Symbolic Music Similarity through a Graph-Based Representation

Federico Simonetta
Department of Information Engineering
University of Padua
simonettaf@dei.unipd.it

Filippo Carnovalini
Department of Mathematics
University of Padua
filippo.carnovalini@studenti.unipd.com

Nicola Orio
Department of Information Engineering
University of Padua
nicola.orio@dei.unipd.it

Antonio Rodà
Department of Information Engineering
University of Padua
roda@dei.unipd.it

ABSTRACT

In this work, a novel representation system for symbolic music is described. The proposed representation system is graph-based and could theoretically represent music both from a horizontal (contrapuntal) and from a vertical (harmonic) point of view, by keeping into account contextual and harmonic information. It could also include relationships between internal variations of motifs and themes. This is achieved by gradually simplifying the melodies and generating layers of reductions that include only the most important notes from a structural and harmonic viewpoint. This representation system has been tested in a music information retrieval task, namely melodic similarity, and compared to another system that performs the same task but does not consider any contextual or harmonic information, showing how the structural information is needed in order to find certain relations between musical pieces. Moreover, a new dataset consisting of more than 5000 leadsheets is presented, with additional meta-musical information taken from different web databases, including author, year of first performance, lyrics, genre and stylistic tags.

CCS CONCEPTS

• **Information systems → Music retrieval**; • **Applied computing → Sound and music computing**;

KEYWORDS

Melodic Similarity, Symbolic Music, Music Reduction

ACM Reference Format:
Federico Simonetta, Filippo Carnovalini, Nicola Orio, and Antonio Rodà. 2018. Symbolic Music Similarity through a Graph-Based Representation. In *Audio Mostly 2018: Sound in Immersion and Emotion (AM'18), September 12–14, 2018, Wrexham, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3243274.3243301

AM'18, September 12–14, 2018, Wrexham, United Kingdom

ACM ISBN 978-1-4503-6609-0/18/09…$15.00
https://doi.org/10.1145/3243274.3243301

1 INTRODUCTION

Several studies concerning music cognition prove that our music perception uses some level of *abstraction* [1, 8, 23]. We believe that the study of symbolic music could help to improve computational systems where user perception is relevant, because, as Vinet points out,

> *"The symbolic representation is content-aware and describes events in relation to formalized concepts of music (music theory)." [27]*

This means that the symbolic level could include information about our music cognition and not just about music notation, through the help of music theory. Indeed, some of the abstractions that a listener naturally performs – e.g. transposition-independent melody recognition, pitch categorization, timbre classification – are already represented in music theory. Also, certain notations include generally accepted cognitive notions; for instance a $B\sharp$ is different from a C because $B\sharp$ includes contextual information that translates to a perceived *tension* and need of *resolution*.

Until now, most of the studies about symbolic level in computational applications focused on pitch information, often represented through `MIDI` values. Sometimes *pitch class* invariant representations were used, while others mantained transposition invariance by using the intervals between pitches, rather than the pitch values themselves. Various degrees of precision have been used in the definition of these intervals. Some studies also focused on chord representations [4, 5, 10].

Other studies used more complex graph and tree structures designed for information retrieval purposes [19–21], sometimes inspired by musicological theories like Schenkerian theory and Lerdahl and Jackendoff's Generative theory of tonal music [9, 15, 17].

In this paper we introduce a modified version of [17]. This is mainly based on `MusicXML` files, which are able to represent the symbolic level enriched by information expressed in traditional musicology that is not immediately present in other formats like `MIDI`, like for example slurs or textual annotations. We also show a novel general graph-based paradigm that, theoretically, could be able to represent both the harmonic and contrapuntal textures in polyphonic works. We test this method with a novel dataset of more than 5000 leadsheets containing songs of different genres, each associated to lyrics, meta information and about 100 statistical features about the symbolic level and meta information.

2 GENERAL FRAMEWORK

There are two main aspects in our approach:

> *Segment based representation*: music is represented as a sequence of tokens: this allows us to highlight contrapuntal relationships between segments in different points of the same part or even between different voices.
> *Harmonic context reduction*: each part is *reduced* by progressively deleting the least *relevant* notes.

The definition of *reduction* is clearly ambiguous, but this basic idea is presented, although with various interpretations, in most of the graph-based representations cited above.

To adequately represent these two aspects, we propose to use a graph structure in which the nodes are segments of the music piece and the edges are labeled with the *transformation* function that changes one segment into the other. In this way, if a computer application is interested in only one kind of these transformations, it can restrict its operations to the corresponding type of edge. Moreover, to be effective in a wider range of computational tasks, the structure could also include information on temporal and spatial relationships between segments.

Transformations could include, for instance, transposition (diatonic and/or chromatic), inversion, retrograde inversion, doubling/halving of durations, etc. From this point of view, reductions could also highlight some transformations that would be difficult to identify otherwise, allowing a better understanding of theme and motif elaborations that occur in the song [15].

We propose the following general procedure:

(1) *Segment* each voice;
(2) *Reduce* each segment;
(3) *Identify transformations* between different segments – both involving the reduced and the original segments;
(4) *Repeat* from point 2 for each reduced segment still containing more than one note.

About the *segmentation* stage, several algorithms have been proposed – see [14] for a review, while the *reduction* stage can be carried out exploiting one of the algorithms proposed in [9, 15, 17, 20]. The representation method – intervals, midi values, etc. – of the segments themselves can be chosen depending on the application. Further details on how these techniques were integrated in our work are presented in the next section.

In the presented implementation, we did not focus yet on the *identification* stage, since we prefer to test the paradigm in an intermediate stage which involves just melodies, segmentation and reductions, leaving other transformations to future work.

3 IMPLEMENTATION

3.1 Segmentation

For the *segmentation* stage we chose to implement the *Local Boundary Detection Model* (LBDM) algorithm, which is fast, widely used and simple to implement, as described in [3]. It is also based on musicological features, making it a natural choice for our proposal that is also musicologically grounded.

LBDM is based on the assignment of scores to each couple of two consecutive notes, that depend on the amount of variation (called *degree of change*) of certain features between the examined couple of notes and the surrounding ones. The basic idea behind this is that a musical segment must have some kind of internal coherence to be perceived as a single united element, so a greater variation is a sign of the disruption of this coherence and thus of the end of a musical phrase. While it is possible to use different features, we used the ones used in [3]: pitch difference, duration of the notes, rests between the notes.

We made some potential improvements to its original definition. The original *LBDM* defines a boundary when a local maxima with values over a fixed threshold is found in the succession of scores – this succession is called *boundary profile*. We substituted the fixed threshold with a single-linkage clustering and we introduced a rule to give different weights to rests and pitch differences based on the percentage of rests in the score. If many rests are present, they receive higher weights, otherwise when few rests are present the weighting scheme favours pitch differences. This is based on the assumption that a musical piece with few rests expresses a hiatus using pitch jumps, while if many rests are present, they are probably also used to define hiatuses. The effect of these changes has been evaluated only qualitatively and without systematic collection of relevance judgments.

3.2 Reduction

We based our reduction algorithm on [17]. There, each note was assigned a score that took into account the metrical position of the note in its measure, and the importance of the note in respect to the underlying chord as well as the importance of said chord in respect to the tonal context. The authors used a sliding window of two notes, and at each step they deleted the least relevant note until a single note remained in the whole piece. In case of a tie between the scores of two consecutive notes, the authors first compared the importance of the underlying chords, then the metrical importance and finally the consonance/dissonance of the note with its underlying chord. In our work we decided to switch the order of the metrical and consonance/dissonance comparison in case of ties.

We also extended the algorithm to make it able to deal with tuplets, ternary meters and tied notes, unlike the original formulation. To this end, we used a sliding window with a variable duration based on the beat of the piece, rather than just being double the duration of the shortest note as in the original formulation. For example, in a 3•4 piece, if the shortest note is an eight note, then the window will be 1•4 long, but if it is a quarter, then the window will be 3•4 long, because in a piece with ternary beat 2•4 is not a reasonable subdivision according to music theory. Then, for each window containing more than one note, the algorithm deletes the note with minimum score, and "covers" the deleted note by expanding either the prevous or the following note, based on which one has the highest score. In case of tie, the previous note is chosen. Tuplets are managed in the first step of each reduction by creating a standalone measure for each tuplet, that is reinserted in the original piece after the reduction. This process ends when only one note remains for each measure, then a simplified version of the algorithm further reduces the notes until only one note remains. Figure 1 shows an example of the reduction procedure.

To apply this algorithm, the input needs to have information about the underlying chord of each note. In [17], the musical pieces

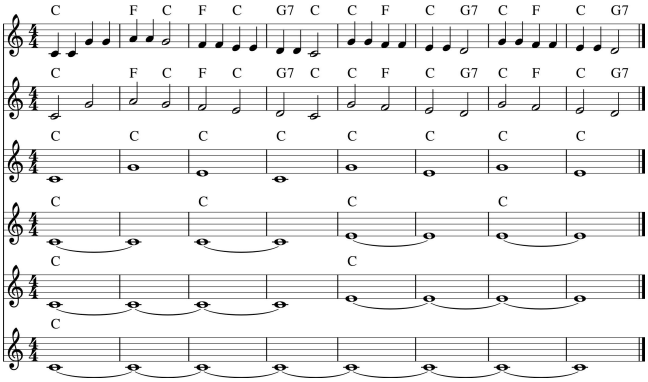

Figure 1: Monophonic reduction: a simple example of reduction procedure of a melody until just one note remains. Notice that the resulting note is the root of the tonality.

used for evaluation were simple monophonic melodies, so the harmonic information was manually added in the form of chord annotations. In the dataset we used in this work – see section 4 – these chords annotations were already present, so there was no need to add them manually or to compute the chords in any way.

3.2.1 Edges Weighting. The greatest modification we applied to the reduction algorithm taken from [17], is to introduce weights to the edges of the graph. Since every reduction step deletes a variable amount of notes, the original segment could be very similar or very different from its reduction. We thus decided to introduce weights to each reduction to keep track of the distance from the original piece. We decided to test a variety of different ways to compute this weights, that are listed below. From now on we will call these functions *weight functions*.

Semantic function is computed between two notes as the difference of the mean of the scores used in the reduction stage; optionally, the *estrada* [22] distance can be weighted together to count chord structure similarity. Since scores are tonality invariant this distance is also diatonic transposition invariant.

MIDI function between two notes is given by the difference of the respective MIDI values but after having subtracted the average MIDI value of each respective segment [26]. This is chromatic transposition invariant.

Boolean function is 1 if the two intervals were different and 0 otherwise. The invariance to transposition depends from the interval representation scheme.

Fuzzy function between two intervals is computed as the difference between values associated to each interval; we associated to musicological intervals one of the following:

- one minus the *consonance* score used in the reduction stage associated to the musicological interval difference;
- the difference between the number of semitones in each interval;

Notice that *semantic* and *MIDI* functions are based on notes, while the other functions are based on intervals. This means that they require different representations of the segments created in the segmentation and reduction phases. Notes representation is simply based on MIDI values. Instead, for the intervals we tested four different representations: musicological – major/minor seconds, thirds and so on –, General Pitch Interval Representation (GPIR) [2] – only the diatonic intervals are considered: for example a third is always a third regardless of it being diminished, minor, major or augmented – , step-leap – interval equal to a major or minor second had values 1 or +1, unison had value 0 and others had values 2 or +2 – and contour, which is similar to step-leap but only 0, 1 and +1. Clearly, these different representations imply different degrees of precision in distinguishing the intervals.

We tested the goodness of each function and representation in the esperiments described in Section 5.

3.3 Score and Dataset Representation

We represent a score – or a leadsheet – as a sequence of segments, each one with its own reductions and weights. The dataset is then represented as a set of scores, thus the same segment may have multiple occurrences in the dataset; however this redundancy can affect efficiency but not effectiveness.

4 DATASET

As mentioned before, our implementation requires scores having chords annotations, i.e. what musicians call leadsheets. To our knowledge, the biggest leadsheet dataset freely available is the *Nottingham Folk Music Database*, which contains more than 1200 leadsheets of British and American folk tunes in ABC format and was originally created by Eric Foxley[1]. Unfortunately, we were not able to access the *LSDB* dataset [18] that contains more than 8.000 leadsheets of jazz songs.

Wanting a database bigger and more varied in genre than the *Nottingham Folk Music Database*, we created the *Enhanced Wikifonia Leadsheet Dataset (EWLD)*, a new leadsheet dataset with more than 5100 scores. Starting from the *Wikifonia* archive[2] , we collected data from discogs.com and secondhandsongs.com. We added information like lyrics, genre, features [16] – extracted with `music21` [7] – correct composer name, correct title, date of first performance, composer date of birth and death and so on. Since *Wikifonia* users were not professional score editors, we had to filter out scores which had no key signature or chord annotations, and those that had multiple parts – or single parts with multiple voices –, key changes or modulations – according to the *Krumhansl-Schmuckler* algorithm [12]. All these tasks were performed automatically through a `python` script. The result is the heterogeneous dataset we desired: Figures 2 and 3 give an idea of the variety represented in this dataset.

5 EXPERIMENTS

We tried to evaluate the goodness of a music symbolic representation based on principles described in section 2, with a *query-by-excerpt* approach: given a segment as query we retrieved a list of similar songs. Since our dataset does not include relevance judgments yet, we followed two approaches: in the first typology of

[1] It is available in ABC format at https://ifdo.ca/~seymour/nottingham/nottingham.html, while the original database in a plain ASCII format at http://www.chezfred.org.uk/freds/music/

[2] While Wikifonia archive is no more directly accessible from http://www.wikifonia.org, you may contact the authors if you are interested in EWLD.

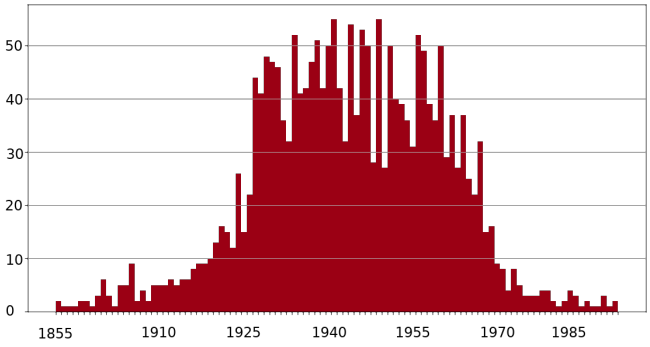

Figure 2: The number of works for each first performance year

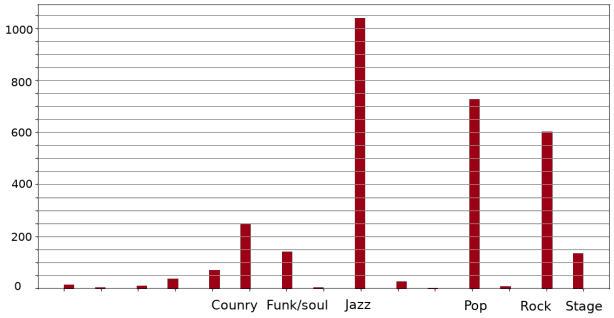

Figure 3: The number of works for each genre

experiments we tried to retrieve the *parent* document of the query segment – later *parent song detection* –. This kind of experiments also helped trying various settings for our system. In the second typology of experiments we compared the results of a query in our system, and the same query (with the same dataset) in a different system, based on the melody alone, to see if the results differ and if keeping into account the harmonic context can give reasonable results that would be overlooked otherwise.

5.0.1 Similarity Measure. The similarity between two segments A and B is computed by summing of the weights of each edge on the path from A to B in the graph generated by the reductions. If A and B have some common reduction, then a path linking them exists and the distance is computed using the already present edges. If A and B have a different top level note in the reduction tree, then an edge is added between the two top level notes, having a weight computed according to some distance measure. In our experiments, these special weights were always set to 1 to avoid increasing computational time, but we checked that results do not vary significantly with weighted edges. Figure 4 shows a demonstrative example of how two segments are compared. The distance between a given segment C and a whole song was computed as the average distance between C and each segment in the song.

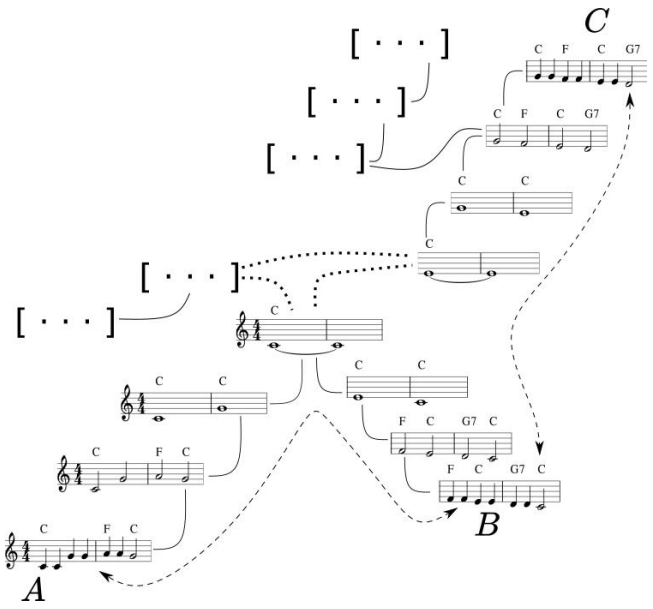

Figure 4: Dataset representation: the distance between two segments – i.e. A and B – is computed as a path (dashed lines) in the graph representation of the dataset; if two segments – i.e. B and C – have a different top note in their own reduction tree, then another edge is added at retrieval time (dotted lines). Weights are not shown in the figure.

5.1 Experimental Setup

For the *Parent Song* detection, we observed results by varying different parameters, such as the type of distance used to compute the dissimilarity between two segments, the representation of each segment itself and the scoring scheme used in reductions. In the setting of the scores used by the reduction algorithm, we were inspired by classical harmonic theory and cognitive studies about tonal hierarchies [11]; tables 1 and 2 show the scores used for the computation of the functional harmony and consonance/dissonance relations, while the metrical score was set according to the number of times the beat's duration had to be divided in half (or in 3 parts in the case of compound meters) to get the note's position. We tested three different scoring schemes, but no relevant differences were found, thus the following results all use the scheme presented in tables 1 and 2.

Moreover, we performed experiments with all these variants of settings both with indexed segments, that is segments that were present in the graph, and random segments, that were generated by randomly extracting segments from songs in the dataset – thus getting segments that were quite different from the ones created by LBDM.

5.2 Measures

Providing that the query documents did not have the corresponding relevance judgements, we had to define some special evaluation metrics that don't rely on such judgements.

Intervals	Scores
perfect unison	1.00
perfect fifth	0.75
diminished fifth	0.50
major third	0.75
minor third	0.75
diminished third	0.20
perfect fourth	0.30
minor sixth	0.40
major sixth	0.40
minor second	0.20
major second	0.20
minor seventh	0.50
major seventh	0.20
diminished seventh	0.50
rest	0.50

Table 1: Consonance scores of a note in respect to the root of a chord.

Scale grade	Scores
I	1.0
II♭	0.2
II	0.5
III	0.4
IV	0.8
IV♯	0.3
V	0.9
VI♭	0.1
VI	0.6
VII	0.7
rest	0.50

Table 2: Functional scores of a chord based on its root grade in the tonal region.

Precision at 20 (P-20): the number of queries upon the total (400) in which the parent document of the query segment appeared in the top-20 documents retrieved; together with MRR it gives an indication of the precision of the retrieval. The usual *average precision* cannot be used in this case since there is always only one relevant document (the parent), thus the precision could be at most the reciprocal of the retrieved documents.

Mean Reciprocal Rank (MRR): if the parent document of the query segment was retrieved in the top-20 documents, we stored its reciprocal rank; then, we computed the mean over all samples stored; this is a typical measure in information retrieval [6].

Mean Normalized Inverted Rank (MNIR): if the parent document of the query segment was retrieved in the top-20 documents, we stored its normalized and negated rank; then, we computed the mean over all samples stored; this is an unusual measure but being completely linear allows to compute the mean rank without losing the *higher is better* approach; it is not top-heavy, while MRR is.

DISTANCE		P-20	MNIR	MRR
semantic, no chords weight		0.56	0.91	0.71
semantic, chord weight = 0.5		0.45	0.91	0.75
MIDI		0.54	0.89	0.71
DISTANCE	**INTERVAL**			
boolean	music21	0.235	0.48	0.13
	GPIR	0.21	0.44	0.11
	step-leap	0.19	0.44	0.12
	contour	0.24	0.51	0.13
fuzzy	music21 (a)	0.20	0.51	0.14
	music21 (b)	0.25	0.45	0.12
	GPIR	0.215	0.51	0.14
	step-leap	0.21	0.44	0.11
	contour	0.175	0.515	0.13

Table 3: *Parent* song detection with indexed segments

DISTANCE		P-20	MNIR	MRR
semantic, no chords weight		0.07	0.9	0.57
semantic, chord weight = 0.5		0.04	0.8	0.49
MIDI		0.06	0.8	0.58
DISTANCE	**INTERVAL**			
boolean	music21	0.03	0.5	0.11
	GPIR	0.03	0.5	0.22
	step-leap	0.03	0.6	0.16
	contour	0.02	0.6	0.19
fuzzy	music21 (a)	0.03	0.5	0.21
	music21 (b)	0.03	0.6	0.18
	GPIR	0.02	0.5	0.17
	step-leap	0.02	0.5	0.20
	contour	0.03	0.6	0.23

Table 4: *Parent* song detection with random segments

5.2.1 Results. For each of the different settings described in section 5 we performed 400 queries. The results, according to the metrics described above, are presented in table 3, that shows the results using indexed segments as queries, and in table 4, that shows the results obtained when random segments were used.

5.3 Comparison with Melody-Based Retrieval

As stated above, we also used another approach to overcome the lack of relevance judgements, that was the comparison with another established algorithm for melodic similarity. By looking at the results of the *Mirex 2015 Symbolic Melodic Similarity challenge* [13], we decided to use *MelodyShape* [25], as it had very good results in the challenge, and a complete implementation is already available [24], saving us the time to reimplement it.

We compared the results that our algorithm – using semantic distance, with no chord weight – and Melshape – using the 2015-shapetime variant – gave in a qualitative way, before setting up another experiment. This explorative comparisons made it clear

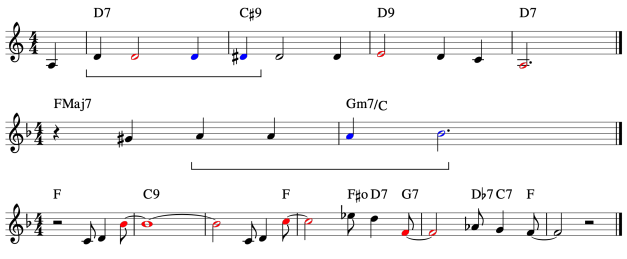

Figure 5: The first line is a query, the second line is the top result retrieved by MelodyShape, the third is the top result retrieved by our system. The underlined notes show an identical melody, with a semitone step (the blue notes). The red notes represent the common reduction of the query and the third line.

that our system often does not retrieve those melodies that are the most similar to the queries. In fact, most of the times the top-10 results given by MelodyShape and the top-10 results given by our implementation had empty intersection. We thus decided not to use the results of MelodyShape as a sort of relevance judgement, but instead to try to understand what is the reason of such different results, and if they both were reasonable.

Figure 5 shows a good example of how the two approaches differ, but are both interesting. MelodyShape only considers the melodies given in input, so in the example it found a very similar melody, characterized by a semitone step – the blue notes in the figure. The red notes are instead the ones in the reduction of the query, and noticeably the semitone step is not included. Since the $C\sharp 9$ is not a very important chord when compared to the song tonality – C Major –, our algorithm considered those $D\sharp$s to be more decorative notes rather than structural, and thus were overlooked. The result given by our algorithm is instead quite different from the query from the point of view of the mere melody, but the longer notes show that the two pieces share the same basic structure – going up by a major second and then down by a perfect fifth –, even if this structure was decorated in the full melody in very different ways.

6 CONCLUSIONS

Our evaluation is a feasibility evaluation of a music symbolic representation based on principles described in section 2. We did not perform any formal evaluation with judgements of relevance and our results should be taken as indicative of the goodness of this approach.

The creation of models took about one hour and a half on a computer cluster with 8 different processes for about 5000 songs. It needed an extensive RAM usage – due also to the not perfect management of shared memory in python – and each dataset file had size of about 100 MB compressed with LZMA encoding.

Aside from the computational side, results show that note based representations, used with the *semantic* weight function, which is able to take into account the functional harmonic context in a hierarchic way as suggested by cognitive studies, had the best results. This shows the importance of the musical context, included functional harmonic context, in music representation and melodic similarity, but still our system is outperformed by many others in terms of a symbolic melodic similarity search, as made clear by the low scores archived in our parent song detection task when random segments were used.

The comparison with a well-performing algorithm like MelodyShape shows indeed that our results do not really reflect the usual concept of melodic similarity. Instead, analyzing results such as the one reported in figure 5, we can see that our system has the interesting ability of finding "hidden" patterns in the melodies, retrieving segments that share a similar structure, even if the melody is quite different.

This leads us to believe that a system based on this framework should not be used for similarity tasks, as other systems are both more effective and more efficient, but could give interesting results if integrated in musicologial analysis tools, to help scholars find similarities among songs that are not evident from the melody itself. Adding trasformations other than the reductions, as we described in section 2, could improve even more this ability to find underlying similarities.

Some interesting applications based on these intuitions would be the creation of an application that makes it visually simple to navigate the graph of the reductions, to find this kind of similarities in an intuitive way. Another application that could be devised is the use of the graph in an algorithmic composition framework, to generate melodies starting from simple structures given by the user.

All these applications, as well as the implementation of the other transformations, is left for future work. We also plan to test this approach with polyphonic music, since our aim is to create a representation system that is able to express both the harmonic and contrapuntal intrinsic features of musical texture, for any kind of music. The proposed framework needs to be further extended to achive this goal.

REFERENCES

[1] Rita Aiello. 1994. Music and Language: Parallels and Contrasts. In *Musical Perceptions*, Rita Aiello and John A. Sloboda (Eds.). Oxford University Press, Oxford, UK, 40–63.
[2] Emilios Cambouropoulos. 1996. A General Pitch Interval Representation: Theory and Applications. *Journal of New Music Research* 25, 3 (1996), 231–251.
[3] Emilios Cambouropoulos. 2001. The Local Boundary Detection Model (LBDM) and its Application in the Study of Expressive Timing.. In *ICMC*.
[4] Emilios Cambouropoulos. 2016. The Harmonic Musical Surface and Two Novel Chord Representation Schemes. In *Computational Music Analysis*, David Meredith (Ed.). Springer International Publishing, Cham, 31–56. https://doi.org/10.1007/978-3-319-25931-4_2
[5] Emilios Cambouropoulos, Maximos Kaliakatsos-Papakostas, and Costas Tsougras. 2014. An Idiom-independent Representation of Chords for Computational Music Analysis and Generation. In *Proceedings of the Joint 11th Sound and Music Computing (SMC) Conference and 40th International Computer Music Conference (ICMC)*. https://doi.org/10.13140/2.1.4128.1281
[6] W Bruce Croft, Donald Metzler, and Trevor Strohman. 2010. *Search engines: Information retrieval in practice*. Vol. 283. Addison-Wesley Reading.
[7] Michael Scott Cuthbert and Christopher Ariza. 2010. music21: A Toolkit for Computer-aided Musicology and Symbolic Music Data. In *ISMIR*.
[8] Diana Deutsch, William F Thompson, Henkjan Honing, and Stephen McAdams. 2013. *Psychology of Music*. Elsevier.
[9] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. 2016. Implementing Methods for Analysing Music Based on Lerdahl and Jackendoff's Generative Theory of Tonal Music.
[10] Christopher Harte, Mark B Sandler, Samer A Abdallah, and Emilia Gómez. 2005. Symbolic Representation of Musical Chords: A Proposed Syntax for Text Annotations.. In *ISMIR*, Vol. 5. 66–71.
[11] Carol L. Krumhansl and Lola L. Cuddy. 2010. A Theory of Tonal Hierarchies in Music. In *Music Perception*, Mari Riess Jones, Richard R. Fay, and Arthur N. Popper (Eds.). Springer New York, New York, NY, 51–87. https://doi.org/10.1007/

978-1-4419-6114-3_3
[12] Carol L Krumhansl. 2001. *Cognitive foundations of musical pitch.* Oxford University Press.
[13] International Music Information Retrieval Systems Evaluation Laboratory. 2015. MIREX 2015 Symbolic Melodic Similarity Results. Retrieved May 29, 2018 from http://www.music-ir.org/mirex/wiki/2015:Symbolic_Melodic_Similarity_Results
[14] Marcelo Rodrıguez López and Anja Volk. 2012. *Automatic Segmentation of Symbolic Music Encodings: A Survey.* Utrecht University.
[15] Alan Marsden. 2010. Recognition of Variations using Automatic Schenkerian Reduction. *ISMIR* (2010).
[16] Cory McKay and Ichiro Fujinaga. 2006. jSymbolic: A Feature Extractor for MIDI Files.. In *ICMC*.
[17] Nicola Orio and Antonio Rodà. 2009. A Measure of Melodic Similarity based on a Graph Representation of the Music Structure.. In *ISMIR*. 543–548.
[18] François Pachet and Jeff Suzda. 2013. A Comprehensive Online Database of Machine-Readable Lead-Sheets for Jazz Standards.. In *ISMIR*.
[19] Alberto Pinto, Reinier H Van Leuken, M Fatih Demirci, Frans Wiering, and Remco C Veltkamp. 2007. Indexing Music Collections through Graph Spectra. In *ISMIR*. 153–156.
[20] David Rizo Valero. 2010. *Symbolic Music Comparison with Tree Data Structures.* Ph.D. Dissertation. Universidad de Alicante.
[21] David Rizo Valero, José Manuel Iñesta Quereda, and Pedro J Ponce de León. 2006. Tree Model of Symbolic Music for Tonality Guessing.. In *Artificial Intelligence and Applications*, Vol. 2006. 299–304.
[22] Thomas Rocher, Matthias Robine, Pierre Hanna, and Myriam Desainte-Catherine. 2010. A Survey Of Chord Distances With Comparison For Chord Analysis.. In *ICMC*.
[23] D. Schön, L. Akiva-Kabiri, and T. Vecchi. 2007. *Psicologia della Musica.* Carocci, Rome, Italy. https://books.google.it/books?id=LZEYAQAAIAAJ
[24] Julián Urbano. 2013. MelodyShape: a Library and Tool for Sym- bolic Melodic Similarity based on Shape Similarity. Retrieved May 29, 2018 from https://github.com/julian-urbano/MelodyShape
[25] Julián Urbano. 2014. MelodyShape at MIREX 2014 Symbolic Melodic Similarity. *10th Music Information Retrieval Evaluation eXchange (MIREX 2014); 2014 Oct 27-31; Taipei, Taiwan. MIREX; 2014.* (2014).
[26] Gissel Velarde, David Meredith, and Tillman Weyde. 2016. A Wavelet-Based Approach to Pattern Discovery in Melodies.
[27] Hugues Vinet. 2003. The Representation Levels of Music Information. In *International Symposium on Computer Music Modeling and Retrieval*. Springer, 193–209.