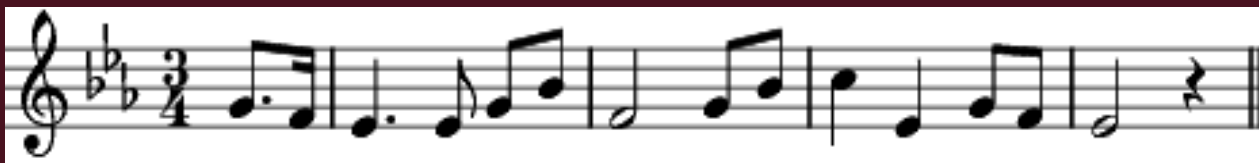
1: (67 0) (65 6) (63 8) (63 16) (67 24) (70 28) (65 32) (65 40) (67 48) (70 52) (72 56) (70 64) (67 72) (65 76) (63 80)

2: (67 0) (65 6) (63 8) (63 20) (67 24) (70 28) (65 32) (65 40) (67 48) (70 52) (72 56) (70 64) (67 72) (65 76) (63 80)

3: (67 0) (65 6) (63 8) (63 20) (67 24) (70 28) (65 32) (\_\_\_\_) (67 48) (70 52) (72 56) (70 64) (67 72) (65 76) (63 80)

4: (67 0) (65 6) (63 8) (63 20) (67 24) (70 28) (65 32) (67 48) (70 52) (63 56) (70 64) (67 72) (65 76) (63 80)

(67 0) (65 6) (63 8) (63 20) (67 24) (70 28) (65 32) (67 48) (70 52) (63 56) (70 64) (67 72) (65 76) (63 80)



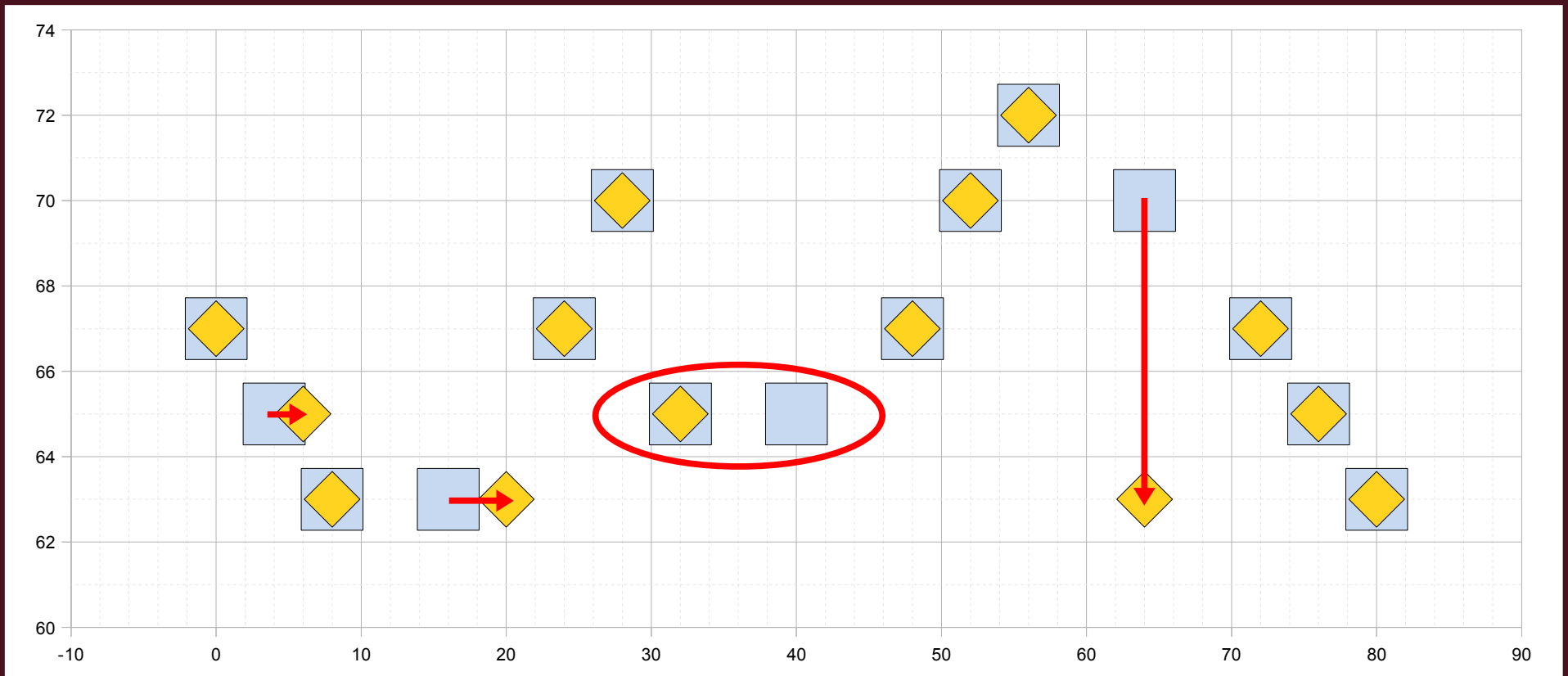
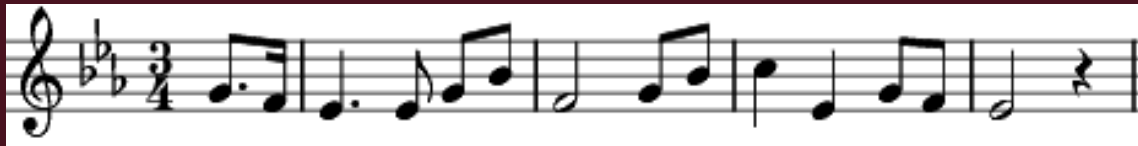
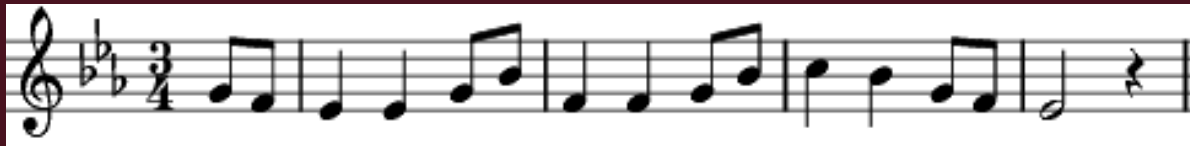
# Edit Distances

- Advantages:
  - Can account for rhythm
  - Edit distance is zero if and only if the melodies are identical
- Disadvantages:
  - Does not account for changes in transposition
  - All substitutions are identical, regardless of how “close” the notes are

# Transportation Distances

- Melodies are represented as 'weighted point sets'
- A note's pitch and onset pair are taken as coordinates
- Each note also has an associated weight value
- The transportation distance of two melodies is the work required to transfer the weight from one location set to the other

# Transportation Distances



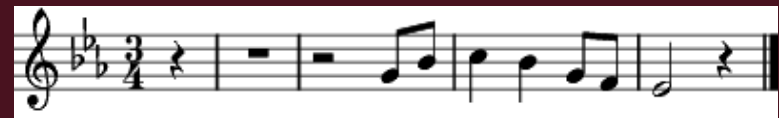
# Transportation Distances

- Earth Mover's Distance (EMD)
  - The minimum work required to transform one melody into another, measured as the product of weight and distance moved
  - Handles partial matching with ease
  - “Triangle inequality” does not hold for unequal weight totals

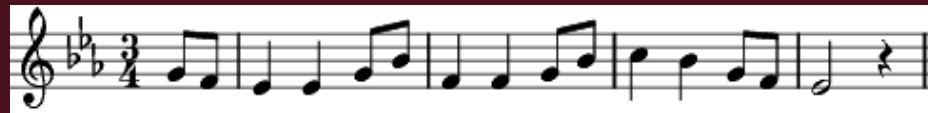
e.g.  $\exists a, b, c: \text{EMD}(a, c) > \text{EMD}(a, b) + \text{EMD}(b, c)$



(a)



(c)

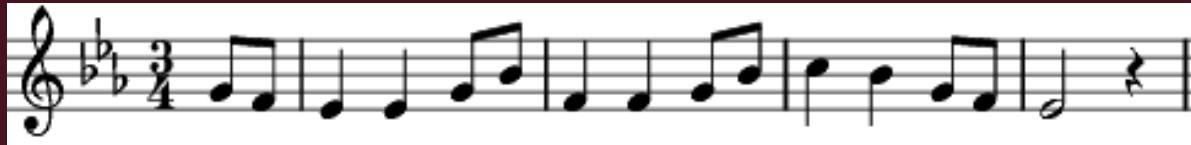


(b)



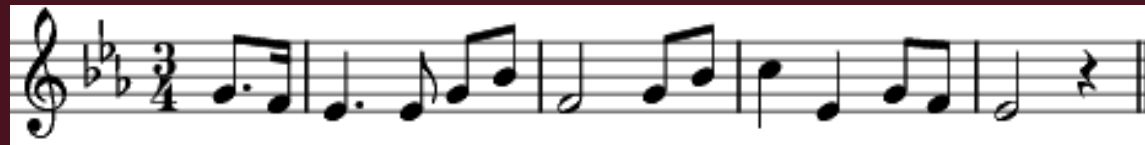
# How about some examples...

NETTLETON ("Come Thou Fount of Every Blessing")



VS.

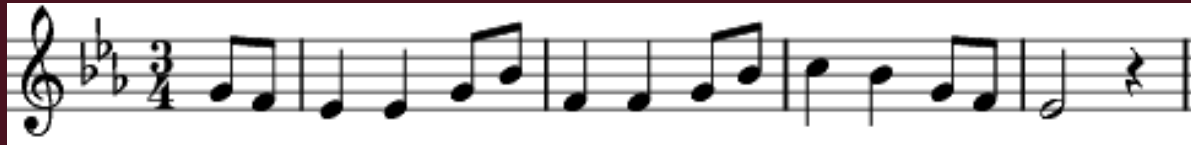
NETTLETON, altered



PTD: 2.24

# How about some examples...

NETTLETON ("Come Thou Fount of Every Blessing")



VS.

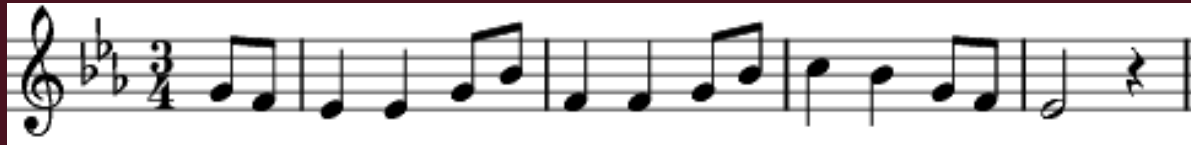
TERRA BEATA ("This is My Father's World")



PTD: 12.62

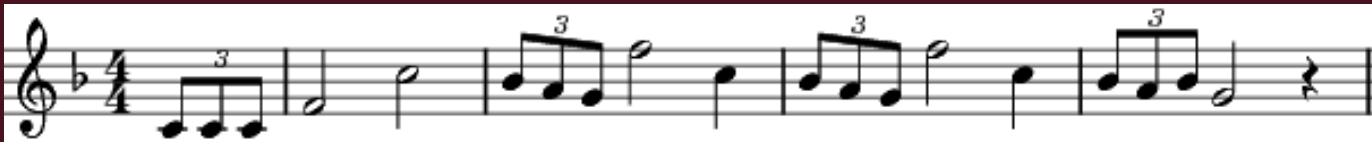
# How about some examples...

NETTLETON ("Come Thou Fount of Every Blessing")



VS.

"Star Wars" Main Theme



PTD: 21.50

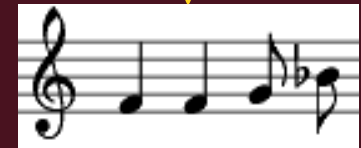
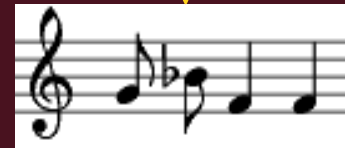
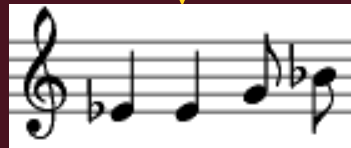
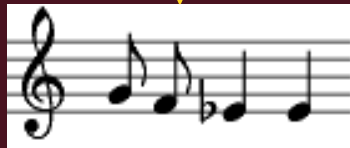
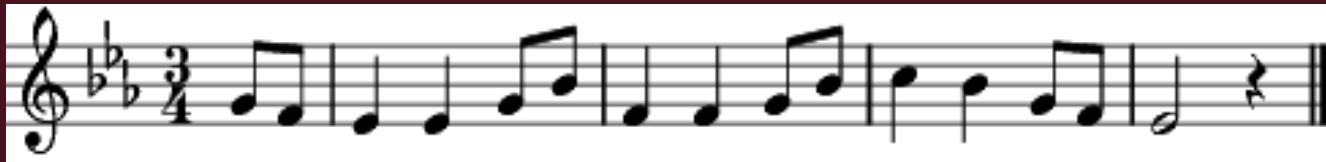
# Searching Efficiently

# Melodic Segmenting

- The Proportional Transportation distance works best with melodies that are roughly the same length.
- It's unlikely a user will input an entire hymn melody – more often, it will be the incipit (opening line)
- So, we can break the melodies into fixed-size segments and then compare those
- Tunes with the most matching segments will be the best matches overall

# Melodic Segmenting

- Many segmenting schemes are possible
- We use sequences of four notes, offset by two



# Vantage Indexing

- Calculating the PTD between two tunes uses the Simplex algorithm – an  $O(a^n)$  operation, worst-case
- For speed, we use a technique called “vantage indexing”
- We generate random segments called “vantages” and create a catalog of PTDs against those

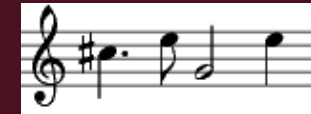
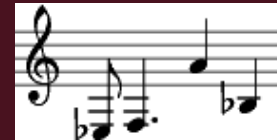
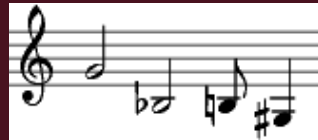


Sample Vantage Segments

# Vantage Indexing

## Vantage

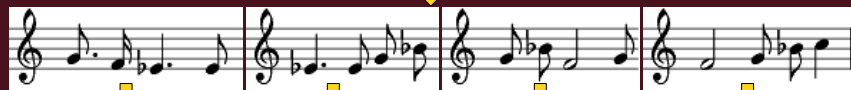
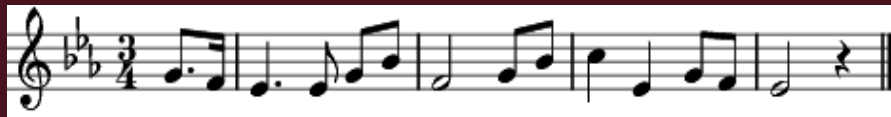
Proportional  
Transportation  
Distance



	6.508818	9.475279	3.724060
	9.818089	5.139505	4.891103
	8.342955	7.935081	3.031250
	8.828067	6.127550	4.007567

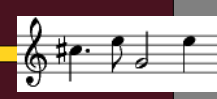
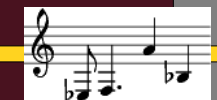
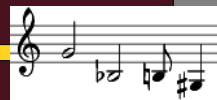
Segment

# Putting It All Together

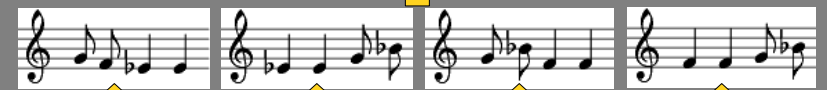
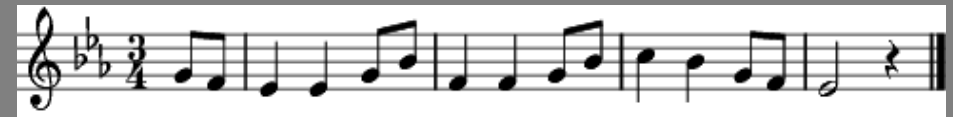


(67, 0)	(63, 0)	(67, 0)	(65, 0)
(65, 6)	(63, 12)	(70, 4)	(67, 16)
(63, 8)	(67, 16)	(65, 8)	(70, 20)
(63, 20)	(70, 20)	(67, 24)	(72, 24)

6.934169	10.002071	9.918364	10.849325
9.798689	6.351939	7.416667	6.537853
3.885391	5.292989	3.496060	6.162902



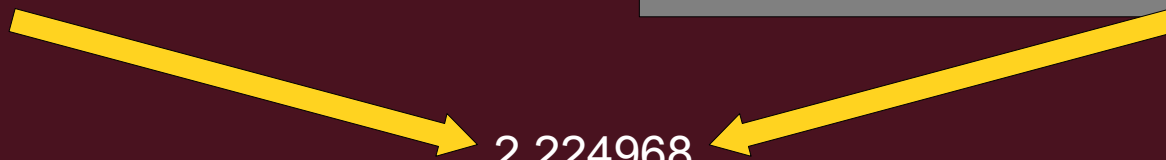
precalculated



(67, 0)	(63, 0)	(67, 0)	(65, 0)
(65, 4)	(63, 8)	(70, 4)	(65, 8)
(63, 8)	(67, 16)	(65, 8)	(67, 16)
(63, 16)	(70, 20)	(65, 16)	(70, 20)

6.508818	9.818089	8.342955	8.828067
9.475279	5.139505	7.935081	6.127550
3.724060	4.891103	3.031250	4.007567

2.224968



# Thank You

## Sponsors:

- Dr. Harry Plantinga
- Mr. Sidney Jansma
- Everyone at CCEL and GLSEC

## Open Source Software:

- Han-Wen Nienhuys and Jan Nieuwenhuizen, [Lilypond 2.12.2](#)
- Yossi Rubner, [Earth Movers Distance - C implementation](#)
- Scott Schiller, [SoundManager 2](#)
- Johannes E. Schindelin, [MediaWiki Lilypond Extension](#)
- Valentin Schmidt, [PHP Midi Class 1.7.5](#)

# References

Clausen, Michael, and Frank Kurth. "A Unified Approach to Content-Based and Fault-Tolerant Music Recognition." *IEEE Transactions on Multimedia* 6, no. 5 (2004): 717-731.

Clausen, Michael, Frank Kurth, and Heiko Körner. "An Efficient Indexing and Search Technique for Multimedia Databases." *SIGIR Workshop on Multimedia Retrieval* (2003).

Typke, Rainer. *Music Retrieval based on Melodic Similarity*. Utrecht, Netherlands: Utrecht University, 2007.

Typke, Rainer, Remko C. Veldkamp, and Frans Wiering. "Searching Notated Polyphonic Music Using Transportation Distances." *Proceedings of the 12th annual ACM international conference on Multimedia* (2004).

Orio, Nicola. "Music Retrieval: A Tutorial and Review." *Foundations and Trends in Information Retrieval* 1, no. 1 (2006): 1-90.