String Matching and Geometric Algorithm for Melodic Similarity

Kjell Lemström, Niko Mikkilä, Veli Mäkinen and Esko Ukkonen
C-BRAHMS Group, Department of Computer Science
P.O.Box 68 (Gustaf Hällströmin katu 2b)
FIN-00014 University of Helsinki, FINLAND
{klemstro,mikkila,vmakinen,ukkonen}@cs.helsinki.fi

Keywords: MIREX'05, Melodic Similarity, String Matching, Geometric Matching

1 INTRODUCTION

This abstract gives an overview on two algorithms, developed earlier in the C-BRAHMS group (Lemström et al. (2003)), that took part in the MIREX'05 melodic similarity contest. Given two excerpts of symbolically encoded music, the *query pattern* and the *target music*, the task of both these algorithms is to find musically relevant occurrences of the query pattern within the target music.

The first algorithm (Lemström and Ukkonen (2000)) is based on the string matching framework. It is suitable for assessing the similarity of monophonic (pitch) sequences. The similarity (or actually distance) between the query and its occurrence is expressed by a number stating the minimum of required editing operations to convert the query sequence into that of the found occurrence.

The other algorithm (Ukkonen et al. (2003)), based on geometric matching, is somewhat more complicated but also more flexible: either (or both) of the sequences may be polyphonic. Using the piano-roll representation, the intuitive idea is to slide the vertical bar-lines representing the query over the piano-roll representation of target and to find the position that gives the maximal common shared time (maximal overlapping of the two sets of vertical bar-lines).

Both the algorithms are computationally very efficient. The former can be calculated in time $O(mn)$ by using dynamic programming, the latter in $O(mn \log m)$. Here m and n denote the number of musical events (notes) in query and target, respectively. In the following two sections we present the fundamental parts of these algorithms. We refer the reader to (Lemström and Ukkonen (2000); Ukkonen et al. (2003)) for more details.

2 String Matching Algorithm

Basing the similarity measure on pitch values, one has two possibilities. Either to use the *absolute* or the *relative* values (i.e. intervals between consecutive pitches). The latter representation has the advantage over the first, that it also finds transposed occurrences of the query. On the other hand, substituting one absolute pitch value corresponds to two substitutions in the relative representation (see e.g. Lemström and Ukkonen (2000)). Thus, in the case of several individual erroneous notes, the user defined error threshold is exceeded much earlier by an algorithm based on relative encoding than it would be by one based on absolute encoding.

The idea in our first candidate is to try to capture the advantages of both these encodings. It always chooses which out of the two encodings would locally give the best match. Let us denote the query and the target by strings $A = a_1, \ldots, a_m$ and $B = b_1 \ldots b_n$, respectively, where each $a_i (1 \leq i \leq m)$ and $b_j (1 \leq j \leq n)$ are taken from an alphabet over pitch values. Our algorithm evaluates the following simple recurrence.

$$d_{0j} = 0 \quad (\forall j,\ 0 \leq j \leq n)$$

$$d_{ij} = \min \begin{cases} d_{i-1,j} + 1 \\ d_{i,j-1} + 1 \\ d_{i-1,j-1} + (\text{if } a_i = b_j \text{ or } a_i - a_{i-1} = b_j - b_{j-1} \text{ then } 0 \text{ else } 1). \end{cases}$$

Evaluating the recurrence above corresponds to calculating all the values of an $m \times n$-sized table (d_{ij}). Any value d_{mj} (value at the bottom row of (d_{ij})) corresponds to the smallest possible distance between the query and any of the substrings of target that ends at position j.

Note that the only refinement to the conventional algorithm (finding only absolute or transposed occurrences), is the last possibility for the minimum operation. Note also that (strictly speaking) our algorithm is not transposition invariant.

3 Geometric Algorithm

Our primary candidate is based on a geometric sweepline technique. As described above, the intuition behind this algorithm is to slide the vertical bar-lines representing the query over the piano-roll representation of target and to find the position that gives the maximal common shared time (see Figures 1 and 2 below).

To this end, the piano-roll representation of target music is given to the algorithm as lexicographically ordered turning points (starting and ending points of each note). The notes of the query are ordered by their starting points.

The algorithm first populates a priority queue with two translation vectors for each turning point in the pattern. In the beginning the vectors point to first starting and ending

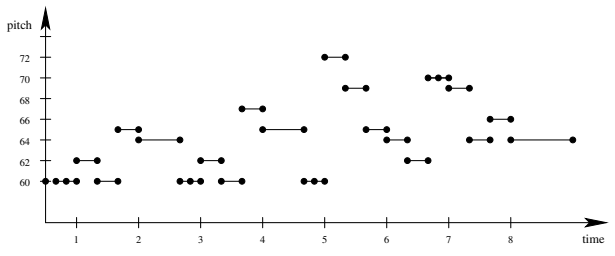

Figure 1: Query in piano-roll representation.

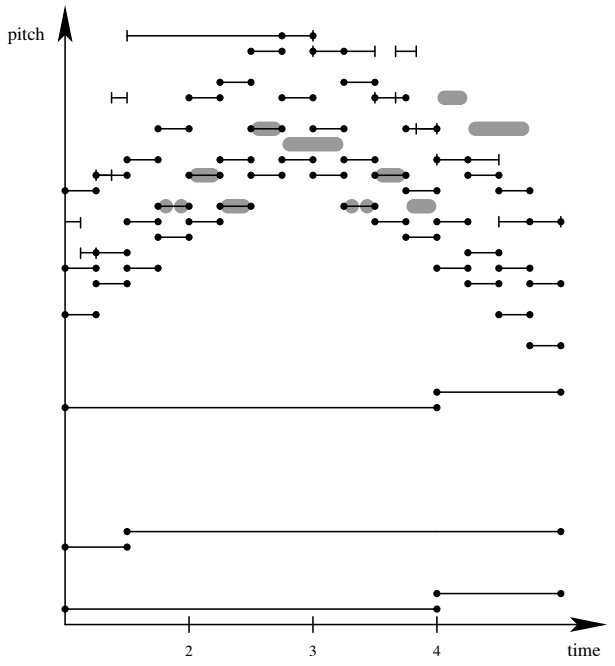

Figure 2: Target in piano-roll representation. The first twelve notes of the query in Figure 1 are shown by shading in a translated position such that the total length of the overlapping is six quarter notes.

point in the target. After this initialization, the algorithm loops through all possible translation vectors between the query and the target. At each iteration the first vector in lexicographic order is retrieved from the queue and replaced by the next corresponding vector (a vector for the same turning point in the query) that has not yet been inserted into the queue.

When the algorithm iterates over translation vectors, it counts common time between the query and the target on each vertical translation level to ensure transposition invariance. Common time is counted by using a linear slope that the translation vectors adjust, and the maximal overlapping is simply checked for at each iteration. Finally, normalizing the maximal overlapping by the combined length of query or target notes (whichever is smaller) results in a value that expresses the similarity of the query and the target.

4 Results

The goal in the MIREX'05 Symbolic Melodic Similarity contest was to retrieve the most similar incipits from the RISM A/II collection, given one of the incipits as a query. The dataset contained 558 MusicXML files of the incipits from the RISM collection. 11 incipits of this collection were chosen as queries. For further information about the contest setting and evaluation, we refer to Typke (2005).

The results for the seven contestants are given in Table 1. For each contestant, five figures were drawn from the experiments; two of them based on precision, two on recall and one related to the running time. The scores were averaged over the 11 queries. In the table we have emphasized the best three contestants in each of the five categories. Our submissions are marked with a bold typeface. DP refers to the string matching algorithm and P3 to the geometric algorithm.

When considering recall, three contestants outperformed the rest. The average dynamic recall and normalized recall for the first three was more than 60 % and 50 %, respectively. For the remaining four, including our algorithms, the figures were in the range of 50 to 60 % and 40 to 50 %, respectively.

As regards to precision, Grachten, Arcos & Mántaras's method outperformed the others: it was the only one having average precision over 50 %. Three more contestant crossed the border of 40 %, including our sweepline technique.

Looking at the running times, our algorithms outperformed the others by an order of the magnitude. That is an issue to be taken into account when dealing with very large databases (recall also that our geometric sweepline method works as well with polyphonic as with monophonic music). Moreover, as the reported running times include times spent for evaluation of the results, the real relative difference is even larger than the reported (it has been announced that the evaluation took some 4 seconds, that is approximately 40 % of the reported times of our algorithms while, for instance, only 5 % of that of Grachten, Arcos & Mántaras's method).

The relatively large difference in running times between our algorithms and the others can be partly explained by more efficient MIDI parsing and some optimized data structures. As the incipits were very short and there were a lot of them, initializing a heavy MIDI parser and the algorithm for each of them separately could easily waste a lot of time. Still, the main reason for the difference is most likely the fact that our algorithms were designed with much larger data sets in mind. It should also be noted that in practice, with long target music and short queries, the geometric algorithm is slower than the string matching algorithm. However, the string matching algorithm does not work well with polyphonic music, which is harmful in some applications. Both algorithms were implemented completely in Java for compatibility reasons, but a faster C version of the geometric algorithm is also available.

Rank	Participant	ADR	NR	Average Precision	Precision at N documents	Input	Runtime (s)	Machine
1	Grachten, Arcos & Mántaras	65.98%	55.24%	51.72%	44.33%	MIDI	80.174 *	B0
2	Orio, N.	64.96%	53.35%	42.96%	39.86%	XML	24.610	B4
3	Suyoto & Uitdenbogerd	64.18%	51.79%	40.42%	41.72%	MIDI	48.133	B3
4	Typke, Wiering & Veltkamp	57.09%	48.17%	35.64%	33.46%	MIDI	51.240	B4
5	**Lemström, Mikkilä, Mäkinen & Ukkonen (P3)**	**55.82%**	**46.56%**	**41.40%**	**39.18%**	**MIDI**	**10.007** *	**B0**
6	**Lemström, Mikkilä, Mäkinen & Ukkonen (DP)**	**54.27%**	**47.26%**	**39.91%**	**36.20%**	**MIDI**	**10.106** *	**B0**
7	Frieler & Müllensiefen	51.81%	45.10%	33.93%	33.71%	MIDI	54.593	B4

Table 1: 2005 MIREX Contest Results - Symbolic Melodic Similarity
ADR is Average Dynamic Recall and NR is Normalized Recall at group boundaries. For our algorithms, DP refers to the string matching algorithm and P3 to the geometric algorithm. All the machines B0, B3 and B4 had the same hardware and software specifications.
Note * : These runs were executed in M2K environment, and thus the runtime includes evaluation time.

References

K. Lemström, V. Mäkinen, A. Pienimäki, Mika Turkia, and E. Ukkonen. The C-BRAHMS project. In *Proceedings of the ISMIR'03 4th International Conference on Music Information Retrieval*, pages 237–238, Baltimore, October 2003. URL `http://www.cs.helsinki.fi/group/cbrahms/`.

K. Lemström and E. Ukkonen. Including interval encoding into edit distance based music comparison and retrieval. In *Proceedings of the AISB'00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, pages 53–60, Birmingham, April 2000.

R. Typke. Symbolic melodic similarity. In MIREXWiki, 2005. URL `http://www.music-ir.org/mirexwiki/index.php/Symbolic_Melodic_Similarity`.

E. Ukkonen, K. Lemström, and V. Mäkinen. Sweepline the music! In *Computer Science in Perspective — Essays dedicated to Thomas Ottmann*, volume 2598 of *Lecture Notes in Computer Science*, pages 330–342. Springer-Verlag, 2003.