A Fast, Randomised, Maximal Subset Matching Algorithm for Document-Level Music Retrieval

Raphaël Clifford
University of Bristol, Merchant Venturers' Building
Woodland Road, Bristol BS8 1UB, UK
clifford@compsci.bristol.ac.uk

Manolis Christodoulakis
King's College, London
The Strand, London WC2R 2LS, UK
manolis@dcs.kcl.ac.uk

Tim Crawford, David Meredith, Geraint Wiggins
Goldsmiths College, University of London
New Cross, London SE14 6NW, UK
{t.crawford,d.meredith,g.wiggins}@gold.ac.uk

Abstract

We present **MSM**, a new maximal subset matching algorithm, for MIR at score level with polyphonic texts and patterns. First, we argue that the problem **MSM** and its ancestors, the SIA family of algorithms, solve is 3SUM-hard and, therefore, subquadratic solutions must involve approximation. **MSM** is such a solution; we describe it, and argue that, at $O(n \log n)$ time with no large constants, it is orders of magnitude more time-efficient than its closest competitor. We also evaluate **MSM**'s performance on a retrieval problem addressed by the OMRAS project, and show that it outperforms OMRAS on this task by a considerable margin.

Keywords: Pattern matching, point set representation.

1. Introduction

1.1. Overview

We present **MSM**, a new fast, randomised maximal subset matching algorithm which is applicable to MIR at the level of (quantised) score encodings. This is a useful level of representation because there is a vast amount of music stored in this form (though not yet digitally), and because such "mid-level" representations constitute a helpful abstraction in understanding perceived musical structure (Bello and Pickens, 2005), and arise as output from, for example, MIDI recorders and transcription systems. We currently focus on idealised representations containing only restricted forms of noise; we will generalise this later.

Our algorithm finds the largest subset of a pattern appearing in a text at an arbitrary offset in any of the dimensions represented. In our representation, this means transposition invariant and time offset tolerant matching.

The rest of this section explains why point-set representations are better for note-level MIR than string-based ones. Section 2 summarises related work in point-set matching and relevant aspects of the OMRAS project, against which we evaluate our results. Section 3 presents first a proof that the problem we solve is 3SUM-hard, and so is unlikely to have an exact solution better than $O(n^2)$ in time, and then a randomised, approximate $O(n \log n)$ time solution. Section 4 presents comparisons, with a baseline of the OMRAS matching method, and Ukkonen et al.'s (2003) P3, applied to query-by-example (i.e., large queries). Section 5 concludes and suggests future work.

1.2. Point-set matching in music

Most approaches to music pattern matching proposed to date presuppose that the music is represented as a string of symbols or a set of such strings. Typically, each voice is represented as a string and each symbol within each string represents either a note or an interval between two consecutive notes in a voice. Usually, the similarity between two patterns is measured by *edit distance*, calculated using dynamic programming (Bellman, 1957). Authors using this approach include Crawford et al. (1998), Lemström (2000), Guo and Siegelmann (2004) and Cambouropoulos et al. (2005).

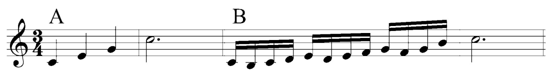


Figure 1: Pattern B is perceived as an elaboration of pattern A, but the musical edit distance between them is 9, which is large, being more than twice the number of notes in A.

However, there are several problems with this approach. Consider the patterns, A and B, in Figure 1. B is clearly perceived to be an embellished version of A. So we would want a musical pattern matching algorithm to identify these two patterns as versions of the same motif. However, if the two patterns in Figure 1 are represented as strings in which each symbol is a note, then the edit distance between them is 9 because 9 notes need to be inserted into A to get B. Since A only contains 4 notes, an edit distance of 9 is large. But if we allow A to match with patterns that differ from it by an edit distance of 9, we obtain many spurious matches.

Another problem with string-based approaches to musical pattern matching arises when we search in polyphonic music. Specifically, if we do not know the voice to which each note belongs or if we are interested in patterns containing notes from two or more voices, then the number of passage-length strings required to represent the passage fully is exponential in the number of distinct onsets; if we don't search *all* of these strings, we risk missing occurrences of the query. This is illustrated in Figure 2, which shows a graph of pitch against time representing the beginning of *Frère Jacques*. Each point in the figure represents a note

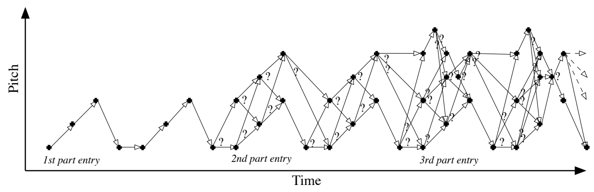


Figure 2: Fully representing a passage of unvoiced polyphonic music produces a combinatorial explosion.

and all pairs of notes that could be consecutive within a single string are linked by an arrow. The number of strings required is multiplied by k each time a note is followed by k notes that start simultaneously.

These problems are avoidable by representing the music *geometrically*, as a set of points (or line segments) in a multi-dimensional Euclidean space. A two-dimensional representation has been used, in which one dimension represents time and the other pitch. We follow this approach.

2. Background

2.1. Point-set pattern matching in music

We now review basic concepts of the geometrical approach and existing work on point-set pattern matching in music.

Let $\mathbf{v}$ be a vector and $\mathbf{p}$ a point in a k-dimensional Euclidean space. Then $\mathbf{p}+\mathbf{v}$ denotes the point that results when $\mathbf{p}$ is translated by $\mathbf{v}$. Similarly, let P be a set of points and $\mathbf{v}$ be a vector in a k-dimensional Euclidean space. Then $P+\mathbf{v}$ denotes the point set that results when all points in P are translated by $\mathbf{v}$. That is, $P+\mathbf{v}=\{(\mathbf{p}+\mathbf{v}) \mid \mathbf{p}\in P\}$. Let P and T be k-dimensional point sets and $\mathbf{v}$ be a k-dimensional vector. Then P is *translatable in T by* $\mathbf{v}$ iff $P\subseteq T$ and $P+\mathbf{v}\subseteq T$. Two point sets, P and Q, are *translationally equivalent* iff there exists a vector, $\mathbf{v}$, such that $P+\mathbf{v}=Q$. P is *the maximal translatable pattern (MTP) in T for the vector,* $\mathbf{v}$, iff P is the largest point set that is translatable in T by $\mathbf{v}$. In other words, the MTP in T for a vector $\mathbf{v}$ is $\{\mathbf{p} \mid (\mathbf{p}\in T)\wedge((\mathbf{p}+\mathbf{v})\in T)\}$.

The *maximal match* for a pattern point set, P, in a text point set, T, for a vector, $\mathbf{v}$, is the ordered pair, $\langle\mathbf{v},S\rangle$, such that S contains all the points in P that can be translated by the vector $\mathbf{v}$ to give points in T. That is, the maximal match for a query set, P, in a text set, T, for a vector, $\mathbf{v}$, is $\langle\mathbf{v},S\rangle$ such that $S=\{\mathbf{p} \mid (\mathbf{p}\in P)\wedge((\mathbf{p}+\mathbf{v})\in T)\}$.

Meredith et al. (2002a) present SIA, an algorithm that finds all non-empty MTPs in a k-dimensional point set of size n in $O(kn^2\log_2 n)$ time and $O(kn^2)$ space. Wiggins et al. (2002) generalise SIA to a general pattern matching algorithm, SIA(M), which finds all maximal matches of a k-dimensional pattern point set, P, of size m, in a k-dimensional text point set, T, of size n, in $O(knm\log_2 n)$ time and $O(knm)$ space. They implement version SIA(M)Ex, which finds all the *complete* occurrences of a pattern point set of size m in a text point set of size n in $O(knm)$ time, and version SIA(M)E which finds all complete *and partial* matches in $O(knm\log(nm))$ time.

The problem of finding the largest subset of P which occurs in the data set under translation has also been studied. A naïve $O(nm\log n)$ algorithm takes the differences between each point in the pattern and each in the data set, and sorts them (Wiggins et al., 2002); the algorithm can be reduced straightforwardly to $O(nm)$ time, by using hashed storage, but the bottleneck of space usage remains (Meredith et al., 2002b). By observing that the differences from any given point in the pattern are already in sorted order, the working space overhead was reduced to $O(m)$ by Ukkonen et al. (2003). Although the time complexity of this approach is $O(nm\log m)$ it is the only practical method for solving large problems of which we are aware. Lubiw and Tanur (2004) also consider the matching problem in sets of horizontal line segments. They use weight functions to measure the quality of a match rather than, e.g., the degree of overlap between line segments in pattern and text as in Ukkonen et al.'s (2003) problem P3. For example, Lubiw and Tanur's (2004) technique allows for a consonant interval between a pattern note and a text note to be matched more strongly than a dissonant one. They give an algorithm which runs in $O(nm\log m)$ time (n and m as before) and show that the problem is 3SUM-hard and so unlikely to have a subquadratic solution.

Other important related contributions are by Clausen and Kurth (2002), who present a general framework and implementations for searching for patterns in polyphonic symbolic and audio music data, and Typke et al. (2004), whose method searches for a polyphonic pattern in a database of polyphonic music represented in the form of weighted point sets, similarity between which is measured in terms of transportation distances such as the Earth Mover's Distance.

2.2. OMRAS

The OMRAS project (`http://www.omras.org`) designed a polyphonic MIR method (Pickens et al., 2003) to capture the general harmonic similarity between a musical query and musical documents sought in a collection. The simplest, 0th-order OMRAS model is built by estimating the relative strength of the members of a lexicon of chords, given the observable notes in a window traversing the music, and by consolidating the set of these partial observations (one for each window position) into an overall harmonic signature for the piece, which, like each of the partial observations, is a probability distribution over the lexical chords rather than a single value. This process is applied to all the files in the collection to be searched and the signature of each file is stored. Then, at search time, the signature of the query is computed along with a measure of the divergence between the query and each file in the database. A list of file names is then returned with the files ranked by decreasing order of divergence from the query.

The chord-lexicon used in most of the OMRAS work is the 24 major and minor triads. To admit approximate matching between harmonic descriptions, it was necessary to represent the harmonic unfolding of the music so that notes in a query different from those in the original document lend appropriate weight to a matching function according to their harmonic distance from it. There is no music-theoretical way to judge the distance between all possible chords arising in the course of a piece of music, but there is such a measure for the set of 24 triads: Krumhansl and Shepard (1979) present the relative positions of the triads in a four-dimensional space derived from rigorous empirical perceptual studies. The Euclidean distance between the triads in this space forms the basis for constructing the model.

As well as this simple harmonic model, OMRAS uses higher-order models to capture not just the local harmonic description of each window on the music, but also the harmonic transitions between windows. The model grows exponentially with order, which severely curtails efficiency, though it adds considerable richness to the model.

The OMRAS results reported in this paper use a parameter set that has been found to perform robustly: the data is gathered from MIDI files using a window of size 4 events and the OMRAS models are of the second order. We emphasise that this method essentially performs approximate matching for document-level retrieval, and was principally designed as a step towards bridging the gap between audio and symbolic representations of music (Bello and Pickens, 2005).

3. MSM

We now show that the problem of finding the largest subset match is 3SUM-hard. A problem is 3SUM-hard if it is at least as hard as the problem of determining if any three integers in a set sum to 0 (Gajentaan and Overmars, 1995, whose notation we use). A quadratic time lower bound is conjectured for the time complexity of this class of problems if exact solutions are required. We then present a randomised algorithm which takes $O(n \log n)$ time and so is able to beat the conjectured bound by giving approximately correct answers with high probability.

3.1. Largest subset match is 3SUM-hard

For two problems PR1 and PR2, $PR1 \lll_{f(n)} PR2$ means that any instance of PR1 can be solved using a constant number of instances of PR2 plus $O(f(n))$ time. A problem PR is 3SUM-hard if $3SUM \lll_{f(n)} PR$ where $f(n) \in o(n^2)$. Our proof is by reduction from EQDIST, a known 3SUM-hard problem (Barequet and Har-Peled, 2001).

Problem 1. EQDIST (EQUAL DISTANCE)[1]*: Given two sets A and B of n and $m = O(n)$ integers, respectively, is there a pair a_1, $a_2 \in A$ and a pair b_1, $b_2 \in B$ such that $a_1 - a_2 = b_1 - b_2$?*

[1] This problem definition has been adapted from the work of Barequet and Har-Peled (2001), since the original considered real values.

Problem 2. MAXSUBSET (MAXIMUM SUBSET MATCH)*: Consider two sets P and Q of m and $n = O(m)$ integers and an integer bound d. Consider also the largest subsets $P' \subseteq P$ and $Q' \subseteq Q$ such that for some vector v, $P' + v = Q'$. The* MAXSUBSET *problem is to determine whether $|P'| \geq d$.*

Theorem 1. EQDIST$\lll_{f(n)}$MAXSUBSET

Proof. Let (A, B) be an instance of EQDIST. We claim that (A, B) is also an instance of MAXSUBSET with the bound $d = 2$. Assume there is a solution to EQDIST. Therefore, there exist a_1, $a_2 \in A$ and b_1, $b_2 \in B$ such that $a_1 - a_2 = b_1 - b_2$. Let $v = b_1 - a_1$, so now we have $v + a_1 = b_1$ and $v + a_2 = b_2$ and therefore the maximum subset match of A and B is at least 2.

Now assume there is a solution to MAXSUBSET. It follows that there exists a vector v such that for two subsets $A' \subseteq A$ and $B' \subseteq B$, $A' + v = B'$ and $|A'| \geq 2$. Therefore, there exist two points $a_1, a_2 \in A'$ and two points $b_1, b_2 \in B'$ so that $a_1 + v = b_1$ and $a_2 + v = b_2$. Therefore $b_1 - b_2 = a_1 - a_2$ as required. □

EQDIST is known to be 3SUM-hard (Barequet and Har-Peled, 2001) and so it follows immediately that MAXSUBSET is also 3SUM-hard.

3.2. How to overcome the complexity limit

We now introduce an $O(n \log n)$ time randomised algorithm that calculates approximately the largest subset match. The idea is to reduce the problem of point set matching in n dimensions to *binary wildcard matching* in 1 dimension.

Definition 1. *Given k-dimensional point sets P and T, the* subset matching *problem is to find the largest subset of P which is a subset match in T.*

Further, we define the hash functions $g(x) = ax \bmod q$, $h(x) = g(x) \bmod s$ and $h2(x) = (g(x) + q) \bmod s$ for some preselected values of a, q and s.

The first two steps 'project and reduce length' (Cardoze and Schulman, 1998; Cole and Hariharan, 2002):

1. Random Projection in 1D To project P and T to one dimension, pick d integers b_i uniformly at random from a large space. For each point, x in $P \cup T$ calculate $\sum b_i x_i$, where x_i is the ith coordinate of x. Call the resulting sets of points in one dimension, P' and T' (Fig. 3(b)).

2. Length Reduction (by hashing) Use universal hashing to reduce the sparsity of the data. Choose q to be a random prime in $[2N, \ldots, 4N]$, where N is the maximum of the projected values of P and T; a uniformly from $[1, \ldots, q-1]$; and s to be kn, where $k > 1$ is a constant. Each non-zero location in P' is mapped to position $h(P'_i)$. Each non-zero location in T' is mapped to four positions: $h(T'_j)$, $h(T'_j) + s$, $h2(T'_j)$ and $h2(T'_j) + s$. Call the resultant arrays of length s

and $2s$ *resp.*, p and t (Fig. 3(c)). The values at the mapped positions in p and t are all set to 1.

3. Binary Wildcard Matching (using FFTs) Set any undefined position in p to 'wildcard'. Set any undefined position in t to 0. Perform binary wildcard matching on p and t. The result is the number of 1s that p and t have in common at each alignment position.

Lemma 1 justifies using these hash functions for matching. Its significance is that if some subset of the pattern P' matches in the text so that $P'_i + c = T'_j$ for some c then $h(P'_i) + h(c)$ matches one of $h(T'_j)$, $h(T'_j) + s$, $h2(T'_j)$ and $h2(T'_j) + s$.

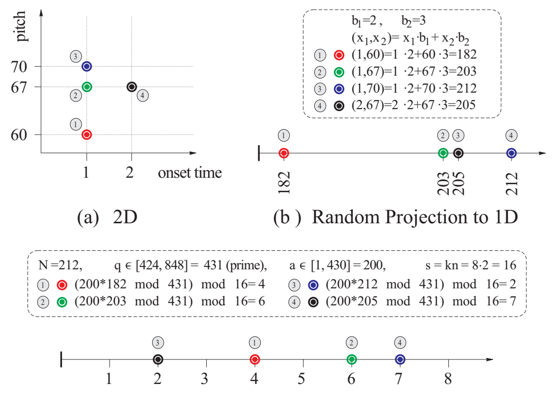
Lemma 1.

$$h(x) + h(y) = \begin{cases} h(x+y) & \text{if } g(x) + g(y) < q \text{ and} \\ & \quad h(x) + h(y) < s, \\ h(x+y) + s & \text{if } g(x) + g(y) < q \text{ and} \\ & \quad h(x) + h(y) \geq s, \\ h2(x+y) & \text{if } g(x) + g(y) \geq q \text{ and} \\ & \quad h(x) + h(y) < s \\ h2(x+y) + s & \text{otherwise.} \end{cases}$$

Proof. It is easy to show that $h(x) + h(y) = h(x+y)$ when $g(x) + g(y) < q$ and $h(x) + h(y) < s$. Now consider the second case, $g(x) + g(y) < q$ and $h(x) + h(y) \geq s$. Then

$$\begin{aligned} h(x) + h(y) &= g(x) \bmod s + g(y) \bmod s \\ &= (g(x) + g(y)) \bmod s + s \\ &= (ax \bmod q + ay \bmod q) \bmod s + s \\ &= (a(x+y) \bmod q) \bmod s + s \\ &= h(x+y) + s. \end{aligned}$$

We can follow similar steps to prove the other cases. $\square$


pitch
70
67
60
1
2
onset time
$b_1=2$, $b_2=3$
$(x_1,x_2)= x_1 \cdot b_1 + x_2 \cdot b_2$
(1,60)=1 ·2+60 ·3=182
(1,67)=1 ·2+67 ·3=203
(1,70)=1 ·2+70 ·3=212
(2,67)=2 ·2+67 ·3=205
182
203
205
212
N =212, q ∈ [424, 848] = 431 (prime), a ∈ [1, 430] = 200, s = kn = 8·2 = 16
(200*182 mod 431) mod 16= 4
(200*203 mod 431) mod 16= 6
(200*212 mod 431) mod 16= 2
(200*205 mod 431) mod 16= 7
1
2
3
4
5
6
7
8

(a) 2D (b) Random Projection to 1D

(c) Length Reduction (by hashing)

Figure 3: Illustration of the main steps of the procedure

Now we estimate the size of the subset matches. At each alignment of p and t we estimate the number of true matches in the original data. By making some simple assumptions, we derive an unbiased estimate of the number of true matches at a given offset. However, this estimate may have unacceptably large variance, so we make a further observation, giving us a biased estimate which is much more accurate in practice, for the type of musical data we analyse.

Our improvement works because we can, with some effort, find at which offset in the original 2-dimensional data the best match in the hashed and length reduced data appears. Having found it, we count the true number of matches at that point, thus entirely removing one source of error. The initial, possibly inaccurate, estimate of the number of matches at a given offset is not reported; only the order of the estimates is used. To achieve a perfect ranking of the documents we now only require that the initial estimates be monotonic in the true number of matches. Indeed, we need even less than this as only the largest subset match is considered for any given pair of pieces.

The relative probability of introducing false positives during the dimension reduction phase is small enough to use P' and T' as the original data from here on. We can add a further stage of lookup to find the true original points in P and T if need be. The outline algorithm runs thus:

1. Find the offset giving the best match of p and t and call it $h(c)$ (c is the currently unknown offset in P' and T').

2. Find which points in p match t at offset $h(c)$ ($O(m)$ time).

3. Look up where the points that match in the hashed pattern, p were mapped from. We must have stored a reverse map from p_i to P'_i when doing the original hashing.

4. We want to avoid calculating $h(c)$ for all c. We know P'_i for each match p_i, and that $t_j = h(P'_i + c)$; so we can look up T'_j in another reverse lookup table. Note that the four images (Lemma 1) created by hashing T' are distinct. Now we have $c = T'_j - P'_i$ for each pair of points that match in the hashed strings at a particular offset. If there are no collisions in the hashing step of P' and T' then we have solved the problem.

5. However, collisions are possible, so look at all p_j and make an array of candidate offsets c. Choose the offset that occurs most frequently as the estimate of the true offset.

The running time is dominated by the time taken to do the binary wildcard matching of p and t, which are both of length $s \in O(n)$. So the total running time is $O(n \log n)$.

4. Results and Evaluation

4.1. Evaluation Method

We compared **MSM** with the OMRAS system on a document-level retrieval task. OMRAS uses an approximate matching method, designed to work with noisy data (in both pitch and time dimensions), whereas **MSM** is an exact-matching method suitable for score-like representations; therefore the comparison is not exact, but we believe it convincingly shows the validity of our new approach.

To compare the two systems, we ran them on a test set of 2338 documents, of which 480 were used as queries. Each query was defined prior to the experiment to be relevant to 39 other queries (and itself), thus partitioning the

480 queries into 12 sets of 40 mutually relevant documents. Each of these sets was generated by artificially corrupting one of Bach's four-part chorale harmonizations by (1) inserting 10, 20, 30, 40 or 50% extra notes at random into the chorale (5 documents); (2) inserting 10, 20, 30, 40 or 50% extra notes so that each inserted note was a perfect fifth away from one of the notes in the original (5 documents); (3) transposing 10, 20, 30, 40 or 50% of the notes by a random interval smaller than an octave (5 documents); (4) transposing 10, 20, 30, 40 or 50% of the notes by a perfect fifth or a perfect fourth (5 documents); (5) deleting 10, 20, 30, 40 or 50% of the notes (5 documents); and (6) removing some combination of voices (15 documents).

Apart from these artificially corrupted documents, which test robustness, all the other documents in the database were derived automatically from the Musedata score encodings (Hewlett, 1997)[2]. The documents were normalised with respect to tactus; they were generated from score encodings, so they contained no expressive timing.

MSM and OMRAS were used to compute the similarity between each query document and each database document. The documents were then ranked by decreasing order of similarity to the query. The rankings of the relevant documents for each query were then combined together to produce an 11-point precision-recall curve for each algorithm.

In this task, the relevant documents for each query were corrupted versions of the query, as above. This models a real-world retrieval task of seeking possibly degraded versions of a symbolic query in a database of symbolic music encodings (i.e., scores). The noise introduced into the relevant documents here is typical of the pitch noise that might be found in MIDI files derived from audio recordings or performed input. However, timing information was not distorted at all, so robustness in this respect was not tested.

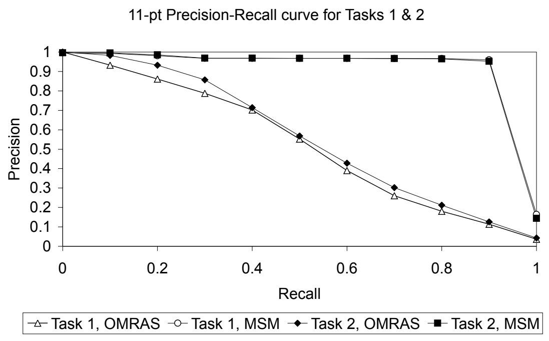
4.2. Test Results

Figure 4 shows the results of the tests with noisy queries, two tests per graph, comparing **MSM** results and OMRAS results for each test; in 4(a) and 4(b), the **MSM** results for each pair of tasks are almost identical. The overall results of the tests are shown in Figure 5. **MSM** consistently outperforms OMRAS on these tasks by a considerable margin.

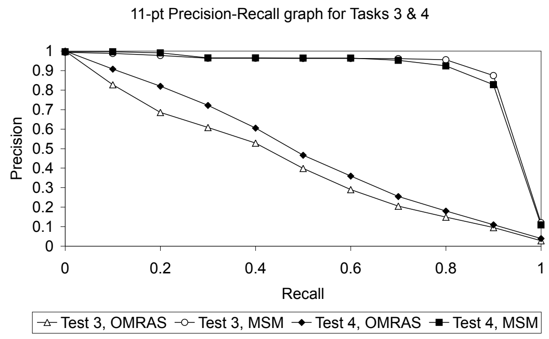
4.3. Execution speed of MSM

Figure 6 shows the execution times (y-axis) of Ukkonen et al.'s (2003) P3 algorithm and MSM, applied to prefixes of various sizes (x-axis) of Beethoven's 3rd Symphony, each prefix being compared with itself; this finds the maximal repeated subset of the prefix. **MSM**'s subquadratic behaviour is clearly visible, and the timings show that no large constants are hidden in our estimates. Further, they show that MSM is faster by between one and two orders of magnitude, improving with data size—the graph is plotted on log scale to allow the comparison to fit on the page. The code was in C++, compiled under g++ 3.4 on an AMD 3000+ machine using the fast Fourier library *fftw* 3.1.1 (Frigo and Johnson, 2005). No attempt at optimisation was made.

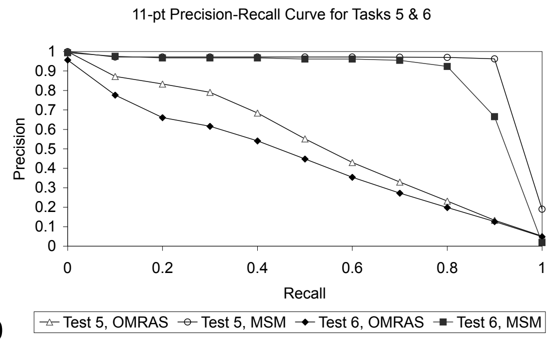
[2] `http://www.musedata.org`; thanks to CCARH for their use.



(a)



(b)



(c)

Figure 4: Precision/Recall graphs for the noisy queries; in (a) and (b), **MSM** results for each pair are almost identical.

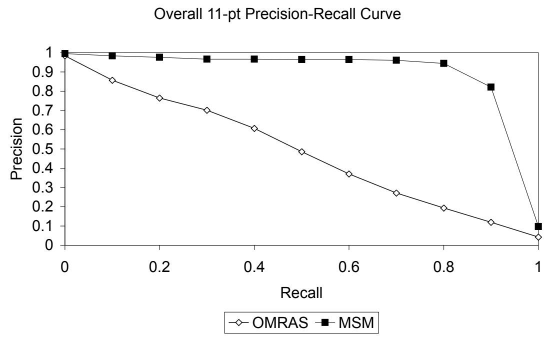


Figure 5: Overall precision/recall graph for all tests.

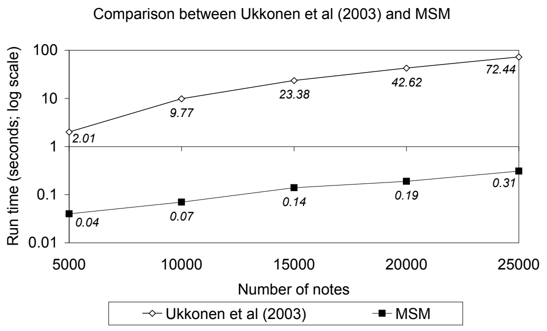


Figure 6: P3 *vs.* **MSM** log-scale CPU time used to detect maximal structure repetition in Beethoven's 3rd Symphony.

4.4. Discussion

MSM is much faster than SIA(M)E, particularly for large queries; the method is practicable where SIA(M)E was generally not. It outperforms Ukkonen et al.'s (2003) P3 algorithm, without loss in accuracy, though we note that this algorithm was intended for small queries. On well-behaved score data, its precision/recall results approach perfection, though we concede that this is a specialist task in which small-scale noise is not an issue, which is not usually the case in real-world data. In this special case, **MSM** significantly outperforms OMRAS.

5. Conclusions and Future Work

We believe the new method is promising, but there is more to do, in rendering the method robust to the kind of noise that is encountered in real-world music data. The technique is also applicable to line segment representations of music; this requires further empirical work. Most important, this algorithm is fast enough to admit non-indexed searching of large score databases for the first time.

References

G. Barequet and S. Har-Peled. Polygon containment and translational min-hausdorff-distance between segment sets are 3sum-hard. *Int. J. Comput. Geometry Appl.*, 11(4):465–474, 2001.

R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

J. P. Bello and J. Pickens. A robust mid-level representation for harmonic content in music signals. In *Proceedings of the 7th International Conference on Music Information Retrieval, ISMIR 2005*. Queen Mary College, University of London, 2005.

E. Cambouropoulos, M. Crochemore, C. Iliopoulos, M. Mohamed, and M.-F. Sagot. A pattern extraction algorithm for abstract melodic representations that allow partial overlapping of intervallic categories. In *Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR 2005, 11–15 September 2005, London)*, 2005.

D. E. Cardoze and L. J. Schulman. Pattern matching for spatial point sets. In *IEEE Symposium on Foundations of Computer Science*, pages 156–165, 1998.

M. Clausen and F. Kurth. A unified approach to content based and fault tolerant music identification. In *International Conference on Web Delivering of Music*, Darmstadt, Germany, 2002.

R. Cole and R. Hariharan. Verifying candidate matches in sparse and wildcard matching. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 592–601, 2002.

T. Crawford, C. S. Iliopoulos, and R. Raman. String matching techniques for musical similarity and melodic recognition. *Computing in Musicology*, 11:73–100, 1998.

M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. special issue on "Program Generation, Optimization, and Platform Adaptation".

A. Gajentaan and M. H. Overmars. On a class of $o(n^2)$ problems in computational geometry. *Computational Geometry*, 5:165–185, 1995.

A. Guo and H. Siegelmann. Time-warped longest common subsequence algorithm for music retrieval. In *Proceedings of the Fifth International Conference on Music Information Retrieval (ISMIR 2004, 10–15 October 2004, Barcelona)*, 2004.

W. B. Hewlett. *MuseData*: Multipurpose representation. In E. Selfridge-Field, editor, *Beyond MIDI: The Handbook of Musical Codes*, pages 402–447. MIT Press, Cambridge, MA., 1997.

C. L. Krumhansl and R. N. Shepard. Quantification of the hierarchy of tonal functions within a diatonic context. *Journal of Experimental Psychology: Human Perception and Performance*, 5:579–594, 1979.

K. Lemström. *String Matching Techniques for Music Retrieval*. PhD thesis, University of Helsinki, Faculty of Science, Department of Computer Science, 2000. Report A-2000-4.

A. Lubiw and L. Tanur. Pattern matching in polyphonic music as a weighted geometric translation problem. In *Fifth International Conference on Music Information Retrieval, Barcelona (ISMIR 2004, 10–15 October 2004)*, 2004.

D. Meredith, K. Lemström, and G. A. Wiggins. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, 31(4): 321–345, 2002a.

D. Meredith, G. A. Wiggins, and K. Lemström. Method of pattern discovery, 2002b. PCT patent application number PCT/GB02/02430, UK patent application number 0211914.7. Applied for by City University, London and filed on 23 May 2002. (Priority date: 23 May 2001, draft available online at `http://www.titanmusic.com/papers.html`).

J. Pickens, J. P. Bello, G. Monti, T. Crawford, M. Dovey, M. Sandler, and D. Byrd. Polyphonic score retrieval using polyphonic audio queries: A harmonic modeling approach. *Journal of New Music Research*, 32(2):223–226, 2003.

R. Typke, R. C. Veltkamp, and F. Wiering. Searching notated polyphonic music using transportation distances. In *Proceedings of the ACM Multimedia Conference (New York, October 2004)*, pages 128–135, 2004.

E. Ukkonen, K. Lemström, and V. Mäkinen. Geometric algorithms for transposition invariant content-based music retrieval. In *Proceedings of the Fourth International Conference on Music Information Retrieval, Baltimore (ISMIR 2003, 26–30 October 2003)*, 2003.

G. A. Wiggins, K. Lemström, and D. Meredith. SIA(M)ESE: An algorithm for transposition invariant, polyphonic, content-based music retrieval. In *3rd International Symposium on Music Information Retrieval (ISMIR 2002), 13–17 September 2002*, pages 283–284, IRCAM, Centre Pompidou, Paris, France., 2002.