A database linking piano and orchestral midi scores with application to automatic projective orchestration

Léopold Crestel, Philippe Esling, Lena Heng, Stephen Mcadams

A DATABASE LINKING PIANO AND ORCHESTRAL *MIDI* SCORES WITH APPLICATION TO AUTOMATIC PROJECTIVE ORCHESTRATION

Léopold Crestel[1] **Philippe Esling**[1]
Lena Heng[2] **Stephen McAdams**[2]
[1] Music Representations, IRCAM, Paris, France
[2] Schulich School of Music, McGill University, Montréal, Canada
leopold.crestel@ircam.fr

ABSTRACT

This article introduces the Projective Orchestral Database (*POD*), a collection of *MIDI* scores composed of pairs linking piano scores to their corresponding orchestrations. To the best of our knowledge, this is the first database of its kind, which performs piano or orchestral prediction, but more importantly which tries to learn the correlations between piano and orchestral scores. Hence, we also introduce the projective orchestration task, which consists in learning how to perform the automatic orchestration of a piano score. We show how this task can be addressed using learning methods and also provide methodological guidelines in order to properly use this database.

1. INTRODUCTION

Orchestration is the subtle art of writing musical pieces for the orchestra by combining the properties of various instruments in order to achieve a particular musical idea [11,23]. Among the variety of writing techniques for orchestra, we define as *projective orchestration* [8] the technique which consists in first writing a piano score and then orchestrating it (akin to a projection operation, as depicted in Figure 1). This technique has been used by classic composers for centuries. One such example is the orchestration by Maurice Ravel of *Pictures at an Exhibition*, a piano work written by Modest Mussorgsky. This paper introduces the first dataset of musical scores dedicated to projective orchestrations. It contains pairs of piano pieces associated with their orchestration written by famous composers. Hence, the purpose of this database is to offer a solid knowledge for studying the correlations involved in the transformation from a piano to an orchestral score.

The remainder of this paper is organized as follows. First, the motivations for a scientific investigation of orchestration are exposed (section 2). By reviewing the previous attempts, we highlight the specific need for a symbolic database of piano and corresponding orchestral scores. In an attempt to fill this gap, we built the *Projective Orchestral Database* (*POD*) and detail its structure in section 3. In section 4, the automatic projective orchestration task is proposed as an evaluation framework for automatic orchestration systems. We report our experiment with a set of learning-based models derived from the Restricted Boltzmann Machine [26] and introduce their performance in the previously defined evaluation framework. Finally, in section 5 we provide methodological guidelines and conclusions.

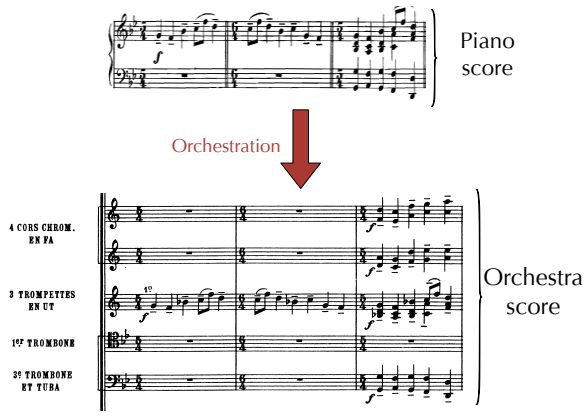

Figure 1. *Projective orchestration* of the first three bars of Modest Mussorgsky's piano piece *Pictures at an Exhibition* by Maurice Ravel. Piano notes are assigned to one or several instruments, possibly with doubling or harmonic enhancement.

2. A SCIENTIFIC INVESTIGATION OF ORCHESTRATION

Over the past centuries, several treatises have been written by renowned composers in an attempt to decipher some guiding rules in orchestration [11, 21, 23]. Even though they present a remarkable set of examples, none of them builds a systemic set of rules towards a comprehensive theory of orchestration. The reason behind this lack lies in the tremendous complexity that emerges from orchestral works. A large number of possible sounds can be created by combining the pitch and intensity ranges of each instru-

ments in a symphonic orchestra. Furthermore, during a performance, the sound produced by a mixture of instruments is also the result of highly non-linear acoustic effects. Finally, the way we perceive those sounds involves complex psychoacoustic phenomena [14, 16, 25]. It seems almost impossible for a human mind to grasp in its entirety the intertwined mechanisms of an orchestral rendering.

Hence, we believe that a thorough scientific investigation could help disentangle the multiple factors involved in orchestral works. This could provide a first step towards a greater understanding of this complex and widely uncharted discipline. Recently, major works have refined our understanding of the perceptual and cognitive mechanisms specifically involved when listening to instrumental mixtures [15, 22, 25]. Orchids, an advanced tool for assisting composers in the search of a particular sonic goal has been developed [8]. It relies on the multi-objective optimization of several spectro-temporal features such as those described in [20].

However, few attempts have been made to tackle a scientific exploration of orchestration based on the study of musical scores. Yet, symbolic representations implicitly convey high-level information about the spectral knowledge composers have exploited for timbre manipulations. In [6] a generative system for orchestral music is introduced. Given a certain style, the system is able to generate a melodic line and its accompaniment by a full symphonic orchestra. Their approach relies on a set of templates and hand-designed rules characteristic of different styles. [19] is a case study of how to automatically transfer the *Ode to joy* to different styles. Unfortunately, very few details are provided about the models used, but it is interesting to observe that different models are used for different styles. Automatic arrangement, which consists in reducing an orchestral score to a piano version that is can be played by a two-hand pianist, has been tackled in [10] and [24]. The proposed systems rely on an automatic analysis of the orchestral score in order to split it into structuring elements. Then, each element is assigned a role which determines whether it is played or discarded in the reduction. To the best of our knowledge, the inverse problem of automatically orchestrating a piano score has never been tackled. However, we believe that unknown mechanisms of orchestration could be revealed by observing how composers perform projective orchestration, which essentially consists in highlighting an existing harmonic, rhythmic and melodic structure of a piano piece through a timbral structure.

Even though symbolic data are generally regarded as a more compact representation than a raw signal in the computer music field, the number of pitch combinations that a symphonic orchestra can produce is extremely large. Hence, the manipulation of symbolic data still remains costly from a computational point of view. Even through computer analysis, an exhaustive investigation of all the possible combinations is not feasible. For that reason, the approaches found in the literature rely heavily on heuristics and hand-designed rules to limit the number of possible solutions and decrease the complexity. However, the recent advents in machine learning have brought techniques that can cope with the dimensionality involved with symbolic orchestral data. Besides, even if a wide range of orchestrations exist for a given piano score, all of them will share strong relations with the original piano score. Therefore, we make the assumption that projective orchestration might be a relatively simple and well-structured transformation lying in a complex high-dimensional space. Neural networks have precisely demonstrated a spectacular ability for extracting a structured lower-dimensional manifold from a high-dimensional entangled representation [13]. Hence, we believe that statistical tools are now powerful enough to lead a scientific investigation of projective orchestration based on symbolic data.

These statistical methods require an extensive amount of data, but there is no symbolic database dedicated to orchestration. This dataset is a first attempt to fill this gap by building a freely accessible symbolic database of piano scores and corresponding orchestrations.

3. DATASET

3.1 Structure of the Database

The database can be found on the companion website [1] of this article, along with statistics and Python code for reproducibility.

3.1.1 Organization

The Projective Orchestral Database (*POD*) contains 392 *MIDI* files. Those files are grouped in pairs containing a piano score and its orchestral version. Each pair is stored in a folder indexed by a number. The files have been collected from several free-access databases [1] or created by professional orchestration teachers.

3.1.2 Instrumentation

As the files gathered in the database have various origins, different instrument names were found under a variety of aliases and abbreviations. Hence, we provide a comma-separated value (*CSV*) file associated with each *MIDI* file in order to normalize the corresponding instrumentations. In these files, the track names of the *MIDI* files are linked to a normalized instrument name.

3.1.3 Metadata

For each folder, a CSV file with the name of the folder contains the relative path from the database root directory, the composer name and the piece name for the orchestral and piano works. A list of the composers present in the database can be found in table 1. It is important to note the imbalanced representativeness of composers in the database. It can be problematic in the learning context we investigate, because a kind of stylistic consistency is *a priori* necessary in order to extract a coherent set of rules. Picking a subset of the database would be one solution, but another possibility would be to add to the database this stylistic information and use it in a learning system.

[1] `https://qsdfo.github.io/LOP/database`

Composer	Number of piano files	Percentage piano frames	Number of orchestra files	Percentage orchestra frames
Arcadelt. Jacob			1	0.07
Arresti. Floriano	3	0.57		
Bach. Anna Magdalena	3	0.43		
Bach. Johann Sebastian	9	4.57	4	0.81
Banchieri. Adriano			1	0.32
Beethoven. Ludwig Van	1	0.60	38	42.28
Berlioz. Hector			1	0.14
Brahms. Johannes			3	0.28
Buxtehude. Dietrich	1	0.21		
Byrd. William	1	0.13		
Charpentier. Marc-Antoine			2	0.38
Chopin. Frederic			2	0.44
Clarke. Jeremiah			1	0.23
Debussy. Claude	1	0.59	6	0.90
Dvorak. Anton			6	2.42
Erlebach. Philipp Heinrich			1	0.10
Faure. Gabriel			1	0.60
Fischer. Johann Caspar Ferdinand	1	0.10		
Gluck. Christoph Willibald			1	1.61
Grieg. Edvard			1	2.10
Guerrero. Francisco	1	0.12		
Handel. George Frideric	4	1.00	1	0.75
Haydn. Joseph			6	1.01
Kempff. Wilhelm	1	1.58		
Leontovych. Mykola			2	0.22
Liszt. Franz	34	39.98		
Mahler. Gustav			1	0.85
Mendelssohn. Felix			2	1.41
Moussorgsky. Modest			1	0.04
Mozart. Wolfgang Amadeus	1	0.71	8	1.45
Okashiro. Chitose	3	1.09		
Pachelbel. Johann	1	0.15		
Praetorius. Michael			2	0.14
Purcell. Henry			1	0.08
Ravel. Maurice	6	6.49	8	6.69
Rondeau. Michel	2	0.25	1	0.14
Schonberg. Arnold			1	0.21
Schumann. Robert			1	0.05
Shorter. Steve	1	0.26		
Smetana. Bedrich			1	0.61
Soler. Antonio	1	0.54		
Strauss. Johann			1	0.04
Strauss. Richard			1	0.22
Stravinsky. Igor			4	0.94
Tchaikovsky. Piotr Ilyich			36	20.08
Telemann. Georg Philipp			2	1.04
Unknown.	107	40.18	28	7.47
Vivaldi. Antonio			4	2.94
Walther. Johann Gottfried	1	0.14		
Wiberg. Steve			1	0.75
Zachow. Friedrich Wilhelm	1	0.32	2	0.23

Table 1. This table describes the relative importance of the different composers present in the database. For each composer, the number of piano (respectively orchestral) scores in the database are indicated in the second (respectively fourth) column. The total number of files is 184 x 2 = 392. As the length of the files can vary significantly, a more significant indicator of a composer's representativeness in the database is the ratio of the number of frames from its scores over the total number of frames in the database.

Figure 2 highlights the activation ratio of each pitch in the orchestration scores ($\frac{\#\{\text{pitch on}\}}{\#\{\text{pitch on}\}+\#\{\text{pitch off}\}}$, where $\#$ is the cardinal of an ensemble) over the whole dataset. Note that this activation ratio does not take the duration of notes into consideration, but only their number of occurrences. The pitch range of each instrument can be observed beneath the horizontal axis.

Two different kinds of imbalance can be observed in figure 2. First, a given pitch is rarely played. Second, some pitches are played more often compared with others. Class imbalance is known as being problematic for machine learning systems, and these two observations highlight how challenging the projective orchestration task is.

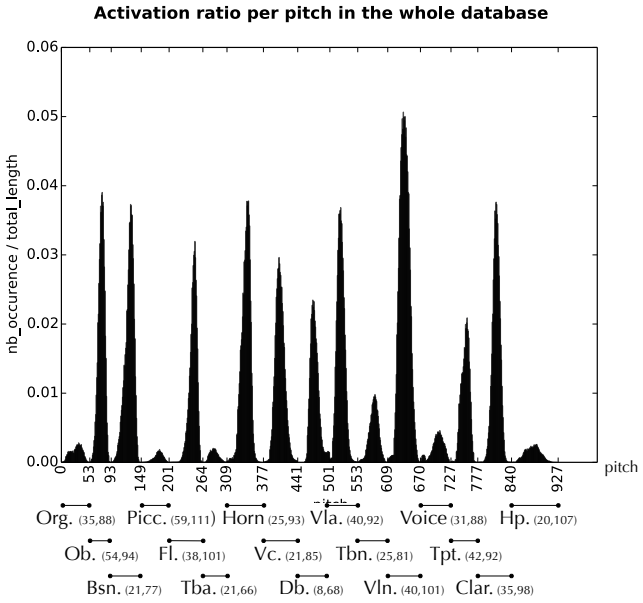

Figure 2. Activation ratio per pitch in the whole orchestral score database. For one bin on the horizontal axis, the height of the bar represents the number of notes played by this instrument divided by the total number of frames in the database. This value is computed for the event-level aligned representations 4.2. The different instruments are covered by the pitch axis, and one can observe the peaks that their medium ranges form. The maximum value of the vertical axis (0.06), which is well below 1, indicates that each pitch is rarely played in the whole database.

More statistics about the whole database can be found on the companion website.

3.1.4 Integrity

Both the metadata and instrumentation *CSV* files have been automatically generated but manually checked. We followed a conservative approach by automatically rejecting any score with the slightest ambiguity between a track name and a possible instrument (for instance *bass* can refer to *double-bass* or *voice bass*).

3.1.5 Formats

To facilitate the research work, we provide pre-computed piano-roll representations such as the one displayed in Figure 3. In this case, all the *MIDI* files of piano (respectively orchestra) work have been transformed and concatenated into a unique two-dimensional matrix. The starting and ending time of each track is indicated in the *metadata.pkl* file. These matrices can be found in Lua/Torch (.t7), Matlab (.m), Python (.npy) and raw (.csv) data formats.

3.1.6 Score Alignment

Two versions of the database are provided. The first version contains unmodified midi files. The second version contains *MIDI* files automatically aligned using the *Needleman-Wunsch* [18] algorithm as detailed in

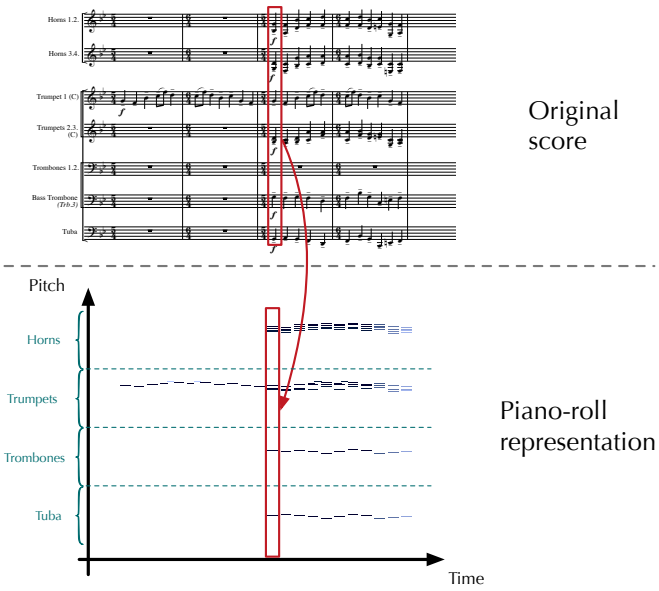

Figure 3. *Piano-roll* representation of orchestral scores. The *piano-roll* pr is a matrix. A pitch p at time t played with an intensity i is represented by $pr(p,t) = i$, where 0 is a note off. This definition is extended to an orchestra by simply concatenating the *piano-rolls* of every instrument along the pitch dimension.

Section 3.2.

3.2 Automatic Alignment

Given the diverse origins of the *MIDI* files, a piano score and its corresponding orchestration are almost never aligned temporally. These misalignments are very problematic for learning or mining tasks, and in general for any processing which intends to take advantage of the joint information provided by the piano and orchestral scores. Hence, we propose a method to automatically align two scores, and released its Python implementation on the companion website [2]. More precisely, we consider the piano-roll representations (Figure 3) where the scores are represented as a sequence of vectors. By defining a distance between two vectors, the problem of aligning two scores can be cast as a univariate sequence-alignment problem.

3.2.1 Needleman-Wunsch

The *Needleman-Wunsch* (*NW*) algorithm [18] is a dynamic programming technique, which finds the optimal alignment between two symbolic sequences by allowing the introduction of gaps (empty spaces) in the sequences. An application of the *NW* algorithm to the automatic alignment of musical performances is introduced in [9]. As pointed out in that article, *NW* is the most adapted technique for aligning two sequences with important structural differences like skipped parts, for instance.

The application of the *NW* algorithm relies solely on the definition of a cost function, which allows the pairwise comparison of elements from the two sequences, and the cost of opening or extending a gap in one of the two sequences.

3.2.2 Similarity Function

To measure the similarity between two chords, we propose the following process:

- discard intensities by representing notes being played as one and zero otherwise.
- compute the pitch-class representation of the two vectors, which flattens all notes to a single octave vector (12 notes). In our case, we set the pitch-class to one if at least one note of the class is played. For instance, we set the pitch-class of C to one if there is any note with pitch C played in the piano-roll vector. This provides an extremely rough approximation of the harmony, which proved to be sufficient for aligning two scores. After this step, the dimensions of each vector is 12.
- if one of the vectors is only filled with zeros, it represents a silence, and the similarity is automatically set to zero (note that the score function can take negative values).
- for two pitch-class vectors $\boldsymbol{A}$ and $\boldsymbol{B}$, we define the score as

$$S(\boldsymbol{A},\boldsymbol{B}) = C \times \frac{\sum_{i=1}^{12} \delta(A_i + B_i)}{max(||\boldsymbol{A}+\boldsymbol{B}||_1, 1)} \quad (1)$$

where δ is defined as:

$$\delta(x) = \begin{cases} 0 & \text{if x = 0} \\ -1 & \text{if x = 1} \\ 1 & \text{if x = 2} \end{cases}$$

C is a tunable parameter and $||x||_1 = \sum_i |x_i|$ is the $\mathcal{L}1$ norm.

Based on the values recommended in [18] and our own experimentations, we set C to 10. The gap-open parameter, which defines the cost of introducing a gap in one of the two sequences, is set to 3 and the gap-extend parameter, which defines the cost of extending a gap in one of the two sequences, is set to 1.

4. AN APPLICATION : PROJECTIVE AUTOMATIC ORCHESTRATION

In this section, we introduce and formalize the automatic projective orchestration task (Figure 1). In particular, we propose a system based on statistical learning and define an evaluation framework for using the *POD* database.

4.1 Task Definition

4.1.1 Orchestral Inference

For each orchestral piece, we define as **O** and **P** the aligned sequences of column vectors from the *piano-roll* of the orchestra and piano parts. We denote as T the length of the aligned sequences **O** and **P**.

[2] https://qsdfo.github.io/LOP/code

The objective of this task is to infer the present orchestral frame knowing both the recent past of the orchestra sequence and the present piano frame. Mathematically, it consists in designing a function f where

$$\hat{O}(t) = \boldsymbol{f}[P(t), O(t-1), ..., O(t-N)] \quad \forall t \in [N, ...T] \tag{2}$$

and N defines the order of the model.

4.1.2 Evaluation Framework

We propose a quantitative evaluation framework based on a one-step predictive task. As discussed in [5], we make the assumption that an accurate predictive model will be able to generate original acceptable works. Whereas evaluating the generation of a complete musical score is subjective and difficult to quantify, a predictive framework provides us with a quantitative evaluation of the performance of a model. Indeed, many satisfying orchestrations can be created from the same piano score. However, the number of reasonable inferences of an orchestral frame given its context (as described in equation 2) is much more limited.

As suggested in [4,12], the accuracy measure [2] can be used to compare an inferred frame $\hat{O}(t)$ drawn from (2) to the ground-truth $O(t)$ from the original file.

$$\text{Accuracy}(t) = 100 \cdot \frac{TP(t)}{TP(t) + FP(t) + FN(t)} \tag{3}$$

where $TP(t)$ (true positives) is the number of notes correctly predicted (note played in both $\hat{O}(t)$ and $O(t)$). $FP(t)$ (false positive) is the number of notes predicted that are not in the original sequence (note played in $\hat{O}(t)$ but not in $O(t)$). $FN(t)$ (false negative) is the number of unreported notes (note absent in $\hat{O}(t)$, but played in $O(t)$).

When the quantization gets finer, we observed that a model which simply repeats the previous frame gradually obtains the best accuracy as displayed in Table 2. To correct this bias, we recommend using an event-level evaluation framework where the comparisons between the ground truth and the model's output is only performed for time indices in T_e defined as the set of indexes t_e such that

$$\mathrm{O}(t_e) \neq \mathrm{O}(t_e - 1)$$

The definition of event-level indices can be observed in Figure 4.

In the context of learning algorithms, splitting the database between disjoint *train* and *test* subsets is highly recommended [3, pg.32-33], and the performance of a given model is only assessed on the test subset. Finally, the mean accuracy measure over the dataset is given by

$$\frac{1}{K} \sum_{s \in \mathcal{D}_{test}} \sum_{t_e \in T_e(s)} Accuracy(t_e) \tag{4}$$

where $\mathcal{D}_{test}$ defines the test subset, $T_e(s)$ the set of event-time indexes for a given score s, and $K = \sum_{s \in \mathcal{D}_{test}} |T_e(s)|$.

4.2 Proposed Model

In this section, we propose a learning-based approach to tackle the automatic orchestral inference task.

4.2.1 Models

We present the results for two models called *conditional Restricted Boltzmann Machine* (*cRBM*) and *Factored Gated cRBM* (*FGcRBM*). The models we explored are defined in a probabilistic framework, where the vectors $O(t)$ and $P(t)$ are represented as binary random variables. The orchestral inference function is a neural network that expresses the conditional dependencies between the different variables: the present orchestral frame $O(t)$, the present piano frame $P(t)$ and the past orchestral frames $O(t-1, ..., t-N)$. *Hidden units* are introduced to model the co-activation of these variables. Their number is a hyper-parameter with an order of magnitude of 1000. A theoretical introduction to these models can be found in [26], whereas their application to projective orchestration is detailed in [7].

4.2.2 Data Representation

In order to process the scores, we import them as *piano-roll* matrices (see Figure 3). Their extension to orchestral scores is obtained by concatenating the *piano-rolls* of each instrument along the pitch dimension.

Then, new events $t_e \in T_e$ are extracted from both piano-rolls as described in Section 4.1. A consequence is that the trained model apprehends the scores as a succession of events with no rhythmic structure. This is a simplification that considers the rhythmic structure of the projected orchestral score to be exactly the same as the one of the original piano score. This is false in the general case, since a composer can decide to add nonexistent events in an orchestration. However, this provides a reasonable approximation that is verified in a vast majority of cases. During the generation of an orchestral score given a piano score, the next orchestral frame is predicted in the event-level framework, but inserted at the temporal location of the corresponding piano frame as depicted in Figure 4.

Automatic alignment of the two *piano-rolls* is performed on the event-level representations, as described in Section 3.2.

In order to reduce the input dimensionality, we systematically remove any pitch which is never played in the training database for each instrument. With that simplification the dimension of the orchestral vector typically decreases from 3584 to 795 and the piano vector dimension from 128 to 89. Also, we follow the usual orchestral simplifications used when writing orchestral scores by grouping together all the instruments of a same section. For instance, the *violin* section, which might be composed by several instrumentalists, is written as a single part. Finally, the velocity information is discarded, since we use binary units that solely indicate if a note is on or off.

Eventually, we observed that an important proportion of the frames are silences, which mathematically corresponds to a column vector filled with zeros in the piano-roll representation. A consequence of the over-representation of silences is that a model trained on this database will lean towards orchestrating with a silence any piano input, which is statistically the most relevant choice. Therefore, orches-

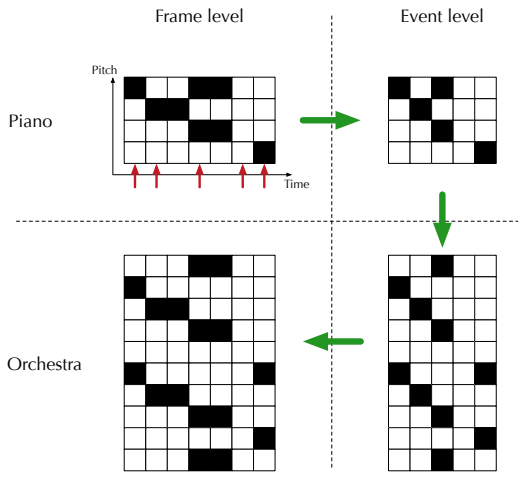

Figure 4. From a piano score, the generation of an orchestral score consists in extracting the event-level representation of the piano score, generating the sequence of orchestral events, and then injecting them at the position of the event from the piano score. Note that the silence in the fourth event of the piano score is not orchestrated by the probabilistic model, but is automatically mapped to a silence in the orchestral version.

tration of silences in the piano score ($P(t) = 0$) are not used as training points. However, it is important to note that they are not removed from the piano-rolls. Hence, silences could still appear in the past sequence of a training point, since it is a valuable information regarding the structure of the piece. During generation time, the silences in the piano score are automatically orchestrated with a silence in the orchestra score. Besides, silences are taken into consideration when computing the accuracy.

4.2.3 Results

The results of the *cRBM* and *FGcRBM* on the orchestral inference task are compared to two naïve models. The first model is a random generation of the orchestral frames obtained by sampling a Bernoulli distribution of parameter 0.5. The second model predicts an orchestral frame at time t by simply repeating the frame at time $t-1$. The results are summed up in Table 2.

Model	Frame-level accuracy (Q = 4)	Frame-level accuracy (Q = 8)	Event-level accuracy
Random	0.73	0.73	0.72
Repeat	61.79	76.41	50.70
cRBM	5.12	34.25	27.67
FGcRBM	33.86	43.52	25.80

Table 2. Results of the different models for the projective orchestration task based on frame-level accuracies with a quantization of 4 and 8 and event-level accuracies.

4.3 Discussion

As expected, the random model obtains very poor results. The repeat model outperform all three other models, surprisingly even in the event-level framework. Indeed, we observed that repeated notes still occur frequently in the event-level framework. For instance, if between two successive events only one note out of five is modified, the accuracy of the repeat model on this frame will be equal to 66%.

If the *FGcRBM* model outperforms the *cRBM* model in the frame-level framework, the *cRBM* is slightly better than the *FGcRBM* model in the event-level framework.

Generations from both models can be listened to on the companion website [3]. Even though some fragments are coherent regarding the piano score and the recent past orchestration, the results are mostly unsatisfying. Indeed, we observed that the models learn an extremely high probability for every note to be off. Using regularization methods such as weight decay has not proven efficient. We believe that this is due to the sparsity of the vectors $O(t)$ we try to generate, and finding a more adapted data representation of the input will be a crucial step.

5. CONCLUSION AND FUTURE WORK

We introduced the Projective Orchestral Database (*POD*), a collection of *MIDI* files dedicated to the study of the relations between piano scores and corresponding orchestrations. We believe that the recent advent in machine learning and data mining has provided the proper tools to take advantage of this important mass of information and investigate the correlations between a piano score and its orchestrations. We provide all *MIDI* files freely, along with aligned and non-aligned pre-processed piano-roll representations on the website `https://qsdfo.github.io/LOP/index.html`.

We proposed a task called automatic orchestral inference. Given a piano score and a corresponding orchestration, it consists in trying to predict orchestral time frames, knowing the corresponding piano frame and the recent past of the orchestra. Then, we introduced an evaluation framework for this task based on a train and test split of the database, and the definition of an accuracy measure. We finally present the results of two models (the *cRBM* and *FGcRBM*) in this framework.

We hope that the *POD* will be useful for many researchers. Besides the projective orchestration task we defined in this article, the database can be used in several other applications, such as generating data for a source-separation model [17]. Even if small errors still persist, we thoroughly checked manually the database and guarantee its good quality. However, the number of files collected is still small with the aim of leading statistical investigations. Hence, we also hope that people will contribute to enlarge this database by sharing files and helping us gather the missing information.

[3] `https://qsdfo.github.io/LOP/results`

6. REFERENCES

[1] Imslp. http://imslp.org/wiki/Main_Page. Accessed : 2017-01-23.

[2] Mert Bay, Andreas F Ehmann, and J Stephen Downie. Evaluation of multiple-f0 estimation and tracking systems. In *ISMIR*, pages 315–320, 2009.

[3] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[4] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

[5] Darrell Conklin and Ian H Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, 1995.

[6] J. Cookerly. Complete orchestration system, May 18 2010. US Patent 7,718,883.

[7] Leopold Crestel and Philippe Esling. Live orchestral piano, a system for real-time orchestral music generation. In *Proceedings of the 14th Sound and Music Computing Conference*, Aalto, Finland, July 2017.

[8] Philippe Esling, Grégoire Carpentier, and Carlos Agon. Dynamic musical orchestration using genetic algorithms and a spectro-temporal description of musical instruments. *Applications of Evolutionary Computation*, pages 371–380, 2010.

[9] Maarten Grachten, Martin Gasser, Andreas Arzt, and Gerhard Widmer. Automatic alignment of music performances with structural differences. In *In Proceedings of 14th International Society for Music Information Retrieval Conference (ISMIR*. Citeseer, 2013.

[10] Jiun-Long Huang, Shih-Chuan Chiu, and Man-Kwan Shan. Towards an automatic music arrangement framework using score reduction. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 8(1):8, 2012.

[11] Charles Koechlin. *Traité de l'orchestration*. Éditions Max Eschig, 1941.

[12] Victor Lavrenko and Jeremy Pickens. Polyphonic music modeling with random fields. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 120–129. ACM, 2003.

[13] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 05 2015.

[14] Sven-Amin Lembke and Stephen McAdams. Timbre blending of wind instruments: acoustics and perception. 2012.

[15] Stephen McAdams. Timbre as a structuring force in music. In *Proceedings of Meetings on Acoustics*, volume 19, page 035050. Acoustical Society of America, 2013.

[16] Stephen McAdams and Bruno L Giordano. The perception of musical timbre. *The Oxford handbook of music psychology*, pages 72–80, 2009.

[17] M. Miron, J. Janer, and E. Gómez. Generating data to train convolutional neural networks for classical music source separation. In *Proceedings of the 14th Sound and Music Computing Conference*, pages 227–233, Aalto, Finland, 2017 2017.

[18] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453, 1970.

[19] François Pachet. A joyful ode to automatic orchestration. *ACM Trans. Intell. Syst. Technol.*, 8(2):18:1–18:13, October 2016.

[20] Geoffroy Peeters, Bruno L Giordano, Patrick Susini, Nicolas Misdariis, and Stephen McAdams. The timbre toolbox: Extracting audio descriptors from musical signals. *The Journal of the Acoustical Society of America*, 130(5):2902–2916, 2011.

[21] Walter Piston. *Orchestration*. New York: Norton, 1955.

[22] Daniel Pressnitzer, Stephen McAdams, Suzanne Winsberg, and Joshua Fineberg. Perception of musical tension for nontonal orchestral timbres and its relation to psychoacoustic roughness. *Perception & psychophysics*, 62(1):66–80, 2000.

[23] Nikolay Rimsky-Korsakov. *Principles of Orchestration*. Russischer Musikverlag, 1873.

[24] Hirofumi Takamori, Haruki Sato, Takayuki Nakatsuka, and Shigeo Morishima. Automatic arranging musical score for piano using important musical elements. In *Proceedings of the 14th Sound and Music Computing Conference*, Aalto, Finland, July 2017.

[25] Damien Tardieu and Stephen McAdams. Perception of dyads of impulsive and sustained instrument sounds. *Music Perception*, 30(2):117–128, 2012.

[26] Graham W Taylor and Geoffrey E Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *Proceedings of the 26th annual international conference on machine learning*, pages 1025–1032. ACM, 2009.