UNSUPERVISED DETECTION OF COVER SONG SETS: ACCURACY IMPROVEMENT AND ORIGINAL IDENTIFICATION

Joan Serrà[†], Massimiliano Zanin[‡], Cyril Laurier[†], and Mohamed Sordo[†]
[†] Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain.
[‡] Universidad Autónoma de Madrid, Madrid, Spain.
joan.serraj@upf.edu, massimiliano.zanin@hotmail.com, {cyril.laurier,mohamed.sordo}@upf.edu

ABSTRACT

The task of identifying cover songs has formerly been studied in terms of a prototypical query retrieval framework. However, this framework is not the only one the task allows. In this article, we revise the task of identifying cover songs to include the notion of sets (or groups) of covers. In particular, we study the application of unsupervised clustering and community detection algorithms to detect cover sets. We consider current state-of-the-art algorithms and propose new methods to achieve this goal. Our experiments show that the detection of cover sets is feasible, that it can be performed in a reasonable amount of time, that it does not require extensive parameter tuning, and that it presents certain robustness to inaccurate measurements. Furthermore, we highlight two direct outcomes that naturally arise from the proposed framework revision: increasing the accuracy of query retrieval-based systems and detecting the original song within a set of covers.

1. INTRODUCTION

Cover song identification has been a very active area of study within the music information research (MIR) community over the last years [1]. Traditionally, cover song identification has been set up as a typical information retrieval (IR) task where queries are processed in a batch mode [2] (p. 74): the user submits a query (a song) and receives an answer back (a list of songs ranked by their relevance to the query). Such a setup has conditioned the way of implementing and evaluating cover song identification systems [1, 3].

Here we take a new qualitative approach by considering cover song sets instead of isolated cover song queries. More concretely, we automatically identify sets (or groups) of covers of the same underlying musical piece in a music collection. We do so by utilizing grouping algorithms on top of an existing cover song identification system. We focus on unsupervised clustering [4, 5] and community detection [6, 7] algorithms.

The usage of grouping algorithms can be intuitively justified from multiple perspectives. First, grouping algorithms are a natural choice given the output of current cover song identification systems. Because this usually consists in a set of pairwise distances [1], we can directly assume that the preliminary issues of a typical pattern grouping task [5], namely feature extraction and distance measurement, are appropriately dealt with. Second, grouping algorithms may help in obtaining more coherent answers for isolated queries. In particular, the answers to any query song belonging to a given cover set would coherently contain the other songs in the set (notice that this property is not ensured by the distance measurements nor by batch-mode query systems). Third, grouping algorithms may profit from second order cover song associations. For instance, if cover song pairs s_i, s_j and s_j, s_k are independently detected, the cover song relation between s_i and s_k could automatically be inferred. This way, the system would take advantage of these collaborative effects and, among other things, increase the overall accuracy. Fourth, grouping algorithms can provide insightful clues to the study of inter and intra-group relations (e.g. by using hierarchical clustering algorithms [4, 5]).

The Music Information Retrieval Evaluation eXchange (MIREX) provides a batch-mode query framework for evaluating cover song identification systems [2]. Nonetheless, some participants have started moving towards the cover set detection framework. This framework has been indirectly and scantily introduced in [8] and, simultaneously, in our previous work in [9]. In [8], a multidimensional scaling analysis was performed on the basis that the configuration of the music collection under study was known (i.e. the number of cover sets and their cardinality was a priori defined, and the latter was assumed to be constant for all sets). This analysis was shown to substantially increase the final system's identification accuracy. A comparable increase was also achieved by the post-processing step mentioned in [9], whose details we now disclose.

Below, we first present the grouping algorithms that we use for detecting cover song sets (Sec. 2). We then summarize the followed methodology (Sec. 3) and subsequently present the obtained results (Sec. 4). We also show two

[1] In this article we pragmatically use the term *distance* to refer to any similarity or dissimilarity measure between cover songs.

[2] `http://www.music-ir.org/mirex/2008/index.php/Audio_Cover_Song_Identification`

natural outcomes of detecting cover sets (Sec. 5): increasing the accuracy of batch-mode query systems (Sec. 5.1) and detecting the original song (Sec. 5.2). We finally close the article with a conclusions section (Sec. 6).

2. STUDIED ALGORITHMS

2.1 K-medoids

The K-medoids algorithm is a common choice when the computation of means is unavailable, as it solely operates on pairwise distances and can exhibit some advantages compared to the standard K-means algorithm [4]. We employ the K-medoids implementation of the TAMO [3] package, which incorporates several heuristics [4] to achieve an optimal K value [4]. We use the default parameters and try all possible heuristics provided in the aforementioned implementation.

2.2 Hierarchical clustering

We also consider four representative agglomerative hierarchical clustering methods [4, 5]: single linkage, complete linkage, group average linkage (UPGMA), and weighted average linkage (WPGMA). We use the hcluster [5] implementation with the default parameters, and we try different cluster validity criteria [4] such as [6] checking descendants for inconsistent values, or considering the maximal or the average inter-cluster cophenetic distance.

2.3 Proposed method 1

The present and subsequent methods perform community detection [6, 7] on a complex network [10]. A weighted complex network [11] can be easily built from pairwise distances. Given a music collection $S=\{s_i\}, i=1,...,N_S$, with N_S songs, we query a cover song identification system for each song and obtain a set of answers $A=\{A_i\}$, where A_i contains N_{A_i} tuples $\{s_j, d(s_i,s_j)\}$, $s_j \in S$, ranked from low to high according to the provided distance measure d. In our case, N_{A_i} can be different for each A_i and it can be significantly lower than N_S. As we do not expect cover songs to have high distances or ranks in A_i, we determine N_{A_i} by applying a distance and a rank threshold d_{Th} and r_{Th}, respectively. From A, we construct a graph G with N_S vertices ($V=\{v_i\}$, $V \leftrightarrow S$) and N_{A_i} weighted edges for each vertex (an edge with a weight $w_{i,j}=\frac{1}{d(s_i,s_j)+\epsilon}$, ϵ being an arbitrary small constant, e.g. $\epsilon=0.01$, is assigned between vertices v_i and v_j if $s_j \subset A_i$ or $s_i \subset A_j$).

The method performs community detection by just looking at connected vertices in G in such a way that all connected vertices are assigned to the same community. Therefore, this algorithm strongly relies on d_{Th} and/or r_{Th} parameters. This approach presents some analogies with the common nearest neighbor clustering approach [5].

[3] http://fraenkel.mit.edu/TAMO

[4] http://fraenkel.mit.edu/TAMO/documentation/TAMO.Clustering.Kmedoids.html

[5] http://code.google.com/p/scipy-cluster

[6] http://www.soe.ucsc.edu/~eads/cluster.html

2.4 Proposed method 2

The aforementioned approach could be further improved by reinforcing triangular connections in the complex network before the last step of checking for connected vertices. Following the intuitive reasonings advanced in Sec. 1, the algorithm proposed here tries to reduce the "uncertainty" generated by triplets of vertices connected by two edges. In other words, it tries to reinforce coherence in a triangular sense.

If v_i, v_j and v_j, v_k are respectively connected, we can induce more coherence by either creating a connection between v_i and v_k (i.e. forcing the existence of a triangle), or by deleting one of the existing edges. This coherence can be measured through an objective function f_{O} which considers complete and incomplete triangles in the whole graph G. We define f_{O} as a weighted difference between the number of complete triangles N_Θ and the number of incomplete triangles N_Φ (three vertices connected by only two links) that can be computed from a pair of vertices: $f_{\text{O}}(N_\Theta, N_\Phi)=N_\Theta-\alpha N_\Phi$. The constant α, which weights the penalization for having incomplete triangles in G, is set experimentally.

The method starts by building a complex network G as described in the previous section. Then, for each pair of vertices v_i, v_j, the value of f_{O} is calculated for two situations: (i) when an edge between v_i and v_j is artificially created and (ii) when such an edge is deleted. Then, the option which maximizes f_{O} is preserved and the adjacency matrix of G is updated as necessary. The process of assigning cover sets is again the connected vertices criterion.

2.5 Proposed method 3

We can substantially reduce the computation time of proposed method 2 by considering for the computation of f_{O} only those vertices whose connections seem to be uncertain. If the distance between two songs is extremely high or low, this means that the cover song detection system has clearly detected a match or a mismatch. Accordingly, we just consider the pairs of vertices whose edge weight is around $w_{\text{Th}}=\frac{1}{d_{\text{Th}}+\epsilon}$. In particular, for each vertex v_i, a pre-selection of adjacent vertices is performed according to a certain weight margin w_{Ma}, which we set manually. This way, v_j is associated to v_i for further processing iff $|w_{i,j}-w_{\text{Th}}|<w_{\text{Ma}}$, where $|\cdot|$ indicates absolute value. The process of building the initial complex network and assigning cover sets is the same as in Sec. 2.3.

2.6 Modularity optimization

We also consider the method in [12], which extracts the community structure from large networks based on the optimization of the network modularity [6, 7, 12]. This algorithm outperforms all other known community detection methods in terms of computational time while maintaining a high accuracy [12]. We use the implementation by Aynaud [7] and we build the initial network as in Sec. 2.3.

[7] http://perso.crans.org/~aynaud/communities/index.html

3. METHODOLOGY

3.1 Data

For the generality of our experiments, we employ two sources of data: artificial and real-world symmetric distance matrices. Artificial distances are randomly generated given a predefined noise level σ_ξ. More concretely, the distance $d(s_i, s_j) = d(s_j, s_i)$ between songs s_i and s_j is drawn from a normal distribution $\mathcal{N}(\mu, \sigma)$ with mean μ and standard deviation σ such that

$$d(s_i, s_j) = \begin{cases} 0 & \text{if } i = j, \\ |\mathcal{N}(0, \sigma_\xi)| & \text{if } i > j \text{ and } s_i, s_j \text{ covers}, \\ 1 - |\mathcal{N}(0, \sigma_\xi)| & \text{otherwise.} \end{cases} \quad (1)$$

Real-world distances are provided by a state-of-the-art cover song identification system [13].

3.2 Experimental setups

A cover song music collection can be characterized by certain parameters constituting a setup [1]: the total number of songs N_S, the number of cover sets N_C the collection includes, the cardinality C of these cover sets, and the number of added noise songs [8] N_N. Because some setups can lead to wrong accuracy estimations [1], it is safer to consider several of them, including fixed and variable cardinalities. In our experiments we use the setups summarized in Table 1. For real-world data we use an extension of the music collection described in [13] which comprises a variety of genres and styles, as well as different types of covers. The characteristics of this music collection correspond to setup 3. For other setups we simply sample cover sets from setup 3 and repeat the experiments N_T times (number of trials, average accuracies reported). We either sample cover sets with a fixed cardinality ($C = 4$, the expected cardinality of setup 3) or without fixing it (variable cardinality, $C = \nu$). When using artificial data, we construct the distance matrix following Eq. (1). As fixed cardinality we use $C = 4$ as well, and as variable cardinality we take $\nu \sim \lfloor 2 + e(1/2) \rfloor$, where $\lfloor \cdot \rfloor$ denotes floor value and $e(1/\lambda)$ corresponds to an exponential distribution [9] with rate parameter λ.

3.3 Evaluation measures

To evaluate batch-mode query systems we employ the mean of average precisions (MAP) over all queries [3, 14]. The average precision for a query s_i (AP_i) is calculated from the retrieved answer A_i as $\text{AP}_i = \frac{1}{C-1} \sum_{r=1}^{N_S-1} P_i(r) I_i(r)$, where P_i is the precision of the sorted list A_i at rank r, $P_i(r) = \frac{1}{r} \sum_{l=1}^{r} I_i(l)$, and I_i is a relevance function such that $I_i(z) = 1$ if the song with rank z in A_i is a cover of s_i, $I_i(z) = 0$ otherwise.

[8] By *noise songs* we mean songs that do not belong to any cover set included in the collection.

[9] An exponential distribution is used to imitate the distribution of C with setup 3 (see [13]). With $\lambda = 2$, an expected value $\langle \nu \rangle = 4$ is obtained.

Setup	Parameters				
	N_C	C	N_N	N_S	N_T
1.1	25	4	0	100	20
1.2	25	ν	0	$\langle 100 \rangle$	20
1.3	25	4	100	200	20
1.4	25	ν	100	$\langle 200 \rangle$	20
2.1	125	4	0	500	20
2.2	125	ν	0	$\langle 500 \rangle$	20
2.3	125	4	400	900	20
2.4	125	ν	400	$\langle 900 \rangle$	20
3	525	ν	0	2135	1

Table 1. Experimental setup summary. The $\langle \cdot \rangle$ delimiters denote expected value.

To evaluate cover set detection we resort to the classical F-measure with even weighting [2, 14]: $F = \frac{2PR}{P+R}$. Here P and R correspond to precision and recall, respectively. For our evaluation, we compute these quantities independently for all songs and average afterwards. More concretely, for each song s_i, we count the number of true positives TP_i (i.e. the number of cover songs of s_i belonging to the the same group [10] as s_i), the number of false positives FP_i (i.e. the number of songs belonging to the same group as s_i that are not covers of s_i), the number of false negatives FN_i (i.e. the number of cover songs of s_i that do not belong to the same group as s_i), and average $P_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}$ and $R_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i}$ across all N_S songs [11].

4. RESULTS

4.1 Artificial data

We first evaluate the algorithms' accuracy as a function of the noise level σ_ξ introduced to the artificial data for setups 1.1 to 1.4. Before computing the reported accuracies, we independently performed a parameter optimization for each algorithm [12], σ_ξ, and setup. Then, using the optimal parameters found for a given σ_ξ, we plot average F over 20 trials versus σ_ξ (Fig. 1). In general, we observe that the accuracy for all algorithms starts to drop for $\sigma_\xi > 0.2$ until it reaches an $F < 0.3$ for $\sigma_\xi > 0.5$. We also see that the K-medoids and single-linkage algorithms are less robust to noise than the others. Low accuracies for the K-medoids algorithm might be explained by the difficulty to automatically set the correct K value. Furthermore, cover sets with variable cardinality, such as the ones used for setups 1.2 and 1.4, might further decrease the accuracy [4]. Low accuracies for the single linkage algorithm with respect to other hierarchical clustering algorithms confirm the findings in the literature [5]. UPGMA, and WPGMA accuracies are slightly higher than other algorithms under noise levels $\sigma_\xi \in [0.2, 0.4]$.

[10] Through this subsection by *group* we mean the cover set detected by the evaluated algorithm.

[11] Note that, unlike other clustering evaluation measures [15], F is not computed on a per-cluster basis, but on a per-song basis.

[12] This parameter optimization was not critical to achieve near-optimal accuracies (see Sec. 4.2). We did not consider proposed method 2 at this stage due to its computational complexity (see Sec. 4.3).

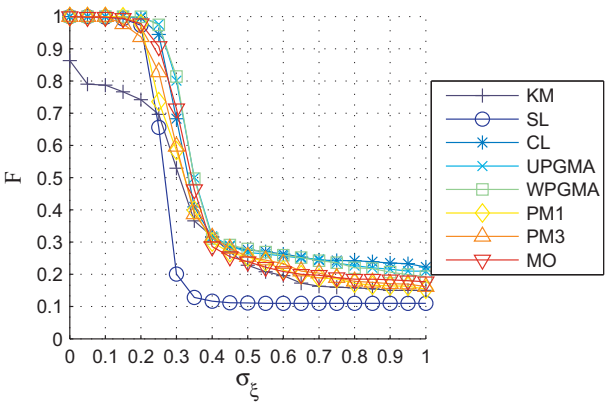

Figure 1. Accuracy under different noise levels for setup 1.4. Other setups lead to similar curves. Here and in subsequent tables and figures KM stands for K-medoids, SL for single-linkage, CL for complete-linkage, UPGMA for group average linkage, and WPGMA for weighted average linkage, PM for proposed method, and MO for modularity optimization.

Algorithm	Setup			
	2.1	2.2	2.3	2.4
KM	0.517	0.497	0.520	0.500
SL	0.656	0.690	0.654	0.683
CL	0.816	0.820	0.814	0.821
UPGMA	**0.895**	**0.900**	**0.897**	**0.902**
WPGMA	0.879	0.889	0.883	0.893
PM1	0.713	0.723	0.716	0.718
PM3	0.665	0.698	0.668	0.704
MO	0.721	0.735	0.716	0.744

Table 2. Accuracy (F) for artificial data with $\sigma_\xi = 0.25$.

To better assess the algorithms' performance, we show the accuracies achieved with setups 2.1 to 2.4 under a fixed noise level of $\sigma_\xi = 0.25$ (Table 2). Here, differences between accuracies can be better estimated, as we are employing a bigger music collection and quite a high noise level. We see that UPGMA and WPGMA definitely perform best under the specified σ_ξ.

4.2 Real-world data

To evaluate the algorithms' accuracy with real-world data we independently optimized all possible parameters for each algorithm on setups 1.1 to 1.4. Within this pre-analysis, we saw that the definition of a distance threshold [13] was, in general, the only critical parameter for all algorithms. Apart from this, all other parameters turned out not to be critical for obtaining near-optimal accuracies. Methods that had specially broad ranges of these near-optimal accuracies were K-medoids, proposed method 2, and all considered hierarchical clustering algorithms.

We report the accuracies with optimal parameters for setups 2.1 to 3 (Table 3). We see that accuracies for proposed methods 1 and 3 are comparable to the ones achieved

[13] Either cophenetic distances, d_{Th}, or r_{Th} (see Sec. 2).

Algorithm	Setup				
	2.1	2.2	2.3	2.4	3
KM	0.637	0.642	0.656	0.666	*n.c.*
SL	0.776	0.783	0.833	0.828	0.676
CL	0.777	0.768	0.860	0.853	0.756
UPGMA	0.797	0.812	0.865	**0.875**	**0.796**
WPGMA	0.800	0.804	**0.866**	0.862	0.788
PM1	0.804	**0.813**	0.853	0.853	0.751
PM2	0.761	0.738	*n.c.*	*n.c.*	*n.c.*
PM3	0.788	0.790	0.841	0.848	0.733
MO	**0.808**	0.809	0.856	0.858	0.762

Table 3. Accuracy (F) for real-world data. Due to algorithms' complexity, some results were not computed (denoted as *n.c.*, see Sec. 4.3).

by the other algorithms and, in some setups, even better. Overall, we corroborate the findings of the previous section, although differences between algorithms are not so large now. We hypothesize that these similar (as well as near-optimal) accuracies are due to the relatively good distance measure provided by the employed cover song identification system (we have already seen that these differences get stressed with artificial data).

4.3 Time performance

In the application of these techniques to big real world music collections, computational complexity is of great importance. To qualitatively evaluate this aspect, we report the average amount of time spent by each algorithm to achieve a solution for each setup (Fig. 2).

We see that K-medoids and proposed method 2 are completely inadequate for processing collections with more than 2000 songs (e.g. setup 3). The steep rise in the time spent by hierarchical clustering algorithms to find a cluster solution for setup 3 also raises some doubts as to the usefulness of these algorithms for huge music collections. Furthermore, the hierarchical clustering algorithms, as well as the K-medoids algorithm, take the full pairwise distance matrix as input. Therefore, with a music collection of, say,

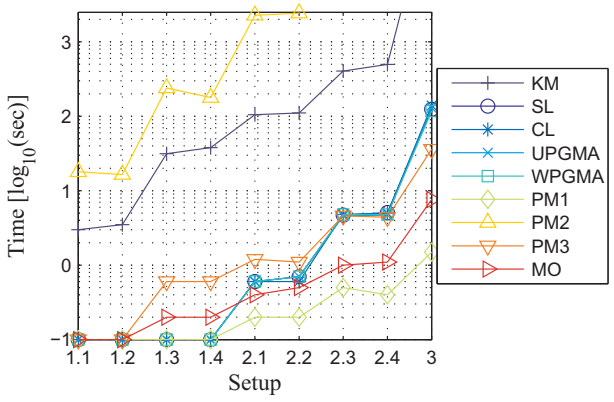

Figure 2. Average time performance for each considered setup. Algorithms were run with an Intel(R) Pentium(R) 4 CPU 2.40GHz with 512M RAM.

10 million songs, this distance matrix might be difficult to handle. In contrast, algorithms based on complex networks such as modularity optimization and proposed methods 1 and 3, only need a single list of answers A. Moreover, the length of the elements of this list can be very small due to d_{Th} and r_{Th} defined above (e.g. in our tests we found unnecessary to consider $r_{\mathrm{Th}} > 3$). It should also be noticed that the resulting network is sparse, i.e. the number of links is much lower than ${N_S}^2$ [10] and, therefore, calculations on such graphs can be strongly optimized both in memory requirements and computational costs as demonstrated, for instance, by [12]. Thus, methods based on complex networks, despite not achieving the highest accuracies for cover set detection, represent a robust and scalable option for processing big music collections.

5. OUTCOMES

5.1 Accuracy increase

In this section we show that the detection of cover song sets can improve the accuracy of batch-mode query systems. In particular, we study the relative MAP increase for the best cover set detection algorithms found. For comparison purposes, we take the output A of an initial batch-mode query system and transform it according to the grouping solution achieved. More concretely, we divide the distances in each A_i by the maximum distance value found and add an arbitrary constant $\beta > 1$ to the songs that are not detected as belonging to the cover set where s_i is included.

We plot the relative MAP increment versus the cardinality C of the cover sets for artificial data in Fig. 3. We show that one can get a MAP accuracy improvement of up to 25%. A comparable improvement can also be achieved in the case $C = \nu$. In general, it can be seen that the higher the cardinality, the higher the improvement we can get by detecting cover sets. Improvements for real-world data are much more modest as we do not have many sets with high C. With setup 2.4, average improvements are 2.8% for UPGMA, 2.1% for WPGMA, 5.1% for proposed method 1, and 4.9% for modularity optimization.

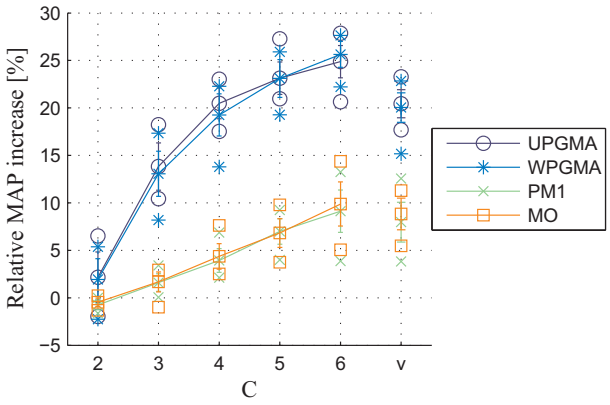

Figure 3. Relative MAP increment with artificial data. We use $\sigma_\xi = 0.25$ and, except C, the same parameters as in setups 2.3 and 2.4.

A further out-of-sample test was done within the MIREX 2008 audio cover song identification contest, where we submitted two versions of the same system [9] and obtained the two highest accuracies [14] achieved to date. The first version corresponded to the algorithm we use here for obtaining the real-world data [13]. The second version comprised the same algorithm, plus an additional post-processing step performing cover set detection based on proposed method 1 (the only method we had available at that time) and the maximum distance value normalization described in the present section. The MAP achieved with the former was 0.66 while the MAP of the latter was 0.75, which corresponds to a 13.6% relative increment. This increment is higher than the one achieved here with real-world data most probably because in the MIREX task $C = 10$.

5.2 Original detection

In the clustering context, many applications exploit compact cluster descriptions such as centroids or medoids [4, 5]. Analogously to cluster centroids and medoids, the cover set centroids and medoids can be determined. This way, the centroid of a cover set would correspond to the "average realization" and the medoid would correspond to the "prototype" [15] of the underlying musical piece. We here focus on the latter and leave the former for future consideration. In general, we could consider this prototype to be the most referential, influential, or inspirational song in a cover set (e.g. the musical piece covered by the majority of the other pieces). We here make an oversimplification and assume that the medoid of a cover song set corresponds to the original version.

To evaluate this option for detecting original songs we manually check for original versions in setup 3 of the used real-world data (we find 426 originals out of 525 cover sets) and we consider an ideal cover set detection algorithm (with all cover sets correctly estimated) as well as the best performing algorithms found in previous sections. For these last ones, we discard for evaluation the detected sets that do not contain an original.

To automatically determine the original song we take all possible pairwise distances within songs in a detected set and select the one which has a minimal distance sum to the other songs in the set. Let $\hat{S} = \{\hat{s}_j\}$, $j = 1, ..., \hat{C}$, be a set of detected covers with cardinality $\hat{C}$. Then, the index i of the prototype song corresponds to [16]

$$i = \underset{j}{\operatorname{argmin}} \left\{ \sum_{\substack{k=1 \\ k \neq j}}^{\hat{C}} d(\hat{s}_j, \hat{s}_k) \right\}. \qquad (2)$$

The results in Table 4 show the percentage of hits and misses, which can be compared to the null hypothesis of randomly selecting one song in the set. We observe that,

[14] http://www.music-ir.org/mirex/2008/index.php/Audio_Cover_Song_Identification_Results

[15] Standard, typical, or best example.

[16] Notice that analogous formulae can be derived by employing the notion of betweenness or closeness centrality in a complex network [10, 11].

Algorithm	C				
	2	3	4	5	6
Ideal	50.0	51.8**	39.2*	36.8*	41.7**
UPGMA	50.0	58.0**	55.0**	60.4**	50.4**
WPGMA	50.0	55.9**	46.6*	63.2**	47.5**
PM1	50.0	62.7**	61.3**	70.0**	50.0**
MO	50.0	62.8**	61.0**	70.0**	50.0**
Null	50.0	33.3	25.0	20.0	16.7
Ref. N_C	187	85	51	38	24

Table 4. Accuracies (%) for the original song detection task depending on C. The * and ** symbols denotes statistical significance at $p < 0.05$ and $p < 0.01$, respectively. The last line shows a referential N_C corresponding to the number of cover sets obtained with an ideal grouping solution.

in general, accuracies are around 50% with all considered cardinalities. This exactly corresponds to the null hypothesis of sets with $C = 2$ but, as soon as $C > 2$, accuracies become higher than the null hypothesis and statistical significance arises (statistical significance is assessed with the binomial test [16]). Discarding cover sets with no original song explains why accuracies for the algorithms studied here become slightly higher than the ones achieved by the ideal grouping algorithm. We hypothesize that our algorithms tend to split the ideal cover sets into "more coherent" or compact subsets and, therefore, within these, the method of Eq. 2 can better determine the original song.

6. CONCLUSIONS

In this paper we propose and study a framework for cover song identification based on the notion of cover sets that subsumes the current batch-mode query framework (the latter is naturally incorporated as an important part of the former). Through comprehensive experiments we show that the detection of cover sets is feasible, that it does not require extensive parameter tuning, and that it is quite robust to noisy distance measurements. Furthermore, we propose three versions of an unsupervised community detection algorithm that, when compared to existing state-of-the-art methods, achieve comparable accuracies with similar computation time (proposed method 3) or even faster (proposed method 1). We evidence that this new framework can provide new outcomes and can give raise to new applications. In addition to showing that cover set detection can substantially increase the accuracy (and coherence) of current systems, we here provide a proof-of-concept application to detect the original song within a cover set, which is inspired by the notion of cluster medoids.

Finally, we would like to highlight that, in spite of focusing on cover songs, the intuitive reasonings followed thoughout the paper could be as well applied to any IR batch-mode query system, and especially to other MIR systems (e.g. query-by-humming, query-by-example, audio fingerprinting, or music similarity).

7. ACKNOWLEDGMENTS

The authors would like to thank E. Gómez, M. Haro, P. Herrera, R. Marxer, and J. Salamon for useful discussions. This research has been partially funded by the EU-IP project PHAROS [17] IST-2006-045035.

8. REFERENCES

[1] J. Serrà, E. Gómez, and P. Herrera. *Audio cover song identification and similarity: background, approaches, evaluation, and beyond*. Springer, 2009. In press.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press Books, 1999.

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

[4] R. Xu and D. C. Wunsch. *Clustering*. IEEE Press, 2009.

[5] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[6] S. Fortunato and C. Castellano. *Community structure in graphs*. Springer, Berlin, 2009.

[7] L. Danon, A. Díaz-Aguilera, J. Duch, and A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics*, 9008:09008, 2005.

[8] A. Egorov and G. Linetsky. Cover song identification with if-f0 pitch class profiles. *MIREX extended abstract*, September 2008.

[9] J. Serrà, E. Gómez, and P. Herrera. Improving binary similarity and local alignment for cover song detection. *MIREX extended abstract*, September 2008.

[10] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. U. Hwang. Complex networks: structure and dynamics. *Physics Reports*, 424:4–5, 2006.

[11] A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *Proc. of the National Academy of Sciences*, 101:3747, 2004.

[12] V. D. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics*, 10:10008, 2008.

[13] J. Serrà, X. Serra, and R. G. Andrzejak. Cross recurrence quantification for cover song identification. *New Journal of Physics*. Under review.

[14] C. D. Manning, R. Prabhakar, and H. Schutze. *An introduction to Information Retrieval*. Cambridge University Press, 2008. Available online: `http://www.informationretrieval.org`.

[15] N. Sahoo, J. Callan, R. Krishnan, G. Duncan, and R. Padman. Incremental hierarchical clustering of text documents. *ACM Int. Conf. on Information and Knowledge Management*, pages 357–366, 2006.

[16] P. H. Kvam and B. Vidakovic. *Nonparametric statistics with applications to science and engineering*. John Wiley and Sons, Hoboken, New Jersey, USA, 2007.

[17] `http://www.pharos-audiovisual-search.eu`