MULTILEVEL MELODIC MATCHING

Chris Walshaw
Department of Computing & Information Systems,
University of Greenwich, London SE10 9LS, UK
c.walshaw@gre.ac.uk

ABSTRACT

This paper describes a multilevel algorithm for matching tunes when performing inexact searches in symbolic musical data. The basis of the algorithm is straightforward: initially each tune in the search database is normalised and quantised and then recursively coarsened, typically by removing weaker off-beats, until the tune is reduced to a skeleton representation with just one note per bar. The same process is applied to the search query and melodic matching between query and data can then take place at every level. The algorithm implemented here uses the longest common substring algorithm at each level, but in principle a variety of similarity measures could be used. The multilevel framework allows inexact matches to occur by identifying similarities at course levels and is also exploited with the use of early termination heuristics at coarser levels, both to reduce computational complexity and to enhance the matching qualitatively. Experimentation demonstrates the effectiveness of the approach for inexact melodic searches within a corpus of tunes.

1. INTRODUCTION

This paper presents a multilevel melodic matching algorithm. In a sister paper a variant is explored as a method for identifying related tunes (Walshaw, 2015) but here it is used to perform inexact melodic searches, in particular within the abc notation music corpus.

Abc notation is a text-based music notation system popular for transcribing, publishing and sharing music, particularly online. Similar systems have been around for a long time but abc notation was formalised and named by the author in 1993. Since its inception he has maintained a website, now at abcnotation.com, with links to resources such as tutorials, software and tune collections.

In 2009 the functionality of the site was significantly enhanced with an online tune search engine which indexes a corpus of around 460,000 abc transcriptions from across the web. Users of the tune search are able to view, listen to and download the staff notation, MusicXML, MIDI representation and abc code for each tune, and the site currently attracts around half a million visitors a year.

Currently, however, the search engine is purely text based: it is possible to search melodically, since abc is a text based notation system, but only for exact text matches, and as a result most of the searches that take place are based on tune titles (and other meta data).

The motivation for this paper is to propose an inexact melodic search which will identify tunes closely related to the search query, an important feature for an index of what is mostly folk and traditional music.

2. RATIONALE: A CASE STUDY

Figure 1 shows two versions of the first 4 bars of Speed the Plough, a tune well-known across the British Isles (at the time of writing the abcnotation.com tune search has 244 tunes with a title which includes the phrase "Speed the Plough"). The first version in Fig. 1 is drawn from an English collection and the second, with the title "God Speed the Plough", from an Irish collection. Clearly these tunes are related but with distinct differences, particularly in the second and fourth bars.

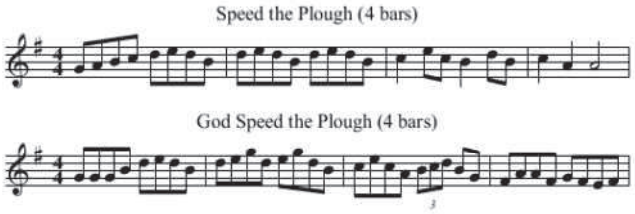

Figure 1. Two tune variants for Speed the Plough.

Subjectively, as a working musician, it is typical in tunes like this (which can be played as a reel, with an even rhythm, or a hornpipe, with a dotted rhythm), that the emphasis is placed on the odd numbered notes, and in particular the first note of each beam. The strongest notes of the bar are thus 1 and 5, followed by 3 and 7.

To capture this emphasis when matching tune variants it might be possible to use some sort of similarity metric which weights stress (so that matching 1st notes carry more importance than, say, 2nd notes, e.g. Typke, 2007). However, in this paper the approach is to build a multilevel (hierarchical) representation of the tunes.

The multilevel paradigm is simple one which involves recursive coarsening to create a hierarchy of increasingly coarse approximations some original representation. As a general solution strategy, the multilevel paradigm is widely used in combinatorial and other optimisation problems and can be extremely effective, both at imparting a global perspective and at accelerating solution techniques, (Walshaw, 2008).

Figures 2 and 3 show multilevel coarsened versions of the original tunes, where the weakest notes are recursively replaced by removing them and extending the length of the previous note by doubling it.

At level 0, i.e. the original, the tunes are quantised to show every note as a sixteenth note, thus simplifying the coarsening process. In addition the triplet in bar 3 of "God Speed the Plough" is simplified by representing it as two eighth notes, the first and last notes of the triplet.

To generate level 1, the 2nd, 4th, 6th and 8th notes are removed from each bar; for level 2, the original 3rd and 7th notes (which are now the 2nd and 4th) are removed; for level 3, the original 5th note (now the 2nd) is removed. As can be seen, as the coarsening progresses the two versions become increasingly similar and thus provide a good scope for melodic comparisons which ignore the finer details of the tunes.

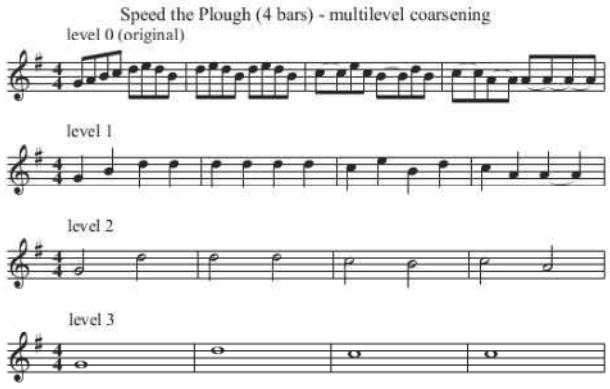

Figure 2. Multilevel coarsening of Speed the Plough

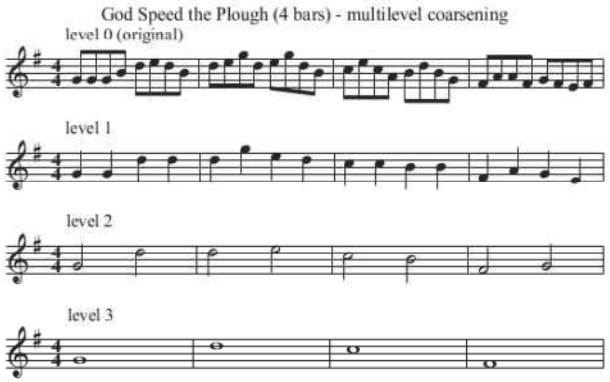

Figure 3. Multilevel coarsening of God Speed the Plough

The technique is related to, although somewhat more general than, the hierarchical contour pattern analysis technique proposed by Anagnostopoulou *et al*., (2013).

3. IMPLEMENTATION

The basis of the multilevel implementation is straightforward. Each tune is initially normalised and quantised and then recursively coarsened down to a skeleton representation with just one note per bar. Melodic matching can then take place at every level with heuristics used to optimise and enhance performance.

3.1 Normalisation

As part of the normalisation process, each tune is cleaned of grace notes, chords and other ornaments. Generally most tunes from the abc corpus are single-voiced (Walshaw, 2014), but if not, only the first voice is used for the matching.

Next, each tune is quantised so that longer notes are replaced with repeated notes (e.g. a half note is replaced with 4 eighth notes). In Western European folk music most tunes are written with eighth notes as the shortest note length; exceptions are, for example, polkas (in 2/4) and 3-time bourrées (3/8) which use sixteenth notes.

The shorter the note used for quantisation, the deeper the multilevel hierarchy and hence the more computationally complex the matching process. On the other hand quantising with longer notes can lose information, so the algorithm uses a long established heuristic in abc notation that if the time signature evaluates as fraction to 0.75 or above, tunes are quantised with eighth notes, otherwise sixteenth notes are used (for example, a tune in 3/4, which evaluates to 0.75, is quantised with eighth notes whereas a tune in 2/4 = 0.5 uses sixteenth notes).

If triplets (i.e. 3 notes in the space of 2) are encountered they are replaced by the first and third notes, but any other tuplets result in the tune being ignored (see Limitations, Section 3.4).

Figures 2 & 3 show examples of the quantisation, here labeled “level 0 (original)” which refers to the fact that no coarsening has yet taken place. In both examples, the tunes are rendered entirely as eighth notes and notice that the triplet `Bcd` in the third bar of God Speed the Plough has been replaced with the notes `Bd`.

3.2 Coarsening

The coarsening works by recursively removing “weaker” notes from each tune to give increasingly sparse representations of the melody. Coarsening progresses until there is one note remaining in each bar; it would be possible to go even further, coarsening down to one single note for a tune, but experimentation suggests that the bar is a good place to stop.

The choice of which notes to remove is subjective, based on the author’s experience of playing Western European folk music and in the current implementation the default coarsening strategy removes every other note, i.e. the off-beats (see Figures 2 & 3).

Exceptions to the “remove every other note” rule are handled with heuristics, typically for tunes in compound time. Thus for jigs in 6/8, 9/8 & 12/8, which are normally written in triplets of eighth notes, the weakest notes are generally the second of each triplet. As an example, for a slip jig in 9/8, once the tune is quantised to 9 eighth notes, the initial coarsening would remove notes 2, 5 & 8. A similar reasoning applies for waltzes, mazurkas and polskas in 3/4, so that for 3 quarter notes in a bar, the weakest is generally the second. The heuristics for dealing with these, and other less common time signatures (e.g. 5/4, 5/8, 7/8, 11/8, …), are summarised in Table 1 (although see also Section 3.4, Limitations).

Note that these heuristics are applied only once during the coarsening, specifically when the number of notes remaining in each bar is equivalent to the number of beats as expressed by the upper number of the time signature. For example, a tune in 3/8 would be quantised as 6 sixteenth notes which initially be coarsened by removing the notes 2, 4 & 6. This leaves 3 notes, 1, 3 & 5, equivalent to the number of beats and so, as indicated by the table, the second of these is removed.

Time signature	Beats per bar	Typical emphasis	Notes to remove
3/2, 3/4, 3/8	3	**1** 2 3	2
5/4, 5/8	5	**1**,2,3 **4**,5	2
6/4, 6/8	6	**1**,2,3 **4**,5,6	2 5
7/8	7	**1**,2,3 **4**,5 **6**,7	2
9/8	9	**1**,2,3 **4**,5,6 **7**,8,9	2 5 8
11/8	11	**1**,2 **3**,4 **5**,6,7 **8**,9 **10**,11	6
12/8	12	**1**,2,3 **4**,5,6 **7**,8,9 **10**,11,12	2 5 8 11

Table 1. Coarsening heuristics.

If the number of notes is not equivalent to the number of beats then evenly numbered notes are removed. For example with a slip jig in 9/8 with notes 1,2,3 4,5,6 7,8,9 at level 0, the initial coarsening using the heuristic from Table 1, removes 2, 5 & 8 leaving 1,3 4,6 7,9 at level 1. The next coarsening removes every other note, 3, 6 & 9, leaving 1, 4, 7 at level 2. This continues to give 1, 7 at level 3 and finally 1 at level 4.

Note that, in contrast to the motivational example coarsenings shown in Figures 2 & 3, this results in an unmusical result: each bar initially contains 9 eighth notes and 3 are removed so unless the remaining six notes are written as tuplets (specifically 2 notes in the time of 3), the tune no longer makes sense. However, since all notes are the same length (because of quantisation), this need not concern the implementation which is really only concerned with pitch rather than duration.

3.3 Similarity Measure

Once the multilevel representations are constructed a variety of methods could be used to actually compare tunes at each level (e.g. Kelly, 2012; Stober, 2011; Typke *et al.*, 2005). In the current implementation, each level is converted to intervals and then matching is done using the Longest Common SubString (LCSS) algorithm. However, in principle various methods can be used (this is a strength of the multilevel paradigm which is not generally tied to a particular local search strategy).

3.3.1 Multilevel Similarity

Because folk and traditional tunes can differ widely at the finest level whilst resembling each other at the coarser levels, a possibility for quantifying the similarity, S_{XY}, between a pair of tunes X and Y is simply to add up the lengths of all the LCSS values at every level. In other words, if S^l_{XY} expresses the similarity between tunes X and Y at level l then $S_{XY} = \Sigma_l S^l_{XY}$.

To illustrate this, Tables 2 & 3 show the semitone intervals for the two tunes in Figures 2 & 3 at all 4 levels. Notice that, because the coarsening is removing alternate notes at every level, every interval at levels 1, 2 & 3 is the sum of the two intervals in the parent level.

Applying the LCSS algorithm to all four levels (in reverse order) gives the following results:

- At level 3, the LCSS is 7,-2 so $S^3_{XY} = 2$
- At level 2, the LCSS is 7,0 so $S^2_{XY} = 2$
- At level 1, the LCSS is -5,-3,-2,-3 so $S^1_{XY} = 4$
- At level 0, the LCSS is 3,2,-2,-3,3,2 so $S^0_{XY} = 6$

Hence the similarity of this particular pair of tunes is quanitified as $S_{XY} = 2 + 2 + 4 + 6 = 14$. This puts value on the structural correspondences at levels 2 and 3 as well as the detailed similarities at levels 0 and 1.

Note that at levels 1 and 0 the LCSS are found at different positions in each tune. At level 1 it compares the notes `eBdcA` in Speed the Plough (bars 3 & 4) with the notes `BF#AGE` in God Speed the Plough (essentially the whole of bar 4), so is somewhat of a false positive as these two phrases are not really related (although see Section 3.3.3 for a way to avoid this). At level 0 it compares `BdedBde` from both tunes and is therefore more representative as the beamed notes `dedB` occur in both tunes (repeatedly in Speed the Plough) and are a characteristic feature of several other versions of the tune.

3.3.2 Multilevel Distance Metric

For convenience, it can be helpful to formulate the matching problem as a minimisation and hence to express the similarity as a distance, $D_{X,Y}$.

	Bar 1	Bar 2	Bar 3	Bar 4
Level 0	2 2 1 2 2 -2 -3 3	2 -2 -3 3 2 -2 -3 1	0 4 -4 -1 0 3 -3 1	0 -3 0 0 0 0 0
Level 1	4 3 0 0	0 0 0 -2	4 -5 3 -2	-3 0 0
Level 2	7 0	0 -2	-1 1	-3
Level 3	7	-2	0	

Table 2. Multilevel interval analysis for Speed the Plough.

	Bar 1	Bar 2	Bar 3	Bar 4
Level 0	0 0 4 3 2 -2 -3 3	2 3 -5 2 3 -5 -3 1	4 -4 -3 2 3 -3 -4 -1	3 0 -3 1 -1 -2 2
Level 1	0 7 0 0	5 -3 -2 -2	0 -1 0 -5	3 -2 -3
Level 2	7 0	2 -4	-1 -5	1
Level 3	7	-2	-6	

Table 3. Multilevel interval analysis for God Speed the Plough.

This is easy to do by computing, at each level l,

$$D^l_{XY} = \min(\text{length}(X^l), \text{length}(Y^l)) - S^l_{XY},$$

where length(X^l) is the length of the array of intervals at level l. Then $D_{XY} = \Sigma_l D^l_{XY}$.

In the case of the two example tunes, length(X^l) = length(Y^l) at every level and so the partial distances are given by:

- At level 3, $D^3_{XY} = 3 - S^l_{X,Y} = 1$
- At level 2, $D^2_{XY} = 7 - S^2_{XY} = 5$
- At level 1, $D^l_{XY} = 15 - S^l_{XY} = 11$
- At level 0, $D^0_{XY} = 31 - S^0_{XY} = 25$

and so the total distance is $D_{XY} = 1 + 5 + 11 + 25 = 42$.

3.3.3 Algorithmic and Other Variants

Although essentially expressing the inverse of S_{XY}, D_{XY} puts much more emphasis on the finer levels, simply because the arrays are much longer. This means that a pair of tunes which match closely at the coarsest levels may still have a relatively large D_{XY} value if the LCSS at the finest level is relatively short.

To compensate for this, a possibility is to normalise the coarser levels so that the length of the LCSS at every level has approximately the same contribution (as it does for S_{XY}). Since, in most cases, the length of the interval arrays is halved at each successive level, the simplest way to do this is just to multiply D^l_{XY} by 2^l so that

$$\underline{D}^l_{XY} = 2^l \,.\, [\min(\text{length}(X^l), \text{length}(Y^l)) - S^l_{XY}]$$

For the two example tunes this gives:

- At level 3, $\underline{D}^3_{XY} = 8 \,.\, [3 - S^l_{X,Y}] = 8$
- At level 2, $\underline{D}^2_{XY} = 4 \,.\, [7 - S^2_{XY}] = 20$
- At level 1, $\underline{D}^l_{XY} = 2 \,.\, [15 - S^l_{XY}] = 22$
- At level 0, $\underline{D}^0_{XY} = 1 \,.\, [31 - S^0_{XY}] = 25$

and so the total distance is $\underline{D}_{XY} = 8 + 20 + 22 + 25 = 75$. Although this is larger in absolute value than D_{XY}, because of the weighting, it can often better distinguish between matches.

To illustrate this further it is helpful to introduce another tune, a Northumbrian version of Speed the Plough, as shown in Figure 4, this time with an 8 bar fragment of the tune in a different key and with an anacrusis. This is a much closer match to the original "Speed the Plough" than "God Speed the Plough" is.

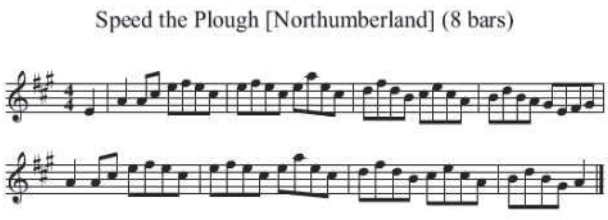

Figure 4. A further tune variant for Speed the Plough.

Denoting "Speed the Plough [Northumberland]" as tune Z (where X refers to "Speed the Plough" and Y to "God Speed the Plough"), the similarity values are $S_{XZ} = 21$ and $S_{YZ} = 18$ as compared with $S_{XY} = 14$. The distance values are $D_{XZ} = 35$ and $D_{YZ} = 38$ as compared with $D_{XY} = 42$ and the normalised distance values are $\underline{D}_{XZ} = 57$ and $\underline{D}_{YZ} = 71$ as compared with $\underline{D}_{XY} = 75$ showing that the normalised distance, $\underline{D}$, gives a better spread than just the distance, D, and clearer discrimination.

Another possible algorithmic variant is the way in which tunes of different lengths are compared. Using the minimum of the two lengths, i.e. min(length(X^l), length(Y^l)), instead of the maximum to calculate the distance D^l_{XY} from the similarity S^l_{XY} seems sensible, particularly in a search context where the query is likely to be a tune fragment. However, in other settings it might not be appropriate and this is explored experimentally in a sister paper (Walshaw, 2015).

With regard to the matching at each level, a potential pitfall of the LCSS algorithm is that it may give false positives by matching short phrases from completely different parts of the tune. One way around this is to include bar markers or even bar numbering within the strings that are to be matched. So, for example, level 2 of "Speed the Plough" which would normally be represented as "7,0,0,-2,-1,1,-3" (see Table 2) could instead be represented as "7,0,|,0,-2,|,-1,1,|,-3" where the "|" symbols represent bar lines. This means that any matched common substrings must respect bar lines (unless they are shorter than the length of a bar).

Furthermore, if the bar symbols are numbered, e.g. "7,0,$|^1$,0,-2,$|^2$,-1,1,$|^3$,-3", then matched common substrings must also respect the position in the tune. (If matching of subsections of the tune is important then the numbering can be restarted at natural breaks such as double bar lines and repeat marks; however, that is not been tested here.)

In terms of implementation, the "strings" of intervals are represented as an array of short integers so that bar markers (or numbers) can easily be included with large integer values outside the possible range of intervals.

This inclusion of bar markers or numbers is more of a representational variant than an algorithmic one and does increase the computational complexity of the multilevel matching (as the strings to be compared by the LCSS algorithm are longer). However, it has a significant effect on the results and is an important component of the multilevel LCSS algorithm.

Finally, the multilevel framework also allows for the use of optimisation heuristics to terminate the matching process early, at the coarser levels, when it looks unpromising. This is discussed in more detail alongside the experimentation (see Section 4.5).

3.4 Limitations

The current implementation is a prototype and there are a number of tune features it cannot handle fully. Some are treated as exceptions, in which case the tune is excluded from the search data, and in most cases it should be possible to improve the handling in future versions:

- **Tunes with no time signature** are treated as exceptions and excluded for the time being. An easy option is to include them and just coarsen by removing alternate notes for, say, 4 or 5 levels.
- **Tunes which change time signature** are treated as exceptions and excluded for the time being. This is

just an implementation issue and in principle the methods should work by taking account of changes and applying the coarsening methods accordingly.

- **Complex time signatures / emphasis patterns** may mean that the heuristics in Table 1 are incorrect. For example, some tunes in 5/4 have a **1**,2 **3**,4,5 emphasis pattern and there are Eastern European tunes in 9/8 with **1**,2,3 **4**,5 **6**,7 **8**,9. It is probably impossible to identify all of these anomalies accurately and anyway these are not common in the corpus (which is mostly comprised of Western European and North American tunes) so they treated as any other tune.
- **Tuplets –** other than triplets, tuplets are not yet handled correctly and any tune with tuplets is excluded.
- **Short bars** (e.g. anacruses & repeat bars) are treated like any other bar meaning that the Table 1 heuristics may not be applied correctly if the bar is shorter than it would normally be.
- **Polyphonic tunes** are currently handled by extracting the first voice. In principle it should be possible to extract each voice and treat it as a separate tune, but this is not yet implemented.
- **Repeat signs** are currently ignored so that two identical tunes, one with repeat signs and one written out in full would not have a distance measure of 0. In principle it would be possible to expand all repeats, but this would significantly add to the computational complexity of the matching.
- **Transcription errors** and extraneous symbols in the abc code are mostly ignored but if the parser really cannot understand the input, the tune is excluded.

4. EXPERIMENTATION

4.1 Experimental Framework

The experimental framework uses two illustrative search queries to assess and demonstrate some of the aspects of the multilevel search algorithm. Both are chosen as examples of tunes which are very well known in the Western European folk canon and which therefore have many variants represented in the abc music corpus.

The first search query is for the first two bars of "Speed the Plough" (see Figure 1), a tunethought to have been composed by John Moorehead, originally from Edinburgh, in around 1800 and named after its inclusion in an eponymous play[1].

The second search query is the first two bars of the tune "Black Joker" (also known as "Black Joke", "Black Jack", "Black Jock", "Black Joak", etc., as well as "But the House and Ben the House" in the Shetland Isles, "Sprig of Shillelagh" in Ireland and "La Badine" in the Netherlands). This is an older tune, dating from at least the early eighteenth century, and still popular for morris dancing[2]. Figure 5 shows the first 4 bars of a number of variants (all drawn from the test dataset, Section 4.3), as well as the search query.

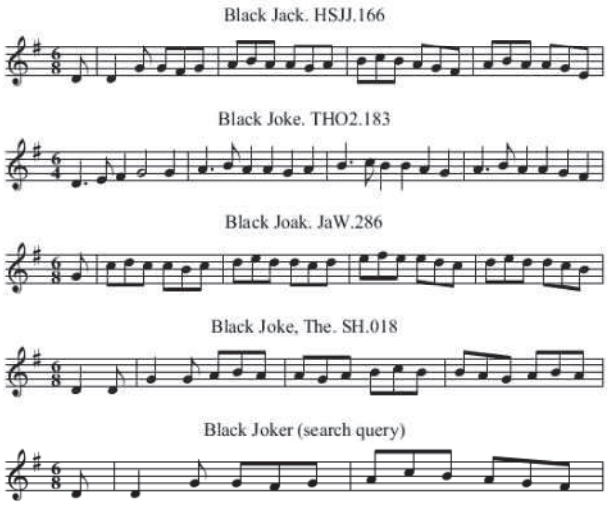

Figure 5. Several variants of the incipit for Black Joker.

The queries are run on two datasets: a small set of test data (5,610 tunes) and the entire corpus (460,000 tunes, reduced to 168,960 after various groups are excluded).

4.2 Variants Tested

A number of algorithmic and representational variants are tested. In particular:

- **Normalisation**: as described in Section 3.3.3 this weights the distance measure so that matching substrings of equal length at different levels of the multilevel representation contribute approximately the same amount to the distance measure.
- **Bar markers/numbers:** as described in Section 3.3.3 bar markers or numbers can be included in the "strings" of intervals matched at each level. This forces the LCSS algorithm to respect the position of notes relative to bar lines (if using markers) or additionally the position of notes within the tune (if using numbers).
- **Single-level search:** in order to provide a comparison, all the tests are also performed with single level searches, i.e. so that the LCSS algorithm is just used at level 0, ignoring the multilevel representation.

4.3 Results – Test Dataset

The initial experimentation uses a small subset of the full corpus consisting of the 5,638 abc transcriptions taken from www.village-music-project.org.uk[3], a collection of social dance music from England mostly transcribed from hand-written manuscript books in museums and library archives. Of these 28 are removed due to implementation limitations (see Section 3.4) leaving 5,610.

One of the advantages of using this dataset is the diversity of the material, sometimes with several versions for each tune. Another advantage is that the tunes include the manuscript they are taken from abbreviated in the tune title, making it easy to identify specific instances.

[1] See http://www.ibiblio.org/fiddlers/speed.htm for a comprehensive history

[2] See http://www.ibiblio.org/fiddlers/BLACK.htm

[3] See http://village-music-project.org.uk/

4.3.1 Search Metrics

Testing a search algorithm, particularly in a corpus where there is no "ground truth" established, is not an easy matter. In the testing below, the aim is to find all the versions of the tune with matching titles, solely by using the musical representation. In other words, the testing considers tunes which contain "Speed the Plough" or "Black Jack/Joke/Joke" (because of the variant spellings) in the title and where they appear in the search results.

In the case of the test data there are 10 instances with "Speed the Plough" in the title and 9 instances with "Black Jack" / "Black Joak" and "Black Joke" (plus 2 with "Sprig of Shillelah" and "Sprig of Shillela" which are not considered). These two sets of instances are each respectively referred to as **target sets** below.

One way of assessing the results is consider the entire dataset and how "distant" each target set is from the query. Two metrics are used for this purpose:

- **Avg. distance**: the simplest measure is just to average all the distances, D_{ST}, of tunes, T, in the target set from the search query, S, so that if {T} represents the target set, calculate $\Sigma_{\{T\}} D_{ST} / |\{T\}|$. However, since the various algorithmic variants can have different scales for the distance calculation, the distances are all normalised by dividing by the maximum possible distance to give values between 0 and 1.
- **Halfway index**: another way to assess the results is to consider how many tunes would need to be shown in the search results before the entire target set appeared. However, this measure could be badly influenced by outliers in the target set (which have a matching tune title but which are not similar musically) so a more robust metric is to consider how many tunes need to be shown in the results before half of the target set appears. This is known as the **halfway index**.

Another way of assessing the results is to use the distance measure to identify a **results set** of tunes close to the search query, i.e. where $D_{ST} < \Delta$ for some chosen value Δ. The results set then contains the tunes offered to the user performing the search. Two further metrics are used to assess the results set:

- **Size**: ideally the results set should be small enough to be useable, but large enough to contain some diversity (e.g. similar tunes that the user might not be aware of).
- **Instances**: a count of how many instances from the target set are contained in the results set.

4.3.2 Results

Table 4 shows the results for the two search queries in the test dataset. The first 3 columns indicate the algorithmic variant (as described in Section 4.2), and in particular: the search type, either multilevel (ML) or single-level (SL); whether normalisation is applied to multilevel distances give each level equal weight; whether bar markers or numbers are included in the tune representation. The results are in the last 4 columns with the "best" values indicated in boldface.

Variant			Target set		Results set (Δ = 0.5)	
Type	Normalisation	Bars	Halfway index	Avg. distance	Size	Instances
Speed the Plough						
ML	yes	numbered	**5**	**0.187**	264/5,610	**10/10**
ML	yes	marked	6	**0.187**	2,612/5,610	**10/10**
ML	yes	ignored	9	0.265	1,310/5,610	**10/10**
ML	no	numbered	6	0.382	16/5,610	8/10
SL	-	numbered	94	0.659	4/5,610	3/10
Black Joker						
ML	yes	numbered	**12**	0.364	290/5,610	6/9
ML	yes	marked	19	**0.299**	2,829/5,610	**7/9**
ML	yes	ignored	24	0.378	1,387/5,610	**7/9**
ML	no	numbered	34	0.500	19/5,610	4/9
SL	-	numbered	187	0.650	6/5,610	3/9

Table 4. Results for the test dataset.

As can be seen, the tables suggest that the multilevel search with normalisation (top 3 rows of results) generally offers the best results in terms of the halfway index, the average distance and the number of target instances in the results set.

The single level scheme means essentially just using LCSS as a search strategy and substantially increases the average distance. This could just be an issue of scaling in the distance function but it also hugely increases the halfway index, fundamentally because single-level LCSS is an exact search with none of the approximate matches provided by the coarser levels in the multilevel versions. As a consequence the results set is tiny and does not contain many target instances (although it might be possible to tailor the value of Δ to improve this).

The multilevel results without normalisation seem to confirm these findings. Normalisation weights the distance contribution to give approximately equal emphasis from each level and if it is not used the distance measure is heavily biased towards the matching at the finest (original level), so it is not too dissimilar from using a single-level LCSS search. Nonetheless, the contributions from coarser levels do seem to discriminate further between different tunes and hence improve the results.

The choice of whether to use bar markers / bar numbering could easily be left to the user and, in particular, bar numbering only makes sense if the search query is known to come from the start of the tune (although it could be adapted to apply to the start of each section of the tune or for user-chosen numbering). However, the results make it clear that it is significantly better to use bar marking or numbering than to ignore the bar lines.

Overall the best results seem to come from using the multilevel algorithm with normalisation and bar numbering. This configuration produces half of the target in-

stances in the first few results (as measured by the halfway index) and the results set for $\Delta = 0.5$ contains a few hundred tunes (around 5% of the dataset). The results set does not always contain all of the target instances but that is because not all of the targets are particularly close matches – in particular the 3 missing for the Black Joker query are shown in Figure 5 with the manuscript identifiers THO2.183, JaW.286 & SH.018; of these the first has, unusually, been transcribed in 6/4, the second is not a close match over the first 2 bars (although would have been if the search query were 4 bars long), whilst the third has been transcribed (on the original manuscript) with the bar lines in the wrong place.

If bar marking is used instead of numbering, the size of the results set increases by a factor of around 10 in both cases, mostly likely as the result of a large number of false positive matches from elsewhere in the tune. These enlarged results sets also account for the fact that they contain the highest number of target set instances.

To tease this out a little further it is worthwhile studying the histograms of distances, shown in Figure 6, here for Speed the Plough, although those for Black Joker are similar (note that the scaling on the *x*-axis is different when bars are ignored as the representation strings are shorter and hence the maximum distance is smaller).

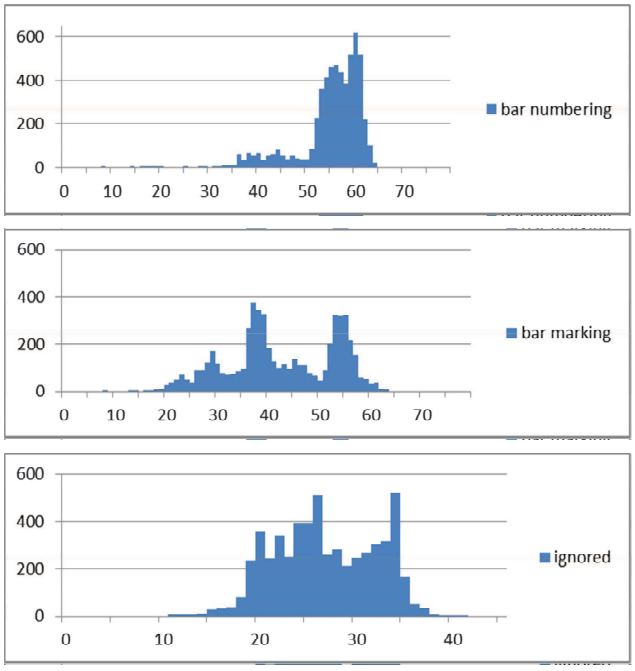

Figure 6. Distance histograms for Speed the Plough.

These histograms show the extent of the results set: for bar marking/numbering, any tune within a distance of 39 ($\Delta = 0.5$) is included in the results set. When bars are ignored the distance is 22.

What is clear from them is that bar numbering is better at separating out the results with a gradual accumulation of matching tunes up to the value 50 (corresponding to $\Delta \approx 0.625$) and the main bulk of tunes coming after that. In contrast, bar marking or ignoring bar lines altogether, spreads the distance results out more evenly and so the results set can grow rapidly as Δ is increased.

4.4 Results – Entire Corpus

The second data set is the entire abc corpus which at the time of writing consists of around 460,000 tunes from across the web (Walshaw, 2014). Of these 244,000 are exact electronic duplicates which are excluded and another 40,000 are potentially copyright and also ignored. A further 7,000 are excluded because of implementation limitations (see Section 3.4), leaving a total of 168,960 used for testing.

The target sets in this case (i.e. those tunes which have closely matching titles) contain 84 versions of Speed the Plough and 88 versions of Black Joker.

Table 5 shows the results for the full corpus set out as previously. These seem to bear out the findings established for the test dataset – the single-level and multilevel with no normalisation give the worst results and, in the multilevel results with normalisation, using bar numbering provides the best results in terms of the halfway index and nearly the best results in terms of the number of target instances in the results set (once again, the best results arise from results sets which are many times bigger).

Variant			Target set		Results set ($\Delta = 0.5$)	
Type	Normalisation	Bars	Halfway index	Avg. distance	Size	Instances
Speed the Plough						
ML	yes	numbered	**91**	0.242	6,651/168,960	69/84
ML	yes	marked	96	**0.216**	71,163/168,960	**73/84**
ML	yes	ignored	117	0.271	37,171/168,960	69/84
ML	no	numbered	129	0.360	318/168,960	60/84
SL	-	numbered	341	0.527	168/168,960	41/84
Black Joker						
ML	yes	numbered	**159**	0.326	7,688/168,960	58/88
ML	yes	marked	273	**0.286**	77,734/168,960	**64/88**
ML	yes	ignored	314	0.332	36,209/168,960	**64/88**
ML	no	numbered	201	0.434	397/168,960	53/88
SL	-	numbered	2882	0.582	179/168,960	35/88

Table 5. Results for the full corpus.

4.5 Optimisations

LCSS is computationally expensive, especially since, in the multilevel context, it may be carried out at 4 or sometimes 5 levels. However the multilevel framework also allows for the use of optimisation heuristics to terminate the matching early, at the coarser levels, when it looks unpromising and this is used to significantly speed up the matching process, as follows.

4.5.1 Distance Threshold Optimisation

If a distance threshold, Δ, is set it provides a natural early termination condition when comparing tunes, so that as each D^{I}_{XY} contribution is added in to the distance meas-

ure D_{XY}, the matching can be ended as soon as the partial sum $\Sigma_l D^l_{XY}$ is greater than the current distance threshold. Since LCSS is an $O(n^2)$ operation this can result in significant savings in computational complexity, especially if, as is typical in the multilevel framework (Walshaw, 2008), the matching is carried out coarsest to finest with the shortest arrays compared first.

4.5.2 Coarse Level Matching Limit (CLML)

Distance threshold optimisation will not change the results set, but if the desire is refine the search accuracy an option is to exclude any tune from the results if the length of the LCSS for the coarsest level L is less than some minimum matching limit M^L or, in other words, $S^L_{XY} < M^L$. With a short search query of 2 bars, as used here, it makes sense to set $M^L = 2$.

Since the coarsest level has one note per bar and the final note remaining in each bar during coarsening is always the first note in the bar, this is effectively the same as saying that candidate tunes must have the same first note as the search query in each bar, for its entire length.

Note also that, in principle, it would be possible to employ this heuristic at any level by setting minimum matching limits for finer levels, e.g. M^{L-1}, M^{L-2}, etc. However this has not been tested.

4.5.3 Results

Table 6 shows the results of the two optimisations applied the multilevel algorithm (using normalisation and bar numbering). Here the runtime gives the time, in milliseconds to search through 168,960 multilevel tune representation (although not to read the tunes in from file).

	Target set		Results set		
Optimisation	Halfway index	Avg. distance	Size	Instances	Runtime (ms)
Speed the Plough					
none	91	0.242	6,651	69/84	1,295
threshold	91	0.242	6,651	69/84	822
CLML	**45**	n/a	5,912	69/84	**250**
Black Joker					
none	159	0.326	7,688	58/88	1,423
threshold	159	0.326	7,688	58/88	853
CLML	**70**	n/a	7,330	57/88	**400**

Table 6. Optimisation results.

As can be seen, the distance threshold optimisation does not change the results qualitatively, but does reduce the runtime by over a third. In contrast the CLML optimisation, by excluding tunes which do not completely match the search query at the coarsest level, not only reduces the runtime still further, but also significantly improves the halfway index of the target sets.

5. CONCLUSION

This paper has presented a multilevel algorithm for matching tunes when performing inexact searches in symbolic musical data.

The basis of the algorithm is very straightforward: each tune is initially normalised and quantised and then recursively coarsened, typically by removing weaker offbeats, until the tune is reduced to a skeleton representation with just one note per bar. Melodic matching can then take place at every level, coarse to fine.

The matching implemented here uses the Longest Common SubString (LCSS) algorithm, but in principle a variety of similarity measures could be used.

The multilevel framework allows inexact matches to occur by identifying similarities at course levels and is also exploited with the use of early termination heuristics at coarser levels, both to reduce the computational complexity and enhance the matching qualitatively.

Initial results suggest that the algorithm, coupled with the LCSS, works very well at performing searches on a corpus of tunes. However, this is only a prototype and further work remains to be done, in particular:

- further testing with a broader, more diverse range of search queries;
- exploration of different similarity measures within the algorithm;
- better handling of exceptions and anomalies (see Section 3.4) to improve the robustness.

6. REFERENCES

Anagnostopoulou, C., Giraud, M., & Poulakis, N. 2013. Melodic contour representations in the analysis of children's songs. In P. van Kranenburg *et al.*, Eds., *3rd Intl Workshop on Folk Music Analysis*pp. 40–43. Amsterdam: Meertens Institute and Utrecht University.

Kelly, M. B. 2012. *Evaluation of Melody Similarity Measures*. Queen's University, Kingston, Ontario.

Stober, S. 2011. Adaptive Distance Measures for Exploration and Structuring of Music Collections. In *Audio Engineering Society Conf.*, pp. 1–10.

Typke, R. 2007. *Music Retrieval based on Melodic Similarity*.

Typke, R., Wiering, F., & Veltkamp, R. C. 2005. A survey of music information retrieval systems. In *Proc. ISMIR*, pp. 153–160.

Walshaw, C. 2008. Multilevel Refinement for Combinatorial Optimisation: Boosting Metaheuristic Performance. In C. Blum, Ed., *Hybrid Metaheuristics - An emergent approach for optimization*pp. 261–289. Springer, Berlin.

Walshaw, C. 2014. A Statisical Analysis of the Abc Music Notation Corpus. In A. Holzapfel, Ed., *4th Intl Workshop on Folk Music* Analysis, pp. 2–9. Istanbul: Bogaziçi University.

Walshaw, C. 2015. A Multilevel Melodic Similiarity Framework. In *Proc. ISMIR (submitted)*.