Melodical Transformation and Evaluation of Music

Elias Moons
KU Leuven, 2016, Leuven, Belgium

This paper describes a method to melodically transform given musical pieces to new ones. There will also be presented a framework on how to evaluate these transformations and musical pieces in general. This framework will be used to calculate the efficiency of the transformations. At last an algorithm is presented that will efficiently combine different melodical transformations to acquire a new musical piece that is optimal under the proposed framework. The algorithm can only make use of a given set of transformations to perform this task.

1 Introduction

One of the most common problems for musicians these days is the *writer's block*. This phenomenon, where a musician momentarily lacks the inspiration to come with ideas for new melodies can be very frustrating. Therefore, it would be very useful to have a tool that gives the songwriter the inspiration that he/she needs. One possible solution to this problem will be explained in this paper. This solutions consists of transforming an original melody into a new one, using transformations.

A lot of research has already been done in the field of musical generation [1]. But the results of these experiments often led to musical pieces that were either musically correct but often less interesting to listen to (often had an 'artificial' sound), or less musically correct but too frustrating. Musical transformation might give an interesting out in this respect. Transforming an original melody, of which it is assumed to be interesting, will hopefully keep this interestingness after transformation. This paper discusses a particular set of transformations, called the melodical transformations. These transformations completely ignore the ryhtmical part of a musical piece and focus on the melody. These transformations will only have an effect on the pitch of the notes in the original melody. The counterpart of melodical transformations is rhythmical transformations. There has already been done research in this domain, with promising results which led to the belief that melodical transformations might work as well [2].

There will also be proposed a framework to judge these transformations and melodies in general. A good framework is necessary to be able to adequately judge these melodical transformations. At last an algorithm will be described which will combine different given transformations to transform a certain melody as efficiently as possible into a new one. This new melody should have the highest possible probability in the proposed framework.

2 Scope

Since research on the topic of musical transformation is still in its early days, it is too difficult to immediately start dealing with all kinds of problems at once. This paper therefore describes transformations only to transform melodies (only one note played at the same time), instead of full polyphonic musical pieces. Also the proposed framework will be built based on the assumption of single-voice problems. The transformation of polyphonic musical pieces is a lot more complex and is subject to further research.

The transformation, reference model and algorithm that will be described in this paper will be tested and based on the Essencorpus [3]. This corpus contains Folk songs in the MusicXML [4] format which makes it easy to use an internal representation of the music based on music theory with notes (instead of a physical one based on frequencies). It is also important to note that the framework that will be presented to assess the quality of musical pieces, will only consider the melody of a musical piece and ignore the rhythmical information in the piece. The reason for this is because the transformation and algorithm that are described also only reason in terms of the melody.

3 Framework for evaluating melodies

In this section, a framework is described that will be used to evaluate transformations later on. There are lots of important parameters that define the quality of a transformation. The framework that is proposed combines three of these important parameters.

3.1 Idea of the RPK-model

The RPK-model is a framework of which the ideas are described in [5]. This model is based on three important pa-

rameters, which are also the basis for the name of the model. For each of the notes in a melody, its probability is modelled as the product of these three parameters.

There is the *range*-parameter, which models the distance of a note to the average pitch in the corpus. Notes that have a closer distance to the average pitch, have a higher probability of occurrence. The distribution of notes in the Essencorpus is given in figure 1 to illustrate this.

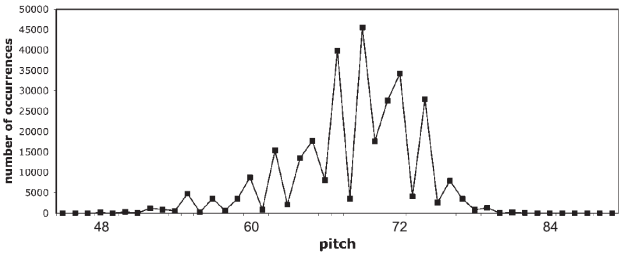

Fig. 1. Distribution of all notes in the Essencorpus. Middle C has value 60, the unit on the x-axis is a semitone.

The second parameter is the *proximity*. This models the distance of a note to the note that is played previous to this note. This is an important parameter because the context of a note (notes played in the immediate neighbourhood of that note) have an influence on the probability of occurrence of a certain note. Notes that differ less in pitch are for example more common to succeed each other than notes that have a bigger difference in pitch. In figure 2, the occurrences of all intervals between consecutive notes in the Essencorpus is illustrated.

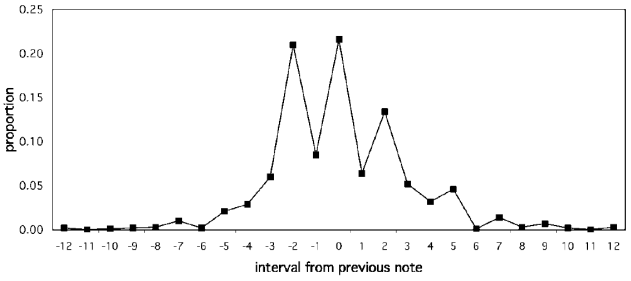

Fig. 2. Proportions of occurrence of all intervals between neighbouring notes in the Essencorpus. Difference expressed in amount of semitones.

At last, the *key*-parameter is introduced. This parameter models the probability of occurrence of a certain note in the key of the musical piece. This is crucial because there is a big difference in occurrence for different notes in the key. There is also a difference in the key-profile for major and minor scales. This becomes clear in figures 3 and 4 for major and minor scales respectively. Here the distribution of notes for all musical pieces in the Essencorpus is displayed.

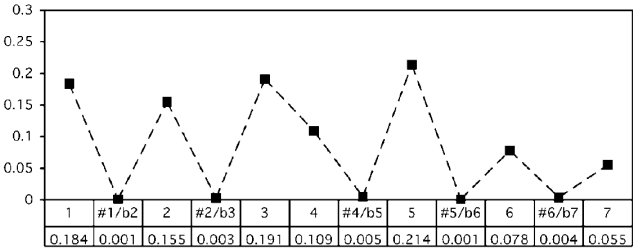

Fig. 3. Average distribution of notes in the Essencorpus for pieces that are in a major key.

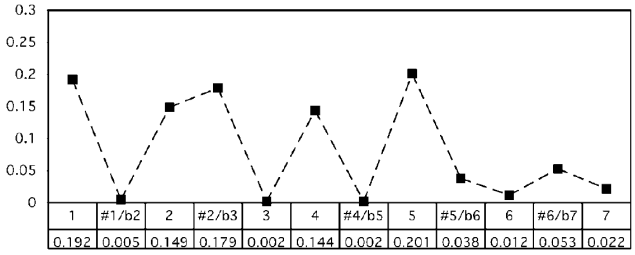

Fig. 4. Average distribution of notes in the Essencorpus for pieces that are in a minor key.

3.2 Using the RPK-model

We can model the probability of a note in a musical piece. Now we can define the probability of a given melody as the mean probability of all notes in that melody. By calculating the probability of a melody before and after a certain transformation, the quality of a given transformation can be assessed.

It is important to note that a transformation that gives better scores on average, is not necessarily a better transformation. The reasoning behind this is that musical pieces that score extremely high on the RPK-model are very boring pieces of music that have a lot of occurrences of the same note that lies close to the central pitch. Musical pieces with very low scores are on average frustrating to listen to, because they will probably consist of a lot of big 'jumps' in the melody.

The solution to this problem is to consider the score of an original melody as a sort of golden number for a melody on this rhythm (since the rhythm of a musical piece will not change under the melodical transformations described in this paper). Now a given transformation will be judged 'better' than another one if, one average, it produces results that in score differ less from the original.

4 Melodical Transformation

The melodical transformation that will be discussed in this paper is one that is dependent on the difference in semitones to the previous one. This means that only the difference in semitones to the previous note will decide which note a given note will transform to.

Diff (mod 8)	0	1	2	3	4	5	6	7
Addition	5	-4	1	-3	1	1	2	3

Table 1. Transformation dependent on the difference in semitones between the current and previous note.

4.1 Idea of the transformation

This difference in semitones (modulo 8) can be seen as an index of a function that gives back a number that represents that amount of semitones that will be added to the current note. An important aspect of the transformation is that this amount of semitones will be 'added' in the other direction of the distance to the previous note. This means that a positive value for addition always leads to a jump of pitch of the current note in the direction of the pitch of the precious note. A benefit of this directive jump is that differences in both directions are handled equally. A certain rising gap will lead to a lowering jump that is as big as a descending gap of equal size would generate in the opposite direction.

4.2 Example

As an example, table 1 is given. A difference in semitones of 6 downwards will for example lead to a rise in semitones of 2 of the current note using this transformation. In figure 5, an example is given of the application of this transformation on a given melody. It is also important to note that after applying the transformation on a note, if the resulting note is not part of the key of the piece (and only then), it will be transformed to its neighbour note (+1 of -1 semitone) which has the highest probability. This means that a jump of +4 semitones in the table can theoretically lead to a jump of +3, +4 or +5 semitones.

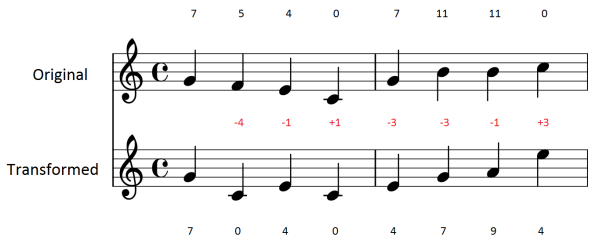

Fig. 5. Example of using transformation described in table 1.

4.3 Discussion

One of the benefits of this transformation that it is still a relatively simple one to use. The concept is easy and the applications doesn't require any complex or heavy computation. It is however, far from optimal since only a small part of the context of a given note (only the previous one) is taken into account. Results might be better if a bigger idea of context could be integrated into the transformation.

An interesting benefit of the transformation is that, although it does not explicitly searches for repetitiveness in the music, it almost always keeps it through the transformation. This means that the structure of the original musical piece is often kept, which can be interesting for giving the music a more natural feel that the listener is more used to.

5 Combining Transformations

In this section, an algorithm will be described that will combine different melodical transformations to transform an original melody into a new one. This new melody should have a probability, as judged by the RPK-model, that is as high as possible. To do this, for each note in an original melody, the algorithm has the choice between either keeping the note as it is, or alternatively to transform the note using one of the given transformations.

5.1 Different parameters

The algorithm is dependent on three different parameters. First, there is the input, the original melody on which the transformation should take place. The impact that the original melody has on the algorithm, performance wise, is firstly in its amount of notes (AN). Second, there are the different given transformations (or the amount of transformations (AN)) that the algorithm can use. The more different transformations, the more options the algorithm has to transform a given note. Lastly, there is also a 'minimum transformation length' (ML) parameter. This parameter states that if the algorithm wants to transform a certain part of the original melody using a certain transformation, it should do so for at least ML consecutive notes with the same transformation.

5.2 High level idea of the algorithm

The algorithm is based on the principles of *dynamic programming* [6]. The main idea of the algorithm is the following. Suppose we are at note n in the melody. This note on position n has a few different notes it can be transformed to (conform the given transformations). Of all the paths that end on such a note, we save the most probable one for each of those notes. There are only a few ways tot construct such a path.

First we can extend any optimal path that ends on note $n-1$ with just a 'non-transformation, this is always a possibility and will not violate any constraint. Second, for a given transformation $\mathcal{F}$, we can extend an optimal path of length $n-1$ that already ended on this transformation, with the same transformation. Third, for a given transformation $\mathcal{F}$, we can extend any optimal path that ends on note $(n-ML)$, with applying this transformation f on the ML next notes.

These three options capture all different ways an optimal path can be constructed. For the algorithm to work correctly, all the paths that are constructed or saved as optimal paths should also be 'valid' paths. This means that at any point in time all saved paths that consist of partially transformed parts, should have these parts transformed over a minimum length of ML notes with the same transformation.

To be able to keep up with all the bookkeeping, all the optimal paths for each transformation (also the 'non-transformation'), for each of the last ML last notes and for each of the notes these last ML notes can be transformed to, the optimal and valid path that ends on that position on that note with that transformation is saved. The same is done with the probabilities, which eases the computation of the probabilities of the different paths. Because the probabilities are stored separately, it is not necessary to recompute the RPK-probabilities by going over the whole stored path.

5.3 Performace and memory complexity

5.3.1 Performance

The performance time of this algorithm is linearly dependent on the amount of notes (AN). Because, if all other parameters are constant, the amount of work is constant for each timestep, and the amount of timesteps is linearly dependent on AN.

The speed of the algorithm is quadratically dependent on the amount of transformations (AT). The amount of work done in each time step is linearly dependent on AT and the work for each transformation again is linearly dependent on AT.

At last the algorithm is dependent on $ML \times (AN - ML)$. Since the amount of work in each time step to compute the extension of length ML to a path of length $(n - ML)$ in step n, is linearly dependent on ML. The amount of times such a path has to be calculated though is linearly dependent on $(AN - ML)$.

This gives a total time complexity for the algorithm of:

$$\underline{\text{Time:}}\ O(AN \times ML \times (AN - ML) \times AT^2)$$

5.3.2 Memory usage

The amount of memory used during the execution of the algorithm is linearly dependent on the amount of notes in the original melody. Since for all different transformations and possible notes that can be reached via transformation, a single optimal path will be saved. This length of this path is, of course, linearly dependent on the length of the original path.

The memory usage is also linearly dependent on the minimum transformation length. The minimum transformation length leads to the amount of optimal paths of past notes that have to be stored for each transformation. This amount is linearly dependent on ML.

At last, the memory usage is also linearly dependent on the amount of transformations given to the algorithm to use. For each of these transformations an array is made to store all the optimal paths that end on using this specific transformation.

This gives a total memory complexity for the algorithm of:

$$\underline{\text{Memory:}}\ O(AN \times ML \times AT)$$

6 Experiments and Results

The RPK-model that was proposed as a verifier for the musical transformations also has some disadvantages. One of which is that for each note in the musical piece, to compute its probability, the product of the three probability values for that note has to be normalized against all other possible notes for that position. This creates a lot of extra work just to be able to normalize at every note of the melody. An experiment was conducted to find a computationally more interesting predictor for this probability of the RPK-model.

Also, a few experiments have been carried out concerning the algorithm for combining transformations. In these experiments the influence of three parameters of the algorithm (minimum transformation length, the amount of iterations of the algorithm on the melody and the amount of different transformations) on the efficiency of the algorithm is checked.

It is interesting to know which parameters can deliver results that have a score that is still close to the original. Therefore in all the experiments the score of the melody that is found by the algorithm is compared to the score of the original melody as well as the score of the melody that is most probable in the given key. The closer the score stays to the original the better. If the score quickly converges in the direction of the theoretically optimal solution, then the parameter that is tested might, when overly used, lead to results that have too consonance high scores. This is not ideal since these melodies are often not that interesting, and on average contain a lot of the same notes.

When, in the results of the experiments the (average) probability of a musical piece is mentioned, this mean the average probability of a note in the musical piece. Which technically is the score that the RPK-model gives to the melody.

6.1 Predictor for RPK-model score

In this experiment, the combination of two different parameters that can be evaluated much more quickly than the RPK-score are tested on how their score for a musical piece compares tot that of the RPK-model itself.

The first parameter is the average distance in semitones between consecutive notes in the musical piece. One could assume that a smaller average distance leads to a higher probability of the melody. The second parameter is a distance of the distribution of notes in the melody, to the distribution of the notes in the whole of the Essencorpus. This distance is modeled as the Bhattacharyya distance [7].

For 100 musical pieces of the Essencorpus, two different values are calculated. The first one is the absolute value of the consonance score(logarithm of the probability) calculated by the RPK-model. The second one is the product of the average distance between successive notes and the Bhattacharyya distance of the note distribution to the average note distribution. The results of this experiment are shown in figure 6.

The results of this experiment show that there is a correlation between these two metrics. A higher value for the product of the average distance and the Bhattacharyya score,

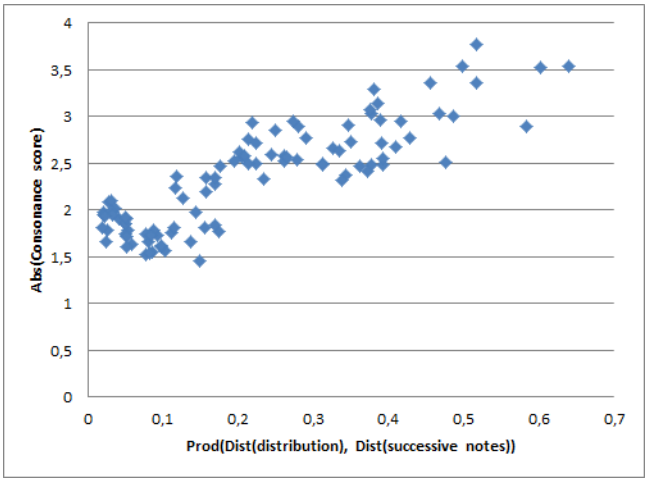

Fig. 6. Product of average distance between successive notes and Bhattacharyya distance of note distribution to average distribution on x-axis. Absolute value of the consonance score on the y-axis.

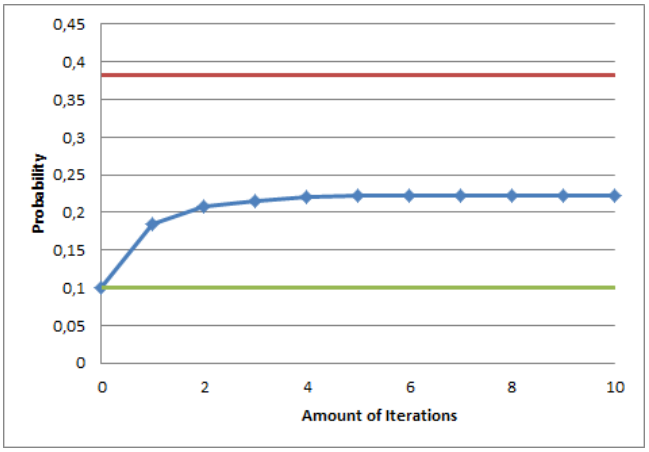

Fig. 7. Average probability of transformed melody after different amounts of iterations in blue. Average theoretical best probability in red. Average probability of the original melody in green.

on average, leads to a higher absolute value for the consonance score and vice versa. This means that for computationally very intensive programs this product metric might be used as a first predictor for the real RPK-value of a musical piece.

6.2 Amount of iterations of the algorithm

In this experiment, the influence of the amount of iterations of the algorithm on the probability of the resulting melody is checked. In this case the algorithm is successively applied a certain number of times to the result of the precious iteration. In this experiment, only the transformation described in table 1 can be used by the algorithm. The minimum transformation length is set to 1.

Now for 100 pieces of the Essencorpus, The average probability of these pieces is computed. Also the average probability of the theoretically best piece in the key of each of those pieces is calculated. And against those two values, for an amount of iterations between 1 and 10 the average scores are compared. The results of this experiment are visible in figure 7.

The figure shows that most of the improvement is made in the first step. After more than five steps there isn't even an improvement any more but at that point the probability of the transformed piece is still closer to that of the original piece than that of the theoretically best one. The reason for this is that in each iteration only the same transformation is available for the algorithm, so the amount of options the algorithm has is small and doesn't change over time in this experiment. This way, most of the improvement is logically in the first step of the algorithm and since the amount of options is limited, it is difficult to reach a score that is close to the theoretically best one.

6.3 Amount of different transformations

This experiment will test the influence of the amount of different transformations available to the algorithm on the probability of the resulting transformed melody.

Diff (mod 8)	0	1	2	3	4	5	6	7
Addition 1	5	-4	1	-3	1	1	2	3
Addition 2	1	3	4	-5	-1	6	5	-1
Addition 3	1	4	5	-3	2	-1	1	0
Addition 4	4	6	-2	4	2	6	-4	2
Addition 5	-3	-2	3	-1	4	-3	2	-4

Table 2. Transformations used in experiment 6.3.

during execution, the algorithm can use the transformations mentioned in table 2. Only one iteration of the algorithm is performed. The algorithms is executed for 5 different values of the amount of available transformations. First the algorithm only has transformation 1 available. Then the algorithm has transformation 1 and 2 that it can use, etc.. The experiment is carried out on 50 pieces of the Essencorpus and the average probability values are calculated for all 5 different values for the amount of transformations.

The results of this experiment are visualised in figure 8. This time the probability of the resulting piece keeps growing for a higher number of available transformations. More so than was the case with a higher number of iterations. This can be explained because the algorithm has more different options for each note and therefore can transform each note to more different target notes that on average lie closer to the optimal solution.

6.4 Minimum transformation length

In a last experiment the influence of the minimum transformation length on the probability of the transformed melody is measured. Now the algorithm can only transform a part of the original melody if that part is at least as long as the minimum transformation length and if every note of that part of the melody will be transformed via the same transfor-

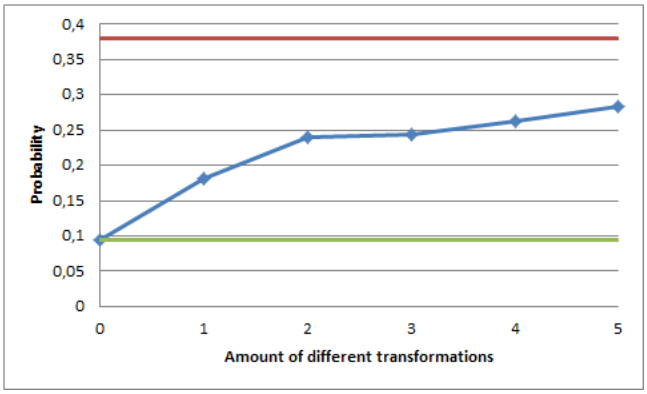

Fig. 8. Average probability of transformed melody for different amounts of available transformations in blue. Average theoretical best probability in red. Average probability of the original melody in green.

Diff (mod 8)	0	1	2	3	4	5	6	7
Addition 1	5	-4	1	-3	1	1	2	3
Addition 2	1	3	4	-5	-1	6	5	-1

Table 3. Transformations used in experiment 6.4.

mation.

During this experiment, the algorithm has two different transformations available which are described in table 3. Only one iteration of the algorithm will be performed each time. For 50 pieces of the Essencorpus, the average probability after transformation is measured for ten different values of the minimum transformation length parameter (1 to 10). This value is then again compared to the average theoretical best probability and the average probability of the original musical piece.

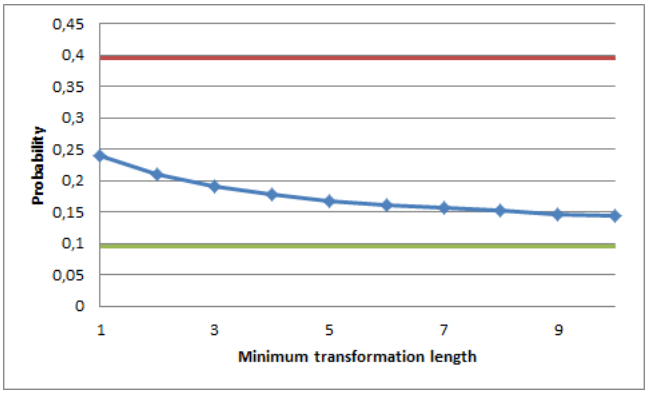

Fig. 9. Average probability of transformed melody for different amounts of minimum transformation lengths in blue. Average theoretical best probability in red. Average probability of the original melody in green.

The results of this experiment are shown in figure 9. Now the average probability of the transformed melody lowers with higher values for the minimum transformation length. This is logical since a higher value for the minimum transformation length constraints the problem more. This result is interesting since choosing a higher value for the minimum transformation length leads to a melody which has an average probability that is relatively close to the average probability of the original piece. One downside is that this is partially the result because some parts of the piece will probably not be transformed. Using a higher amount of different transformations can help against this problem but then also the average probability of the transformed piece will be higher. Depending on what is valued most, the ML parameter can be set accordingly.

7 Conclusions

In this paper, a melodical transformation is described that can transform a given melody into a new one. To judge this type of transformations or a combination of them, the RPK-model was introduced. This model serves as a reference model for musical pieces (of which only the melody is considered, not the rhythm). The model grants a probability to the melody based on the three described parameters that are important in the melodic aspect of music.

Because the RPK-model has some computational disadvantages, which are located in a normalization step for the probability of each note, a predictor has been constructed. This predictor is computationally more interesting and led to promising results which makes it a candidate to use as when a quick prediction of the PRK-probability is necessary for a relatively large piece of music.

Lastly, an algorithm was described which combines different transformations to transform an original melody into a new one with highest reachable probability. In experiments it became clear that, to get resulting melodies that have either a high probability or a probability close to the original, the different parameters of the model can tweaked to influence the average probability into the desired direction.

References

[1] Kroger, P., 2012. *Music for Geeks and Nerds*.
[2] Pattyn, T., 2015. “Op zoek naar inspiratie: Transformatie en evaluatie van muziek”. Master’s thesis, KU Leuven, Belgium.
[3] of Oxford Text Archive, U. Essen corpus of german folksong melodies. http://ota.ox.ac.uk/desc/1038.
[4] Musicxml. http://www.musicxml.com/.
[5] Temperley, D., 2007. *Music and Probability*. The MIT Press, Cambridge, Massachusetts.
[6] A. Lew, H. M., 2006. *Dynamic Programming*. Springer.
[7] Thacker, N. A., Aherne, F. J., and Rockett, P. I., 1998. The bhattacharyya metric as an absolute similarity measure for frequency coded data.