The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.88 e-ISSN: 2251-7545

Compiling Hierarchical Dependency Graph for Large-Span Musical Expressive Feature Analysis Using Multi-Scaling Probabilistic Graphical Models

Ren Gang, Xuchen Yang, Zhe Wen, Dave Headlam, Mark F. Bocko
Dept. of Electrical and Computer Engineering, Edmund A. Hajim School of Engineering and Applied Sciences, Univ. of Rochester
Dept. of Music Theory, Eastman School of Music, Univ. of Rochester
Rochester, NY 14627, USA
g.ren@rochester.edu, xuchenyang@rochester.edu,zhe.wen@rochester.edu,dheadlam@esm.rochester.edu, mark.bocko@rochester.edu

***Abstract*—Music performance conveys profound music understanding and artistic expression in musical sound. These performance-related dimensions can be extracted from audio and encoded as musical expressive features, which is based on a high-dimensional sequential data structure. In this paper we propose a structure learning based method using probabilistic graphical models that obtains a hierarchical dependency graph from musical expressive features. The hierarchical dependency graph we proposed serves as an intuitive visualization interface of the internal dependency patterns within feature data series and helps music scholars identify in-depthconceptual structures.**

Keywords- knowledge engineering; feature analysis; probabilistic graphical model; music performance analysis

I. INTRODUCTION

Music performance conveys profound music understanding and artistic expression in musical sound. These performance-related dimensions can be extracted from audio using various signal processing algorithms [1]. In this feature extraction process, performance-related information is encoded as musical expressive features, which is based on a high-dimensional sequential data structure. Currently the analysis of music performance features still poseschallenges in both music theoretical study and multimedia datasetanalysis.One of key challenges in the analysis ofmusical expressive feature lies in itscomplex dependency structure. In this paper we introduce the hierarchical dependency graph as a visualization tool for the internal structures of musical expressive features. These hierarchical dependency graphs learned then propose interesting musical patterns to music scholars to facilitate in-depth manual analysis.

Graphical analysis has been a long-time tradition in diverse areas of music study since antiquity [2]. In fact, the music score is developed from a graphical representation of melodic contour (as a graph, instead of the more "symbolic" score we use today). In modern music theoretic analysis, graphical representations are extensively applied. For example, Heinrich Schenker developed the analytic graph (which is subsequently called Schenkerian analysis graph, see [3] for examples) to encodea hierarchical relation of music objects. Model formulations of geometrical music models [4] ypically apply a graphical representation to describe the complex roles played by elementary music events. The common point in these graphical analyses is that the graph represents a level of abstraction where themusical connections can be observed at a higher level: the graph links provide connections to music entities that are not connected otherwise. On the other hand, an absence of graphical link in adjacent musical features helps to segment music patterns. In this paper we extend these concepts of music graphical analysis to the analysis tasks of musical expressive features. Instead of encoding manually formed structures, we apply structure learning algorithms to discover graphical patterns from data series. The proposed method serves as a human-data-interaction tool that helps music scholars find interesting data patterns.Here the proposed framework serves as a formalization interface that transforms the feature series (numbers) into a knowledge representation format that is more suitable for human-knowledge dissemination (graphs).

Related works on computational analysis of music performance are summarized in [5,6]. Compare to these systems, our proposed methods provide additional modeling flexibility by introduce probabilistic graphical model as pattern analysis tool. Related manual analysis methods are also compared in [5,6]. Currently human analysis capabilities clearly excel machine analysis results in various areas of music theoretic research as detailed in [5]. Thus many music scholars repel computational approaches [5] and insist on an all-manual approach.Other scholars point out the importance of integrating human listening capabilities since a matured computational "listening" model is not available [6]. Obviously these manual analysis procedures also have important limitations such as:

- Analysis results biased from subjective interpretations
- Large-scale patterns are not easily identified by manual inspection.
- The data volume can be overwhelming;a moderate task usually takes decades to study and analysis.

The aim of this work is thus to resolve this manual-computational analysis gap by introducing computational elements into existing manual analysis procedures.In designing this framework we also consider the compatibility to existing working environment of music scholars to ensure an effective human-date interaction. Extracting Musical Expressive Features from Performance Audio

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.88 e-ISSN: 2251-7545

When a matching music score of the performance audio is available, the performance-level features are obtained by comparing the score with the audio. First we perform the score-audio alignment algorithm as in [1]. This alignment algorithm maps the score music event to the time-frequency locations of performance audio using dynamic time warping, which optimally aligns the variation patterns of pitch and timing features obtained from score and audio. The score pitch is converted to a fundamental frequency using a temperament system which is derived from a reference frequency point $\bar{f}_R$ with symbolic pitch value p_R as:

$$\bar{f}_m = \text{tpa}(\bar{f}_R, p_R; p_m) \quad (1)$$

where p_m is the symbolic pitch value of frequency point $\bar{f}_m$, tpa() indicates a temperament function. For equal temperament scale $\bar{f}_m$ could be calculated as:

$$\bar{f}_m = \text{tpa}_e(\bar{f}_R, p_R; p_m) = 2^{\frac{p_m - p_R}{12}} \cdot \bar{f}_R \quad (2)$$

The logarithmic value of $\bar{f}_m$ is:

$$12 \cdot \log_2 \bar{f}_m = 12 \cdot \log_2 \bar{f}_R + p_m - p_R \quad (3)$$

Here p_m and p_R is specified in MIDI value. Since human frequency discernment is most acute at mid-frequency region, the reference point $[p_R: \bar{f}_R]$ could be selected at this frequency region. In our implementation an initial reference point is selected as [69:440Hz]. Then we shift the frequency reference point in 160 small steps within 1/6 of a semitone interval and find the best reference frequency point $\bar{f}_R + \Delta\bar{f}_R^*$ where a F0 alignment cost is minimized. The F0 alignment cost here is a weighted sum of frequency mis-alignments d_l between the audio F0 and the score pitch according to the temperament grid as:

$$C(\Delta\bar{f}_R) = \sum_{l=1}^{L} \eta(f_l) \left| d_l(\Delta\bar{f}_R) \right| \quad (4)$$

Here $\left| d_l(\Delta\bar{f}_R) \right|$ denotes the frequency distance of l th alignment event when the reference point shift is $\Delta\bar{f}_R$. The small variation $\Delta\bar{f}_R$ is incorporated into the reference pitch to shift the temperament grid so the d_l values are changing with $\Delta\bar{f}_R$. The weights $\eta(f_l)$ are based on the frequency discrimination model as introduced in [1] and f_l is the F0 of of lth alignment event. Larger $\eta(f_l)$s are assigned for higher frequencies (especially for frequencies higher than 2kHz) since human can discern frequency better at these frequencies. Using the optimal reference frequency point $\bar{f}_R + \Delta\bar{f}_R^*$ where the alignment result is minimized, we can calculate the pitch deviation of each music event by comparing the audio pitch and the score pitch. The pitch deviation (P) of music event l in the units of cents (A cent represents 1/100 of a semitone) is calculated as:

$$\Delta p_m = 1200 \cdot \log_2 \frac{d_l(\Delta\bar{f}_R^*)}{\bar{f}_l} \quad (5)$$

The score-audio alignment algorithm also provides score-aided music event segmentation functionalities. For monophonic music the segmentation results provide the onset and offset of each music events. For polyphonic music the segmentation results further group sonic partials into instrument tracks. For monophonic music or an instrument track of polyphonic music the segmentation result is represented as $\{[e_s, t_s] | s \in 1, \dots S\}$, where e_s denotes a music event prescribed by the music score and t_s denotes its onset time location. The expressive timing features are obtained by comparing the score timing and the performance timing. The time deviation (T) of music event e_s is calculated as the normalized difference between audio onset timing $t(e_s)$ and the interpolated score timing $\hat{t}(e_s)$:

$$F_T(e_s) = \frac{t(e_{s+1}) - t(e_s)}{\hat{t}(e_{s+1}) - \hat{t}(e_s)} \quad (6)$$

Here onset time deviation is normalized by the interpolated score note duration and the deviation value of previous notes is deduced. $t(e_{s+1})$ denotes the next onset location. $F_T(e_s)$ can be viewed as an indicator of the extension ($F_T(e_s) > 1$) or compression ($F_T(e_s) < 1$) of the audio segment of current notes. From different interpolation settings of score timing this method produces an expressive timing hierarchy. If the score timing interpolation is based on a long audio segment, macro-scale timing is obtained. We can then shorten the interpolation range to music phrase or individual meter for a micro-scale analysis. Other performance-level feature dimensions including loudness (L), timbre (B), articulation (A) and vibrato (V) could be obtained using the algorithms as in [1].These algorithms arebriefly summarized in Table 1. These feature values are quantized to three levels for subsequent analysis.

II. Compiling Hierarchical Dependency GraphUsing Multi-Scaling Probabilistic Graphical Models

A. *Discover Patterns from Template Samples*

Suppose the music expressive features are structured as a multi-dimensional data sequence as:

$$\boldsymbol{F} = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1t} & \cdots & f_{1T} \\ f_{21} & f_{22} & \cdots & f_{2t} & \cdots & f_{2T} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ f_{k1} & f_{k2} & \cdots & f_{kt} & \cdots & f_{kT} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ f_{K1} & f_{K2} & \cdots & f_{Kt} & \cdots & f_{KT} \end{bmatrix} \quad (7)$$

where $k = 1, \cdots, K$ is the feature dimension index, and $t = 1, \cdots, T$ is the temporal (time) index. The structure template is defined as a sub-matrix of $\boldsymbol{F}$and represented as :

$$\boldsymbol{S} = \begin{bmatrix} s_{11} & \cdots & s_{1Q} \\ \vdots & \ddots & \vdots \\ s_{P1} & \cdots & s_{PQ} \end{bmatrix} \quad (8)$$

When applied as a shifting template, the structure template $\boldsymbol{S}$ draws feature variables from data sequence$\boldsymbol{F}$uniformly.Specifically the sample values of a template at location (m, n) are represented as:

$$s_{ij}(m, n) = f_{i+m, j+n} \quad (9)$$

The resultant sampled template points $s_{ij}(m, n)$ is uniformly located at sampling points around shifting template location (m, n) with template size (P, Q). This sampled template serves as an instance for the template variables. An example of shifting templates is illustrated in Fig. 1(a), with $m = 0, \dots, 11; n = 0$.The template samples from multiple sample points are simply treated as i.i.d samples. Denoting these samples as instantiations $\boldsymbol{D} = \{D_1, \dots, D_N\}$, where $D_n = \boldsymbol{S}(m, n)$. In this setting we only shift the template at temporal (time) direction.

In this implementation the temporal length Q and feature depth Phave to be limited because the dependency graph based

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/iscse.v3.n3.88 e-ISSN: 2251-7545

Feature	**Acoustical Description**	**Musical Description**	**Feature Extraction Algorithms**	**Typical Value**
Pitch Deviation "P"	The difference between the performance frequency and the score specified frequency	Pitch Bending	(1) The fundamental frequency (F0) of an audio segment is detected using a pitch analysis algorithm. (2) When a score is available the score F0 is obtained from the symbolic score value and a temperament system; when only performance audio is available F0 is obtained from a quantization process. (3) The pitch deviation is calculated by comparing the audio F0,f and the score F0,p, The pitch deviation is calculated in units of cents (A cent represents 1/100 of a semitone.): $$\Delta p = 1200 \cdot log_2 \frac{f}{p}$$	-15 cents to +15 cents
Timing "T"	The time difference of musical events between the score and the audio.	Rubato	(1) The audio onset time is detected by applying threshold detection to the audio wave file envelope . The score timing is interpreted from an existing score or from rhythmic analysis and quantization of the performance. (2) The time deviation of onset n is calculated as the difference between audio onset timing $t(n)$ and the score timing $\hat{t}(n)$: $$\Delta t(n) = t(n) - \hat{t}(n)$$ (3) The onset time deviation is normalized by the interpolated score note duration [1]. as: $$F_T(n) = \frac{t(n+1) - t(n)}{\hat{t}(n+1) - \hat{t}(n)}$$	From 0.6 (compression) to 1.5 (extension)
Auditory Loudness "L"	The perceptual intensity of sound	Dynamic Level	(1) Calculate the strength of auditory response [1] of an audio segment based on its energy distribution in the frequency domain, using a computational auditory model. First the audio segment is partitioned into short analysis frames of 20ms. For each analysis frame $s(l)$, the auditory loudness can be approximately evaluated by summing up the auditory responses of individual frequency components as: $$v(l) = \sqrt{\sum_{k=1}^{K} \xi(k) S_M^2(l,k)}$$ where$\xi(k)$ is the frequency-response weighting obtained from a computational auditory model. $S_M(l,k)$is the STFT magnitude, which can also be calculated as the magnitude of the spectrum of a time frame centered at time point l. (2) Calculate the time average of auditory loudnesses of the short frames within the analysis range. Suppose the analysis range is composed of analysis frame 1 to L, the average loudness level of this audio section is: $$v(i) = \sqrt{\frac{1}{L}\sum_{l=1}^{L} v^2(l)}$$	A dynamic range of 30 dB. (depending on the normalization)
Timbre "B"	The pattern of the energy magnitude in the frequency domain	Sound color (bright vs. dark)	(1) Calculate the short time Fourier analysis $S_M(i,k)$, where k is the frequency bin index. $i$is the time frame index. (2) The timbre centroid is calculated as the "weight center" of the frequency spectrum of a analysis segment as: $$c(i) = \sum_{k=1}^{K} k S_M^2(i,k) \Big/ k_F \sum_{k=1}^{K} S_M^2(i,k)$$ Here the timbre centroid is normalized by the frequency bin index of fundamental sonic partial k_F. (3) Timbre width is defined as the frequency width $b(i)$ required to include a pre-defined portion η (with a typical value of 90%) of the total energy. Its integer part $b_I(i)$ can be calculated by adding in frequency components $S_M^2(i,1), \ldots, S_M^2(i, b_I(i)+1)$ until the threshold η is surpassed. We also add in a decimal part to improve the sensitivity of $b(i)$ as: $$b_D(i) = \frac{\eta \sum_{k=1}^{K} S_M^2(i,k) - \sum_{k=1}^{b_I(i)} S_M^2(i,k)}{S_M^2(i, b_I(i)+1)}$$ The timbre width is normalized using the fundamental frequency by: $$b'(i) = \frac{b_I(i) + b_D(i)}{k_F}$$	Timbre centroid from 1.2 to 4. Timbre width from 1.5 to 3.
Attack "A"	onset transient characteristics	hard / soft, sharp / dull	The attack feature is calculated as the ratio of the energy content of the first 1/3 of the note duration and the energy content in the latter 2/3's of the note duration.	from 0.5 to 3.
Vibrato "V"	The amplitude and frequency modulation inside a musical note	shallow/wide , rapid/slow	(1) Perform vibrato recognition using the algorithms as in [1]. (2) A band-pass filter is implemented to extract a single sonic partial as a quasi-monochromatic componentfrom the complex harmonic sound. (3) Amplitude and frequency modulation components are extracted from quasi-monochromatic components using analytic signal methods. (4) The *AM/FM modulation depth* (AMD/FMD) is calculated as the average peak/valley distance in the amplitude and frequency modulating components. AMD represents the "vibration" of the amplitude envelope. FMD is the average pitch deviation.	AMD from 0.1 to 0.4. FMD from 10 Hz to 40 Hz.

Table 1: Summary of the definitions and feature extraction algorithms for musical expressive features. For the feature dimensions of timbre and vibrato, multiple feature descriptors are implemented.

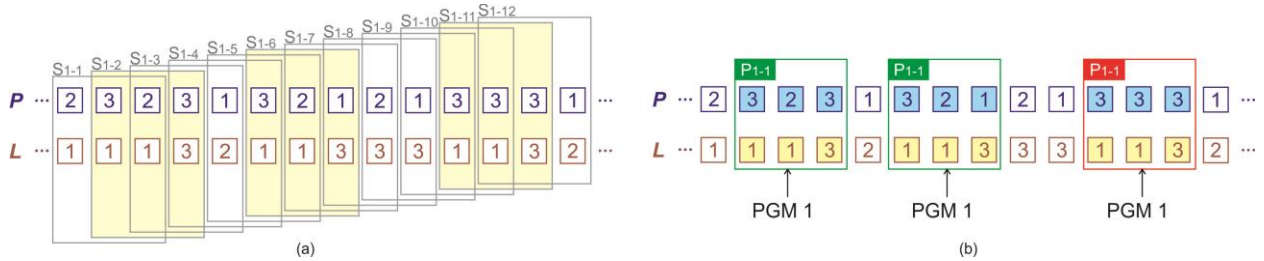

Figure 1. First level structure learning based on probabilistic graphical model (PGM) in the two feature dimensions of pitch deviation (P) and loudness (L). (a) template sampling results and (2) PGM Learning and the support regions of PGM model. The green box indicates support regions with high acceptance level of 'PGM 1", in this case these areas is a perfect pattern. The red box indicate a lower acceptance level of 'PGM 1' but still accepted as support region. In this case this area contains an imperfect pattern.

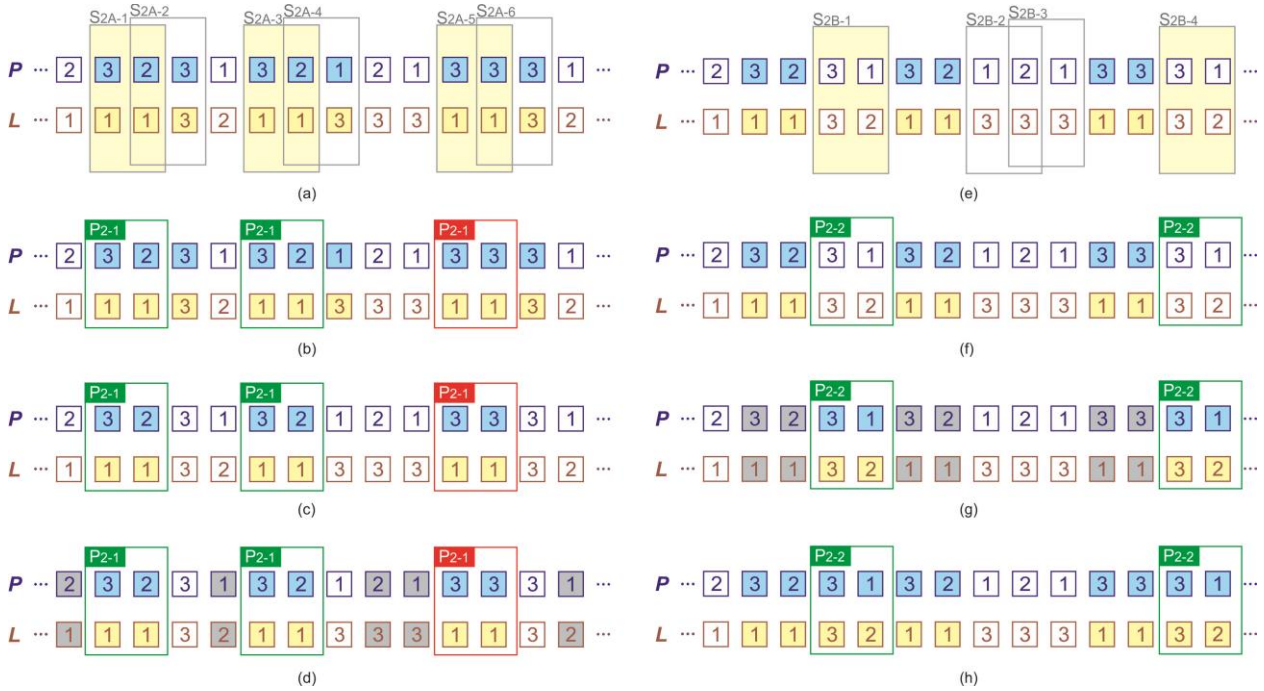

Figure 2. Learning hierarchical dependence graph in the deeper levels. The data region in the first level (Figure 1) is divided into pattern region (blue-yellow data squares) and non-pattern region (white data squares). In (a)-(d) smaller pattern in the pattern region is identified. This second level data region is then further split into pattern region (c) and non-pattern region (d). The grey-color square indicate the non-pattern region repeated from level 1 and will not be included here. In (e)-(h) smaller patterns in non-pattern region is identified. This second level data region is then split into the second-level pattern region (g) and non-pattern region. Note in (g) the gray-color squares indicate the pattern region identical in the first level and not included in this level.

to the number of variables in the template.The analysis method of large-span dependency over the template size is covered in Sec. III.B.

A heuristic search algorithmis applied to find the candidate PGM set $\boldsymbol{\mathcal{L}} = \{\mathbb{L}_1, \dots, \mathbb{L}_M\}$for the data $\boldsymbol{D}$.we compute the score for a PGM $\mathbb{L}$ as [7]:

$Score(\mathbb{L}: \boldsymbol{D}) = P(\boldsymbol{D}|\mathbb{L})$(10)

The PGM with the highest score in the candidate PGM set is selected as the model for data $\boldsymbol{D}$.

$\mathbb{L}^* = \text{augmax}_{\boldsymbol{\mathcal{L}}} P(\boldsymbol{D}|\mathbb{L})$ (11)

B. *Hierarchical Dependency Patterns*

Using the template learning methods in Sec. II we only obtain part of the dependency patterns within the structured data $\boldsymbol{D}$ because:

- The learned PGM model is just an approximation of the dependency structure data. Since it is not a complete description of data, other internal structures are also possible.

For example,the dependency pattern contained in $\boldsymbol{D}_1, \boldsymbol{D}_3, \boldsymbol{D}_5$ might be different from the patterns in $\boldsymbol{D}_2, \boldsymbol{D}_6, \boldsymbol{D}_{11}$. The optimal PGM $\mathbb{L}^*$ thus may only capture the pattern in$\boldsymbol{D}_2, \boldsymbol{D}_6, \boldsymbol{D}_{11}$part of all data instances $\boldsymbol{D}$. The PGM learning algorithm in Sec. III.A also does not guarantee to have captured all patterns within templates D_2, D_6, D_{11}.

To enable a more detailed analysis of the data instances $\boldsymbol{D}$, The optimal graphical model $\mathbb{L}^*$is then fit back to the templates by calculating the probability of the data given the optima model. Supposethe$i$-th instantiation is denoted as$\boldsymbol{D}_i = \{D_{i,1}, \dots, D_{i,N}\}$, where $\{D_{i,1}, \dots, D_{i,N}\}$ is a vector of the re-organized template

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3n3.88 e-ISSN: 2251-7545

sampling variables $s_{ij}(m,n)$. Thus $\boldsymbol{D}_i$ contains a feature

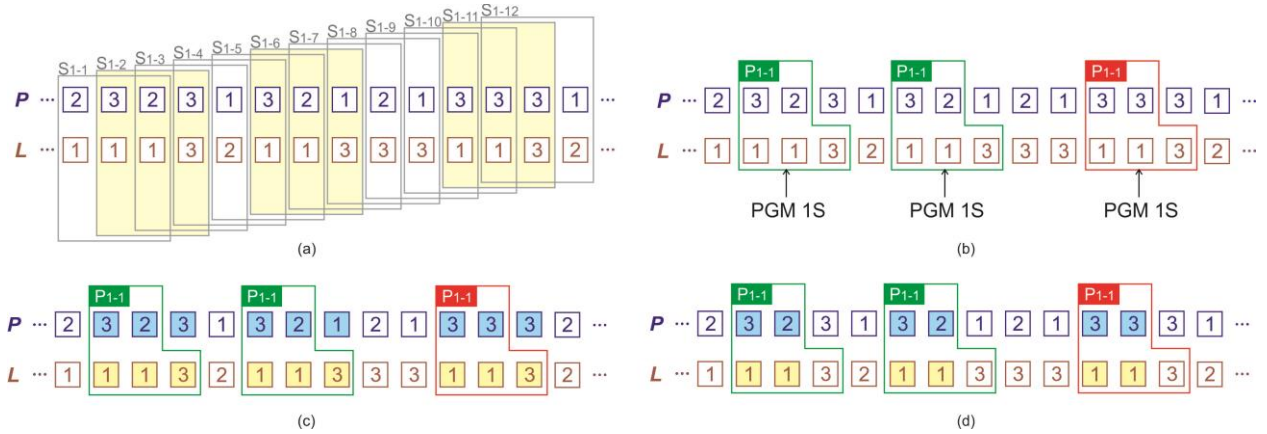

Figure 3. Applying link-strength methods to first level structure learning. (a) Template sampling results. (b) PGM Learning and the support regions of PGM model. Here the data point in the pattern only includes the nodes with high-strength links. The latent PGM model is only the significant parts of the PGM model learned from the data sequence in (a). (c) is the pattern region for the next level of PGM learning. (d) is the non-pattern region for the second level of PGM learning.

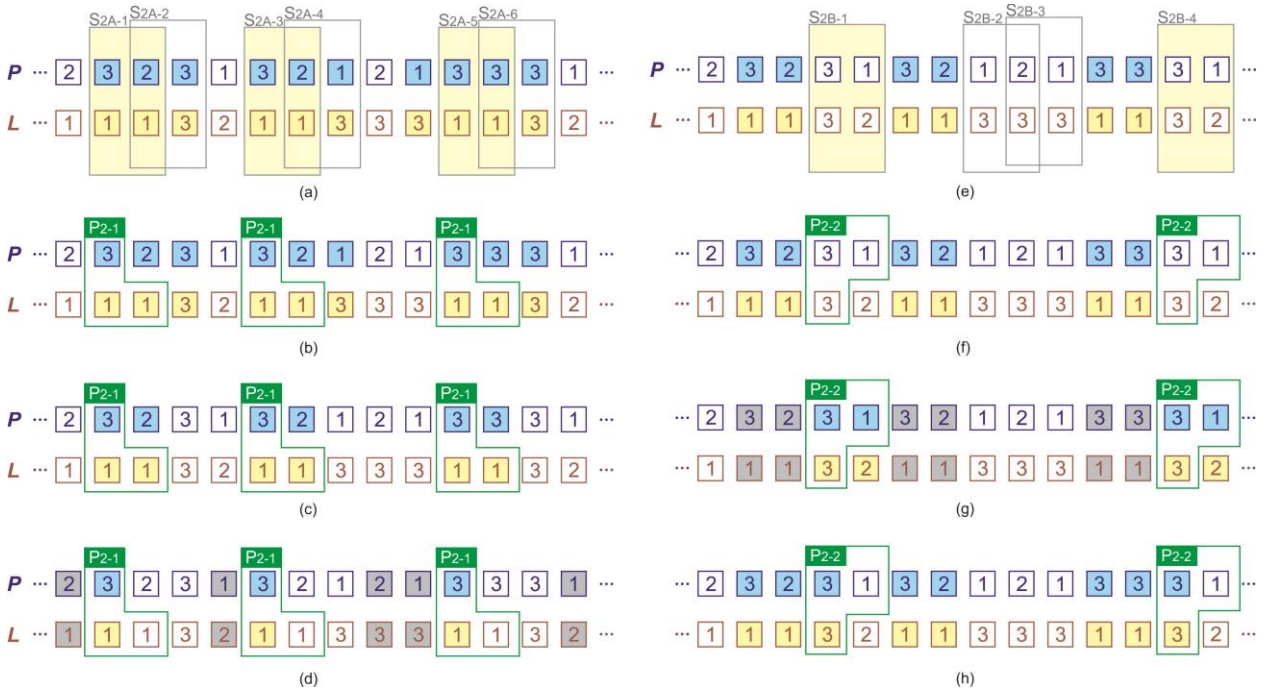

Figure 4. Learning hierarchical dependence graph in the deeper levels with link-strength option. The data region in the first level (Figure 3) is divided into pattern region (blue-yellow data squares) and non-pattern region (white data squares) and applies to second level of PGM learning. In (a)-(d) smaller pattern in the pattern region is identified and only the significant part of the dependency structure is kept. In (e)-(h) smaller patterns in non-pattern region is identified with the high-strength linked part circled. This second level data region is then further split into pattern region and non-pattern region as (b) >> (c)(d) and (f)>>(g)(h).

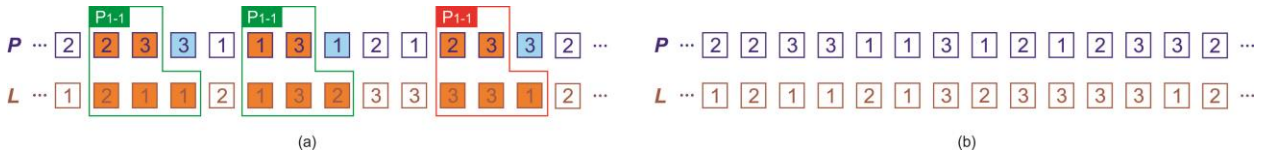

Figure 5. Simplified pattern masking method. The pattern region is replaced with random numbers (orange data squares in (a)). These random numbers erase the dependency structures in these data region. The updated sequence (b) is then applied to another round of PGM learning.

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.88 e-ISSN: 2251-7545

The acceptance level of $A(\boldsymbol{D}_i, \mathbb{L}^*)$ is calculated as the probability of observing $\boldsymbol{D}_i$ given the PGM model $\mathbb{L}^*$:

$$A(\boldsymbol{D}_i, \mathbb{L}^*) = \text{Log}P(\boldsymbol{D}_i|\mathbb{L}^*) \quad (12)$$

Here $P(\boldsymbol{D}_i|\mathbb{L}^*)$ is calculated using the chain rule as:

$$P(D_{i,1}, \dots, D_{i,N}|\mathbb{L}^*) = P(D_{i,1}|\mathbb{L}^*)P(D_{i,2}|D_{i,1}, \mathbb{L}^*) \cdot P(D_{i,3}|D_{i,1}, D_{i,2}, \mathbb{L}^*)P(D_{i,N}|D_{i,1}, \dots, D_{i,N-1}, \mathbb{L}^*) \quad (13)$$

$$\text{Log}P(D_{i,1}, \dots, D_{i,N}|\mathbb{L}^*) = \text{Log}P(D_{i,1}|\mathbb{L}^*) + \text{Log}P(D_{i,2}|D_{i,1}, \mathbb{L}^*) + \text{Log}P(D_{i,3}|D_{i,1}, D_{i,2}, \mathbb{L}^*) + \text{Log}P(D_{i,N}|D_{i,1}, \dots, D_{i,N-1}, \mathbb{L}^*) \quad (14)$$

In practical implementation, (17) can be calculated by appending evidence (observed nodes) and evaluate the marginal probability distributions. We first evaluate $P(D_{i,1}|\mathbb{L}^*)$ by use null evidence in the inference engineer, and evaluate the marginal distribution of $D_{i,1}$. $P(D_{i,2}|D_{i,1}, \mathbb{L}^*)$ is evaluated by add in evidence of a $D_{i,1}$ value, then evaluate the marginal distribution of $D_{i,2}$.

To discover addtional patterns within the data, we exclude the instances with high acceptance level where$A(\boldsymbol{D}_i, \mathbb{L}^*) > \delta$. These data instances with high acceptance level are defined as the support region of $\mathbb{L}^*$ here. In Fig. 1. The learned PGM pattern is matched to templates 'S1-1,6,11'. We call the matched parts of templates the pattern region of this level. The other part is called non-pattern region. The pattern part here provides a contour of large-span connections. This method of large-span analysis is more flexible than other similarity-based methods by admitting latent patterns instead of pattern in the original feature data.

The remaining instances in both pattern region and non-pattern regionare applied for another round of PGM learning as illustrated in Fig. 2. Here we also introduce the variable template size as in Fig. 2(a),(e) for both pattern region and non-pattern region at each PGM learning stage. Specifically we discover smaller-scale models from both the pattern part and the non-pattern part of the previous learning task. For pattern regions, the smaller patterns learned at the support region contains the more important part of a larger pattern.In the non-pattern part, we excluded the current model $\mathbb{L}^{1^*}$by excluding the data instances with high acceptance levels. The second round of PGM learning would find another model $\mathbb{L}^{2^*}$ and then its support region.The data region is then split into the pattern region and non-pattern region in this level.

In Fig. 3 and 4 we further apply link-strength methods [7] to improve the flexibility of learned dependency graph.This link-strength helps prioritize important dependency structures when forming a structure hierarchy. The masking procedure on original data is then based on the nodes with high-strength links as in Fig. 4. Fig. 5 further illustrates a simplified masking method that replaces the pattern region with random patterns.The pattern leaning process in Fig. 2 and 3 forms a hierarchical dependency graph as illustrated in Fig. 6. Here the template in Fig. 6(a) includes 3 consecutive feature values such as 'A', 'A+1' and 'A+2'. Then smaller pattern in (b) is then obtained from the pattern regionby applying templates including 2 consecutive feature values. These patterns form a hierarchical dependence graph and are proposed as candidate conceptual music models.

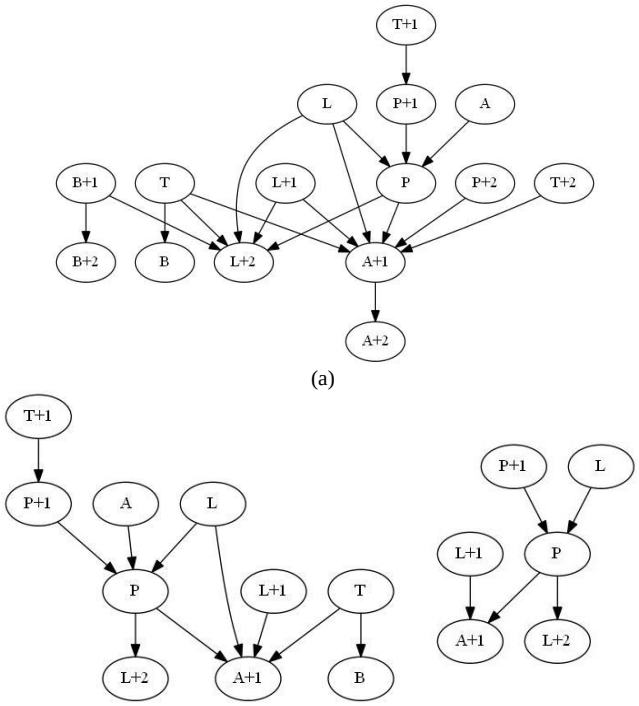

Figure 6. Examples of hierarchical dependency graph obtained from musical expressive features of 5 feature dimensions including pitch deviation (P), loudness (L), timing (T), timbre (B) and articulation (A). Each feature dimension includes feature values from three consecutive notes.

III. SUMMARY

In this paper, we introduce hierarchical dependency graph as a human-data interface for music expressive feature analysis. The hierarchical dependency graph we proposedproposes interesting musical patterns and providesaalgorithmic-enhanced working environment for music scholars.The long term goal of this research is to further improve the automation level and minimize user intervention. At current stage this method serves as an investigation tool for manual analysis, to speech up the analysis, scale up the analysis, and eases the procedure.

REFERENCES

[1] G.Ren, J. Lundberg, G. Bocko, D. Headlam, and M. F. Bocko, "What makes music musical? a framework for extracting performance expression and emotion in musical sound", Proceedings of IEEE Digital Signal Processing Workshop, Sedona, AZ, 4-7 Jan. 2011, pp. 301 – 306.

[2] M.E. Bonds, A History of Music in Western Culture, 3rd ed., Prentice Hall: Upper Saddle River, NJ, 2009, pp 2-15.

[3] B.M. Ayotte, Heinrich Schenker, a Giude to Research, Routledge: New York, NY, 2004, pp 5-39.

[4] D. Tymoczko, A Geometry of Music, Oxford University Press: New York, NY, 2011, pp 63-115.

[5] A. Gabrielsson, "Music Performance Research at the Millennium," Psychology of Music, Vol. 31, July 2003, pp. 221-272.

[6] G. Widmer; W. Goebl, Computational Models of Expressive Music Performance: The State Of The Art, Journal od New Music Research, Vol. 33, 2004, pp. 13-26.

Doi: 10.7321/jscse.v3.n3.88 e-ISSN: 2251-7545

[7] I. Ebert-Uphoff,LinkStrength Package for Bayesian network toolboxwww.dataonstage.com/BNT/PACKAGES/LinkStrength/index.htm .