BLAST FOR AUDIO SEQUENCES ALIGNMENT: A FAST SCALABLE COVER IDENTIFICATION TOOL

Benjamin Martin[1], **Daniel G. Brown**[2], **Pierre Hanna**[1], **Pascal Ferraro**[1]
[1]Université de Bordeaux
CNRS, LaBRI, UMR 5800
{bmartin,hanna,ferraro} @labri.fr
[2]University of Waterloo
Cheriton School of Computer Science
dan.brown@uwaterloo.ca

ABSTRACT

Searching for similarities in large musical databases is common for applications such as cover song identification. These methods typically use dynamic programming to align the shared musical motifs between subparts of two recordings. Such music local alignment methods are slow, as are the bioinformatics algorithms they are closely related to. We have adapted the ideas of the Basic Local Alignment Search Tool (BLAST) for biosequence alignment to the domain of aligning sequences of chroma features. Our tool allows local music sequence alignment in near-linear time. It identifies small regions of exact match between sequences, called seeds, and builds local alignments that include these seeds. Seed determination is a key issue for the accuracy of the method and closely depends on the database, the representation and the application. We introduce a particular seeding approach for cover detection, and evaluate it on both a 2000-piece training set and the million song dataset (*MSD*). We show that the heuristic alignment drastically improves time computation for cover song detection. Alignment sensitivity is still very high on the small database, but is dramatically weakened on the *MSD*, due to differences in chroma features. We discuss the impact of different choices of these features on alignment of musical pieces.

1. INTRODUCTION

During the last decade, an increasing number of large music datasets have become available. One may now access a huge amount of music audio, stored for instance on personal computers, mobile devices or online. In this context, Music Information Retrieval (MIR) focuses on automatic classification, organization, description of music content.

To assess musical similarities between pieces, for example, a major challenge of MIR is analysing acoustic content. Using signal processing techniques, music features are first infered from audio content. Each of them are related to a specific aspect of music. A fairly typical MIR approach consists in obtaining such a feature for short frames of audio signal, hence computing a symbolic string representation that carries the change in some musical property along a given music track. Such symbolic representations can be further analysed and compared to detect meaningful similarities. MIR studies typically employ comparison techniques either custom built or adapted from other fields of information science [6]. Many such techniques account for slight variations in musical features. The way we perceive music is known to work at different time scales, and allows for slight differences in music information received over time. For instance, one may easily recognize a chorus sung twice in a song although the singer may sing different lyrics, the melody may have changed, or the instruments playing may be different in both occurrences.

Among the many applications of automatic assessment of musical similarities in music datasets, cover song identification has been of major concern over the last few years [16]. Cover songs are usually defined as multiple renditions of the same original music piece. They may be played by other performers from different music genres or with distinct recording environments [16]. Although two cover versions of an original song may differ widely in instrumentation, singing voices, background noise, key, structural arrangement, genre, *etc.*, they should hold enough musical similarity for human perception to identify them as renditions of the same piece. To detect such similarities, most retrieval systems use dynamic programming [16]. A key drawback of such systems is their inability to efficiently scale to the range of musical data available in musical platforms, *i.e.* to the order of tens of millions of tracks [6, 16].

We propose an indexing method that substantially increases the efficiency of alignment-based retrieval systems. Our method uses a widely known bio-sequence indexing technique, BLAST [2]. We adapt this method by investigating the distribution of symbols among features sequences, and deducing a strategy for efficient indexing. An empirical study and an evaluation are performed on a custom-built cover song dataset, as well as on the Million Song Dataset (*MSD*) [5]. The remainder of this paper is organized as follows. Previous works are described in Section 2. Section 3.1 presents audio representations and alignment techniques used, and describes the principle of the bio-indexing tool we propose to adapt. Section 3 details the investigation over feature sequences, and describes the particular settings of our music application. Finally, Section 5 presents results obtained for a practical cover song identification, while concluding remarks and perspectives are depicted in Section 6.

2. RELATED WORK

Several heuristics for reducing the computational cost of dynamic programming have been introduced for MIR applications. Dannenberg and Hu [7] proposed to discover patterns in audio sequences by partially exploring the search space around bands, relying on global thresholds that indicate limits on the deviation from diagonals, implicitly assuming that insertions and deletions are rare. Combined to a clustering technique, this process was used to deduce suboptimal alignments and infer the structure of audio pieces.

Kilian and Hoos [12] already introduced BLAST in MIR context, in order to search for approximate patterns in symbolic music. They tested the technique on MIDI excerpts and emphasized the efficiency of the alignment of similar patterns with the bi-directional extension of exact match regions. Authors infered a high potential for BLAST in a non-symbolic input configuration, but did not test it explicitly [13]. Although the work presented in this paper adapts the same BLAST algorithm, it is subtantially different from this study. First, our technique is applied to feature sequences obtained from audio signals. More importantly, we aim at comparing distinct songs with possibly similar regions, not discovering similar patterns inside a single piece. Finally, the tool is used in our case as a filtering technique to allow efficient identification of cover songs.

Analogously, many studies adapt exact audio identification systems to account for musical variations (see [14] and references therein). For instance, Kurth and Muller [14] presented an efficient matching technique robust to typical variations in interpretation of classical music. The approach here is reversed relative to our: an exact fast fingerprinting system is adapted to account for local slight variations, whereas we propose to enhance the efficiency of an accurate slow approximate identification technique. They report an acceleration of the matching process by a factor of 15 to 20 while keeping a high robustness to interpretation variations. However, they emphasize that such a speed-up factor is suitable for efficiently handling datasets in the order of tens of thousands tracks [14]. To handle fast search in larger datasets, an indexing system for cover song identification on the *MSD* was recently proposed [4]. Landmarks estimations are performed from *MSD* chroma features, and heuristic jumpcodes between landmarks encode variations of pitch content along cover versions. Authors substantially reduce the problem to binary identification tasks, and report a fair effectiveness/efficiency tradeoff with a speed of about 200 seconds to query the *MSD*. The method we propose in this paper aims at reducing this cover song querying time to the order of a few seconds while keeping a good accuracy in a standard cover retrieval task.

3. COMPARING MUSIC SEQUENCES

3.1 Music representation

Pitch content plays an important role in the structure of audio pieces, in particular for Western music genres. Common compositional processes in such music are organized around melodic and harmonic sequences that listeners identify, consciously or not, as independent phrases or themes.

Pitch Class Profiles (PCP), also known as *chroma features*, are frequently used to describe these types of information. These features classify spectral energies into bins corresponding to the frequency class where they appear, each class taking into account the cyclical perception of pitch in human auditory system. The number of pitch classes p corresponds to the number of frequency bands considered in each octave. The parameter p is usually set to 12 to respect the common note scale, but higher values (generally multiples of 12) can improve the robustness to tuning issues [9]. Chroma features are usually considered as the most robust representation for cover song detection [16].

3.2 Music sequences alignment

This symbolic representation as a sequence of symbols or *string* can be used to define a similarity measure between two audio pieces. Such a metric is expected to isolate significantly similar sections, or repetitions, assessing their resemblance.

3.2.1 Relevance for music information

Repetitions in strings have been studied extensively, either for locating exact repeats or for identifying substrings that are duplicated within a certain tolerance. In the context of music sequences, musical similarity does not rely on exact matches since variations, such as transpositions, interpretation variations, rhythmic irregularities, background noise, may alter the representing sequences.

Alignment approaches are well suited for an accurate recognition of such variations. Widely used for biological sequences, such techniques have been extremely successful in identifying approximate repetitions between local patterns in DNA or RNA strings that reflect, for instance, gene homologies. The relevance of such methods for music information lies in the same "evolutionary" aspect of musical patterns that may slightly change along a single piece or accross similar pieces. Playing variations of some musical theme implies changing sound events such as notes, rhythms, or lyrics, which echoes the mutation of nucleotides of proteins during biological evolution. Therefore, alignment techniques are frequently used for identifying similar patterns in cover songs.

3.2.2 Alignment and dynamic programming

The first accurate distance measure for approximate string comparison in the context of biogical sequences is often credited to Needleman and Wunsch [15]. A string u of length n can be transformed into a string v of length m by applying *edit operations* on the symbols of u and v. These operations are insertions, deletions or substitutions, and each is assigned a cost. The *edit distance* between u and v is defined by the minimum total cost of edit operations required to transform u into v. The *global alignment* of u and v identifies the positions in the sequence u that are not changed during the process of transforming u to v, and their new position in that sequence. A variant of this comparison method, *local alignment* [18], allows finding and extracting a pair of regions, one from each string, which exhibit the highest similarity according to the scoring scheme assigned to edit operations. In a musical context, this might correspond to finding matches between

two verses of a song, or between smaller approximately duplicated harmonic figures. Its computation is typically performed by a dynamic programming algorithm filling a $(n+1)\times(m+1)$ matrix. The local alignment of u and v can be seen as the best scoring path in the dynamic programming matrix. This edit path is easily computed in practice by tracing back the series of operations performed. More information about alignment algorithms can be found in [10].

In the context of pitch content, an improvement of local alignment [1] allows taking into account a frequent variation of musical patterns in terms of harmony, namely local transposition. In Western popular music, for instance, local transposition happens when an occurrence of a structural pattern (*e.g.* chorus) is be played a few semitones higher than usual. The improvement of the alignment technique consists in adding a new edit operation, the local transposition of a string versus another. Consequently, we compute several matrices that estimate every possible local transposition, and allow a jump from one matrix to another by paying a corresponding transposition cost [1]. This variant yields more accurate alignments of pitch content, but it is much slower to compute in practice. If the alphabet of symbols has a symbols, the slowdown is a factor of a.

3.2.3 Complexity

Dynamic programming gives $\Theta(nm)$ running time for computing optimal alignment scores. Tracing back the effectively aligned substrings requires $\mathcal{O}(n+m)$. Therefore, to compare a new song of length n with a database of k pieces, each of average length m, requires $\Theta(knm)$ time. The naive space complexity is $\Theta(nm)$, where we store the entire dynamic programming matrix, although a simple trick allows reducing it to $\Theta(\max(m,n))$ by keeping only the last computed lines [10].

Local alignment techniques are particularly useful for the accurate identification of strong similarities between sequences [16]. However, the slowness of the dynamic programming makes them heavy to compute and unadapted to fast querying of large-scale datasets that comprise millions of sequences. Facing a similar challenge, bioinformatics researchers developed a fast heuristic-based search tool dedicated to efficient indexing for local alignment.

3.3 BLAST

The Basic Local Alignment Search Tool (BLAST) [2], reduces the computational cost of local alignment. BLAST relies on the observation that when querying a new sequence to a large database, there are likely only a small number of good alignments, so it filters the database to avoid computing irrelevant alignments of unrelated sequences. BLAST partially explores the dynamic programming search space to filter out many irrelevant comparisons before computing local alignments. It consists of several heuristic layers of rules for refining the potential regions of strong similarity, as described in the next sections.

3.3.1 Seeding the search space

The main heuristic of BLAST lies in the assumption that significant local alignments include small exact matches. As represented in Fig. 1-(i), the dashed edit path of the local alignment of u and v contains diagonal sections, that

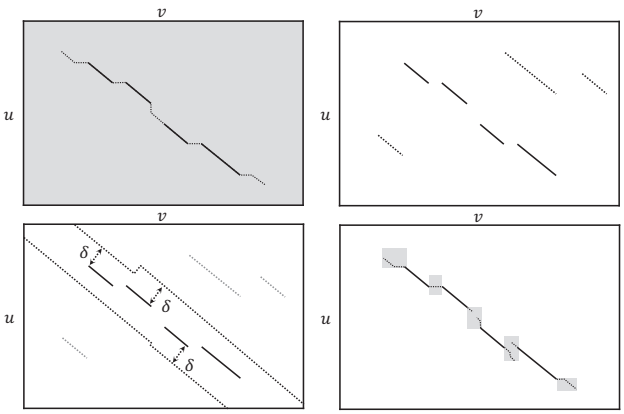

Figure 1. Seeding the search space. Top-Left: exact similar sections (plain lines) inside a local alignment path (dashed line). Top-Right: automatic seeding of the search space, where identified regions may belong to the local alignment path (plain lines) or not (dashed lines). Bottom-left: as result of $\mathcal{F}_1$, seeds are quickly clustered and isolated grey dashed seeds are eliminated. Bottom-right: as a result of $\mathcal{F}_2$, seed extensions roughly depict the local alignment path. Background grey parts highlight the number of dynamic programming computations required.

partially correspond to runs of matches (plain lines), *i.e.* exact repetitions between small sections of u and v. The first step of BLAST finds these small common substrings in order to seed the search space for later local alignments. A practical way of indexing the search space is to fix a seed length N, and index every N-length substring (or words) of every sequence of the dataset in a fast access data structure.

3.3.2 Filtering seeds

Once seeded as in Fig. $1-(ii)$, the search space includes hit regions that may correspond to high-scoring local alignment of sequences (plain segments) or to suboptimal regions (dashed segments), where the exact seed match arose due to coincidence, not true similarity. The second step of BLAST filters out most seeds that do not correspond to desired local alignments.

A first filtering technique $\mathcal{F}_1$ consists in quickly clustering seeds among the search space, and identifying isolated hits. As illustrated in Fig.1, every correct heuristic alignment should have several seeds around a diagonal (plain lines) that sketches the actual local alignment. Consequently, a pair of sequences that does not have regions comprising a significant number of hits may be filtered out. We use a threshold δ to stand for the maximum inter-diagonal distance allowed between two consecutive seeds to be considered as around the same diagonal, *i.e.* potentially belonging to the same alignment.

A more accurate, also common, filter $\mathcal{F}_2$ is seed extension. Each seed is extended in both directions to determine wether it corresponds to a local similar region or not. We denote by (i,k) and (j,l) the coordinates of a seed in the search space, *i.e.* the hit is between the exactly matching substrings $u[i\cdots j]$ and $v[k\cdots l]$. We compute two small alignments, one starting from (i,k) rolling up towards the top left corner of the search space, and the other one starting from (j,l) going towards the bottom right corner. Each

of these alignments, that may be gapped or ungapped, are quickly stopped if their score drops off under a threshold value X. This way, extending unrelated sequences quickly results in stopping the computation, while seeds from locally similar regions grow towards alignments [3].

4. INDEXING MUSIC DATA

4.1 Evaluation framework

4.1.1 Database

Two datasets were used to evaluate our system. The first set, *TSD*, consists of $2,514$ Western popular music pieces comprising 514 cover songs distributed in 17 cover classes, coming from personal music collections [1]. We elected as a second set the million song dataset (*MSD*) [5], that comprises one milliong songs of various musical genres [2]. It includes the Second Hand Songs dataset [3], which comprises $18,196$ cover songs distributed in $5,854$ cover classes. This set is to our knowledge the largest available set of cover song audio features.

Both datasets provide chroma feature sequences. However, it is worth noting that the implementation of such features significantly differs in both cases. In *TSD*, we used our own implementation of *Harmonic Pitch Class Profiles* [9] with a constant frame size and a resolution of 36 subdivisions per octave. This representation was successfully applied for past MIREX tasks [4]. In *MSD*, chroma features were computed using The EchoNest API [5], and consist of segment-synchronized 12-dimensional chroma features, as described in [11]. Thus, each *MSD* chroma represents pitch content on 12 dimensions for a variable audio frame (generally between 80 to 300ms with no overlap), whereas each *TSD* chroma represents pitch content on 36 dimensions for a constant audio frame (743ms with half overlap). The difference between these two approaches turns out to have major implications.

4.1.2 Alphabet definition

In its general definition, each B-dimensional chroma feature consists of B bins that may take any positive value, so their domain is infinite. However, as explained in Section 3.3, the heuristic alignment relies on the identification of exact similar regions in discrete sequences. Hence, it is critical for a BLAST approach to first project representing sequences onto a finite alphabet. A natural quantization is to detect the most probable chord, by looking for the highest scoring triad (root, major/minor third, fifth), and assigning a symbol corresponding to the index of this best triad. We represent each chroma feature over a 12-letter alphabet $\Delta = \{a, b, c, \ldots, k, l\}$ by the root of the predominant chord played. The chord mode (major/minor) is not taken into account since a root-exclusive representation seems to provide sufficiently meaningful audio representations (see Section 4.2). The 36-dimensional chroma features are also reduced down to 12 possible root chord symbols by keeping the multiple of 3 closest to the index of the highest triad (reduction to a 12 note scale with robustness to de-tuning).

[1] See http://www.labri.fr/perso/bmartin/ISMIR12 for the list
[2] http://labrosa.ee.columbia.edu/millionsong/
[3] http://www.secondhandsongs.com
[4] MHRAF submissions in Struct.Seg-11, Cover song-10
[5] http://the.echonest.com/

		TSD		*MSD*	
	Seed size	False negative	False positive	False negative	False positive
(i)	3	0.00	8.91	0.11	4.70
	4	0.03	3.69	4.28	1.15
	5	0.84	1.68	18.3	0.39
	6	4.50	0.82	37.2	0.16
	7	11.7	0.44	54.7	0.08
	8	21.8	0.25	68.6	0.04
(ii)	3	0.00	1.06	0.84	1.33
	4	0.21	0.41	7.08	0.23
	5	2.59	0.18	22.6	0.06
	6	6.67	0.08	41.8	0.02
	7	14.9	0.04	58.9	0.01
	8	26.0	0.02	72.1	.003

Table 1. Sensitivity/specificity tradeoff on *TSD* and *MSD*. Scores are given as percentages. In (i), all words are indexed in the dataset. In (ii), mono-symbolic words are not indexed.

This projection is likely to introduce inconsistencies in compared sequences, due to such a simplistic analysis. However, in practice the method only requires small sections of aligned sequences to be identical in order to assess their similarity, and is tolerant to sparse analysis errors to some extent.

4.1.3 Transposition invariance

Transposition is a very common variation among cover versions. Either globally on an entire rendition or locally accross structural parts, transpositions have to be taken into account in similarity analysis [17]. As highlighted in [14], the indexing strategy should hash harmonic progressions instead of absolute pitch content. While looking for exact matches between two sequences, compared regions are thus transposed down to a common key. For instance, the sequences `bcbbfd` and `deddhd` are in fact an exact match since they describe the same chord variations regardless of their local key, which differ by a major second. Practically, this can be seen as translating all words such that they start with the symbol `a`.

4.2 Seed determination

The heuristic alignment strongly relies on the selection of meaningful parts. A key issue for the method is to determine a seed both *sensitive* enough to correctly index alignments between cover songs, and *specific* enough to index as few spurious hits as possible. The first parameter for optimizing the sensitivity/specificity tradeoff is the length of the seed. In the following, given a seed length N, we denote by Δ^N the set of all possible words of length N over Δ.

4.2.1 Sensitivity evaluation

Assessing the sensitivity of a seed can be done by analyzing the alignments of similar sequences. Let N be a seed length. We must determine how many alignments actually contain seeds of size N, *i.e.* how many alignments contain at least one run of N matches. Thus, we first computed on both *TSD* and *MSD* local alignments between cover songs using the local transposition variant described in 3.2.2. By tracing back the alignments, we were able to identify exact runs of matches. An alignment is considered validated if

TSD		*MSD*	
Word	%	Word	%
aaaaaaa	6.36	aaaaaaa	14.1
ahhhhhh	0.55	aaaaaah	1.45
aaaaaah	0.53	aaaaaaf	1.31
afffffff	0.49	afffffff	1.13
aaaaaaf	0.46	ahhhhhh	0.98

Table 2. The five most probable words in *TSD* and *MSD* and their frequency of occurrence (in % of the words in each database) for $N = 7$ and a 12-letter alphabet.

it can be indexed, *i.e.* if at least one long enough run is found. The second and fourth columns of Tab. 1-(i) show the probability Pr[false negative] that an alignment can not be indexed by the method, as a function of the seed length, on both of the datasets.

4.2.2 Specificity evaluation

To evaluate the specificity of a seed, we need to estimate the probability of finding two identical runs in unrelated audio sequences. Practically, for a given word w over Δ^N, we count the number of occurrences of w in the database of unrelated sequences (no cover songs). This computation is stored in a list L and repeated for each possible word. In the end, $L[j]$ contains the number of instances of a particular word, and $\sum_i L[i]$ is the total number of N-long words in the dataset. The probability of finding *one* word w in a random chunk of the database is given by $\frac{L[j]}{\sum_i L[i]}$, where j is the index in L corresponding to w. Subsequently, the overall probability of finding *two* identical words in the database is given by Pr[false positive] $= \frac{1}{(\sum_i L[i])^2} \sum_j L[j]^2$. The third and fifth columns of Tab. 1-(i) provides the Pr[false positive] computed for each seed length and for both of the datasets, as percentages.

4.2.3 Word distribution

Table 2 shows the five most probable words in both datasets for a fixed seed length of 7. The most probable bin is the mono-symbolic word. Following bins correspond to frequent intervals in tonal music: up-by-fifth (a→h) or down-by-fifth (a→f). In both of the datasets, mono-symbolic words occur far more often than other words, representing 6.36% and 14.1% of the words in *TSD* and *MSD*, respectively. Thus, mono-symbolic words are likely to be responsible for many false positive hits while not capturing very sensitive regions in true alignments. We hence re-evaluated in Tab. 1-(ii) the sensitivity and specificity tradeoff on both datasets without indexing mono-symbolic words. As a result, for a seed length of 7 symbols, specificity is increased by a factor between 8 and 10, while sensitivity is reasonably decreased by around 3% in both datasets.

5. RESULTS AND DISCUSSION

Statistical results emphasize a significant difference between sequences in both datasets. First, cover song alignments share much fewer words in *MSD* than in *TSD*. For instance, for a seed size of 7, only 41.1% of the cover song alignments in *MSD* share common multi-symbolic words, as compared to 85.1% in *TSD*. Subsequently, the characterization of cover songs in *MSD* should be more difficult than in *TSD*. Moreover, false negative rates suggest that the distributions of words between both datasets are different, hence it would not be relevant to put them in common for evaluation purpose.

To test the relevance of our system in comparison with alignment methods, we implemented the alignment techniques described in 3.2.2. Since indexing is computed in a transposition invariant manner, we tested local alignments with the accurate local transposition variant [1]. We evaluated the identification technique using the Mean of Average Precision (MAP), the standard metric for cover song retrieval evaluation [8, 16, 17].

From every cover class in *TSD*, we computed alignments of each member to the rest of the class and to confusing songs. Average MAP values are presented in Tab. 3-(i), and detailed among 8 cover classes of *TSD* in Fig. 2. We experimented with the same approach on *MSD*, more precisely on a subset that made the alignment computation practicable. This subset MSD_{2k} was formed by randomly choosing 30 cover song classes and 2000 confusing songs from *MSD*. We discovered that the identification method did not extend, as indicated by the MAP scores (Tab. 3-(ii)). We identify two possible reasons: 1) Cover songs are particularly different from each other in *MSD*; 2) *MSD* chroma features are not as suitable as *TSD* features for sequence alignment. To isolate the second possibility, we recomputed *TSD* with the audio features used in *MSD*, via the EchoNest API[6]. We repeated the same cover identification experiment on this new dataset $\widetilde{TSD}$ and obtained MAP scores indicated in Tab. 3-(iii). The significant drop in MAP scores between *TSD* and $\widetilde{TSD}$ evaluations suggest that *MSD* chroma features do not make for alignable sequences. This result, already infered in [4], is substantiated by the high false negative rate found in *MSD* sequences (Tab. 1). We think this is due to critical implementation differences in chroma features between both datasets such as chroma dimension, temporal filtering and segment synchronism, which may not be adapted to standard alignment techniques, as highlighted in [16] for instance.

We implemented the indexing strategy described in 4 on *TSD* and *MSD* sequences. Logically, previous results made *MSD* heuristic alignments ineffective. *MSD* is to be considered here only as a computation performance indicator. From Tab. 1, we chose the seed length $N = 7$, that features a reasonable false negative rate of 11.7%, eliminating the most dissimilar cover songs, for a low false positive rate of 0.44% that guarantees high performance with few spurious hits. For each dataset, we built a table hashing every N words in sequences and storing their positions. Then, for each query, all matching songs were filtered using either $\mathcal{F}_1$ or $\mathcal{F}_1$ and then $\mathcal{F}_2$. Resulting MAP scores are given in 3-(iv) and (v). Highly depending on the dataset, our scores are not intended to be compared to state-of-the-art results. Their relevance lies in the comparison between basic method and heuristic alignments. As shown in Fig. 2, BLAST filters seem to slightly decrease the accuracy of the cover identification. Combining both filters seems effective for quickly identifying most covers. Indeed, the overall accuracy of BLAST method on *TSD* reaches a MAP score of 30.11%, corresponding to a loss of 14.71% as compared to an accurate sequence alignment.

[6] http://developer.echonest.com/

	Method	Dataset	MAP (%)	Runtime (s/query)
(i)	Alignment	*TSD*	44.82	129
(ii)		MSD_{2k}	5.71	388
(iii)		$\widetilde{TSD}$	7.20	273
(iv)		*MSD*	-	193,765
(v)	BLAST-$\{\mathcal{F}_1\}$	*TSD*	18.06	0.24
(vi)		*MSD*	-	12.20
(vii)	BLAST-$\{\mathcal{F}_1, \mathcal{F}_2\}$	*TSD*	30.11	0.33
(viii)		*MSD*	-	16.9

Table 3. MAP results and computing times for cover identification on *TSD* and *MSD*.

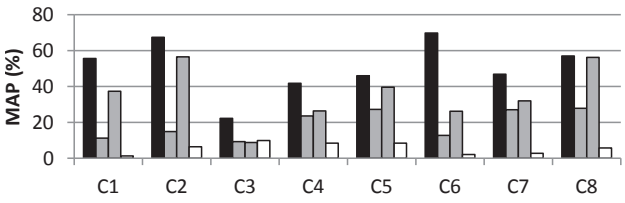

Figure 2. MAP scores obtained on 8 cover classes of *TSD*. Black: alignment, grey: BLAST-$\{\mathcal{F}_1\}$, BLAST-$\{\mathcal{F}_1, \mathcal{F}_2\}$, white: alignment with *MSD* chroma features ($\widetilde{TSD}$).

5.1 Computational efficiency

Due to the very high number of entries in the hash table, we implemented an efficient memory key/value lookup system in C. Building the index from chord sequences required about 16 minutes for the whole *MSD* on our server [7]. The average querying runtimes for each approach are given in Tab. 3. Note that we did not use parallel computing in this study. As expected, alignments imply slow computation, resulting in about 129 seconds per query on *TSD* and 388 seconds per query on MSD_{2k}. By counting the number of symbols in the whole *MSD* dataset, we infer that approximately 53 hours would be required to compute alignments. Note that removing the local transposition variant would speed-up by a factor of 12, still inadequate for practical computation. Thanks to BLAST heuristics, computation is drastically improved with around 12.2 seconds per query on *MSD* (0.24 seconds on *TSD*) with the coarse filter $\mathcal{F}_1$, and 16.9 seconds per query with both filters on *MSD* (0.33 seconds on *TSD*), on average.

6. CONCLUSION

We presented a new method for practical cover identification on large scale datasets. Inspired by bioinformatics heuristics, we applied BLAST to audio features and investigated the distribution of music sequences. Results obtained on our dataset suggest a reasonable loss of accuracy of the retrieval system in exchange for a substantial gain in computing time: estimating cover songs of a 3 minutes feature sequence can be acheived in less than 15 seconds in a million song database. We see this outcome as a significant step towards the practical search for approximate patterns in large datasets. Another main result of our study, although quite unexpected, is the apparent limitation of *MSD* chroma features regarding sequence alignment. Future studies on the *MSD* involving alignment techniques should investigate further the distribution of chroma sequences, and maybe combine them to other data (*e.g.* loudness, timbre). Future work will be particularly focused on enhancing the sensitivity of the identification of cover songs, considering for instance spaced or variable length seeds for BLAST.

7. REFERENCES

[1] J. Allali, P. Ferraro, P. Hanna, and C. Iliopoulos. Local transpositions in alignment of polyphonic musical sequences. In *String Processing and Information Retrieval*, volume 4726, pages 26–38. Springer Berlin / Heidelberg, 2007.

[7] Hardware: Intel Xeon X5675 @ 3.07GHz, 12M cache, 32GB RAM

[2] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, D.J. Lipman, et al. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[3] S.F. Altschul, T.L. Madden, A.A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[4] T. Bertin-Mahieux and D.P.W. Ellis. Large-scale cover song recognition using hashed chroma landmarks. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 117–120, 2011.

[5] T. Bertin-Mahieux, D.P.W. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proc. of the 12th International Conference on Music Information Retrieval*, 2011.

[6] M.A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. Content-based music information retrieval: Current directions and future challenges. *Proc. of the IEEE*, 96(4):668–696, 2008.

[7] R.B. Dannenberg and N. Hu. Pattern discovery techniques for music audio. In *Proc. of the 3rd International Conference on Music Information Retrieval*, pages 63–70, 2002.

[8] J.S. Downie, M. Bay, A.F. Ehmann, and M.C. Jones. Audio cover song identification: Mirex 2006-2007 results and analyses. In *Int. Symp. on Music Information Retrieval (ISMIR)*, pages 468–473, 2008.

[9] E. Gómez. *Tonal Description of Music Audio Signals*. PhD thesis, Universitat Pompeu Fabra, pages 63–100, 2006.

[10] D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, pages 215–253, 1997.

[11] T. Jehan. *Creating Music by Listening*. PhD thesis, Massachusetts Institute of Technology, pages 57–59, 2005.

[12] J. Kilian and H.H. Hoos. Musicblast - gapped sequence alignment for mir. In *Proc. of the 5th International Conference on Music Information Retrieval*, pages 38–41, 2004.

[13] J.F. Kilian. *Inferring Score Level Musical Information From Low-Level Musical Data*. PhD thesis, Darmstadt University of Technology, pages 54–60, 2004.

[14] F. Kurth and M. Müller. Efficient index-based audio matching. *IEEE Trans. on Audio, Speech, and Language Processing*, 16(2):382–395, 2008.

[15] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[16] J. Serrà, E. Gómez, and P. Herrera. *Audio cover song identification and similarity: background, approaches, evaluation, and beyond*, volume 274 of *Studies in Computational Intelligence*, chapter 14, pages 307–332. 2010.

[17] J. Serrà, E. Gómez, P. Herrera, and X. Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Trans. on Audio, Speech and Language Processing*, 16:1138–1151, 2008.

[18] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.