Maialib: uma biblioteca de funções para musicologia e análise musical assistida por computador

MODALIDADE: COMUNICAÇÃO

SUBÁREA: Teoria e Análise Musical

Nycholas Maia
Núcleo Interdisciplinar de Comunicação Sonora - NICS/UNICAMP
nyckmaia@gmail.com

Resumo. *Maialib* é uma biblioteca de funções computacionais desenvolvida em C++ e Python com a finalidade de analisar, pesquisar e transformar dados simbólicos contidos em arquivos de partituras musicais digitalizadas. A partir da implementação de elementos básicos como "Nota", "Intervalo", "Acorde" entre 11 outras classes, a biblioteca permite que músicos, professores e pesquisadores com pouco conhecimento técnico em programação escrevam *scripts* simples em Python de uma maneira rápida e modular, utilizando-os em outros projetos de maior complexidade. O objetivo da biblioteca é principalmente fornecer ferramentas de software integradas com a teoria musical para músicos com pouca experiência em programação (especialmente musicólogos) e para programadores com habilidades modestas de teoria musical, possibilitando o desenvolvimento de trabalhos entre esses dois tipos profissionais. Este artigo apresenta a biblioteca *Maialib*, demonstrando como usá-la e os tipos de problemas para os quais ela é adequada.

Palavras-chave. *Maialib*, Análise musical assistida por computador, Musicologia, Música simbólica, Python

Title. **Maialib: A Toolkit for Musicology and Computer-Aided Music Analysis**

Abstract. *Maialib* is a library of computational functions developed in C++ and Python with the purpose of analyzing, searching and transforming symbolic data contained in files of digitized musical scores. From the implementation of basic elements such as "Note", "Interval", "Chord" among 11 other classes, the library allows musicians, teachers and researchers with little technical knowledge in programming to write simple scripts in Python in a fast and modular way, using them in other projects of greater complexity. The aim of the library is mainly to provide software tools integrated with music theory for musicians with little programming experience (especially musicologists) and for programmers with modest music theory skills, enabling the development of work between these two professional types. This article introduces the *Maialib* library, demonstrating how to use it as well the types of problems it is suitable for.

Keywords. *Maialib*, Computer-aided music analysis, Musicology, Symbolic music, Python

1 Introdução

1.1 Musicologia auxiliada por computador

Nos dias atuais, os avanços na tecnologia dentro do domínio de *"Music Information Retrieval - MIR"* permitiram análises computacionais abrangentes de amplas coleções musicais (DOWNIE, 2003). Por exemplo, na Etnomusicologia Computacional, a MIR tem sido utilizada para facilitar o estudo comparativo da música de diversas culturas, principalmente as não-ocidentais (TZANETAKIS *et al.*, 2007). Também a Musicologia Computacional vem apresentando uma grande integração com a área de *Music Information Retrieval - MIR"* (VOLK; WIERING; KRANENBURG, 2011). Por causa da velocidade de processamento de dados musicais, abre-se um leque maior de possibilidades dentro das várias subáreas da musicologia, como por exemplo as análises etnomusicológicas de Kranenburg sobre canções folclóricas analisadas sob a ótica de altura, estrutura métrica, estrutura de frases e melodias completas (KRANENBURG, 2010).

Além dos recursos já mencionados, a análise musical assistida por computador fornece recursos para agilizar, organizar e testar os pontos de vista do analista. Por esse motivo P. Smith afirma que muitos pesquisadores dessa área descobriram que o processo de automatizar a análise musical tem como pré-requisito solucionar grandes desafios, para os quais são aplicadas uma grande gama de ferramentas numéricas, estatísticas, processamento de linguagem natural, inteligência artificial e outras (SMITH, 2012). Dito de outra forma, com técnicas e algoritmos computacionais, os musicólogos podem analisar grandes volumes de dados musicais de maneira mais eficiente, facilitando a sua interpretação.

Este artigo tem como finalidade apresentar uma nova ferramenta no âmbito da musicologia, análise e composição musical assistida por computador, a saber, a biblioteca de funções musicais *Maialib*, mostrando a sua integração com dados musicais simbólicos, a saber, classes de objetos como *'Score', 'Note', 'Chord'*, funções e outras estruturas, com exemplos tanto de análise musical como de composição musical assistida por computador.

1.2 Informação musical e formatos computacionais

Na segunda metade do século XX começaram a surgir várias propostas de armazenamento e comunicação de dados musicais no âmbito computacional, porém o formato digital que se tornou universalmente aceito foi o MIDI 1.0 anunciado na Audio Engineering Society em 1982 (PETER, 2013). Mesmo com a grande adesão dos músicos ao MIDI, este

formato na versão original possuía muitas limitações de representação de dados musicais, não armazenando, por exemplo, articulações, acentuações, expressões, ligaduras de fraseado, *crescendos* e *diminuendos*, entre outras. Por esse motivo, fez-se necessário a criação de um novo formato de dados musicais que pudesse representar a maioria dos símbolos conhecidos pelos músicos e compositores, a qual, em geral, é conhecida como *Common Music Notation (CMN).*

Com esse propósito surge em 2004 o formato MusicXML 1.0 (MUSICXML 1.0, 2004). Esse formato de arquivo foi sendo aprimorado, adicionando-se novas possibilidades de notação musical e representações simbólicas, como fórmulas de compasso irregulares, variedade de glissandos, trêmulos, notas mudas, *arpeggio*, ornamentos entre muitos outros. Também foram adicionados símbolos mais raros encontrados na notação de música contemporânea do século XXI, chegando na versão 4.0 lançada em 2021 (MUSICXML 4.0, 2021). O formato MusicXML é o formato de dados musicais utilizado pelo Maialab.

Atualmente, os meios mais comuns de se obter uma partitura musical no formato de arquivo MusicXML, são:

1. A busca dos arquivos em bancos de dados online como (MUSESCORE, [*s. d.*]), (IMSLP, [*s. d.*]), (THE MUTOPIA PROJECT, [*s. d.*]), e (CHORAL PUBLIC DOMAIN LIBRARY, [*s. d.*])
2. A entrada manual dos dados musicais utilizando um software de edição de partituras tal como MuseScore, Avid Sibelius, MakeMusic Finale, LilyPond entre outros.

Não havendo a possibilidade de obter os arquivos pelos meios convencionais acima mencionados, o pesquisador musical também pode utilizar outros meios menos acurados e depois corrigir manualmente os erros notacionais pela conversão de um arquivo MIDI para MusicXML ou pela obtenção de uma fotografia da partitura desejada e processá-la em um software de conversão de dados chamado de *Optical Music Recognition* (OMR). Diversos modelos e métodos de OMR estão sendo estudados, cada qual com seu grau de acurácia e limitações (REBELO *et al.*, 2012).

Ao trabalhar com dados musicais simbólicos, os pesquisadores podem extrair e manipular dados musicais específicos, permitindo uma análise e exploração detalhadas através de algoritmos tais como: classificação automática de música simbólica (KARYDIS, 2006), modelos de expectativa melódica (SCHELLENBERG, 1997), algoritmos de medida de

similaridade melódica (ALOUPIS *et al.*, 2006), além da criação de representações e modelos estatísticos de composição de melodia e harmonia a quatro vozes (WHORLEY, 2013) entre outros.

2 *Maialib*: uma nova ferramenta para análise musical e musicologia computacional

Atualmente existe uma grande variedade de ferramentas de criação e análise assistida por computador, cada qual com seu propósito específico e recursos próprios. Neste trabalho vamos focar nas ferramentas de MIR específicas para tratamento e análise de música simbólica. Nesse recorte podemos citar ferramentas de linha de comando como "Music21" para Python (CUTHBERT; ARIZA, 2010) e o "MIDI Toolbox" para MATLAB (EEROLA, 2004) e também ferramentas que possuem interface gráfica como o "OpenMusic" (VINJAR; BRESSON, 2014) e o "jSymbolic" (MCKAY; CUMMING; FUJINAGA, 2018).

O *Maialib* é uma biblioteca computacional em desenvolvimento razoavelmente avançado (versão 1.3.0) com a finalidade de auxiliar músicos, professores e pesquisadores na análise musical assistida por computador como também em composição, sendo que esta última se encontra em estágio menos avançado. Embora existam ferramentas similares ao *Maialib*, todas apresentam vantagens e desvantagens entre si. Assim surgiu o interesse de criar uma nova biblioteca que pudesse combinar as principais vantagens de cada uma delas e tentar reduzir as desvantagens em uma única solução. Neste cenário, descrevemos abaixo o objetivo de construção do *Maialib*, sua arquitetura de software e seus benefícios musicais e computacionais.

2.1 Principais características do *Maialib*

1. O diferencial mais importante da biblioteca *Maialib* é o seu foco em uma arquitetura de software voltada a alta performance computacional. Tendo em vista o crescimento dos bancos de dados de partituras disponíveis na internet, surge a possibilidade de realizar análises e estudos musicológicos envolvendo grandes quantidades, ou coleções, de partituras simultaneamente. Para que esse tipo de análise fosse possível, as estruturas básicas do *Maialib*, como as classes *'Note', 'Interval', 'Chord'*, entre outras 11, foram escritas em C++. Usamos C++ para eficiência de processamento e Python para a interface com o usuário.

2. O uso de *Common Music Notation* que facilita a criação de scripts de análise para usuários que não possuem alto conhecimento técnico em programação. O *Maialib* possui um processo de compilação que permite exportar todas as suas funções em um módulo que pode ser importado nativamente em um ambiente Python. Dessa forma, o *Maialib* abstrai a complexidade da linguagem de baixo nível (C++), proporcionando ao usuário interagir com a biblioteca em um ambiente mais fácil e amigável (Python).
3. Facilidade de integração com as outras bibliotecas de cunho estatístico. Como o *Maialib* pode ser compilado como um módulo Python, bibliotecas científicas como Numpy, Pandas, Seaborn e Matplotlib podem ser facilmente integradas para manipular dados musicais provindos da análise de partituras pelo *Maialib*.
4. Compilação de uma biblioteca C++ para integração em outras aplicações musicais escritas em outras linguagens de programação como C++, Java, Python, etc. Desse modo, é possível utilizar a biblioteca *open-source* não apenas para a investigação científica, mas também para o desenvolvimento de produtos finais. Destaca-se também a possibilidade de compilar o código C++ para WebAssembly (Wasm), permitindo a utilização do *Maialib* em ambientes web mantendo uma performance próxima a do código C++ nativo.
5. Também, na área de composição assistida por computador, embora ainda em estágio mais básico, o Maialib permite que modelos formais de composição, implementados em scripts Python, sejam realizados musicalmente, ou seja com a apresentação de uma ou várias partituras.

Considerando as ferramentas de MIR específicas para tratamento e análise de música simbólica, talvez a mais conhecida e utilizada até o momento seja o Music21 a qual oferece uma larga gama de funções de análise e recursos de composição musical. Dado que as classes do *Maialib* foram escritas em C++ e o Music21 foi escrito totalmente em Python, é esperado que o Maialib tenha uma performance computacional melhor em processos de análise que possam exigir longos períodos de processamento computacional. Em trabalhos futuros esperamos medir e analisar essa diferença de performance entre as bibliotecas que têm o potencial de realizar os mesmos tipos de análise.

2.2 Acesso e contribuições do usuário ao *Maialib*

Com o computador conectado a internet, o módulo Python do *Maialib* pode ser obtido da mesma maneira que são instalados quaisquer outros módulos para Python, ou seja, digitando no terminal o comando: *pip install Maialib*.

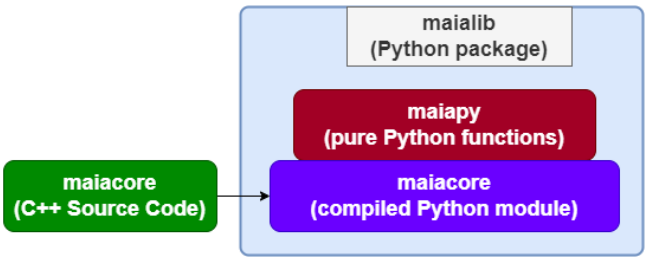
Essa é a forma mais fácil de se obter o módulo, pois o *Package Installer for Python* (PIP) faz a gestão do download e instalação do *Maialib* para o usuário, de modo que ao terminar a instalação, o usuário já possa utilizar o *Maialib* nos seus scripts Python.

Como a biblioteca é *open-source*, usuários com conhecimento técnico em C++ e/ou Python podem fazer o download do código-fonte do *Maialib* através do repositório https://github.com/nyckmaia/maialib e fazer customizações e melhorias. Essas melhorias podem ser submetidas ao repositório original e, caso aprovadas pelo moderador, poderão se tornar parte integrante das novas versões da biblioteca.

2.3 Arquitetura de software do Maialib

A biblioteca é dividida internamente em dois sub-módulos chamados *"maiacore"* e *"maiapy"*. O *maiacore* é a base de construção do *Maialib*, o qual é escrito em C++ visando performance computacional. O *maiapy* é um sub-módulo escrito em Python, o qual utiliza algumas classes anteriormente definidas no *maiacore* combinando-as com outras bibliotecas científicas usuais do Python, como por exemplo o Pandas para manipulação de dados tabulares e o Plotly para a geração de gráficos interativos. A Figura 1 mostra um diagrama da estrutura interna básica do *Maialib*.

Figura 1 – Diagrama da estrutura interna básica do *Maialib*



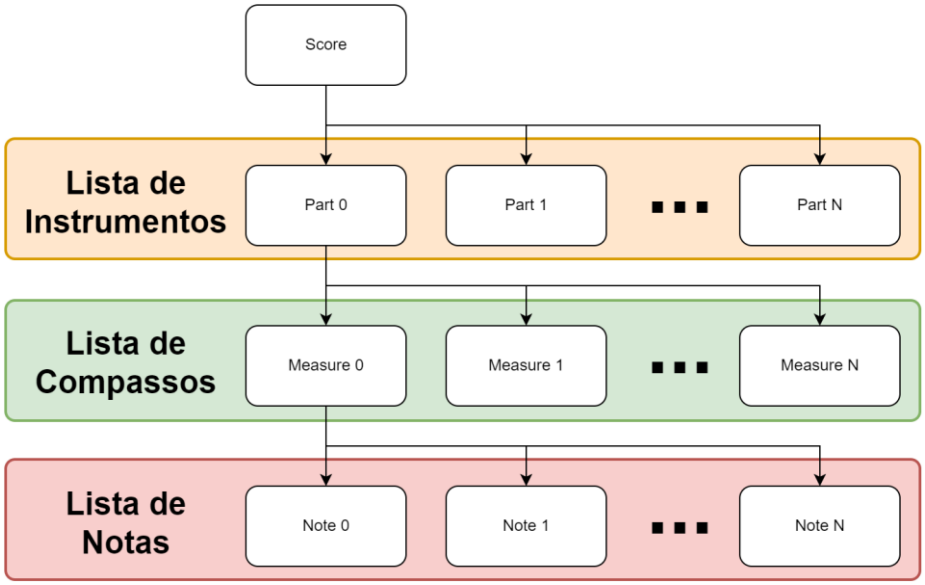
Fonte: De autoria própria

O *Maialib* segue o paradigma de Orientação a Objetos. Além disso, as classes implementadas na versão atual (v1.3.0) foram nomeadas utilizando o vocabulário cotidiano do

meio musical, de modo a facilitar a compreensão intuitiva do que cada classe pode fazer. São elas: *Note*, *Chord*, *Interval*, *Measure*, *Part*, *Score*, *ScoreCollection*, *Clef*, *Key*, *Barline* e *Helper*.

A classe responsável pela importação ou criação de uma partitura no *Maialib* é chamada de *Score*, a qual possui um sistema de árvore interna, permitindo o usuário navegar em cada objeto pertencente a partitura selecionada. A Figura 2 apresenta uma representação dessa estrutura hierárquica.

Figura 2 – Estrutura hierárquica das classes do *Maialib*



Fonte: De autoria própria

2.4 Funções de Alto Nível de Abstração

Além das classes referentes a elementos básicos da notação musical tradicional, o *Maialib* também dispõe de funções mais abstratas, as quais foram criadas combinando-se as classes básicas para criar algoritmos que realizam operações de análise musical mais complexas.

Um exemplo desse tipo de função é a *'plotPartsActivity'* a qual gera um gráfico 2D a partir das informações contidas dentro de uma partitura no formato MusicXML exibindo os momentos no tempo em que cada instrumento musical está ativo (soando) e em qual a região de altura (*pitch*) ele está sendo tocado. No exemplo abaixo (Figura 3) temos um script Python de 4 linhas o qual:

1. Importa o *Maialib* de modo a ser utilizado a partir da sigla "ml"

2. Cria um objeto do tipo *Score* a partir dos dados de um arquivo MusicXML
3. Cria o gráfico da ativação dos instrumentos no intervalo entre os compassos 200 e 300
4. Exibe o gráfico na tela do usuário (Figura 4)

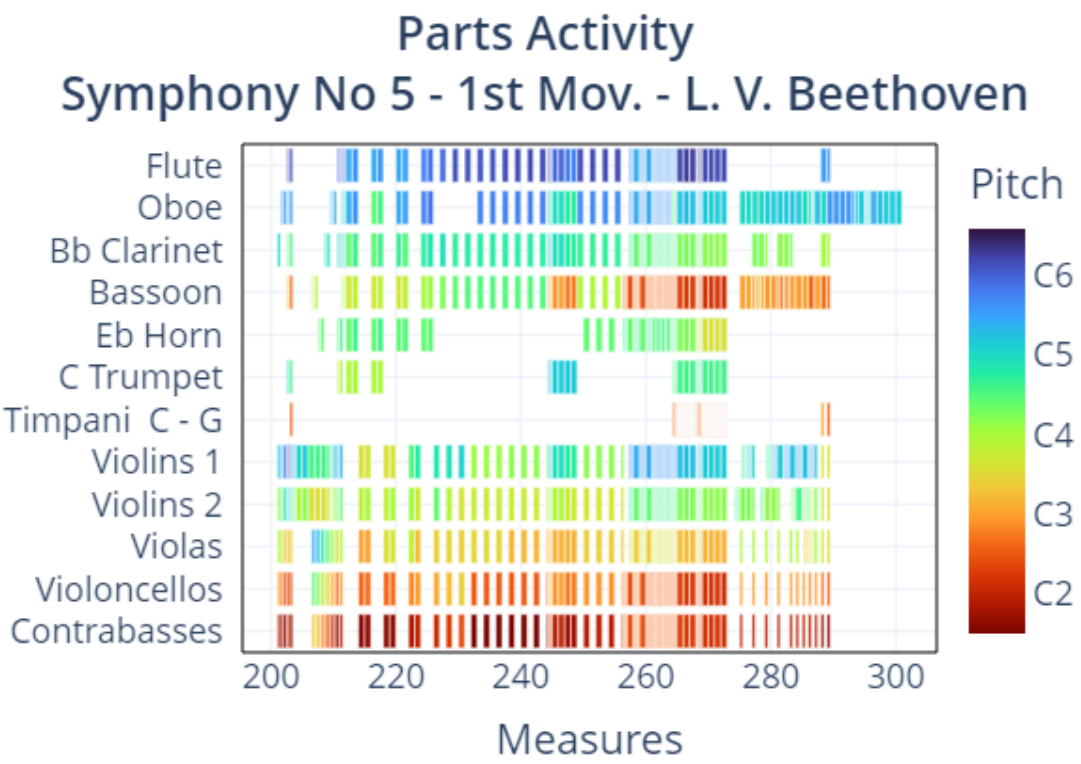
Figura 3 – Script Python 01: Utilização da função *'plotPartsActivity'*

```
import maialib as ml

myScore = ml.Score("./Beethoven-5th-Symphony-1Mov.xml")

fig, df = ml.plotPartsActivity(myScore, measureStart=200, measureEnd=300)
fig.show()

✓ 1.0s
```

Fonte: De autoria própria

Como pode-se observar no canto inferior esquerdo da Figura 3, o tempo total de processamento deste script, o qual inclui, o carregamento de todas as informações do 1º Movimento da Quinta Sinfonia de Beethoven (incluindo todos os instrumentos e todos os compassos), a criação do gráfico e a sua exibição em tela foi de apenas 1.0 segundo. Para este teste utilizamos um computador com processador Intel i7 2.20GHz, 32GB de RAM e Python 3.10. A Figura 4 exibe o gráfico gerado.

Figura 4 – Resultado gráfico da função *'plotPartsActivity'*



Fonte: De autoria própria

Um segundo exemplo de função de alto nível de abstração do *Maialib* é mostrado na Figura 5, no qual utiliza-se a função *'plotChordsNumberOfNotes'* para visualizar graficamente a variação da quantidade de notas próprias de cada acorde (empilhamentos orquestrais) no tempo, no trecho acima mencionado desta obra de Beethoven. Com a finalidade de obtermos um gráfico de linha suavizado, é passado um parâmetro opcional *'numPoints'*, o qual controla o número de pontos a serem exibidos no gráfico.

Figura 5 – Script Python 02: Utilização da função *'plotChordsNumberOfNotes'*

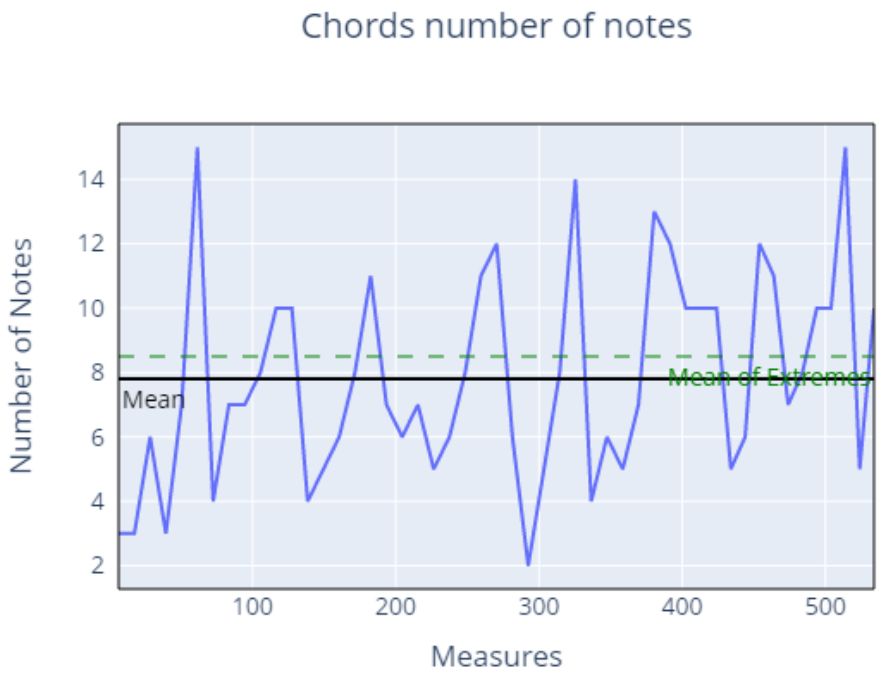
```
fig, df = ml.plotChordsNumberOfNotes(myScore, numPoints=50)
fig.show()
✓ 1.2s
```

Fonte: De autoria própria

O processamento é realizado em 1.2 segundos e na Figura 6 exibimos o resultado gráfico. Além da curva do "Número de Notas" em cada empilhamento orquestral, o gráfico

também exibe a média desses valores (linha preta contínua) e a média dos extremos (linha verde pontilhada).

Figura 6 – Resultado gráfico da função *'plotChordsNumberOfNotes'*



Fonte: De autoria própria

Com auxílio de recursos visuais como esse, pesquisadores podem, em poucos segundos, obter dados que facilitam a análise musical fornecendo novas perspectivas e possibilidades de pesquisa. Com o desenvolvimento do *Maialib*, novas funções de baixo e alto nível serão acrescentadas, aumentando a gama de possibilidades de análise e criação de novos algoritmos musicais.

2.4 Conclusão e Desafios Futuros

Neste trabalho apresentamos o *Maialib*, uma nova ferramenta de análise e composição musical assistida por computador, suas principais características, arquitetura de software e dois exemplos de utilização da biblioteca no ambiente Python. Esperamos que o desenvolvimento dessa biblioteca computacional possa trazer ao músico, professor e pesquisador musical as vantagens da computação de alta performance às suas análises e composições assistidas por

computador dentro de um ambiente de fácil interação (*user-friendly*), e integrado com outras bibliotecas de análise de dados científicos já consolidadas no meio acadêmico.

No futuro esperamos ampliar o número de funções, bem como aprimorar a documentação e criar um processo automatizado de compilação do código C++ do *Maialib* para WebAssembly (Wasm). Desse modo, desenvolvedores poderão criar aplicações musicais executadas na internet (*browser*) utilizando as mesmas classes já definidas.

3 Referências

ALOUPIS, Greg; FEVENS, Thomas; LANGERMAN, Stefan; MATSUI, Tomomi; MESA, Antonio; NUÑEZ, Yurai; RAPPAPORT, David; TOUSSAINT, Godfried. Algorithms for Computing Geometric Measures of Melodic Similarity. **Computer Music Journal**, v. 30, n. 3, p. 67–76, 2006. DOI 10.1162/comj.2006.30.3.67. Disponível em: https://www.jstor.org/stable/4617944. Acesso em: 14 jun. 2023.

CHORAL PUBLIC DOMAIN LIBRARY. [*s. d.*]. Disponível em: https://www.cpdl.org/wiki/index.php/Main_Page. Acesso em: 14 jun. 2023.

CUTHBERT, M.; ARIZA, C. Music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data. *In*: INTERNATIONAL SOCIETY FOR MUSIC INFORMATION RETRIEVAL CONFERENCE, 1 ago. 2010. [*S. l.*: *s. n.*], 1 ago. 2010. Disponível em: https://www.semanticscholar.org/paper/Music21%3A-A-Toolkit-for-Computer-Aided-Musicology-Cuthbert-Ariza/d1bacc1a26df8a3f78c78ba39193eac398c590de. Acesso em: 14 jun. 2023.

DOWNIE, J. Stephen. Music Information Retrieval (Chapter 7). **Annual Review of Information Science and Technology**. [*S. l.*: *s. n.*], 2003. p. 295–340. Disponível em: http://www.music.mcgill.ca/~ich/classes/mumt611_08/downie_mir_arist37.pdf. Acesso em: 23 jun. 2023.

EEROLA, Tuomas. MIDI toolbox: Matlab tools for music research. , p. 95, 1 jan. 2004. Disponível em: https://www.academia.edu/462611/MIDI_toolbox_Matlab_tools_for_music_research. Acesso em: 14 jun. 2023.

IMSLP. [*s. d.*]. Disponível em: https://imslp.org/. Acesso em: 14 jun. 2023.

KARYDIS, Ioannis. Symbolic Music Genre Classification Based on Note Pitch and Duration. 2006. **Advances in Databases and Information Systems** [...]. Berlin, Heidelberg: Springer, 2006. p. 329–338. DOI 10.1007/11827252_25. Disponível em: https://link.springer.com/chapter/10.1007/11827252_25. Acesso em: 14 jun. 2023.

KRANENBURG, Peter van. **A computational approach to content-based retrieval of folk song melodies**. 2010. 188 f. Utrecht University, The Netherlands, 2010. Disponível em:

https://dspace.library.uu.nl/handle/1874/179892. Acesso em: 13 jun. 2023.

MCKAY, C.; CUMMING, J.; FUJINAGA, Ichiro. JSYMBOLIC 2.2: Extracting Features from Symbolic Music for use in Musicological and MIR Research. *In*: INTERNATIONAL SOCIETY FOR MUSIC INFORMATION RETRIEVAL CONFERENCE, 2018. [*S. l.*: *s. n.*], 2018. Disponível em: https://www.semanticscholar.org/paper/JSYMBOLIC-2.2%3A-Extracting-Features-from-Symbolic-in-McKay-Cumming/bc72943095987e1013d2c73b28ab241bd8b6684a. Acesso em: 14 jun. 2023.

MUSESCORE. Musescore.com | The world's largest free sheet music catalog and community. [*s. d.*]. Disponível em: https://musescore.com/. Acesso em: 14 jun. 2023.

MUSICXML 1.0. jan. 2004. **W3C Community Group**. Disponível em: https://www.w3.org/2021/06/musicxml40/version-history/10/. Acesso em: 13 jun. 2023.

MUSICXML 4.0. jun. 2021. **W3C Community Group**. Disponível em: https://www.w3.org/2021/06/musicxml40/version-history/40/. Acesso em: 20 jul. 2023.

PETER, Manning. Electronic music and Computer music. **Electronic music and Computer music**, Oxford University Press. n. 4, p. 570, 2013. Disponível em: https://www.academia.edu/38087738/Electronic_music_and_Computer_music_Peter_Manning_Oxford_University_Press_2013. Acesso em: 13 jun. 2023.

REBELO, Ana; FUJINAGA, Ichiro; PASZKIEWICZ, Filipe; MARCAL, Andre R. S.; GUEDES, Carlos; CARDOSO, Jaime S. Optical music recognition: state-of-the-art and open issues. **International Journal of Multimedia Information Retrieval**, v. 1, n. 3, p. 173–190, 1 out. 2012. DOI 10.1007/s13735-012-0004-6. Disponível em: https://doi.org/10.1007/s13735-012-0004-6. Acesso em: 14 jun. 2023.
SCHELLENBERG, E. Glenn. Simplifying the Implication-Realization Model of Melodic Expectancy. **Music Perception: An Interdisciplinary Journal**, v. 14, n. 3, p. 295–318, 1997. DOI 10.2307/40285723. Disponível em: https://www.jstor.org/stable/40285723. Acesso em: 14 jun. 2023.

SMITH, Peter Anthony. **Computer Aided Statistical Analysis of Motive Use and Compositional Idiom**. 2012. 88 f. Thesis – University of Sydney, Sydney, 2012. Disponível em: https://ses.library.usyd.edu.au/handle/2123/9776. Acesso em: 13 jun. 2023.

THE MUTOPIA PROJECT. [*s. d.*]. Disponível em: https://www.mutopiaproject.org/. Acesso em: 14 jun. 2023.

TZANETAKIS, George; KAPUR, Ajay; SCHLOSS, W.; WRIGHT, Mathew. Computational Ethnomusicology. **Journal of Interdisciplinary Music Studies**, v. 1, p. 1–24, 1 jan. 2007. .

VINJAR, Anders; BRESSON, Jean. OpenMusic on Linux. *In*: LINUX AUDIO CONFERENCE, 2014. Germany: [*s. n.*], 2014. p. 8. Disponível em: https://hal.science/hal-01075235. Acesso em: 14 jun. 2023.

VOLK, A.; WIERING, F.; KRANENBURG, P. V. Unfolding the potential of computational

musicology. *In*: INTERNATIONAL CONFERENCE ON INFORMATICS AND SEMIOTICS IN ORGANISATIONS, jul. 2011. Leeuwarden, The Netherlands: Rene J. Jorna, Kecheng Liu, Niels Faber, jul. 2011. p. 5. Disponível em: https://www.semanticscholar.org/paper/Unfolding-the-potential-of-computational-musicology-Volk-Wiering/3850b52002b34459e5e529b0e639b69f8805ff10. Acesso em: 13 jun. 2023.

WHORLEY, R. **The construction and evaluation of statistical models of melody and harmony**. 2013. 405 f. Thesis – University of London, London, 2013. Disponível em: https://www.semanticscholar.org/paper/The-construction-and-evaluation-of-statistical-of-Whorley/1a6a4f48687b0c719709b9bbfc43c576d74fb995. Acesso em: 13 jun. 2023.