Maelis4Skrid: an Approximate Query Engine for an Online Graph-Based Musical Score Library

Adel Aly, Olivier Pivert, Virginie Thion

Maelis4Skrid: an Approximate Query Engine for an Online Graph-Based Musical Score Library

Adel Aly
adel.aly@irisa.fr
Univ. Rennes
Lannion, France

Olivier Pivert
olivier.pivert@irisa.fr
Univ. Rennes
Lannion, France

Virginie Thion
virginie.thion@irisa.fr
Univ. Rennes
Lannion, France

Abstract

Digital Score Libraries are libraries dedicated to the storing, the management and the dissemination of musical scores. While document-oriented *sheet musical scores* have been the traditional way to preserve and share Western musical works, the emergence of modern digital formats opens the way to services that leverage the musical content itself, including that of retrieving music pieces based on their musical content. In this demonstration, we present the **Skrid platform, an online Digital Score Library (DSL) that utilizes a graph-based storage for the scores' musical content**. We also present **Maelis, a flexible querying module implemented inside Skrid**, that **enables melodic pattern approximate search**, **ranks the results by relevance**, and **provides a detailed explanation of the answers**. The demonstration is composed of two scenarios that showcase the key features of Maelis embedded in Skrid. The scenarios are accompanied by an online interaction inviting the audience to engage with the system.

CCS Concepts

• **Information systems** → **Information retrieval**; **Database design and models**; **Digital libraries and archives**; • **Applied computing** → **Arts and humanities**.

Keywords

Digital musical score library, Graph representation, Approximate melodic pattern matching

ACM Reference Format:
Adel Aly, Olivier Pivert, and Virginie Thion. 2025. Maelis4Skrid: an Approximate Query Engine for an Online Graph-Based Musical Score Library. In *Proceedings of the ACM Web Conference 2025 (WWW '25), May 13–17, 2024, Sydney, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/n

1 Introduction

Music is an indispensable component of the world's cultural heritage. While sheet scores have traditionally been used for engraving music scores, the advent of modern digital representation formats (such as semi-stuctured ones [3, 6, 13], or graph-based ones [5, 12]) has transformed music notation into a valuable resource for music information processing. Today, digital score libraries provide access to large collections of digitally encoded musical scores, leveraging the encoding of musical content to enable advanced retrieval features, as demonstrated here. In the demonstration, **we first present the Skrid platform, an online digital score library (DSL) that stores musical scores in a property graph database**. The graph model provides a topological-oriented intuitive representation of the scores, leveraged by expressive graph-pattern queries (see [12] for details). To our knowledge, Skrid is the only DSL of the literature that relies on such a data model.

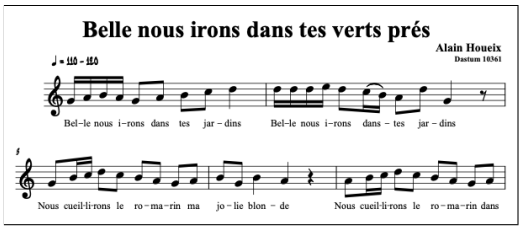

Figure 1: Beginning of *Belle nous irons dans tes verts prés* [11]

The intended primary users of a DSL are music performers, scholars, music students, and specialized users like musicologists, music theorists, music engravers, composers, and librarians. These users make use of the library in diverse ways, driven by distinct goals. Some users focus on retrieving music pieces based on metadata. For example, music performers may search for pieces by title or composer. Other users are more interested in retrieving pieces based on their musical content, such as searching for occurrences of a specific melodic pattern (Fig. 6 p. 4 gives two examples of melodic patterns), which is the problem we focus on. This is a complex task [1], explaining why only a few DSLs offer this feature as of now (see Section 5). Moreover, such DSLs typically consider an exact matching process, which can easily lead to empty or plethoric results. They lack the ability to retrieve musical fragments that are different but close to the considered pattern, and to rank the retrieved answers. In the demonstration, **we also present a querying tool, called Maelis, that is implemented inside Skrid. Maelis (1) retrieves, from (graph-based) musical score collections, data that approximately match a given melodic pattern, (2) computes a similarity score (referred to as *satisfaction degree* from now) between each answer and the given melodic pattern, (3) ranks the answers based on their satisfaction degrees, and (4) provides a detailed explanation for each answer**. To our knowledge, no other DSL implements an approximate melodic pattern retrieval resulting in ranked and explained answers.

The proposed demonstration consists of two parts: a scripted one, and an online interactive one. *The scripted part* is twofold: we present both the Skrid platform and its associated flexible querying

WWW '25, May 28– May 2, 2025, Sydney, Australia

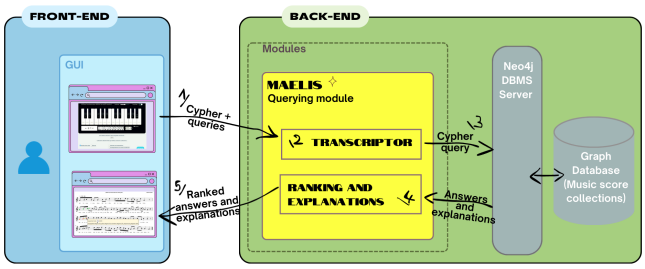

Figure 2: System architecture and content-based querying

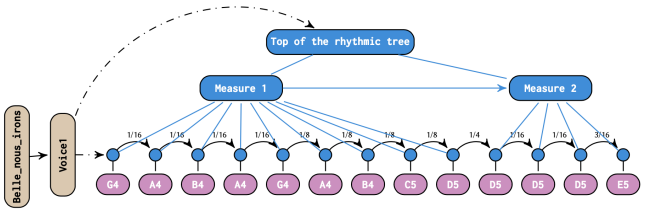

Figure 3: Graph-based representation of a part of Fig. 1

module Maelis, through two scripted scenarios. These scenarios highlight Skrid's and Maelis's key features, and allow illustrating their internal processing. Then comes the *online interaction* segment of the demonstration, during which the audience will engage directly with the system and experiment its features (mainly the approximate melodic searching).

2 Musical notation

Let us first introduce some basic notions of Western music notation, kept to the bare minimum for understanding the demonstration. Even though our model can also deal with polyphonic music, we restrict the presentation to monophonic music for the sake of simplicity. We consider the musical score from Fig. 1 as an example, to illustrate the introduced concepts.[1]

In Western music notation, a finite set of frequencies is used to refer to the sounds that may appear in a musical piece. Each frequency is associated with a pitch class (a letter C, D, E, F, G, A, and B) and an index known as the octave (in the set $\{1, \ldots, 7\}$). On a music sheet, *musical events* are graphically represented to define the sounds that compose the music piece. The symbols are written all along a group of horizontal lines called *staves*. A main symbol that appears in a musical score is the *note* (e.g. a symbol among 𝅗𝅥, 𝅗𝅥, ♪, 𝅘𝅥𝅯, 𝅝, etc.), which denotes a sound to be played. A note symbol encodes both the information of the frequency and the duration of the denoted sound. The vertical position of the note on the staff determines the frequency (e.g. if we consider the first staff of Fig. 1, the first note is a *G*4, then follows an *A*4, and a *B*4, etc.), while the shape of a note (including the head, stem, and flags) determines the duration (for instance, a 𝅘𝅥𝅯 note is two times shorter than a 𝅘𝅥 note). Time is divided into frames called *measures*, synchronized over the staves, and visually delineated by vertical *bars*. In Fig. 1, we can see five measures. Additional symbols exist, including the rest symbols, like 𝄾 that appears near the end of the second measure. The whole musical content of a score is composed of several synchronized sequences of musical events associated with instruments. The digitized content of a musical score also contains additional information in the form of metadata (the title, the composer, the collector, the composition date, etc.), as well as rendering information (margins for the music sheet, spacing between the elements, direction of the stems, fonts, etc.).

[1]In the corpus this musical score belongs to, time signature is voluntary not imposed (See [11] for the explanation by musicologists who collected and encoded the corpus).

3 System Overview

Skrid is both a research project and a DSL platform dedicated to the preservation and the dissemination of traditional musical scores from Brittany (France).[2] Fig. 2 illustrates the architecture of the DSL. Skrid is composed of two parts: a *back-end* (in green in Fig. 2), where data is stored, managed, and processed, and a *front-end* (in blue in Fig. 2), which contains the modules the users interact with.

3.1 Back-end

The back-end of the Skrid platform (in green in Fig. 2) is composed of a data management layer and functional modules, including the Maelis querying module the demonstration focuses on.

Data Model. The back-end of the application embeds a storage layer managed by a graph DBMS, namely Neo4j [8] (in grey in Fig. 2). The graph database stores the musical scores modeled as a graph. Roughly speaking, each musical voice is represented by the flow of notes it is composed of (edges connect each note to the following one). Additional edges also represent how notes are organized into measures.[3] The graph model provides a topology-oriented intuitive representation of the scores (see Fig. 3, where the graph corresponds to a part of the musical score from Fig. 1), leveraged by expressive graph-pattern queries. Due to lack of space, we cannot give more details about the model,[4] and we rather focus on the flexible querying functionality, presented hereafter.

Flexible Querying. Maelis (yellow box in Fig. 2) is the querying module that implements exact and approximate melodic pattern matching. In the approximate case, flexibility is highly parametrizable: it may concern (i) the pitch, (ii) the duration, and (iii) the sequencing of the notes in the pattern. In short, Maelis relies on fuzzy set theory for modeling the gradual criteria associated with an approximate pattern-based search, leading to the calculation of a satisfaction degree in $(0, 1]$ for each answer. Maelis takes the form of an extension of the Cypher graph-pattern query language, denoted by Cypher$^+$ hereafter. This extension allows (i) constructing the membership functions associated with the fuzzy features of the search pattern (for modeling the user-accepted tolerance on the pitch, duration or sequencing of the notes present in the pattern), and (ii) defining a global threshold $\alpha \in (0, 1]$ to prune the set of retrieved elements (and keep only those that are sufficiently

[2]The Skrid DSL platform is a collaborative effort with *Dastum* [2], an association that strives to collect, preserve and disseminate such data. Skrid is available at *https://shaman.enssat.fr/skrid/*

[3]A module allows to populate the graph database with MEI files as input.

[4]The interested reader may refer to [12] for more information about the data model and the exact search facilities that come with this representation.

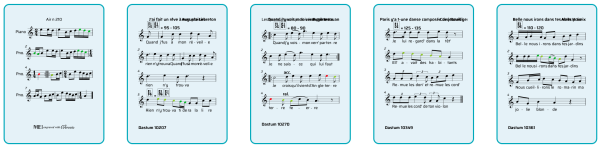

Figure 4: Approximate matches (UseCase$_{Mathieu}$)

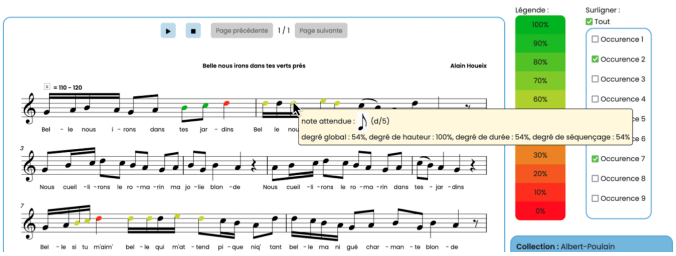

Figure 5: Explanation of an approx. matching (UseCase$_{Aya}$)

satisfactory, i.e., whose satisfaction degree is $\geq \alpha$). The membership function associated with a musical aspect (e.g., the pitch) allows computing a satisfaction degree that assesses the resemblance –wrt. this aspect– between the user-given pattern and the approximate occurrence of the pattern in the database. For each occurrence, the satisfaction degrees related to pitch, duration and sequencing are aggregated into a global satisfaction degree. A user may choose to be tolerant on only some of the three dimensions.

Let us now turn to Maelis's implementation. In Skrid, Maelis is implemented as an open-source[5] add-on layer, developed in Python, which interacts with the graph database. Maelis is composed of two components, detailed hereafter.

The *Transcriptor* component takes as an input a Cypher$^+$ query (stage 1 in Fig. 2) involving both a melodic pattern and flexibility parameters that set the maximal tolerance thresholds allowed for the pitch, the duration, and the sequencing. A scoring function —in the form of a triangular fuzzy membership function— can be inferred from each threshold. Based on this information, the *Transcriptor* converts the Cypher$^+$ query into a (classical) Cypher query that can retrieve the answers (the occurrences of the musical pattern in the database), and the information needed to compute the satisfaction degrees (stage 2 in Fig. 2). This query is sent to the Neo4j DBMS that queries the musical score collections (stage 3 in Fig. 2).

The *Ranking and explanations* component calculates a satisfaction degree for each retrieved answer (stage 4 in Fig. 2). This degree is based on the aggregation of the partial satisfaction degrees associated with the different notes of the answer, which themselves combine the satisfaction degrees related to the different dimensions of the matching (pitch, duration, and sequencing). The overall satisfaction degree makes it possible to rank-order the answers before presenting them to the user. For their part, the partial satisfaction degrees allow providing a detailed explanation for each answer (stage 5 in Fig. 2), detailed in Section 3.2.

3.2 Front-end

Users do not need any knowledge regarding the graph data model, formal query languages, or the process that retrieves the data. They interact with the system through user-friendly modules of the front-end layer (in blue in Fig. 2), which provide an online (web) access to the DSL. The front-end layer of the Skrid platform is a JavaScript thin client. It implements two main graphical user interface (GUI) modules. One of them is the querying GUI module, which is a user-friendly entry point for querying music score collections, based on the musical content of the score. In this module, the user can (i) select a set of collections to search in, (ii) define the musical pattern to be retrieved via a virtual piano keyboard interface, and (iii) set the flexible parameters (in simple text areas). Default values are proposed for the flexible parameters but may be changed by the user. The flexible parameters are the tolerance thresholds for pitch, duration, and sequencing (see Section 3.1). The user may also authorize or not musical transposition. The GUI module converts the entries of the user into a Cypher$^+$ query, which is sent to the back-end querying module of the application (stage 1 in Fig. 2).

Another GUI module displays the musical scores retrieved by the Cypher$^+$ query (stage 5 in Fig. 2), with the global satisfaction degree associated with each score answer (Fig. 4 is an example of such interface). For each matching occurrence of the pattern, a detailed explanation of the satisfaction degrees can also be given for each note, displayed when the user hovers over a note in an answer (Fig. 5 is an example of such another interface). In terms of development, the visualization uses Verovio, an open-source library [15] for displaying the musical scores on a web page generated in JavaScript.

4 Demonstration Scenario

1. Scripted demonstration. We first showcase the capabilities of Maelis through two use cases that illustrate its key features. The use cases consider two user tasks that involve a melodic pattern approximate search, in a (real) database of folk music from Brittany that contains 862 musical scores, modeled as a graph database of 76,674 nodes and 177,681 edges (during the demonstration, the audience will visualize the content of such a graph database through the Neo4j Desktop application [9]).

The first use case, UseCase$_{Mathieu}$, considers Mathieu, an accordion teacher. He would like to retrieve musical scores that contain a given melodic pattern for his students' practicing exercises: five descending notes — $E5$, $D5$, $C5$, $B4$, $A4$ — preferably with a rhythm of five sixteenth notes (see Fig. 6.a). Mathieu enters the pattern using the virtual piano GUI. The exact search retrieves only one answer. Then, Mathieu decides to be more tolerant on the pattern rhythm, with a factor of 2, meaning that he authorizes the duration of a note in a musical score to be at most the double and at least the half of that specified in the pattern. For each note, the more the duration diverges from that of the corresponding note in the pattern, the lower the satisfaction degree (over the duration dimension) for this note will be. Mathieu only makes the rhythm criterion tolerant, while the pitch criterion remains exact (as the pitches of the notes correspond to a gesture meant to be practiced

[5]https://github.com/aa196883/compilation_requete_fuzzy

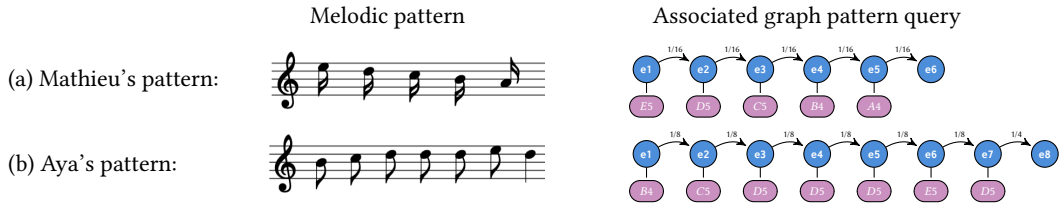

Figure 6: Melodic patterns of the scenarios

on the musical instrument). Mathieu enters these preferences in the GUI, in a dedicated frame under the piano. Then, 37 approximate answers are retrieved, ranked in decreasing order of their satisfaction degrees.[6] Fig. 4 shows the top answers retrieved for Mathieu's query, which includes the exact answer that matches at 100% (its satisfaction degree is equal to 1), followed by the approximate ones.

The second use case, UseCase$_{Aya}$, considers Aya, who is a musicologist. She looks for musical pieces that contain a given pattern, to study the positioning of this pattern in the corresponding scores. This pattern is given in Fig. 6.b. The exact search returns 5 answers only. Then, Aya decides to authorize approximate matches, introducing some tolerance into the pitch condition (allowing a maximal difference of 0.5 tone for each note), the duration one (up to the double or down to half of the initial duration for each note), and the sequencing condition (up to a 𝅘𝅥𝅯 can be inserted between each pair of searched notes). Many answers are then returned to Aya, but this is not a problem as these answers are ranked according to their satisfaction degree (Aya could also have chosen to specify a value for the α parameter, for instance 0.5, to get the sole answers where the satisfaction degree is $\geq \alpha$). For each musical score result, Aya has access to a detailed explanation of the occurrences of the pattern in the score. In Fig. 5, we can see the occurrences of the searched pattern in a result score (which is the score of Fig 1). The occurrences of the pattern are displayed in color. For each note of an occurrence, the global adequacy of the note with the pattern is visually rendered by means of a color scale, and a popup text appears when the cursor hovers over a note, that explains the satisfaction degree of the note wrt. each expected feature. Aya can export the answers and their explanations to a tabular file in order to use the results as she wishes.

During the demonstration, based on the scenarios described above, the audience will see how the user interacts with the system and how internal data circulates in the back-end of Skrid, especially in terms of implementation of the flexible querying functionalities and the underlying graph database querying. More specifically, the audience will see the Cypher$^+$ query that is generated by the GUI, and the transcribed Cypher query that is sent to Neo4j (see Fig. 2).[7]

2. Online interaction. In the second part of the demonstration, the audience will be invited to interact with the system, by specifying a melodic pattern along with flexibility parameters.

[6] By default, the satisfaction degree of a pattern occurrence is the average of the degrees of each note, where the degree of a note is itself calculated as the minimum of the degrees over the musical dimensions. The global satisfaction degree of a musical score is the maximum degree amongst the occurrences of the pattern in the score. The aggregation operators (average, min., max.) can be parameterized in the framework.

[7] The use cases' queries are provided at https://doi.org/10.5281/zenodo.14198722.

5 Related Systems

Other systems of the literature have tackled the task of melodic pattern searching. However, most of them consider an exact matching process, for instance the virtual piano querying interfaces of Musipedia [7], IMSLP [4] or NEUMA [10], with some allowing search via only some musical aspects (e.g. melody only, rhythm only, diatonic search). As argued above, an exact search can easily lead to empty or plethoric sets of answers. A few systems allow a form of approximate searching, for instance Musipedia [7]'s contour or thesession.org [14]. However, in these systems, flexibility is not parameterizable, no matching degree is attached to the answers, and no explanation functionality is provided.

Acknowledgments

We thank Lannion Trégor Communauté for their funding. We also thank the Dastum association, in particular Anne-Marie Nicol and Gwenaël Piel for their involvement. We extend our gratitude to Vincent Barreaud and Tommaso Padovano for their work on the first version of Skrid, and Louis Thomas-Girardey for his contributions on the current version.

References

[1] Michael A. Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. 2008. Content-based music information retrieval: current directions and future challenges. *Proc. IEEE* 96, 4 (2008), 668–696.
[2] Dastum 1993. Dastum Association. https://www.dastum.bzh/association/.
[3] Michael Good. 2001. *The Virtual Score: Representation, Retrieval, Restoration.* MIT Press, Chapter MusicXML for Notation and Analysis, 113–124.
[4] IMSLP 2024. IMSLP Music Search. www.peachnote.com. (consult. date).
[5] Jim Jones, Diego de Siqueira Braga, Kleber Tertuliano, and Tomi Kauppinen. 2017. MusicOWL: the music score ontology. In *Proc. of the Intl Conf. on Web Intelligence (WI).* 1222–1229.
[6] MEI 2024. Music Encoding Initiative (MEI) web site. http://www.music-encoding.org. (consult. date).
[7] Musipedia 2024. Musipedia. www.musipedia.org. (consult. date).
[8] Neo Technology. 2024 (consult. date). Neo4j web site. www.neo4j.org.
[9] Neo4j Desktop 2024. Neo4j Desktop. "https://neo4j.com/docs/desktop-manual/current/". (consult. date).
[10] Neuma 2024. The NEUMA platform. http://neuma.huma-num.fr. (consult. date).
[11] Albert Poulain. 2011. *Carnets de route – Chansons traditionnelles de Haute-Bretagne.* Presses Universitaires de Rennes, co-édition Dastum.
[12] Philippe Rigaux and Virginie Thion. 2024. Topological querying of music scores. *Data Knowl. Eng.* 153 (2024).
[13] Perry Rolland. 2002. The Music Encoding Initiative (MEI). In *Proc. of the Intl. Conf. on Musical Applications Using XML.* 55–59.
[14] The Session 2024. The Session Web Site. "www.thesession.org". (consult. date).
[15] Verovio 2024. Verovio web site. "https://www.verovio.org". (consult. date).