

Melody Characterization by a Genetic Fuzzy System

Pedro J. Ponce de León*, David Rizo*, Rafael Ramirez†, José M. Iñesta*

*Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante, Spain

†Music Technology Group, Universitat Pompeu-Fabra, Barcelona, Spain

Abstract—We present preliminary work on automatic human-readable melody characterization. In order to obtain such a characterization, we (1) extract a set of statistical descriptors from the tracks in a dataset of MIDI files, (2) apply a rule induction algorithm to obtain a set of (crisp) classification rules for melody track identification, and (3) automatically transform the crisp rules into fuzzy rules by applying a genetic algorithm to generate the membership functions for the rule attributes. Some results are presented and discussed.

I. INTRODUCTION

Melody is a somewhat elusive musical term that often refers to a central part of a music piece that catches most of the listener's attention, and which the rest of music parts are subordinated to. This is one of many definitions that can be found in many places, particularly music theory manuals. However, these are all formal but subjective definitions given by humans. The goal in this work is to automatically obtain an objective and human friendly characterization of what it is considered to be a melody.

The identification of a melody track is relevant for a number of applications like melody matching [1], motif extraction from score databases, or extracting melodic ringtones from MIDI files. In this work we approach the problem of automatically building a model that characterizes melody tracks. Such a model is tested in experiments on finding a melody track in a MIDI file. The melody model is a set of human-readable fuzzy rules automatically induced from a corpora of MIDI files by using statistical properties of the musical content.

To our best knowledge, the automatic description of a melody has not been tackled as a main objective in the literature. The most similar problem to the automatic melody definition is that of finding a melody line from a polyphonic source. This problem has been approached mainly for three different objectives and with different understandings of what a melody is. The first objective is the extraction of the melody from a polyphonic audio source. For this task it is important to describe the melody in order to leave out those notes that are not candidates to belong to the melody line[2]. In the second objective, a melody line (mainly monophonic) must be extracted from a symbolic polyphonic source where no notion of *track* is used [3]. The last objective is to select one track containing the melody from a list of input tracks of symbolic polyphonic

music (e.g. MIDI). Ghias et al. [1] built a system to process MIDI files extracting a sort of melodic line using simple heuristics. Tang et al. [4] presented a work where the aim was to propose candidate melody tracks, given a MIDI file. They take decisions based on single features derived from informal assumptions about what a melody track may be. Madsen and Widmer [5] try to solve the problem by the use of several combination of the entropies of different melody properties like pitch classes, intervals, and IOI.

A. What's a melody?

Before focusing on the machine learning methodology to extract automatically the characterization of a *melody*, the musical concept of melody needs to be reviewed.

Melody is a concept that has been given many definitions, all of them complementary. The variability of the descriptions can give an idea on the difficulty of the task to extract a description automatically.

From the music theory point of view, Ernst Toch [6] defines it as “a succession of different pitch sounds brighten up by the rhythm”. He also writes “a melody is a sound sequence with different pitches, in opposition to its simultaneous audition that constitutes what is named as chord”. He distinguishes also the term “melody” from the term “theme”.

A music dictionary [7] defines melody as: “a combination of a pitch series and a rhythm having a clearly defined shape”.

An informal survey was carried out where the subjects were asked to answer the question *What is a melody?*. Both musicians and non-musicians took part in the survey. The following list is a *compendium* of shared melody traits found in answers gathered on that survey:

- (finite) succession of notes
- *cantabile* pitch range
- monophonic
- lead part
- identifies/characterices the piece, song
- unity
- diversity
- contains repeating patterns
- often linked to text
- done by humans
- understandable, memorizable by humans

The music theory literature lacks the same amount of works about melody than can be found about counterpoint, harmony, or "form" [8]. Besides, the concept of melody is dependant on the genre or the cultural convention. The most interesting studies about melody have appeared in recent years, mainly influenced by new emerging models like generative grammars [9], artificial intelligence [10], and Gestalt and cognitive psychology [11]. All these works place effort on understand the melody in order to generate it automatically.

The types of tracks and descriptions of *melody* versus *accompaniment* is posed in [8]. The author distinguishes:

- *compound* melodies where there is only a melodic line where some notes are principal, and others tend to accompany, being this case the most frequent in unaccompanied string music.
- *self-accompanying* melodies, where some pitches pertain both to the thematic idea and to the harmonic (or rhythmic) support
- *submerged* melodies consigned to inner voices
- *roving* melodies, in which the theme migrates from part to part
- *distributed* melodies, in which the defining notes are divided between parts and the prototype cannot be isolated in a single part.

From the audio processing community, several definitions can be found about what a melody is. Maybe, the most general definition is that of Kim et al. [12]: "melody is an auditory object that emerges from a series of transformations along the six dimensions: pitch, tempo, timbre, loudness, spatial location, and reverberan environment".

Gómez et al. [13] gave a list of mid and low-level features to describe melodies:

- Melodic attributes derived from numerical analysis of pitch information: number of notes, tessitura, interval distribution, melodic profile, melodic density.
- Melodic attributes derived from musical analysis of the pitch data: key information, scale type information, cadence information.
- Melodic attributes derived from a structural analysis: motive analysis, repetitions, patterns location, phrase segmentation.

Another attempt to describe a melody can be found in [14]. In that book, Temperley proposes a model of melody perception based on three principles:

- Melodies tend to remain within a narrow pitch range.
- Note-to-note intervals within a melody tend to be small.
- Notes tend to conform to a key profile (a distribution) that depends on the key.

All these different properties a melody should have can be a reference to compare the automatic results.

The rest of the paper is organized as follows: first, the methodology used in this work is presented. Second, the experimentation framework is outlined. Next, results on several datasets for both crisp and fuzzy rule systems are discussed and compared to related work results. Finally, conclusions and further work are presented.

II. METHODOLOGY

The goal of this work is to obtain an human-readable characterization of MIDI tracks containing melody lines, against other kind of tracks. A fuzzy rule system has been chosen as the technique to obtain such a characterization. These fuzzy models should achieve good performance in discriminating melody tracks when compared to other non-fuzzy or non-rule based crisp models.

The methodology applied to obtain such fuzzy models is sketched as follows: first, MIDI tracks are described by a set of statistical features on several properties of the track content. This is presented in section II-A. Next section briefly describes different rule extraction methods used to obtain crisp rule systems that characterize melody tracks. Finally, these rule systems are then converted to fuzzy rule systems applying a fuzzyfication process to the input domain. This is discussed in section II-C.

A. MIDI track content description

MIDI track content is described by a collection of statistics on several properties of musical note streams, such as pitch, pitch interval or note duration, as well as track properties such as number of notes in the track, track duration, polyphony rate or occupation rate. As a result, MIDI tracks are represented by vectors $v \in \mathbb{R}^{34}$ of statistical values. This representation has been used to characterize melody tracks in previous works [15], [16].

This set of statistical descriptors is presented in Table I. The first column indicates the category being analyzed, and the second one shows the kind of statistics describing properties from that category. The third column indicates the range of the descriptor¹.

Four features were designed to describe the track as a whole and fifteen to describe particular aspects of its content. For the latter descriptors, both normalized and non-normalized versions have been computed. Only non-normalized ones are displayed in table I. Normalized descriptors are defined in [0, 1] and computed using the formula

$$(v_i - \min) / (\max - \min)$$

where v_i is the descriptor value to be normalized corresponding to the i -th track, and \min and \max are, respectively, the minimum and maximum values for this descriptor for all the tracks of the target midifile. This allows to represent these properties proportionally

¹ $[x..y]$ denotes integer domains and $[x, y]$ denotes real domains.

TABLE I
MIDI TRACK DESCRIPTORS

Category	Descriptors	Domain
Track info.	Normalized duration	[0, 1]
	Number of notes	[0 .. +∞[
	Occupation rate	[0, 1]
	Polyphony rate	[0, 1]
Pitch	Highest	[0 .. 127]
	Lowest	[0 .. 127]
	Mean	[0, 127]
	Standard deviation	[0, +∞[
Pitch intervals	Number of distinct intv.	[0 .. 127]
	Largest	[0 .. 127]
	Smallest	[0 .. 127]
	Mean	[0, 127]
	Mode	[0 .. 127]
	Standard deviation	[0, +∞[
Note durations	Longest	[0, +∞[
	Shortest	[0, +∞[
	Mean	[0, +∞[
	Standard deviation	[0, +∞[
Syncopation	No. of syncopated notes	[0 .. +∞[
Class	IsMelody	{true, false}

to other tracks in the same file, using non-dimensional values. This way, a total number of $4 + 15 \times 2 = 34$ descriptors were initially computed for each track.

The track information descriptors are normalized duration (using the same scheme as above), number of notes, occupation rate (proportion of the track length occupied by notes), and the polyphony rate (the ratio between the number of ticks in the track where two or more notes are active simultaneously and the track duration in ticks).

Pitch descriptors are measured using MIDI pitch values. The maximum possible MIDI pitch is 127 (pitch G_8) and the minimum is 0 (pitch C_{-2}).

The interval descriptors summarize information about the difference in pitch between consecutive notes. Absolute pitch interval values are computed.

Finally, note duration descriptors are computed in terms of beats, so they are independent from the MIDI file resolution. *Syncopations* are notes that start at some place between beats (usually in the middle) and extend across them.

B. A rule system for melody characterization

In this work, a rule system obtained using the RIPPER algorithm [17] is used as the basis to induce a fuzzy rule system. Briefly, the RIPPER constructs a rule set RS by considering each class, from the less prevalent one to the more frequent one. It builds RS until the description length (DL) of the rule set and examples is 64 bits greater than the smallest DL met so far, or there are no positive examples, or the error rate $\geq 50\%$. Rules are constructed by greedily adding antecedents to the rule until the rule is perfect (i.e. 100% value of each attribute and selects the condition with highest information gain (for details see [17])). We applied the RIPPER algorithm and obtained a rule

system from the SMALL dataset (see section III), so it is called the RIPPER-SMALL rule system. Table II shows the rules in this system. Note that only 13 out of 34 initial statistical descriptors have been selected by the algorithm to characterize melody tracks. Figures about this rule system performance are presented in section V.

TABLE II
RIPPER-SMALL (CRISP) RULES.

Name	Rule
R1	if (AvgPitch ≥ 65.0) and (TrackOccupationRate ≥ 0.51) and (AvgAbsInterval ≤ 3.64) and (TrackNumNotes ≥ 253) then IsMelody=true
R2	if (AvgPitch ≥ 62.6) and (TrackOccupationRate ≥ 0.42) and (TrackPolyphonyRate ≤ 0.21) and (NormalizedDistinctIntervals ≥ 1) then IsMelody=true
R3	if (AvgPitch ≥ 65.4) and (TrackNumNotes ≥ 284) and (ShortestNormalizedDuration ≤ 0.001) and (ShortestDuration ≥ 0.02) and (NormalizedDistinctIntervals ≥ 1) then IsMelody=true
R4	if (AvgAbsInterval ≤ 2.72) and (TrackSyncopation ≥ 16) and (AvgPitch ≥ 60.5) and (TrackOccupationRate ≥ 0.42) and (StdDeviationPitch ≤ 5.0) then IsMelody=true
R5	if (AvgAbsInterval ≤ 3.87) and (TrackSyncopation ≥ 24) and (LowestNormalizedPitch ≥ 0.14) and (DistinctIntervals ≥ 25) and (TrackNormalizedDuration ≥ 0.95) then IsMelody=true
R6	if (AvgAbsInterval ≤ 2.44) and (TrackNumNotes ≥ 130) and (AvgPitch ≥ 55.2) and (TrackOccupationRate ≥ 0.31) and (TrackPolyphonyRate ≤ 0.001) then IsMelody=true

C. From crisp to fuzzy rule system

Although informative, this rule system is not easily readable or even understandable at first sight, at least for people as musicians or musicologists. Also, being *melody* such a vague concept, the authors find that a fuzzy description of melody would be more sensible in the imprecise domain of music characterization.

In order to produce such a fuzzy description, a fuzzyfication process is applied to a crisp rule system, such the one presented in Table II.

Two basic steps must be carried out for the fuzzyfication of the crisp rule system. First, the data representation must be fuzzified. That is, numerical input and output values must be converted to fuzzy terms. Second, the rules themselves must be translated into fuzzy rules, substituting linguistic terms for numerical boundaries.

D. Fuzzyfying attributes

As stated above, a MIDI track is described by a set of statistical descriptors (called attributes from herein). The very first step of the attribute fuzzyfication process is to define the domain for every attribute. Most attributes have a finite domain. For practical application of the fuzzification method, infinite domains should be converted to finite domains. Appropriate upper and lower bounds are so defined for these domains.

In order to fuzzify crisp attributes (statistical descriptors), linguistic terms (such as *low*, *average*, or *high*) for every attribute domain are defined. Then the shape of the fuzzy set associated with each linguistic term is selected and, finally, the value of each fuzzy set parameter within the attribute domain is set.

Fuzzyfication of numerical attributes usually involves the participation of an human expert who provides domain knowledge for every attribute. The expert usually takes into consideration the distribution of values for an attribute in a reference data collection, as well as any other information available to her.

Our approach in this paper is to replace the human expert by a genetic algorithm (GA) which, given the linguistic term definitions for each attribute, automatically learns the fuzzy set parameters. Such combination of a fuzzy system with a genetic algorithm is known as a *genetic fuzzy system* [18].

In order to select the number of linguistic terms per attribute, a number of different crisp rule systems have been induced by different algorithms from the SMALL dataset. The presence of each attribute in those rule systems has been accounted for. Five terms have been assigned to most frequently used attributes. Three terms have been assigned to the rest of attributes. Table III shows these linguistic terms for attributes used in the RIPPER-SMALL crisp rule system.

TABLE III
FUZZY LINGUISTIC TERMS

Attribute	Linguistic terms
TrackNormalizedDuration	<i>Shortest, Average, Largest</i>
TrackNumNotes	<i>low, average, high</i>
TrackOccupationRate	<i>void, low, average, high, full</i>
TrackPolyphonyRate	<i>none, low, average, high, all</i>
LowestNormalizedPitch	<i>low, average, high</i>
AvgPitch	<i>veryLow, low, average, high, veryHigh</i>
StdDeviationPitch	<i>low, average, high</i>
DistinctIntervals	<i>few, average, alot</i>
NormalizedDistinctIntv.	<i>lowest, average, highest</i>
AvgAbsInterval	<i>unison, second, third, fourth, high</i>
ShortestDuration	<i>low, average, high</i>
ShortestNormalizedDur.	<i>shortest, average, longest</i>
TrackSyncopation	<i>few, average, alot</i>

Every linguistic term has a fuzzy set or membership function associated to it. This is a probability function from the attribute crisp input domain to the range $[0, 1]$ that, for every possible attribute crisp value, outputs the probability for this value to be named with that specific linguistic term. Figure 1 shows an example.

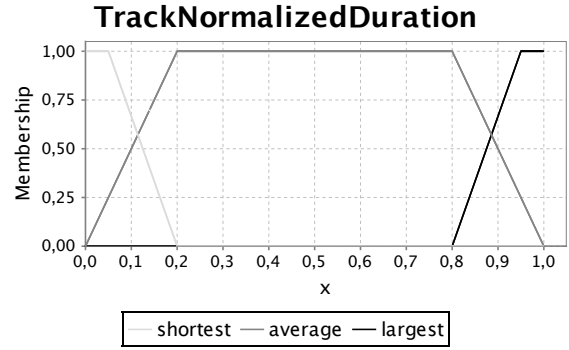


Fig. 1. Fuzzy set example for attribute *TrackNormalizedDuration*

For efficiency reasons, the shape for a fuzzy set in this work is restricted to be either trapezoidal or triangular, being the latter a special case of the former. Each fuzzy set is modeled by four points, corresponding to the extreme points of the *core (prototype)* and *support* of a fuzzy set, as depicted in Fig. 2. The support of a fuzzy set defines the range of the input domain where the fuzzy set membership probability is not zero. These fuzzy set parameters would be inferred from data by the GA.

The objective for the genetic fuzzy system presented here is to optimize fuzzy set parameters for every attribute in a fuzzy rule system. This optimization process is guided by a fitness function that, given a reference fuzzy rule system, tests potential solutions against a reference dataset.

1) *Fuzzy set representation scheme*: An individual's chromosome encodes all attributes of the fuzzy rule system. This means to encode fuzzy sets associated with linguistic terms for every attribute. The fuzzy set support is considered the most important part of a fuzzy set, while its shape is considered a subjective and application-dependent issue. The fuzzy set core is defined as a function of its support. So, the only fuzzy set parameters we need to optimize are the support points of each fuzzy set for every attribute. Figure 3a shows how an attribute domain is partitioned in overlapping fuzzy partitions, each corresponding to a fuzzy set. Let X be such attribute domain, we define

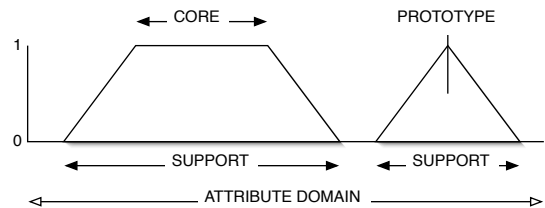


Fig. 2. Fuzzy set parts

a fuzzy partition of X as

$$X^i = [x_L^i, x_R^i], X^i \subset X, \quad 1 \leq i \leq m \quad (1)$$

where x_L^i and x_R^i are the *left* and *right support points* of fuzzy set i , respectively. m is the number of fuzzy sets for the attribute. Partitions are defined so that $X = \bigcup X^i$, that is, every input value belong to at least one partition. We also force the overlapping between adjacent partitions i and $i+1$ to be not void:

$$Z^{i,i+1} = X^i \cap X^{i+1} = [x_L^{i+1}, x_R^i] \neq \emptyset \quad (2)$$

Given these definitions, the set of parameters to optimize for a given attribute is

$$\Theta = \{x_L^1, x_L^2, x_R^1, \dots, x_L^m, x_R^{m-1}, x_R^m\} \quad (3)$$

In order to have a uniform GA representation for every attribute, their domains are normalized to range $[0, 1]$, so every parameter is a value in that range. For the sake of simplicity, let express Θ as

$$\Theta = \{p_0, p_1, p_2, \dots, p_{2m-1}\} \quad (4)$$

From the partitioning scheme definition, it follows that $p_0 = x_L^1 = 0$, so we can drop this first parameter. In order to make Θ suitable to crossover and mutation operations, a relative parameter representation scheme is used in the GA. Such scheme is defined as follows

$$\theta = \{p_1, r_2, r_3, \dots, r_{2m-1}\} \quad (5)$$

where $r_i = p_i - p_{i-1}$. Figure 4 depicts the representation scheme used in the GA. Note that

$$Z^{i,i+1} = r_{2i}, \quad 1 \leq i < m$$

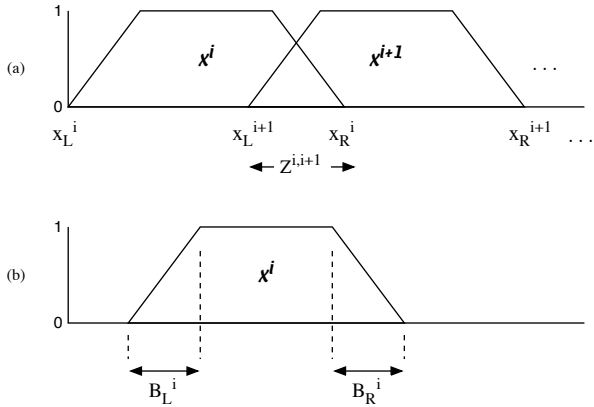


Fig. 3. (a) Fuzzy set partitions overlapping. (b) Boundaries of a fuzzy set.

Once the support points are known, left and right boundaries (figure 3b) are set. They are restricted to lie inside the overlapping section of their corresponding partition. For right boundaries,

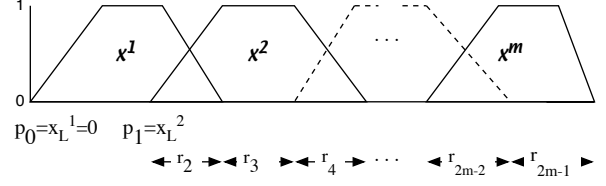


Fig. 4. Representation scheme of fuzzy sets.

$$0 \leq B_R^i \leq Z^{i,i+1} = r_{2i}, \quad 1 \leq i < m$$

and $0 \leq B_R^m \leq r_{2m-1}$.

For left boundaries

$$0 \leq B_L^i \leq Z^{i-1,i} = r_{2i-2}, \quad 1 < i \leq m$$

and $0 \leq B_L^1 \leq p_1$. This ensures that the core of a fuzzy set is equal or greater than zero.

E. Fitness function

The fitness function for the GA consists of testing each individual in a fuzzy inference system (FIS) using the fuzzy rule system discussed in section II-F on a reference dataset (see section III). The better the performance of the rule system, given the fuzzy set definitions provided by the individual's chromosome, the better the individual's score. This is possible because rule fuzzification is a process independent from fuzzy set definition. Several metrics can be used to measure the performance of the FIS. In this work two different metrics have been tested: 1) number of hits and 2) F measure (geometric mean of precision and recall of class *IsMelody=true*).

F. Crisp rule system fuzzyfication

The goal of the rule system presented above is to identify MIDI tracks as melody or non-melody tracks. The objective of this work is to convert this crisp rule system, which perform fairly well for the task at hand, in a human-friendly description of melody tracks.

The final step in this method is to fuzzify the rule system. Antecedents of the form $(x \odot v)$ where \odot is an inequality operator, are translated into one or more antecedents of the form $(x \text{ IS } T)$, where T is a linguistic term defined for attribute x . The value v partitions the attribute domain in two subsets, and the direction of the inequality guides the selection of the fuzzy terms to be included in fuzzy antecedents.

In the present work, the crisp RIPPER-SMALL rule system (section II-B) has been fuzzified in order to present a proof of concept of the methodology applied. A disjunctive fuzzy rule set is then obtained. Table IX shows fuzzy rules corresponding to those shown in section II-B.

III. EXPERIMENTS

A. Datasets

Table IV shows information about all the datasets used to test the fuzzy rule system. They consist of MIDI files, where melody tracks were tagged with a special string in their track name. These tracks have been manually or automatically tagged, depending on the dataset. The automatic tagging process is based on a dictionary of frequent melody track names. The manual tagging was carried out by experts on the different music genres present in the datasets.

The SMALL reference dataset has been used to obtain the crisp rule system from which the fuzzy rule system has been derived. It is also the dataset used in the GA fitness function to test the performance of potential solutions. The rest of datasets are used for testing the system: RWC-G [19], RWC-P [20], LARGE and AJP are all multi-genre datasets of academic, popular, rock and jazz music, among more than ten genres.

TABLE IV
DATASETS.

Dataset	Tracks	Songs	Melody tracks
SMALL	2775	600	554
LARGE	15168	2513	2337
RWC-P	801	75	74
RWC-G	311	48	44
AJP	3732	762	760

B. FIS Optimization Experiment setup

Our genetic fuzzy system has six free parameters that let configure different experiment setups. Table V shows these parameters and the values chosen to build a set of experiments. Parameter values have been restricted to at most three different values. This allows the use of an orthogonal array [21] to explore the free parameter space. Briefly, an orthogonal array of level L , strength n and M runs ensures that, given any n parameters with L values each, all their respective values will appear in combination in an equal number of experiments. This avoids testing all possible combinations, while remaining confident that every combination of n parameters appears at least once in some experiment. In this work, an orthogonal array of strength 2 and 18 runs has been used to setup experiments.

TABLE V
FIS OPTIMIZATION SETUP PARAMETERS

Experiment parameter	Values
GA population size	100, 500, 1000
GA no. of generations	100, 500, 1000
GA mutation ratio	none, 0.05, 0.1
GA selection strategy ²	Best one, Best 10%, Best 20%
GA fitness metric	Hit count, F-measure
Defuzzification threshold ³	0.5, 0.6, 0.7

IV. FUZZY INFERENCE SYSTEM OPTIMIZATION RESULTS

Table VI shows the performance of evolved FIS versus the RIPPER-SMALL crisp rule system performance. Average results from the eighteen experiments performed are shown. Figures in parenthesis are standard deviations. Precision, recall and F-measure are computed for the class 'IsMelody'. Also, the performance of the best evolved FIS are presented. Note that the best evolved FIS performance is very close to that from the crisp rule system. The definition of fuzzy sets for the best evolved FIS, as well as other information and examples on this work can be found on the web at the following address: <http://grfia.dlsi.ua.es/cm/worklines/smc08>.

TABLE VI
BEST AND AVERAGE PERFORMANCE OF EVOLVED FIS V. CRISP
RIPPER-SMALL RULE SYSTEM PERFORMANCE.

Rule system	Precision	Recall	F	Error rate
<i>crisp</i>	0.89	0.87	0.88	0.05
Best FIS	0.81	0.83	0.82	0.06
Avg. FIS	0.80 (.03)	0.77 (.09)	0.78 (.05)	0.08 (.01)

V. RESULTS ON TEST DATASETS.

Table VII presents results from applying both the crisp rule system and the best evolved FIS to test datasets. In these test experiments, a track is classified as a melody track if it fires at least one rule with probability greater than 0.5. Otherwise, the track is classified as non-melody.

TABLE VII
MELODY TRACK CLASSIFICATION RESULTS.

Dataset	Precision	Recall	F	Error rate
LARGE (<i>crisp</i>)	0.79	0.80	0.80	0.06
LARGE (<i>fuzzy</i>)	0.70	0.74	0.72	0.09
RWC-P (<i>crisp</i>)	0.95	0.80	0.87	0.02
RWC-P (<i>fuzzy</i>)	0.51	0.64	0.57	0.09
RWC-G (<i>crisp</i>)	0.54	0.77	0.64	0.13
RWC-G (<i>fuzzy</i>)	0.43	0.43	0.43	0.16
AJP (<i>crisp</i>)	0.88	0.89	0.88	0.05
AJP (<i>fuzzy</i>)	0.88	0.83	0.86	0.06

As the results show, the fuzzyfied rule system performs poorly than the original crisp rule system. However, the recall is consistently better for the fuzzy classifier. It follows that most errors are false positives, that is, some non-melody tracks are classified as melody tracks.

VI. COMPARISON OF CRISP AND FUZZY SYSTEMS ON SOME EXAMPLES

This section discuss several example characterization of melody and non-melody tracks. The example excerpts are shown in Table VIII in the appendix. The words 'Crisp' and 'Fuzzy' under the music systems indicate which rules from the crisp and fuzzy systems were fired, respectively. The fuzzy rule system used with

these examples was the best evolved FIS using the rules in Table IX.

The first three tracks are melody tracks that were correctly identified by the fuzzy rule system. Crisp rules failed at characterizing the first one. This first track almost fulfills rule *R2*, except that it has not the largest pitch interval variety (its *NormalizedDistinctIntervals* value is .85), as the last condition of the rule imposes. The next three tracks in Table VIII are non-melody tracks correctly identified by both rule systems (neither track fire any rule). The last two examples are tracks were both rule systems disagree. The melody track from *Satin Doll* is unusual in the sense that is supposed to be played by a vibraphone (a polyphonic instrument), has one chorus of improvisation and the melody reprise (which is the part shown in the example) is played in a polyphonic *closed chord* style. The last example is a piano accompaniment part, played in *arpeggiato* style, which the fuzzy rules incorrectly identified as a melody track. This track almost fired crisp rule *R6*, except for the last condition of the rule, because its *TrackPolyphonyRate* value is .097. This is a clear example of why a fuzzy version of a crisp rule fires while the crisp rule don't. The value is accepted by the fuzzy rule as linguistic term *none* for the *TrackPolyphonyRate* attribute. This is because it lies into the support of the fuzzy set corresponding to that term. See figure 5 for some fuzzy set examples from the best evolved FIS.

VII. CONCLUSIONS AND FURTHER WORK

A fuzzy rule system for melody track characterization in MIDI files has been outlined. The somewhat poorer performance of the fuzzy rule system could be improved by rising the probability threshold for firing a fuzzy rule. Also, enforcing more than one fuzzy rule to be fired could help improve the results. A smarter way of fuzzyfying rules (using information theory measures) could be applied.

ACKNOWLEDGMENTS

The authors want to thank Pablo Cingolani and the rest of contributors to the *jFuzzyLogic* package (<http://jfuzzylogic.sourceforge.net>) we used to implement our fuzzy rule systems. Also we want to thank Klaus Meffert et al. as major contributors to the *Java Genetic Algorithms Package, JGAP* (<http://jgap.sourceforge.net>), the one we used to implement our GA experiments. Last, but not least, thanks to the people at University of Waikato behind the *weka* project (<http://www.cs.waikato.ac.nz/ml/weka/>), used to build our crisp rule models.

This work is supported by the projects: GV06/166 and CICYT TIN2006-14932-C02, partially supported by EU ERDF.







REFERENCES

- [1] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith, "Query by humming: Musical information retrieval in an audio database," in *Proc. of 3rd ACM Int. Conf. Multimedia*, 1995, pp. 231-236.
- [2] J. Eggink and G. J. Brown, "Extracting melody lines from complex audio," in *ISMIR*, 2004.
- [3] I.Karydis, A.Nanopoulos, A.Papadopoulos, E. Cambouropoulos, and Y. Manolopoulos, "Horizontal and vertical integration/segregation in auditory streaming: a voice separation algorithm for symbolic musical data," in *Proceedings 4th Sound and Music Computing Conference (SMC'2007)*, Lefkada, 2007. [Online]. Available: <http://delab.csd.auth.gr/papers/SMC07knpcm.pdf>
- [4] M. Tang, C. L. Yip, and B. Kao, "Selection of melody lines for music databases," in *Proceedings of Annual Int. Computer Software and Applications Conf. COMPSAC*, 2000, pp. 243-248.
- [5] S. T. Madsen and G. Widmer, "Towards a computational model of melody identification in polyphonic music," in *20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, 2007, pp. 459-464.
- [6] E. Toch, *La melodia (translation of 'Melodielehre', 1923)*. Span-Press Universitaria, 1997.
- [7] S. Sadie and G. Grove, *The New Grove Dictionary of Music and Musicians*. Macmillan, 1084.
- [8] E. Selfridge-Field, *Conceptual and representational issues in melodic comparison*, ser. Computing in Musicology. Cambridge, Massachusetts: MIT Press, 1998, vol. 11, pp. 3-64.
- [9] M.Baroni, *Proposal for a Grammar of Melody: The Bach Chorales*. Les Presses de l'Université de Montréal, 1978.
- [10] D. Cope, *Experiments in Musical Intelligence*. New York, NY, USA: Cambridge University Press, 1996, vol. 2, no. 1.
- [11] E.Narmour, *The Analysis and Cognition of Basic Melodic Structures*. University Of Chicago Press, 1990.
- [12] Y. E. Kim, W. Chai, R. Garcia, and B. Vercoe, "Analysis of a contour-based representation for melody," in *ISMIR*, 2000.
- [13] A. E. Gomez, A. Klapuri and B.Meudic, "Melody description and extraction in the context of music content processing," *Journal of New Music Research (JNMR)*, vol. 32-1, 2003.
- [14] D. Temperley, *The Cognition of Basic Musical Structures*. The MIT Press, 2004.
- [15] P. J. Ponce de León, D. Rizo, and J. M. Iñesta, "Towards a human-friendly melody characterization by automatically induced rules," in *Proceedings of the 8th International Conference on Music Information Retrieval, S. Dixon, D. Brainbridge, and R. Typke, Eds.* Vienna: Austrian Computer Society, September 2007, pp. 437-440.
- [16] D. Rizo, P. J. Ponce de León, C. Pérez-Sancho, A. Pertusa, and J. M. Iñesta, "A pattern recognition approach for melody track selection in midi files," in *Proc. of the 7th Int. Symp. on Music Information Retrieval ISMIR 2006*, T. A. Dannenberg R., Lemström K., Ed., Victoria, Canada, 2006, pp. 61-66, ISBN: 1-55058-349-2.
- [17] W. W. Cohen, "Fast effective rule induction," *Machine Learning: Proceedings of the Twelfth International Conference*, 1995.
- [18] O. Cordon and F. Herrera, "A general study on genetic fuzzy systems," in *Genetic Algorithms in Engineering and Computer Science*, J. Smith, Ed. John Wiley & Sons, 1995, ch. 3, pp. 33-57.
- [19] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Music genre database and musical instrument sound database." in *ISMIR*, 2003.
- [20] —, "RWC music database: Popular, classical and jazz music databases." in *ISMIR*, 2002.
- [21] A. Hedayat, N. J. A. Sloane, and J. Stufken, *Orthogonal Arrays: Theory and Applications*, 1st ed. Springer, 1999.

APPENDIX

TABLE VIII

TRACK CLASSIFICATION EXAMPLES.

<p>True positive examples</p> <p><i>Air In F, Watermusic</i>, Handel (<i>Baroque</i>)</p> <p>Melody</p>  <p>Crisp: - Fuzzy: FR6</p>
<p><i>There Is No Greater Love</i>, I. Jones (<i>pre-Bop Jazz</i>)</p> <p>Melody</p>  <p>Crisp: R2, R5 Fuzzy: FR4, FR6</p>
<p>True negative examples</p> <p><i>Air In F, Watermusic</i></p> <p>Bass</p>  <p>Crisp: - Fuzzy: -</p>
<p><i>There Is No Greater Love</i></p> <p>Piano (accompaniment)</p>  <p>Crisp: - Fuzzy: -</p>
<p>False negative example</p> <p><i>Satin Doll</i>, D. Ellington (<i>pre-Bop Jazz</i>)</p> <p>Melody</p>  <p>Crisp: R2 Fuzzy: -</p>
<p>False positive example</p> <p><i>Sonata no. 3 K545, 2nd Mov.</i>, W.A. Mozart (<i>Classicism</i>)</p> <p>Piano (accompaniment)</p>  <p>Crisp: - Fuzzy: FR6</p>

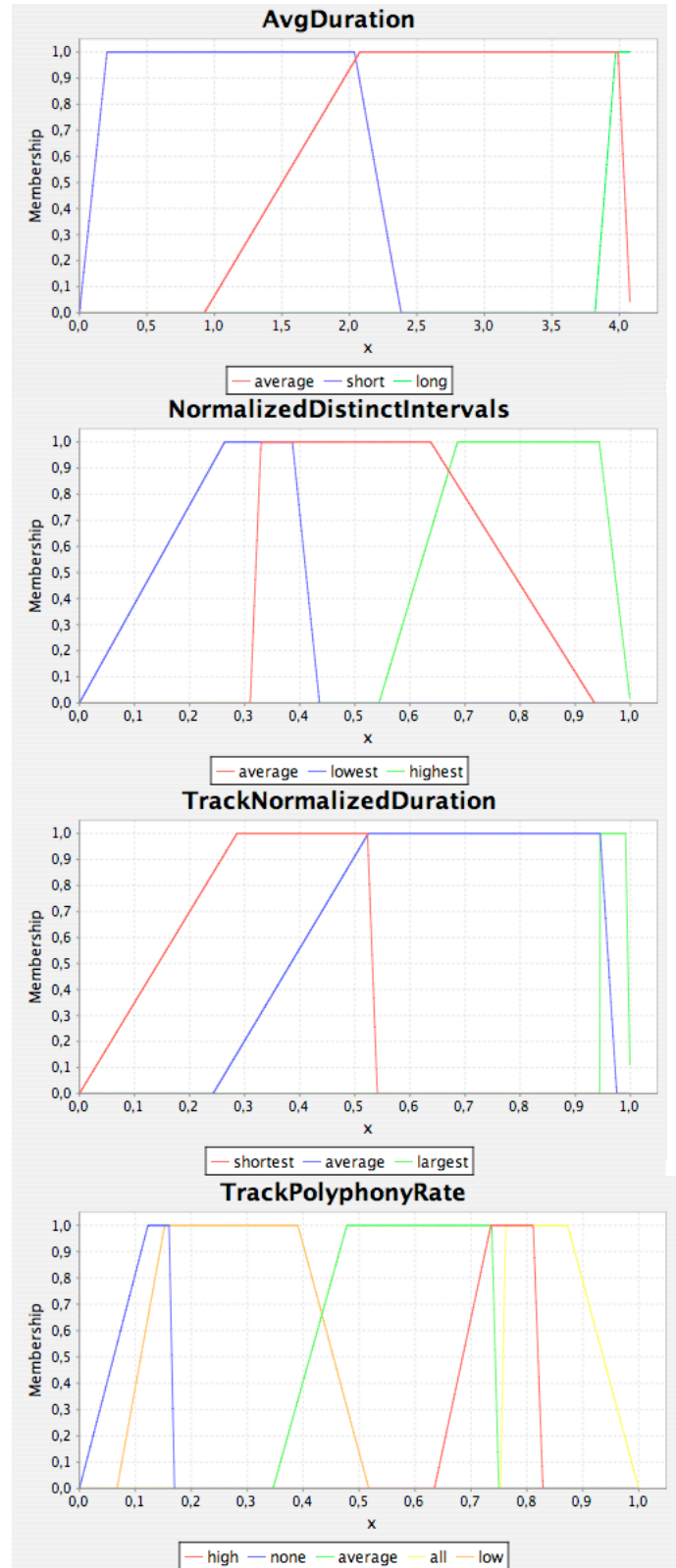


Fig. 5. Fuzzy set examples from the best evolved fuzzy rule system.

TABLE IX
FUZZY RULES EQUIVALENT TO THOSE SHOWN IN TABLE II

Name	Rule	Name	Rule
FR1	IF (AvgPitch IS high OR AvgPitch IS veryHigh) AND (TrackOccupationRate IS NOT void) AND (TrackOccupationRate IS NOT low) AND (AvgAbsInterval IS NOT fourth) AND (AvgAbsInterval IS NOT high) AND (TrackNumNotes IS high) THEN IsMelody IS true	FR2	IF (AvgPitch IS high OR AvgPitch IS veryHigh) AND (TrackOccupationRate IS NOT void) AND (TrackOccupationRate IS NOT low) AND (TrackPolyphonyRate IS NOT average) AND (TrackPolyphonyRate IS NOT high) AND (TrackPolyphonyRate IS NOT all) AND (NormalizedDistinctIntervals IS highest) THEN IsMelody IS true
FR3	IF (AvgPitch IS high OR AvgPitch IS veryHigh) AND (TrackNumNotes IS high) AND (LowestNormalizedDuration IS shortest) AND (ShortestDuration IS NOT low) AND (NormalizedDistinctIntervals IS highest) THEN IsMelody IS true	FR4	IF (AvgPitch IS high OR AvgPitch IS veryHigh) AND (TrackOccupationRate IS NOT void) AND (TrackOccupationRate IS NOT low) AND (AvgAbsInterval IS NOT third) AND (AvgAbsInterval IS NOT fourth) AND (AvgAbsInterval IS NOT high) AND (TrackSyncopation IS NOT few) AND (StdDeviationPitch IS NOT high) THEN IsMelody IS true
FR5	IF (AvgAbsInterval IS NOT fourth) AND (AvgAbsInterval IS NOT high) AND (TrackSyncopation IS alot) AND (LowestNormalizedPitch IS NOT low) AND (DistinctIntervals IS alot) AND (TrackNormalizedDuration IS largest) THEN IsMelody IS true	FR6	IF (AvgPitch IS NOT veryLow) AND (AvgPitch IS NOT low) AND (TrackOccupationRate IS NOT void) AND (TrackOccupationRate IS NOT low) AND (AvgAbsInterval IS NOT third) AND (AvgAbsInterval IS NOT fourth) AND (AvgAbsInterval IS NOT high) AND (TrackPolyphonyRate IS none) AND (TrackNumNotes IS NOT low) THEN IsMelody IS true