UNIVERSIDADE FEDERAL DE PERNAMBUCO

MESTRADO EM CIÊNCIA DA COMPUTAÇÂO

CENTRO DE INFORMÁTICA

2007.2

# COCHONUT: UM PROCESSO PARA RECONHECIMENTO DE ACORDES EM SEQÜÊNCIAS CAPTURADAS POR VIOLÕES MIDI

DISSERTAÇÃO DE MESTRADO
EM INTELIGÊNCIA ARTIFICIAL E
COMPUTAÇÃO MUSICAL

Aluno: Ricardo Enrique Pereira Scholz

Orientador: Geber Lisboa Ramalho

Recife, 11 de março de 2008.

*Um dia eu vi numa estrada*
*um arvoredo caído,*
*não era um tronco qualquer.*
*Era madeira de pinho*
*e um artesão esculpia*
*o corpo de uma mulher.*
*Depois eu vi pela noite*
*o artesão nos caminhos*
*colhendo raios de lua.*
*Fazia cordas de prata*
*que, se esticadas, vibravam*
*o corpo da mulher nua.*
*E o artesão, finalmente,*
*nesta mulher de madeira*
*botou o seu coração.*
*E lhe apertou contra o peito*
*e deu-lhe um nome bonito*
*e assim nasceu o violão.*

"O Violão"
Sueli Costa e Paulo César Pinheiro

## Agradecimentos

A minha mãe, pelo apoio em todas as minhas caminhadas e pela iniciativa e incentivo constante nos meus estudos de música.

A meu pai e a meu irmão por estarem sempre ao meu lado, mesmo quando eu não percebo.

A Dani, pelos tantos anos juntos, e pelos poucos separados, por todas as alegrias e tristezas, por todos os aprendizados, pela importância em parte do que sou hoje e pela amizade sempre.

Aos meus amigos, pelo apoio, pelos conselhos, pelas alegrias divididas e experiências vividas, e com quem eu aprendi e aprendo até hoje.

Aos professores do Conservatório Pernambucano de Música, especialmente a Leonardo Saldanha, pela grande importância e influência na minha formação musical.

Aos professores do Centro de Informática, por todo conhecimento transmitido e aos funcionários pela infra-estrutura proporcionada.

A Geber, em especial, por todo conhecimento e experiência compartilhados, além da importância na formação intelectual que busco.

# Resumo

A recuperação de informações musicais (*MIR – Music Information Retrieval*) é uma área de pesquisa extensa e útil para um conjunto de aplicações como identificação automática de intérprete, compositor ou gênero musical, busca e navegação em bibliotecas musicais, dentre outras.

O projeto "Um País, Um Violão", realizado no Centro de Informática da Universidade Federal de Pernambuco, se propõe a estudar a interpretação violonística no acompanhamento da bossa nova e, neste contexto, o reconhecimento de acordes tem papel fundamental. Dado que a bossa nova possui harmonia jazzística, favorecendo a utilização recorrente de dissonâncias, e os intérpretes freqüentemente realizam re-harmonizações durante a execução das peças, a maneira com a qual tais aspectos são executados torna-se uma informação importante na caracterização de um dado intérprete e no estudo de sua interpretação.

Entretanto, para realização de tal estudo, os requisitos do reconhecimento de acordes são mais exigentes do que os considerados na literatura atual. O processo proposto lida com harmonia jazzística, a mais complexa estrutura harmônica na música ocidental. Nenhum trabalho encontrado na literatura considera tal complexidade.

Infelizmente, a tecnologia atual de captura MIDI em violões é muito sensível a diversos fatores, e ainda insere uma quantidade grande de ruído nos dados capturados. Portanto, o processo inclui uma etapa de pré-processamento das seqüências capturadas para minimização de ruídos. Além disso, por motivos que serão discutidos no quinto capítulo, realizou-se um estudo para garantir que os atrasos gerados pelo captador MIDI são aceitáveis.

Para minimizar os ruídos inerentes à captura realizada pelos violões MIDI, foi proposta uma técnica baseada em duas bases de regras, executadas em série, porém realizando-se uma segmentação simples antes da execução da segunda base. A primeira base contém regras simples e elimina eventos claramente ruidosos, além de marcar potenciais ruídos para análise na segunda etapa quando mais informações estão disponíveis. A segunda base contém regras mais complexas, que consideram informações harmônicas e

contextuais locais para tratar eventos marcados na primeira fase, eliminando ou corrigindo o ruído restante.

O processo proposto para detecção de acordes nas seqüências MIDI divide o problema em três fases: primeiramente, realiza-se a segmentação da seqüência, tentando identificar os pontos de mudança de acorde através de um algoritmo que utiliza conhecimento específico do domínio. Em seguida, utiliza-se uma função de utilidade e um conjunto de padrões de acordes recorrentes em harmonia jazzística e posições comuns no braço do violão para identificar os acordes candidatos em cada segmento, construindo-se um grafo. Por fim, particiona-se o grafo encontrado em regiões de incerteza, ou seja, regiões onde há mais de um acorde candidato para cada camada, cercadas por regiões de certeza, e utiliza-se um conjunto de regras contendo padrões de seqüências recorrentes em harmonia jazzística para tentar resolver os casos ambíguos em cada sub-grafo.

Além disso, realizou-se um estudo, inédito na literatura, sobre a confiabilidade da precisão temporal da captura realizada por violões MIDI, comparando-a com a precisão obtida pelos algoritmos de detecção de ataques em sinais de áudio.

O pré-processamento obteve bons resultados na minimização de eventos ruidosos nas seqüências MIDI. Os resultados chegaram a 74,44% de precisão em reconhecimento de acordes, e foram bastante satisfatórios, especialmente com respeito à completude harmônica envolvida quando considera-se a quantidade de dissonâncias usadas na harmonia jazzística.

**Palavras-chave:** extração de informações musicais, reconhecimento de acordes, extração de acordes, segmentação musical, análise musical, violão MIDI, bossa-nova.

## *Abstract*

*Music information retrieval (MIR) is a large research area, which is useful to perform tasks such as automatic identification of musician, composer, or genre, search, retrieval and navigation on music libraries, and others.*

*The project "A Country, A Guitar"[1], from the Informatics Centre at Federal University of Pernambuco, aims to study Brazilian guitar interpretations when accompanying bossa nova and, in this context, chord recognition has a key role. Given that bossa nova has jazz harmony, favouring recurrent utilization of dissonances, and musicians often carry through re-harmonisations during the execution of the pieces, the re-harmonisations and the way in which the dissonances are used turn to be important information on characterizing a given musician and in the study of his/her interpretation. However, for the purpose of this study, requirements on chord recognition are far more rigorous than the requirements considered in current literature. The proposed process deal with jazz harmony, which is the more complex harmony structure in western music. There is no work on literature that deals with such a complexity.*

*Unfortunately, the current technology of MIDI guitars is very sensitive to several factors, and inserts a huge amount of noisy data in the captured sequences. Therefore, the proposed process includes a step for pre-processing the sequences aiming to minimize noise. Besides that, for reasons that will be explained on chapter five, a study to guarantee that the delays on MIDI capture were not greater than the precision of the current audio signal onset detection algorithms has been done.*

*In order to minimize noise, which are inherent to MIDI guitar capture technology, a rule based approach were proposed. The approach uses two rule bases, run in series, with a simple segmentation phase between then. The first rule base has simple rules aiming to eliminate clearly noisy events, and highlight potential noisy events and for detailed analysis on second phase, when more information is available. The second rule base contains more complex rules, which consider harmonic and local contextual information to minimize remaining noisy events.*

[1] **"Um País, Um Violão"**, inspired on the beginning of the lyrics of Corcovado, by Tom Jobim.

*The proposed process to chord recognition in MIDI sequences splits the problem in three phases: first, an expert knowledge based segmentation algorithm is run, trying to identify the possible points where chords change. Then, a utility function is applied to a set of recurrent chord patterns, which have been built according to jazz harmony and guitar often used chords, in order to identify the most probable chords for each segment. After that, a graph is built. Finally, the graph is split into uncertainty regions, i.e., regions in which there are more than one candidate chord for each layer, surrounded by certainty regions, and a rule base, containing common chord sequences patterns in jazz harmony, is used to resolve ambiguous cases in each part of the split graph.*

*Besides that, an original study were carried through in order to measure the reliability of time accuracy in MIDI guitar captures, comparing it with the precision obtained from attack detection algorithms in audio signals.*

*Pre-processing has reached good results on minimizing noisy events in the MIDI sequences. Results reached 74,44% of precision in chord recognition and were very satisfactory, specially regarding harmonic completeness involved when considering the large number of dissonances used in jazz harmony.*

***Keywords**: musical information extraction, chords recognition, chords extraction, music segmentation, music analysis, MIDI guitar, bossa nova.*

# Sumário

# Índice de Figuras

# Índice de Tabelas

# Índice de Quadros

# 1. Introdução

A maneira como um determinado músico interpreta uma música é uma de suas características mais importantes e de difícil aprendizagem, devido ao seu caráter intrinsecamente subjetivo e pessoal. O aprendizado automático da interpretação musical é um problema de aprendizagem de máquina que oferece diversos desafios computacionais [43, 44, 46, 47, 51], dentre eles, a aquisição de informação simbólica confiável e precisa sobre os dados do conjunto de aprendizagem, a estruturação do conhecimento e detecção dos eventuais padrões utilizados, além do próprio processo de aprendizagem, uma vez que tais informações foram adquiridas.

O projeto "Um País, Um Violão", realizado no Centro de Informática da Universidade Federal de Pernambuco, visa montar um *framework* para extrair características musicais de diversas peças executadas livremente (sem partitura) por um determinado intérprete, analisá-las e assim conseguir estruturar e entender o conhecimento subjetivo envolvido na tarefa. Além disso, para fins de validação, pretende-se aprender a reproduzir automaticamente a interpretação musical das peças estudadas de maneira que se torne impossível, ou ao menos difícil, para um ser humano distinguir se uma nova peça foi executada pelo intérprete ou pelo computador.

Algumas questões interessantes que o projeto se propõe a responder são: dada a infinidade de ritmos possíveis, que fatores levam os músicos, compositores e intérpretes, a considerarem certas combinações rítmicas em detrimento de outras? Estes fatores são os mesmos em todos os estilos musicais (por exemplo, na música erudita dos diversos períodos e nos estilos populares, como a bossa nova), isto é, estes fatores são universais? Será que existe alguma influência cultural na determinação de tais fatores?

Além disso, na música, elementos subjetivos como criatividade, emoção e intuição convivem simultaneamente com elementos formais e estruturais, que permitem a classificação de uma obra como pertencente a um determinado estilo ou compositor. Por que é possível diferenciar músicas compostas por Beethoven e por Vinícius de Moraes, ou mesmo diferenciar músicas compostas pelo mesmo compositor, em diferentes fases composicionais, ou ainda diferenciar a execução de uma mesma música por diferentes

intérpretes, como João Gilberto e Baden Powell executando a "Garota de Ipanema", de Tom Jobim e Vinícius de Moraes? Será que isto se deve apenas a um elemento cultural ou intuitivo, ou existirá algum elemento estrutural, intrínseco à própria música, ou à maneira com a qual ela é executada, que torna possível tal distinção?

Uma das maiores limitações das pesquisas sobre interpretação musical é o tamanho do *corpus* analítico utilizado pelos pesquisadores. A quantidade de dados utilizada é relativamente pequena para que se consiga de fato modelar o fenômeno e generalizar os resultados [8, 9, 31]. Por outro lado, o exame de uma grande quantidade de dados é inviável sem o auxílio do computador. A pesquisa sobre Charlie Parker, conduzida por Owens [28], visando identificar eventuais padrões melódicos em suas improvisações, levou dezesseis anos, devido ao extenso *corpus* musical que Owens analisou manualmente. Além disso, o longo tempo para realizar a pesquisa leva a algumas considerações importantes: será que o pesquisador consegue manter a coerência metodológica durante tanto tempo? Dado que este tipo de pesquisa envolve uma grande quantidade de dados, a análise manual dos mesmos não seria muito vulnerável a erros?

Por estes aspectos, faz-se necessário o uso do computador para realizar a análise dos dados. Entretanto, antes de analisá-los, é necessário obter informações simbólicas sobre os mesmos, tais como a transcrição do que foi tocado, que envolve descobrir em que instante e com que intensidade o músico tocou cada nota, os acordes[1] escolhidos, uma vez que mesmo seguindo uma mesma cifra[2], a seqüência de acordes pode ser diferente nos casos em que o músico realiza re-harmonizações[3] durante a execução da peça. Além disso, informações de *beat tracking*[4], e meta-informações como

[1] **Acorde** é um conjunto de notas relacionadas entre si soando simultaneamente ou intercaladamente. A seqüência de acordes de uma música chama-se **harmonia**. Numa peça musical, a harmonia é uma seqüência temporal fortemente dependente de contexto, ou seja, cada acorde tem forte ligação com seus vizinhos anteriores e posteriores. A harmonia tem influência direta no "humor" da música, causando tensão ou relaxamento [26].

[2] Uma **cifra** consiste numa seqüência de acordes que formam uma música, normalmente acompanhada de informação temporal de baixa granularidade (a partir de um pulso inteiro).

[3] **Re-harmonização** consiste na inclusão, exclusão e/ou substituição de acordes na harmonia de uma música, considerando-se regras que garantem a manutenção da intenção harmônica original [26].

[4] ***Beat tracking*** é a inferência da pulsação, ou andamento, em uma determinada música. Seres humanos tendem a retardar e adiantar o andamento durante a execução de peças, por isso o *beat tracking* não é uma tarefa computacional trivial, sendo tema de diversas pesquisas em computação musical.

análise harmônica funcional[1] da seqüência de acordes utilizada, e descoberta da estrutura musical[2] também são importantes.

Tais informações podem ser adquiridas tanto do áudio, quanto dos dados simbólicos, obtidos através de captura em um violão MIDI[3]. O projeto "Um País, Um Violão" possui uma base de dados contendo tanto arquivos de áudio, quanto suas respectivas seqüências MIDI[4], captadas simultaneamente por um violão MIDI.

Uma abordagem de detecção de acordes sobre as seqüências MIDI capturadas, além de ser uma tarefa mais simples que o processamento do áudio, já que faz uso de informações simbólicas, possibilita sua utilização futura em uma abordagem híbrida, utilizando MIDI e áudio gravados simultaneamente. Ou ainda permite gerar de maneira automática bases de dados rotuladas, e utilizá-las na extração de acordes em sinais de áudio baseadas em aprendizagem supervisionada.

Para poder tratar o problema de reconhecimento de acordes nas seqüências MIDI, e dado que as tecnologias atuais de captura MIDI ainda apresentam um elevado número de ruídos nas seqüências resultantes, tais como inserção de notas inexistentes, ou alteração da freqüência de algumas notas, faz-se necessário uma fase de correção de ruídos antes de qualquer processamento para descoberta de informação. Neste trabalho, uma técnica para a correção de ruídos baseada em regras foi proposta, atingindo-se bons resultados, embora o problema não tenha sido resolvido completamente.

O problema da descoberta de harmonia, tema principal desta dissertação, é dividido em dois subproblemas: segmentação e classificação. O primeiro consiste em, dada uma peça musical, identificar possíveis pontos de mudança de acordes, isto é, particionar a peça de maneira que cada segmento entre

---

[1] **Análise harmônica funcional** consiste na análise da harmonia de uma peça musical visando identificar a **função harmônica** de cada acorde, ou seja, o papel do acorde no contexto no qual está inserido [26].

[2] **Estrutura musical** é a macro estrutura harmônica de uma peça, normalmente representada por letras do alfabeto (ex.: A A B A), onde cada letra significa uma determinada seqüência de acordes, com duração específica medida em compassos ou frações de compassos (normalmente não menores que um pulso) [26].

[3] **Violão MIDI** é um violão contendo um captador MIDI, que processa o áudio captado e gera como saída uma série de eventos MIDI.

[4] **MIDI** (*Musical Instrument Digital Interface*) é um protocolo que visa permitir a comunicação, controle e sincronização entre instrumentos musicais eletrônicos, computadores e outros equipamentos. Uma **seqüência MIDI** é formada por um conjunto de **eventos MIDI** que indicam, por exemplo, quando uma determinada nota deve começar ou parar de ser executada. Assim, o áudio proveniente da execução das seqüências MIDI não é o áudio original capturado, e sim uma síntese do mesmo, realizada através de um sintetizador.

duas partições contenha somente um acorde. O segundo subproblema, dado que a peça já foi particionada, consiste em classificar o conjunto de notas existentes dentro do segmento como um acorde. Vale ressaltar que tal classificação é ambígua, pela natureza ambígua dos acordes, já que um mesmo conjunto de notas pode representar diversos acordes. Tal classificação deve levar em consideração também informações de contexto, ou seja, os acordes que precedem e os acordes que sucedem o segmento em questão, já que é sabido que uma seqüência harmônica numa música tem forte dependência contextual.

Este problema pode ser mapeado como um problema de *clustering*[1] de objetos de um mesmo grupo numa seqüência temporal, seguido de uma busca de uma função de utilidade para mapeamentos entre os elementos de cada *cluster*, que formam uma imagem, e elementos de um domínio; e uma vez que se tem uma lista de prováveis elementos do domínio pra cada instante de tempo, uma busca do caminho mais adequado num grafo.

A técnica proposta neste trabalho tenta resolver primeiramente o problema da identificação das mudanças de acorde, de maneira isolada. Isto é, identificam-se trechos em que não há mudança de acorde, para só então tentar descobrir dentre os acordes possíveis em cada segmento, qual deles é o mais adequado no contexto em questão.

A técnica de identificação das mudanças de acorde, ou particionamento e segmentação da música, sofreu diversos ajustes à medida que os experimentos revelavam deficiências, como o superparticionamento[2] e a hipersensibilidade à adição ou subtração de notas.

Devido a características particulares das gravações MIDI, como a não-quantização[3] ou existência de ruídos, que não ocorrem nas transcrições de partituras para seqüências MIDI, o algoritmo utilizado como referência [30] apresenta uma hipersensibilidade à adição e subtração de notas num segmento, identificando uma quantidade muito maior de segmentos do que os

[1] ***Clustering*** é uma técnica de mineração de dados para fazer agrupamentos automáticos de dados segundo seu grau de semelhança. **Problemas de *clustering*** são problemas cuja solução exige o agrupamento automático de dados.
[2] **Superparticionamento**, no contexto deste trabalho, significa a identificação exagerada de pontos de partição.
[3] **Quantização** é o processo de alinhamento dos ataques de uma seqüência MIDI a uma grade temporal precisa.

segmentos de fato existentes. Assim, foram realizadas diversas adaptações sobre o algoritmo inicial, chegando-se a resultados bastante satisfatórios no particionamento das seqüências, considerando a originalidade da solução, já que não existem soluções na literatura para as dificuldades envolvidas no processamento de dados adquiridos por violões MIDI.

A definição da função de utilidade[1] para escolher os melhores acordes para cada segmento foi baseada na abordagem de Birmingham e Pardo [30], embora se tenha substituído quase que inteiramente os padrões de acordes propostos pelos autores por novos padrões, de forma a considerar a harmonia jazzística [1, 2, 4] e as inversões[2] mais comuns à execução violonística [11, 13]. Tal técnica consiste em definir padrões de acordes genéricos que são comparados com cada conjunto de notas, para cada tônica[3] possível, ou seja, para cada uma das doze notas da escala cromática, predominante na música ocidental, e para cada um dos padrões definidos, a função de utilidade utiliza três critérios: notas comuns a ambos os conjuntos (padrão e segmento), notas faltantes no padrão e notas faltantes no segmento sendo analisado.

A simples escolha pelo acorde com maior valor de utilidade leva, em diversos casos, a classificações erradas, o que sugere que a função de utilidade não é ótima. Entretanto, não seria de se esperar que o fosse, já que se trata de uma função local, a qual não leva em consideração os segmentos anteriores nem posteriores ao segmento em questão. Para chegar a um resultado mais confiável e lidar com situações ambíguas, é preciso definir os acordes mais prováveis para cada segmento, e em seguida construir um grafo, realizando-se então uma busca pelo caminho mais adequado, através de técnicas que façam uso de informação contextual. A abordagem proposta seleciona o conjunto de acordes mais prováveis em cada segmento definindo um limiar de pontuação acima do qual uma dada classificação é considerada provável.

[1] **Função de utilidade** é uma função matemática, e portanto objetiva, utilizada para mapear o nível de adaptação de elementos de um domínio, a elementos de uma imagem.
[2] As **inversões** de um acorde são as diversas combinações de organização das notas do acorde para execução de fato no instrumento, muitas vezes suprimindo algumas das notas pela inviabilidade ergonômica da execução do acorde completo num determinado instrumento [26].
[3] **Tônica** é a nota principal de um acorde, que define seu nome e a partir da qual são calculados todos os intervalos para as outras notas do acorde [26].

Observou-se, entretanto, que em boa parte das classificações corretas, os acordes vencedores obtinham pontuação extremamente maior do que os outros acordes, não deixando que nenhum outro acorde chegasse ao limiar definido. Considerando então que a função de utilidade proposta consegue em alguns casos definir com certo grau de certeza o acorde representado por um segmento, mesmo sem o uso de informação contextual, a abordagem proposta utiliza os acordes cujo grau de certeza é alto, dividindo o grafo construído em sub-grafos que englobam as zonas de incerteza, cercadas por zonas de certeza. Realiza-se então a análise de tais sub-grafos através da aplicação de regras, que consideram os aspectos contextuais da seqüência em questão e fazem uso da informação confiável das zonas de certeza no início e final de cada sub-grafo.

A abordagem proposta para reconhecimento de acordes foi implementada e executada em três gravações de bossa-nova ao violão, rotuladas manualmente por um especialista. Os resultados obtidos no reconhecimento de acordes nas músicas do conjunto de teste chegaram a 74,44% de precisão e foram bastante satisfatórios, especialmente considerando-se a complexidade envolvida no problema, inerente à harmonia jazzística.

Além disso, visando obter uma medida da confiabilidade temporal das informações descobertas a partir das seqüências MIDI, realizaram-se, no decorrer deste trabalho, experimentos inéditos na literatura que garantiram que a precisão temporal dos eventos MIDI capturados é maior ou igual à precisão dos métodos de detecção de ataques em arquivos de áudio. Portanto, tornam-se desnecessárias quaisquer tentativas de melhoria da qualidade de tais dados.

No capítulo a seguir o problema do reconhecimento de acordes em informações simbólicas será descrito em mais detalhes. No capítulo 3, será realizada uma análise das abordagens atuais para o problema da extração de harmonia, tanto em sinais de áudio quanto em dados simbólicos. No quarto capítulo, a abordagem proposta nesta dissertação será apresentada em detalhes. No capítulo 5, os experimentos sobre a confiabilidade temporal das capturas realizadas por violões MIDI será detalhado. Em seguida, no sexto capítulo, a implementação da abordagem proposta será explicada. No capítulo 7, os experimentos e os resultados obtidos serão expostos, juntamente com

uma análise dos mesmos, confrontando-os com os requisitos de uma boa solução para o problema. Por fim, no oitavo capítulo, será realizada uma análise da abordagem proposta e dos pontos que podem permitir a obtenção de resultados melhores futuramente.

# 2. Extração de Informação Harmônica

As partituras musicais utilizam um conjunto de símbolos para representar uma música. Entretanto, tal conjunto proporciona um mapeamento apenas aproximado e subjetivo da peça a ser executada. Na divisão rítmica[1] de uma música não se escrevem as notas em tempos ligeiramente posteriores ou anteriores aos tempos exatos da divisão, tanto por não existir símbolos suficientes para tal, quanto por tornar sua leitura inviável; logo, escreve-se uma divisão rítmica aproximada, suficiente para que um intérprete execute a música corretamente e consiga realizar a leitura com certa facilidade. Além disso, os símbolos de interpretação, como os símbolos de andamento (*rallentando*, *accelerando*, *etc*)[2] ou de intensidade (*pp*, *mp*, *p*, *f*, *mf*, *ff*, *<*, *>*, *etc*)[3], sugerem comportamentos, mas não explicitam exatamente como devem ser executados, deixando a cargo do intérprete executá-los da maneira que achar mais adequada. A figura 2.1 exemplifica o uso de símbolos de interpretação e intensidade.

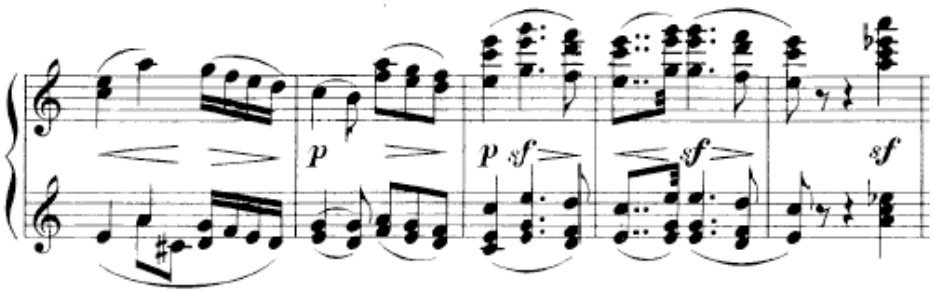


Figura 2.1 – Trecho do primeiro piano da *overture* de *Prometheus*, de Ludwig Van Beethoven, para piano a quatro mãos: uso freqüente de símbolos de interpretação de intensidade.

Em boa parte dos casos, dependendo do estilo musical e muito freqüentemente na música popular, as partituras sequer fornecem símbolos de interpretação, e em alguns casos, as músicas são executadas com base numa

[1] A **divisão rítmica** de uma peça é uma convenção sobre a divisão da pulsação da mesma, na maioria dos casos de forma exponencial numa base *m*, onde normalmente $m = 2$, ou seja, as figuras musicais têm sempre duração $p \div m^n$ ou $p \times m^n$, com $n$ inteiro, $n \geq 0$ e $p$ sendo a duração de um pulso [26].
[2] Os termos ***rallentando*** e ***accelerando***, do italiano, sugerem que o músico diminua ou aumente, respectivamente, o andamento da execução [26].
[3] Os símbolos ***pp*** (*pianíssimo*), ***mp*** (*meso piano*), ***p*** (*piano*), ***f*** (*forte*), ***mf*** (*meso forte*) e ***ff*** (*fortíssimo*) sugerem que o músico modifique a intensidade da execução adequadamente. Já os símbolos **>** e **<** sugerem que o músico diminua ou aumente, respectivamente, a intensidade de execução de maneira gradual [26].

grade de acordes[1] contendo apenas informação harmônica e temporal – neste caso assume-se que o intérprete já conhece a melodia da música.

A interpretação de uma partitura ou grade de acordes por um músico é uma das características mais marcantes na execução humana de peças musicais, e também uma das mais difíceis habilidades a serem desenvolvidas pelos músicos iniciantes, dado o seu caráter subjetivo e pessoal, e fortemente ligado ao sentimento do músico sobre a peça em questão e às suas experiências musicais anteriores.

O aprendizado automático de tal interpretação, e a análise, estruturação e entendimento do conhecimento subjetivo envolvido na mesma, são problemas de aprendizagem de máquina e reconhecimento de padrões que oferecem diversos desafios computacionais [43, 47]. Dentre eles, a aquisição de informações simbólicas confiáveis e precisas sobre os dados do conjunto de aprendizagem, ou o reconhecimento dos padrões utilizados pelos intérpretes, além da própria aprendizagem, uma vez que tais informações foram adquiridas. Idealmente, o resultado deste processo seria um sistema que explica, de forma objetiva, o conhecimento subjetivo implícito na interpretação de um determinado músico, ou ainda, para fins de validação, interpreta peças musicais escritas de forma que um ouvinte humano não seria capaz de identificar se o intérprete é um humano ou uma máquina, uma espécie de Teste de Turing da interpretação musical, como mostrado na figura 2.2. Gravações de um violonista são utilizadas como entrada para o sistema “Um País, Um Violão”. A partir de então, uma nova peça, inexistente no conjunto inicial é executada pelo intérprete e sintetizada pelo sistema, que explica o conhecimento utilizado para a síntese, e ambos os áudios resultantes são submetidos a um ouvinte, que tenta distinguir por que fonte cada um foi gerado.

[1] Uma **grade de acordes** consiste numa seqüência de acordes que formam uma música, normalmente acompanhada de informação temporal de baixa granularidade (a partir de um pulso inteiro).


Figura 2.2 – “Teste de Turing” da interpretação musical: o computador aprende com gravações de um músico e um leigo tenta identificar se um dado arquivo de áudio foi gravado por um intérprete ou gerado por um computador.

Um intérprete utiliza as informações simbólicas contidas numa partitura ou grade de acordes, e as meta-informações inferidas a partir delas, para compor a sua própria interpretação da peça musical. De maneira inversa, as informações simbólicas adquiridas da transcrição da execução de um intérprete, bem como as meta-informações inferidas a partir delas, podem ser então utilizadas para explicitar os aspectos particulares de sua interpretação. Portanto, para viabilizar a utilização de qualquer técnica de aprendizagem de máquina visando aprender sobre a maneira como um determinado intérprete executa uma peça musical, se faz necessário obter dados simbólicos confiáveis e precisos sobre as execuções do intérprete estudado, para que se estabeleça um mapeamento entre entrada e saída a ser utilizado pelo mecanismo de aprendizagem, entre a partitura ou grade de acordes e a execução realizada.

No projeto “Um País, Um Violão”, realizaram-se gravações de intérpretes acompanhando bossa nova em violões MIDI, e gravaram-se simultaneamente às seqüências MIDI, os sinais de áudio capturados. No caso da captação do áudio, o resultado é um arquivo de áudio contendo as freqüências inferidas a cada instante de tempo. Já no caso da captação MIDI, a informação de áudio é processada em *hardware*, em tempo real, gerando eventos que descrevem de forma simbólica as notas executadas. Na literatura, pode-se dividir as abordagens sobre reconhecimento de acordes em dois grandes grupos: o reconhecimento sobre seqüências MIDI [30, 41] e o reconhecimento sobre sinais de áudio [3, 15, 19, 20, 21, 22, 37, 38,39, 41, 50].

Considerando que o projeto “Um Pais, Um Violão” possui tanto as saídas MIDI quanto os sinais de áudio capturados simultaneamente, optou-se por realizar o reconhecimento de acordes a partir das seqüências MIDI por dois motivos principais: o primeiro é baseado no princípio da Navalha de Occam, ou seja, dado que é mais simples inferir tais informações em seqüências de dados simbólicos que em sinais de áudio, é evidente que se tente realizar a tarefa partindo das seqüências MIDI. O segundo, e principal motivo, é a possibilidade de utilizar as informações harmônicas obtidas em uma abordagem híbrida, considerando seqüências MIDI e sinais de áudio gravados simultaneamente, ou ainda partir do reconhecimento de acordes nas seqüências MIDI para gerar automaticamente dados rotulados, e utilizá-los em

abordagens baseadas em aprendizagem supervisionada para reconhecimento de acordes em sinais de áudio [19, 20, 21].

Um problema que antecede o processamento de fato das seqüências geradas pela captura MIDI consiste em saber se a precisão temporal dos ataques[1] é satisfatória. Ou seja, uma vez que o intérprete tocou uma nota, o evento NOTE ON aparece na seqüência MIDI com um atraso, e deve-se mensurar se tal atraso é significante ou pode ser considerado desprezível. É razoável considerar que a precisão temporal da captura MIDI seja satisfatória se esta for melhor ou igual à precisão temporal de um bom algoritmo de reconhecimento de ataques em arquivos de áudio. Assim, pode-se garantir que qualquer informação inferida a partir das seqüências MIDI geradas será tão ou mais confiável, do ponto de vista temporal, que as informações adquiridas através dos algoritmos de detecção de ataques em arquivos de áudio. A figura 2.3 a seguir ilustra melhor o que é preciso mensurar: as diferenças de tempo entre o momento ataque realizado pelo intérprete e o momento do ataque detectado tanto no áudio quanto no MIDI.

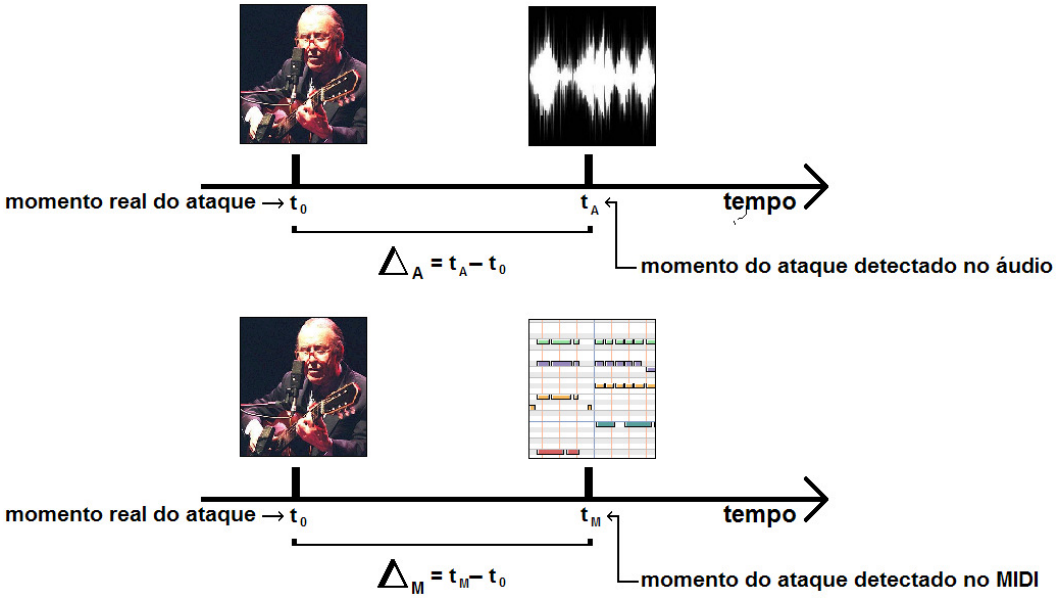


Figura 2.3 – Medições sobre a precisão da detecção em áudio e em MIDI.

A tecnologia atual de captura em violões MIDI apresenta limitações que inserem ruídos nas seqüências resultantes. Este processo é extremamente sensível a fatores difíceis de controlar, como temperatura do ambiente,

[1] Considera-se como **ataque** o início da execução de uma nota em um instrumento.

afinação do violão, maneira como o músico executa as peças, dentre outros[1] [49], e mesmo que tais fatores sejam ideais, a quantidade de ruídos diminui, mas não desaparece. Isto torna a construção do ambiente de gravação ótimo uma tarefa penosa, complicada e de difícil realização. Os principais ruídos são eventos NOTE ON e NOTE OFF inexistentes adicionados à seqüência MIDI resultante, e que portanto devem ser excluídos da mesma, ou eventos de nota cujas freqüências foram reconhecidas erroneamente, devendo-se então corrigi-los. Na figura 2.4 são mostrados exemplos de ruídos comuns inseridos nas seqüências capturadas por violões MIDI.

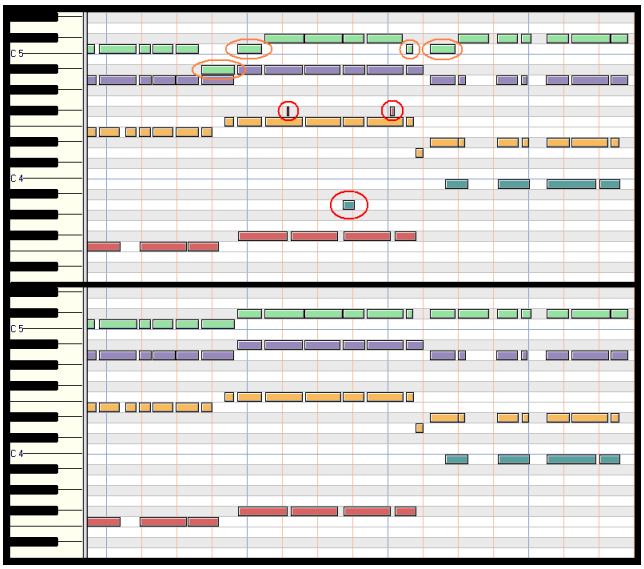

Figura 2.4 – Exemplos de ruídos comuns em capturas de violões MIDI: notas indevidamente adicionadas (em vermelho) e notas capturadas com engano de freqüência (em laranja).

A base inicialmente gravada no projeto “Um País, Um Violão” teve os ruídos manualmente corrigidos por Lima [24]. Segundo o próprio autor, cada minuto de música levou em torno de oito horas para ser corrigido. Assim, a correção manual compromete a escalabilidade do projeto, além de não dar garantias que o especialista conseguiria manter a coerência durante a correção de todas as músicas gravadas.

Isso justifica a realização de um pré-processamento nas seqüências geradas, a fim de minimizar os ruídos mais comuns nas gravações de violões MIDI. Entretanto, é importante que haja a garantia que o pré-processamento

[1] Além das discussões no fórum Yahoo MIDI Guitar [49], o fabricante Gibson®, através de contato por e-mail, também forneceu informações neste sentido.

não irá modificar as notas não ruidosas, mesmo que isso acarrete em um pior desempenho na correção das notas ruidosas. Afinal, não faz sentido a utilização de pré-processamento para correção de ruídos, se este insere novos ruídos em pontos onde não havia ruído anteriormente, ou modifica os ruídos existentes sem de fato corrigi-los.

O problema do reconhecimento de harmonia pode ser modelado como um problema inicial de *clustering* de uma seqüência temporal de símbolos, seguido de um problema de classificação dos *clusters* encontrados em elementos de uma imagem, onde há dependência contextual, ou seja, a classificação de um *cluster* depende das classificações de seus vizinhos anteriores e posteriores. A figura 2.5 ilustra este processo. A seqüência inicial é dividida em conjuntos de eventos que contém características em comum (domínio), em seguida, cada conjunto é classificado como pertencente a um elemento de um conjunto de símbolos (imagem).

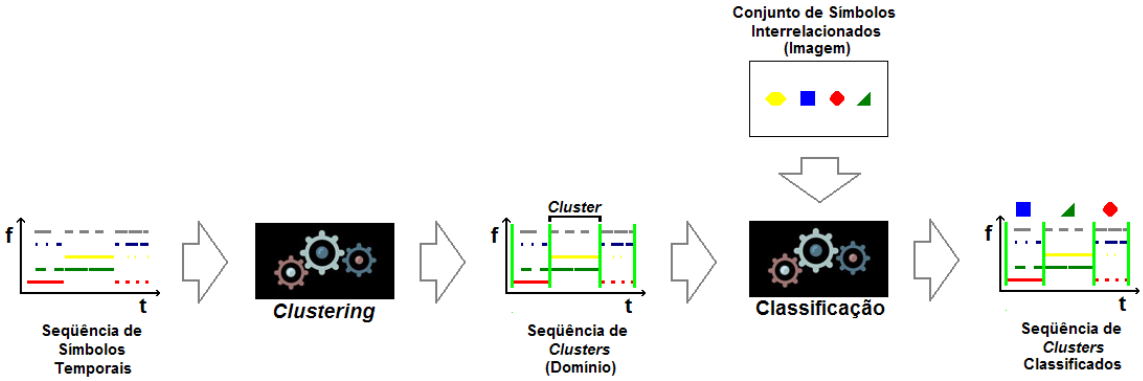


Figura 2.5 – Descrição genérica do problema

O problema abordado por esta dissertação consiste em extrair um determinado tipo de informação musical, a informação harmônica, a partir de seqüências MIDI gravadas por intérpretes acompanhando bossa-nova ao violão, como mostrado na figura 2.6. A partir da informação harmônica, é possível também inferir meta-informações sobre a estrutura harmônica, através do uso de sistemas de análise harmônica automática [33, 34, 35].

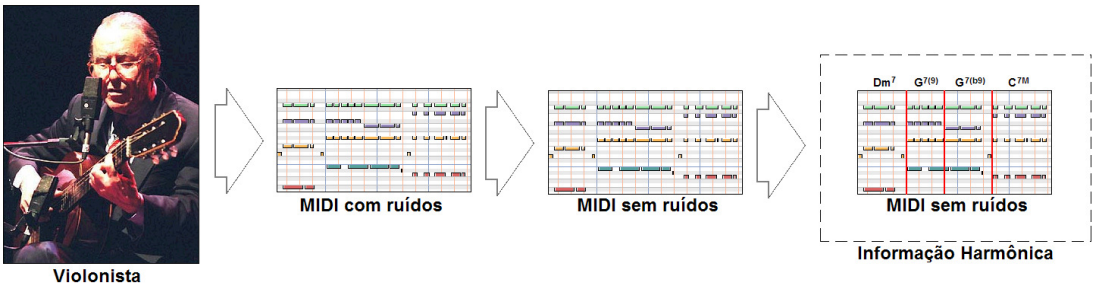


Figura 2.6 – Descrição do problema: obtenção de informação harmônica (reconhecimento de acordes) a partir de seqüências capturadas por violões MIDI.

Observam-se então dois problemas distintos: o primeiro consiste na segmentação da música em trechos que correspondem a um único acorde, ou seja, a identificação dos pontos em que há mudanças de acorde, como exemplificado pelas linhas verticais vermelhas na figura 2.6; e o segundo, consiste na classificação de tais trechos como representando um determinado acorde, dos vários que um conjunto de notas pode representar.

O primeiro problema, de segmentação, apresenta como principais dificuldades computacionais o caráter subjetivo do que pode ser considerado como uma mudança de acorde, em alguns casos, e a existência de notas de passagem[1].

Alguns músicos tendem a não executar durante toda a duração de um acorde as dissonâncias sugeridas pela grade de acordes, ou ainda executar tais alterações de maneira intermitente, o que torna este tipo de situação difícil de distinguir das situações onde de fato o músico deve alternar as dissonâncias por sugestão da grade de acordes, devido ao efeito auditivo peculiar normalmente causado por tal alternância. Sendo assim, no caso da alternância espontânea a grade de acordes exibe apenas um acorde, enquanto que nos casos de alternância sugerida, ambos os acordes são exibidos. A tabela 2.1, a seguir, mostra um exemplo de cada caso. Nos casos de alternância espontânea, o intérprete modifica uma ou mais dissonâncias do acorde, acabando por gerar uma seqüência de acordes diferente da original.

[1] **Notas de passagem** são notas que não fazem parte de um acorde, e são inseridas na execução por questões estéticas, normalmente por curtos períodos de tempo quando da eminência de mudança de acordes ou inversão [26].

Tabela 2.1 – Exemplo de alternância obrigatória e alternância espontânea.

| | Alternância Obrigatória | Alternância Espontânea |
|---|---|---|
| Grade de Acordes | $Dm^7$ $G^{sus4(7)(9)}$ $G^{7(9)}$ $C^{7M}$ | $Dm^7$ $G^{7(b9)}$ $C^{7M}$ |
| Execução | $Dm^7$ $G^{sus4(7)(9)}$ $G^{7(9)}$ $C^{7M}$ | $Dm^7$ $G^{7(\#9)}$ $G^{7(b9)}$ $C^{7M}$ |

Além disso, muitos músicos utilizam notas de passagem durante a execução que, por sua natureza alheia ao acorde, dificultam tanto a segmentação da peça em trechos onde não há mudança de acordes, já que se torna difícil reconhecer quando uma determinada nota é uma nota de passagem ou uma mudança de fato de acorde, quanto a classificação dos segmentos encontrados, pois não se sabe *a priori* que notas num segmento são parte do acorde e que notas são notas de passagem. As notas de passagem normalmente acontecem nos tempos fracos, têm curta duração e nunca devem ser consideradas na classificação dos acordes. Na figura 2.7, é apresentado um exemplo da ocorrência de tais notas.

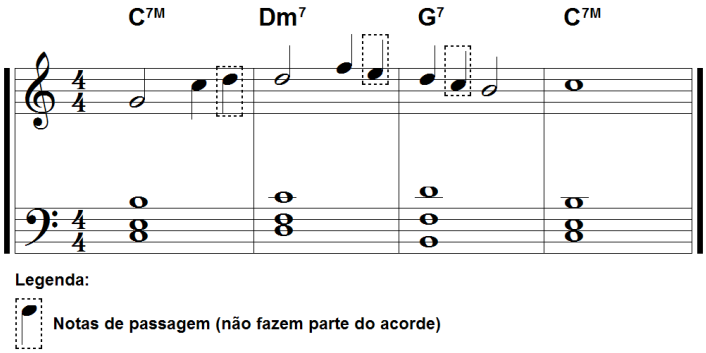


Figura 2.7 – Exemplos da utilização de notas que não fazem parte do acorde, as chamadas notas de passagem.

Particularmente, dados adquiridos de violões MIDI não são quantizados. Um dos motivos para tal são os atrasos e adiantamentos de ataque inerentes à execução humana, quando mais de uma nota é tocada simultaneamente, como no caso das puxadas[1]. Na figura 2.8 a seguir, percebe-se que em (A) todos os ataques e as durações das notas estão totalmente alinhados à grade.

[1] **Puxada** é a execução de duas ou mais cordas simultaneamente, no acompanhamento de alguns ritmos. Tal comportamento é típico na bossa nova, onde o dedo polegar executa os baixos, e os dedos indicador, médio e anular executam as notas dos acordes de maneira simultânea, através de puxadas.

Já em (B), os ataques e as durações das notas sofrem ligeiras discrepâncias com relação à grade, e entre si.

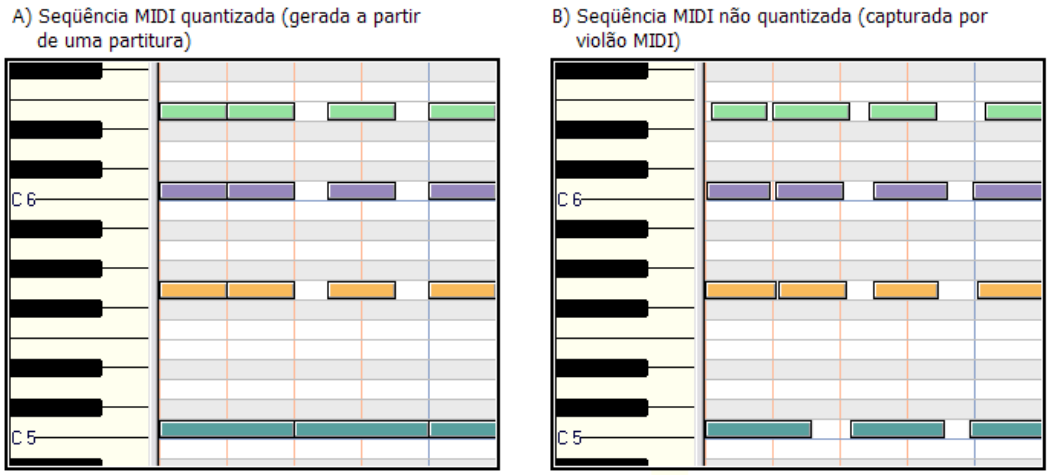


Figura 2.8 – Comparação entre o MIDI quantizado (A) e o MIDI contendo os atrasos e adiantamentos inerentes à execução humana (B).

O problema da classificação dos conjuntos de notas de cada segmento em acordes tem como principal desafio computacional a ambigüidade e dependência de contexto dos acordes, ou seja, um dado conjunto de notas pode representar diversos acordes, dependendo do contexto no qual está inserido. Além disso, a própria escolha do acorde que um conjunto de notas representa num dado contexto, em alguns casos, é subjetiva e pode variar de músico para músico. Normalmente, um acorde é formado por uma tônica (nota principal e que dá nome ao acorde), uma terça e uma quinta (notas que, juntas, definem se o acorde é maior, menor, diminuto ou aumentado) e outras notas, chamadas de dissonâncias. Eventualmente, a terça pode não fazer parte do acorde (no caso dos acordes suspensos, onde a quarta justa substitui a terça) [26]. Um exemplo clássico de um conjunto de notas que pode representar vários acordes é o caso dos acordes diminutos. C°, Eb°, Gb° e A° têm exatamente as mesmas notas (dó, mi bemol, sol bemol e lá – simplificando-se as enarmonias[1]). A escolha do acorde mais adequado, ou seja, a definição sobre qual dessas notas é a tônica, só pode ser realizada considerando-se o contexto no qual o segmento está inserido, pela utilização

[1] Notas **enarmônicas** são notas semanticamente diferentes, mas que uma vez executadas produzem a mesma freqüência e, portanto, o mesmo som. Pode-se estender este conceito a acordes, quando os conjuntos de notas de ambos os acordes são enarmônicos [26].

de informações harmônicas, caso contrário, qualquer das classificações pode ser considerada correta.

Um outro exemplo comum é o conjunto de notas E, G, Bb e D, que podem representar um $Em^{7(b5)}$ ou um $Gm^{13}$. Situações mais complexas podem acontecer quando a quantidade de dissonâncias consideradas na análise é grande, como no conjunto de notas C, D, F e A, que pode representar um $Dm^{7}$, um $C^{sus4(9)(13)}$, um F6 ou mesmo um $A^{sus4(\#9)(b13)}$, dependendo do contexto em questão. Além disso, escolhas de nomenclatura entre acordes enarmônicos também só podem ser realizadas considerando-se informação contextual: $C\#m^{7}$ e $Dbm^{7}$, por exemplo, são acordes harmonicamente iguais, porém semanticamente diferentes, e a decisão sobre qual deles deve ser aplicado num determinado trecho de música depende exclusivamente do contexto harmônico em que estão inseridos. Entretanto, o problema da enarmonia já foi resolvido satisfatoriamente por abordagens que realizam análise harmônica na seqüência de acordes, considerando padrões de seqüências de acordes recorrentes em harmonia jazzística e regras para identificar casos como empréstimos modais e modulações, e modificam a nomenclatura dos acordes de acordo com a tonalidade identificada, ou com as tonalidades já utilizadas anteriormente na mesma música [33].

Por fim, em alguns casos raros onde o contexto harmônico permite mais de uma solução, mantendo a coerência harmônica, a escolha do acorde mais adequado torna-se subjetiva e dependente das experiências anteriores do especialista.

Uma boa solução para o problema de extração de harmonia deve contemplar requisitos de corretude e completude, temporais e harmônicas. A *completude temporal* diz respeito ao reconhecimento de todos os pontos de mudança de harmonia, enquanto que a *corretude temporal* deve garantir que nenhum ponto de mudança de harmonia será reconhecido se não representar uma mudança real de harmonia, ou seja, a solução deve identificar os menores números possíveis de falso-positivos e falso-negativos no reconhecimento dos pontos de mudança de harmonia, e com a menor diferença de tempo possível para o ponto real de mudança.

A abordagem proposta deve ser também *harmonicamente correta*, ou seja, deve reconhecer os acordes corretos para cada conjunto de notas

representado por um segmento. Um acorde pode ser considerado correto quando sua tônica, terça[1] e quinta, foram reconhecidas corretamente, o que já permite classificá-lo como um dos tipos básicos de acorde possíveis: maior, menor, diminuto ou meio-diminuto e aumentado.

Para a maioria dos estilos musicais ou dos campos de estudo, a classificação entre acordes maiores, menores e diminutos pode ser considerada uma boa modelagem do problema do reconhecimento de acordes. Entretanto, para o estudo da bossa-nova, cuja harmonia é predominantemente jazzística, e especialmente o estudo de sua execução ao violão, faz-se necessário o reconhecimento completo dos acordes executados, incluindo-se todas as possíveis alterações[2], já que a maneira como são utilizadas tem papel fundamental na caracterização da interpretação.

Além disso, a execução da bossa-nova ao violão tem severas restrições quanto ao número máximo possível de notas soando simultaneamente, que em geral se limita a quatro. Assim, com freqüência, os padrões de acordes no braço do violão fazem soar preferencialmente a tônica e as dissonâncias, em detrimento de notas cuja importância harmônica é menor, como as quintas. O exemplo da figura 2.9 ilustra a supressão da quinta, num acorde maior com sétima maior e nona, para que se façam soar a tônica, a terça e as dissonâncias (a sétima menor e a nona maior).

[1] A **terça** de um acorde é a nota que define se o acorde será maior ou menor, de acordo com o intervalo de terça maior ou terça menor, respectivamente, entre ela e a tônica do acorde. De maneira análoga pode-se definir a quinta, sétima, nona, décima primeira e décima terceira de um acorde. Pela combinação da terça e da quinta pode-se classificar um acorde em maior (terça maior e quinta justa), menor (terça menor e quinta justa), diminuto (terça menor, quinta diminuta e sétima diminuta), meio diminuto (terça menor, quinta diminuta e sétima menor) ou aumentado (terça maior e quinta aumentada) [26].

[2] **Alteração** é a adição de notas a um acorde de um tipo básico (maior, menor, diminuto ou aumentado). Tais notas são também chamadas de **dissonâncias** ou **acidentes**. São consideradas alterações as seguintes notas: sétimas, nonas, décimas primeiras e décimas terceiras, além de suspensões (adição da quarta justa e retirada da terça) [26].

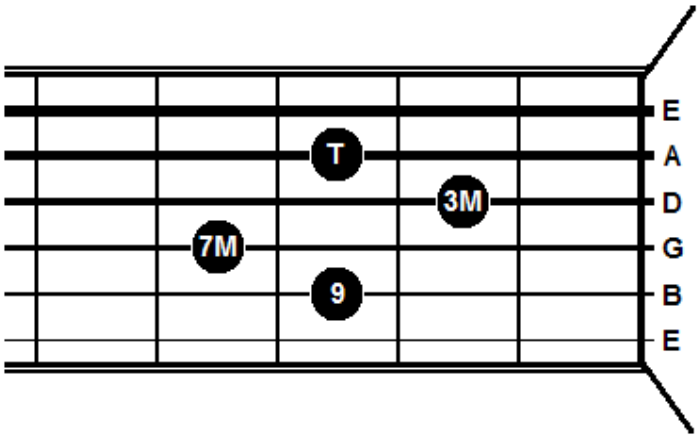


Figura 2.9 – Uma das posições do acorde $C^{7M(9)}$ no braço do violão. Nota-se que a quinta é suprimida, em favor de notas harmonicamente mais importantes, como a terça, a sétima e a nona, além, é claro, da tônica.

Desta maneira, é interessante que a solução proposta seja, por fim, *harmonicamente completa*, isto é, o reconhecimento de um dado acorde inclui não só suas notas fundamentais (tônica, terça e quinta), como todas as alterações possíveis: sétimas, nonas, décimas primeiras, décimas terceiras e suspensões[1].

De maneira sintetizada, o problema de reconhecimento de acordes consiste na divisão da seqüência MIDI em pontos de particionamento que indicam mudanças de acorde, seguida da classificação de cada conjunto de notas em um dado acorde. A solução deve ser temporalmente correta e completa, ou seja, deve identificar com exatidão os pontos de mudanças de acorde, e harmonicamente correta e completa, isto é, deve classificar cada conjunto de notas corretamente e incluir na classificação todas as dissonâncias existentes no acorde em questão.

[1] **Suspensão** é um tipo de alteração onde troca-se a terça do acorde por uma quarta justa. Esta alteração difere da adição de uma décima primeira (enarmonicamente equivalente à quarta justa) porque no acorde suspenso não há terça, enquanto que no acorde com décima primeira adicionada existe uma terça maior ou menor [26].

# 3. Estado da Arte

A extração de características musicais, a partir de sinais de áudio ou de seqüências MIDI, é um passo importante para a realização de diversas outras tarefas na computação musical, como busca ou segmentação de músicas, identificação de similaridade, dentre outros. Além disso, grande parte das abordagens que utilizam aprendizagem de máquina para extrair informações musicais a partir de sinais de áudio, faz uso de bases etiquetadas geradas a partir de dados simbólicos [19, 20, 21], normalmente seqüências MIDI.

Particularmente, com respeito à extração de acordes, a grande maioria das abordagens se restringe a um conjunto reduzido de tipos de acordes, sendo capaz de reconhecer apenas acordes maiores, menores e diminutos [3, 19, 20, 21]. Em alguns poucos casos, acordes com sétima e aumentados podem ser reconhecidos [30, 37].

Conforme exposto no capítulo anterior, a completude harmônica é um fator importante no estudo da execução da bossa-nova ao violão, e o reconhecimento de acordes com mais dissonâncias, como nonas, décimas primeiras e décimas terceiras ou acordes suspensos é requisito necessário de uma boa solução para este problema. Entretanto, nenhuma das abordagens estudadas, tanto sobre sinais de áudio quanto sobre seqüências MIDI, considerou tal complexidade [3, 5, 19, 20, 21, 22, 30, 37, 41, 50].

As abordagens que visam extrair informação harmônica a partir de dados simbólicos, isto é, seqüências MIDI, normalmente consideram que os dados de entrada estarão quantizados [30, 41], tendo a performance bastante afetada negativamente quando isto não acontece, como mostrado no sétimo capítulo.

Nos sub-capítulos que se seguem, algumas das principais linhas de pesquisa em reconhecimento de acordes serão detalhadas. Primeiramente, as abordagens que visam reconhecer acordes a partir de sinais de áudio, muitas vezes utilizando dados simbólicos como entrada para técnicas de aprendizagem supervisionada, e em seguida, as abordagens sobre dados simbólicos propriamente ditas.

## 3.1. Reconhecimento de Acordes em Sinais de Áudio

As características baseadas em PCP (*Pitch Class Profile*) [15], uma transformação matemática que utiliza a DFT (*Discrete Fourier Transform*), ou *croma*, outra transformação matemática muito semelhante ao PCP, têm sido quase que exclusivamente utilizadas como ponto de partida para o reconhecimento de acordes ou tonalidade a partir de sinais de áudio. É consenso no meio acadêmico que tais transformações são capazes de melhor ressaltar as informações harmônicas contidas no sinal de áudio, embora existam abordagens que utilizam variações, como o EPCP (*Enhanced PCP*), que deriva do HPS (*Harmonic Product Spectrum*), e não da DFT [22].

Os PCPs são vetores de características instantâneas de baixo nível representando a intensidade de cada uma das doze classes de freqüências, ou semitons, da escala tonal, predominante na música ocidental, e são largamente usados em aplicações envolvendo conteúdo harmônico. Embora os passos principais no cálculo do PCP sejam muito similares em toda a literatura científica, alguns detalhes de implementação podem variar [5]. A figura 3.1 ilustra os principais passos utilizados no cálculo do PCP de um dado sinal de áudio.

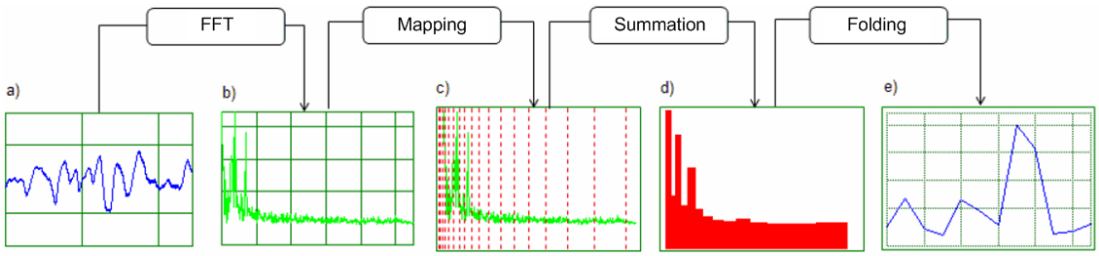


Figura 3.1 – Fases do cálculo do PCP (*Pitch Class Profile*) para um sinal de áudio (adaptado de CABRAL, G. et al [5]).

Fujishima [15] desenvolveu um sistema para reconhecimento de acordes que deriva um PCP a partir da DFT do sinal de áudio, e utiliza casamento de padrões com padrões binários de acordes. Percebeu-se que o simples casamento do PCP com padrões binários de acordes não era suficiente para reconhecer acordes de forma satisfatória, embora a idéia do casamento de padrões tenha sido usada futuramente por Lee [22] sobre uma variante do PCP, que o autor chamou de EPCP *(Enhanced PCP*).

A grande maioria das demais abordagens envolve técnicas de aprendizagem de máquina [3, 37], por vezes utilizando dados simbólicos previamente analisados e sintetizados para construção dos conjuntos de treinamento e validação, enquanto os conjuntos de teste normalmente contêm gravações reais, rotuladas manualmente por um especialista [19, 20, 21].

Sheh e Ellis [37] propuseram um método de aprendizagem estatística para segmentação e reconhecimento de acordes. Eles utilizaram HMMs (*Hidden Markov Models*) treinados com o algoritmo EM (*Expectation Maximization*) e trataram os rótulos dos acordes como valores escondidos dentro do EM. Durante o treinamento, eles utilizaram somente a seqüência de acordes como entrada, e aplicaram o algoritmo *forward-backward*, ou Baum-Welch, para estimar os parâmetros do modelo. A precisão obtida foi de 76% para o problema da segmentação e 22% para reconhecimento dos acordes. A baixa performance em reconhecimento pode ter sido causada por insuficiência de dados. Além disso, Sheh e Ellis trataram apenas acordes de sete tipos: maiores, menores, maiores com sétima maior, menores com sétima menor, dominantes (maiores com sétima menor), aumentados e diminutos. Ainda assim, este é um dos trabalhos que considera a maior quantidade de tipos de acorde no reconhecimento.

Bello e Pickens [3] também usaram HMMs treinados com o algoritmo EM. Entretanto, eles incorporaram conhecimento musical no modelo, considerando encadeamentos simples de acordes pela definição de uma matriz de transição de estados baseada na distância das tônicas, num círculo de quintas, e evitando a inicialização aleatória do vetor médio e da matriz de covariância da distribuição da observação, que foi modelada por uma Gaussiana simples. Além disso, no treinamento dos parâmetros do modelo, eles atualizaram os parâmetros de interesse seletivamente, assumindo que um padrão de acorde ou distribuição é quase universal. Assim, obtiveram 75% de precisão utilizando segmentação sincronizada com o pulso, com um conjunto menor de tipos de acorde. Em particular, eles argumentaram que a precisão cresceu em até 32% quando o ajuste dos parâmetros da distribuição de observação é proibido. Novamente, isto pode ter ocorrido porque eles utilizaram não somente um modelo não-supervisionado, mas também os dados de treinamento foram insuficientes para estimar apropriadamente os parâmetros do modelo. Embora

a precisão tenha aumentado, a utilização de um conjunto menor de tipos de acordes, contendo apenas acordes maiores e menores, torna a abordagem incompleta harmonicamente quanto aos requisitos expostos no capítulo anterior.

Lee e Slaney [19, 20] propuseram uma abordagem baseada no trabalho de Sheh e Ellis [37] e Bello e Pickens [3], onde os estados no HMM representam tipos de acordes, e tentaram achar um caminho ótimo, ou seja, uma seqüência de acordes com um fator de probabilidade máximo. A diferença mais importante é a utilização de dados de treinamento rotulados a partir dos quais os parâmetros do modelo podem ser diretamente estimados. Além disso, Lee e Slaney propuseram um método para obter dados de treinamento automaticamente rotulados baseado na síntese de seqüências MIDI, fazendo uso de uma abordagem de reconhecimento de acordes a partir de dados simbólicos [41]. Um dos principais fatores negativos é, novamente, a baixa completude harmônica; Lee e Slaney consideraram apenas três tipos de acordes em seus estudos: maiores, menores e diminutos.

Ainda utilizando aprendizagem supervisionada em modelos escondidos de Markov (HMMs), Lee [21] propôs uma abordagem construindo um modelo escondido de Markov distinto para cada gênero musical, e mostrou que os resultados foram melhores quando o HMM treinado especificamente para o gênero da entrada era utilizado, sugerindo que o gênero tem certa influência global nas informações do sinal de áudio. Lee considerou seis gêneros musicais: teclados, música de câmara, música orquestrada, rock, jazz e blues. Entretanto, apesar de considerar o jazz como um dos gêneros musicais, o que poderia sugerir uma gama maior de padrões de acordes, pela complexidade harmônica do estilo, Lee utilizou apenas os acordes maiores, menores e diminutos.

O EDS (*Extractor Discovery System*) [38], desenvolvido pelo *Sony Computer Science Lab* [39], em Paris, propõe uma técnica alternativa, baseada em heurística para extrair automaticamente descritores musicais de alto nível a partir de sinais de áudio. A abordagem utiliza programação genética para construir funções de extração como composições de operações matemáticas e de processamento de sinais básicas. A partir de uma base de sinais de áudio rotulada, o EDS pode generalizar um descritor construído

através da execução de uma busca genética para encontrar características de processamento do sinal relevantes para casar com o problema descrito, seguida da execução de algoritmos de aprendizagem de máquina para combinar as características num modelo descritivo generalizado.

Cabral et al. [5] realizaram comparações entre o EDS e métodos tradicionais de extração de descritores para o caso do reconhecimento de acordes no violão executando bossa-nova. Os autores utilizaram duas abordagens tradicionais como *benchmark*, KNN (*K-Nearest Neighbors*) e *PCP Templates*, e consideraram três classes de acordes: um conjunto reduzido com acordes maiores e menores, numa tônica fixa, um conjunto maior com acordes maiores, menores, com sétima maior, com sétima menor, e diminutos, também com tônica fixa, e por fim, um terceiro conjunto, idêntico ao segundo, porém com tônicas variáveis. Na última classe de acordes o EDS chegou a apenas 40,31% de acerto, contra 53,85% do *PCP Templates* e 63,93% do KNN. Embora as abordagens tradicionais utilizadas tenham obtido melhores resultados em todos os casos, os autores acreditam que a combinação de funções de baixo nível é capaz de encontrar diferentes padrões para uma mesma tônica, mas a paleta atual de funções de processamento do EDS não é suficiente para generalizar informações harmônicas.

Yoshioca et al. [50] propuseram uma abordagem baseada em busca num espaço de hipóteses para resolver simultaneamente os problemas de reconhecimento de acordes e segmentação. O método proposto gera hipóteses sobre tuplas de uma seqüência de acordes e uma tônica, calculando um valor de certeza para cada tupla, e selecionando a hipótese com maior valor de certeza. Para o cálculo do valor de certeza das hipóteses, são utilizados três tipos de informações: características acústicas (através de vetores *croma* de 12 dimensões), padrões de progressões de acordes e baixos (tons predominantes em regiões de baixas freqüências). O uso de padrões de progressões de acordes já foi bem sucedido em abordagens para análise harmônica funcional [29, 34], e faz bastante sentido neste tipo de problema, por tratar-se de uma meta-informação harmônica extremamente rica.

Dentre todas as abordagens estudadas, o problema da baixa completude harmônica é recorrente. Nenhuma abordagem é capaz de reconhecer o

conjunto completo de acordes, incluindo as dissonâncias recorrentes em harmonias jazzísticas, embora algumas abordagens cheguem a reconhecer sétimas.

Além disso, as abordagens baseadas em aprendizagem supervisionada precisam de grandes bases de dados rotuladas para treinamento, o que sugere a utilização de abordagens para reconhecimento de acordes sobre dados simbólicos, como seqüências MIDI, para gerar bases rotuladas automaticamente através da sintetização dos dados simbólicos. Assim, a completude harmônica de tais abordagens fica limitada à quantidade de tipos de acordes que as abordagens sobre dados simbólicos podem identificar.

## 3.2. Reconhecimento de Acordes em Seqüências MIDI

O reconhecimento de acordes em dados simbólicos normalmente pressupõe dados quantizados [30, 41]. Quantização é o processo de alinhamento de um conjunto de notas musicais, ou eventos NOTE ON e NOTE OFF, no caso das seqüências MIDI, em uma grade temporal precisa. Tal processo resulta em ataques exatos em pulsos, ou frações exatas de pulsos.

Embora a quantização simplifique bastante o processamento de dados simbólicos, trata-se de uma transformação irreversível e que trás perda de informação. Ou seja, uma vez que uma dada seqüência foi quantizada, é impossível obter a seqüência original pela inexistência de uma inversa única. A informação perdida quando da quantização de uma seqüência são os tempos exatos de ataque de cada evento, tão precisos quanto for a precisão dos dados simbólicos.

Nos dados simbólicos adquiridos a partir da execução do violão MIDI no projeto “Um País, Um Violão”, o andamento, i.e., a velocidade da execução, é variável. Assim, sem utilização de metrônomo ou qualquer meio de induzir a velocidade de execução, o músico tinha total liberdade para realizar sua interpretação, evitando qualquer viés sobre a mesma. Sendo assim, mesmo que a quantização fosse um procedimento considerado razoável, não poderia ser aplicado aos dados adquiridos no projeto “Um País, Um Violão”. Além disso, aspectos importantes da interpretação, como as micro-variações de tempo nos ataques, seriam perdidos.

### *3.2.1. Birmingham e Pardo*

Considerando dados quantizados, Birmingham e Pardo [30] propuseram uma abordagem para particionamento, segmentação e detecção de acordes baseada em casamento de padrões de acordes e busca num grafo acíclico direcionado. Inicialmente, a abordagem de Birmingham e Pardo cria um ponto de partição na seqüência para cada início ou final de nota, i.e., cada evento NOTE ON ou NOTE OFF. Os segmentos formados são vetores contendo a quantidade de vezes que cada nota soa entre cada par de pontos de partição, uma espécie de PCP simbólico. Os segmentos mínimos, que ocorrem entre pontos de partição consecutivos, são pré-calculados. Os demais segmentos são calculados sob demanda, somando-se os elementos correspondentes dos segmentos mínimos contidos entre seus pontos de partição. A figura 3.2 representa graficamente a definição de segmentos, segmentos mínimos e pontos de partição.

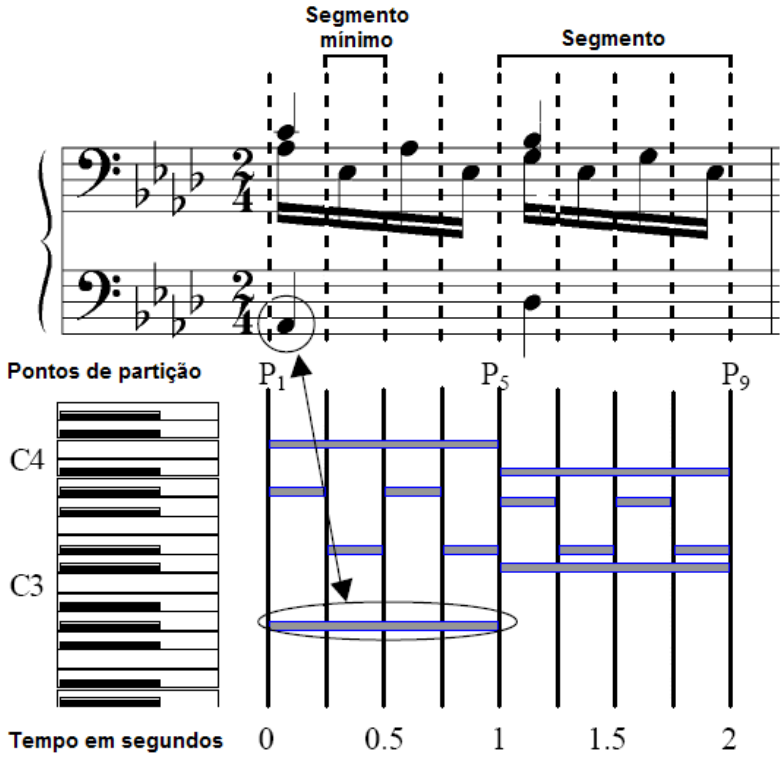


Figura 3.2 – Segmentos, segmentos mínimos e pontos de partição (adaptado de Birmingham & Pardo [30]).

Em seguida, cria-se um grafo acíclico direcionado, onde cada nó representa um ponto de partição, e executa-se o algoritmo HarmAn [30] para

achar o melhor caminho, utilizando-se as informações de cada segmento e um conjunto de padrões de acordes para calcular o peso das arestas. Para tal, utiliza-se uma função de utilidade, aplicada ao segmento para todas as combinações de padrão e tonalidade.

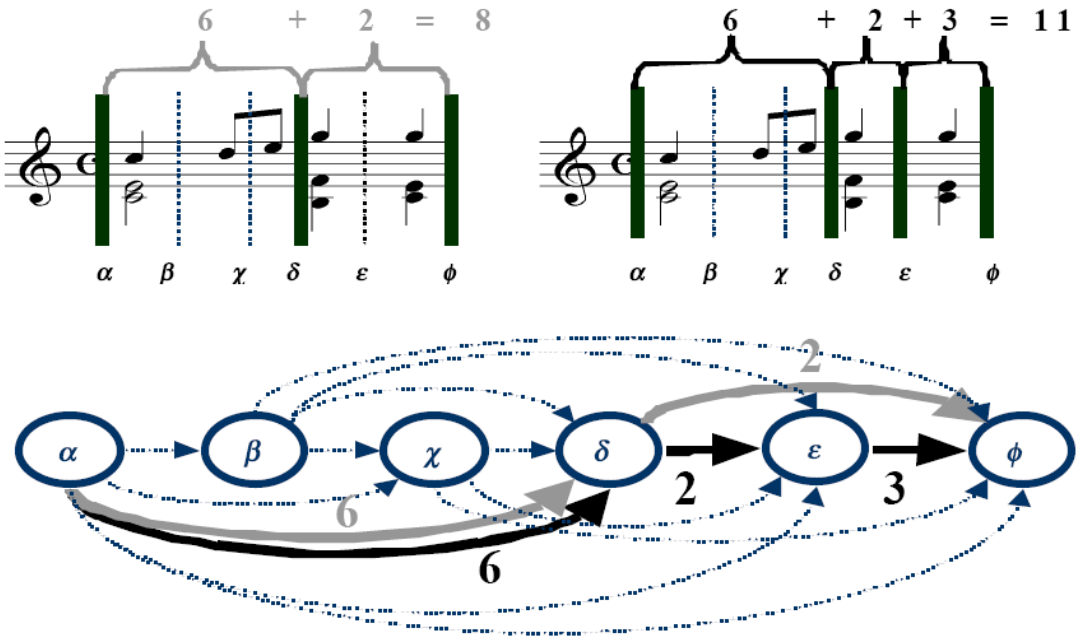


Figura 3.3 – Abordagem de Birmingham e Pardo: utilização do algoritmo HarmAn, que executa podas sobre o grafo acíclico direcionado, gerado a partir dos pontos de partição, utilizando uma função de utilidade baseada em padrões de acordes, para chegar ao resultado final (adaptado de Birmingham e Pardo [30]).

A função de utilidade definida por Birmingham e Pardo utiliza informações exclusivamente locais para calcular a adaptação de um segmento a um padrão, e o algoritmo HarmAn escolhe o caminho com maior valor de utilidade, podando os nós inutilizados. Ou seja, após a construção do grafo, tem-se um problema de otimização.

A proposta de Birmingham e Pardo, apesar de obter bons resultados tanto de segmentação quanto de reconhecimento de acordes em dados quantizados e harmonias simples [30], não chegou a bons resultados com os dados não quantizados do projeto “Um País, Um Violão”, além de ter demonstrado dificuldade no reconhecimento completo da harmonia jazzística utilizada na bossa-nova, como mostrado no sétimo capítulo. Ambos os problemas já eram previsíveis. Uma vez que a abordagem parte do

pressuposto que os dados de entrada são quantizados, a utilização de dados não quantizados causa um superparticionamento, ou seja, a identificação de uma quantidade muito maior que a real de pontos de partição, o que aumenta consideravelmente a complexidade do grafo gerado.

Apesar do conjunto de padrões proposto por Birmingham e Pardo considerar uma gama de acordes maior que a maioria das demais abordagens estudadas [3, 19, 20, 21, 38] contendo também acordes meio-diminutos, maiores com sétima maior e sétima menor e menores com sétima menor, as demais dissonâncias não estão incluídas no conjunto, bem como os acordes suspensos.

### *3.2.2. Melisma Music Analyser*

Partindo de dados simbólicos, o *Melisma Music Analyser* [41] é um dos sistemas mais utilizados para construção de bases rotuladas pelas abordagens baseadas em treinamento supervisionado [19, 20, 21]. O Melisma (*Modular Event-List-Input System for Music Analysis*) é um sistema para análise e extração de informações a partir de uma lista de eventos (por exemplo, seqüências MIDI), capaz de extrair informações como métrica, harmonia e tonalidade.

O módulo de extração de harmonia do Melisma utiliza como entrada, além da lista de eventos, a saída do módulo de *beat tracking*. Este último, antes de realizar de fato a inferência de pulsação, quantiza os dados numa precisão aproximada de 35 milésimos de segundo. O algoritmo de reconhecimento de acordes do Melisma utiliza as informações de *beat tracking* para definir segmentos mínimos do tamanho dos pulsos detectados, realizando uma nova quantização nos dados, desta vez para que se ajustem aos pulsos. A política considerada ajusta os eventos NOTE ON e NOTE OFF para coincidirem com o pulso precedente, com exceção das situações onde ambos os eventos seriam ajustados para o mesmo pulso, tornando a duração da nota zero. Neste caso, o evento NOTE OFF é ajustado para coincidir com o pulso posterior.

O Melisma utiliza algumas políticas durante o reconhecimento de acordes, como por exemplo, a preferência por seguir uma linha de quintas entre acordes consecutivos, ou considerar mudanças de harmonia em tempos fortes. Além disso, há uma enorme gama de parâmetros que pode ser

configurada pelo usuário, de acordo com suas necessidades. Entretanto, o Melisma não faz o reconhecimento de acordes propriamente dito, somente reconhece a tônica de cada segmento, e fornece uma lista de notas presentes no segmento. Embora pareça simples calcular os intervalos de tais notas para a tônica e montar o acorde de fato, tal tarefa implica em tomadas de decisão em casos ambíguos. O discernimento entre notas de passagem e acidentes do acorde, ou a decisão sobre o tipo do acorde quando notas essenciais para tal definição não estão presentes, como as terças, são tarefas cujo método de resolução pode influenciar contundentemente os resultados, e que o Melisma deixa a critério do usuário.

Portanto, as atuais implementações, apesar de apresentarem bons resultados quando aplicadas a dados quantizados e de harmonia menos complexas, não são capazes de resolver o problema do projeto “Um País, Um Violão”, pela falta de quantização dos dados, que afeta negativamente a performance do reconhecimento de acordes [30, 41], e pela baixa completude harmônica no reconhecimento da harmonia jazzística, inerente à bossa-nova [30]. Ou ainda pela realização incompleta da tarefa, não proporcionando todas as informações desejadas [41].

# 4. O Processo COCHONUT para Reconhecimento de Acordes em Seqüências MIDI

A abordagem proposta nesta dissertação para a extração de acordes em seqüências MIDI divide o problema em três:

- a segmentação da seqüência através da identificação de pontos de particionamento que marcam as mudanças na harmonia da música;
- a escolha de um conjunto de acordes mais prováveis para cada segmento, de acordo unicamente com informações locais; e por fim,
- a escolha do acorde mais adequado a cada segmento, considerando as informações contextuais.

Os dados de entrada utilizados para validação dos resultados desta dissertação foram gravados por dois intérpretes, que executaram peças musicais de bossa-nova em um violão MIDI sem que fossem fornecidas as grades de acordes. Simultaneamente às seqüências geradas pelo violão MIDI, foram gravados arquivos de áudio no formato WAV.

As seqüências MIDI possuem seis trilhas, uma para cada corda do violão, como mostrado na figura 4.1. Observa-se que o canal onze é reservado para a primeira corda, o canal doze para a segunda, e assim sucessivamente até o canal dezesseis, reservado para a sexta corda. Pode-se observar também, no canto superior direito da figura 4.1, que as gravações foram realizadas sempre a 100 BPM (*Beats per Minute*), mesmo quando o intérprete não utilizava de fato este andamento, o que invalida as informações simbólicas de tempo e compasso fornecidas automaticamente pelo editor MIDI.

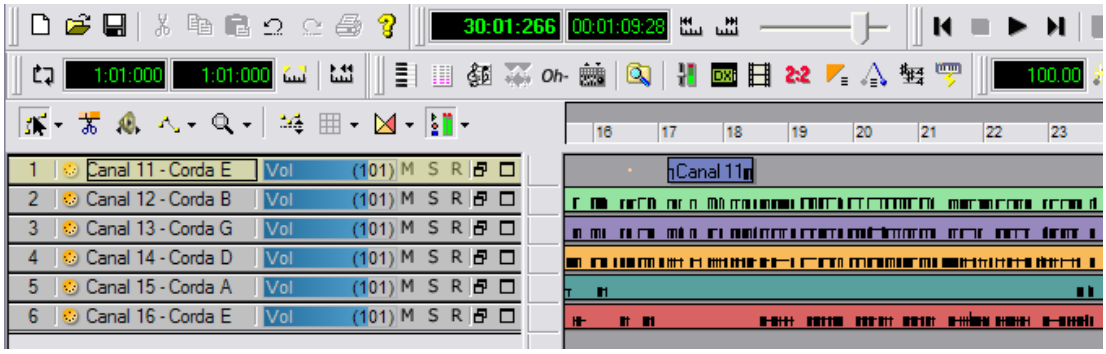


Figura 4.1 – Seqüência MIDI com seis trilhas, uma para cada corda do violão.

O processo proposto por esta dissertação integra várias técnicas de inteligência artificial, como teoria da decisão, otimização, casamento de padrões e reconhecimento baseado em regras [32], e é composto por três etapas, precedidas por uma etapa de pré-processamento que consiste na limpeza dos ruídos nas seqüências geradas pelo violão MIDI, como ilustrado pela figura 4.2.

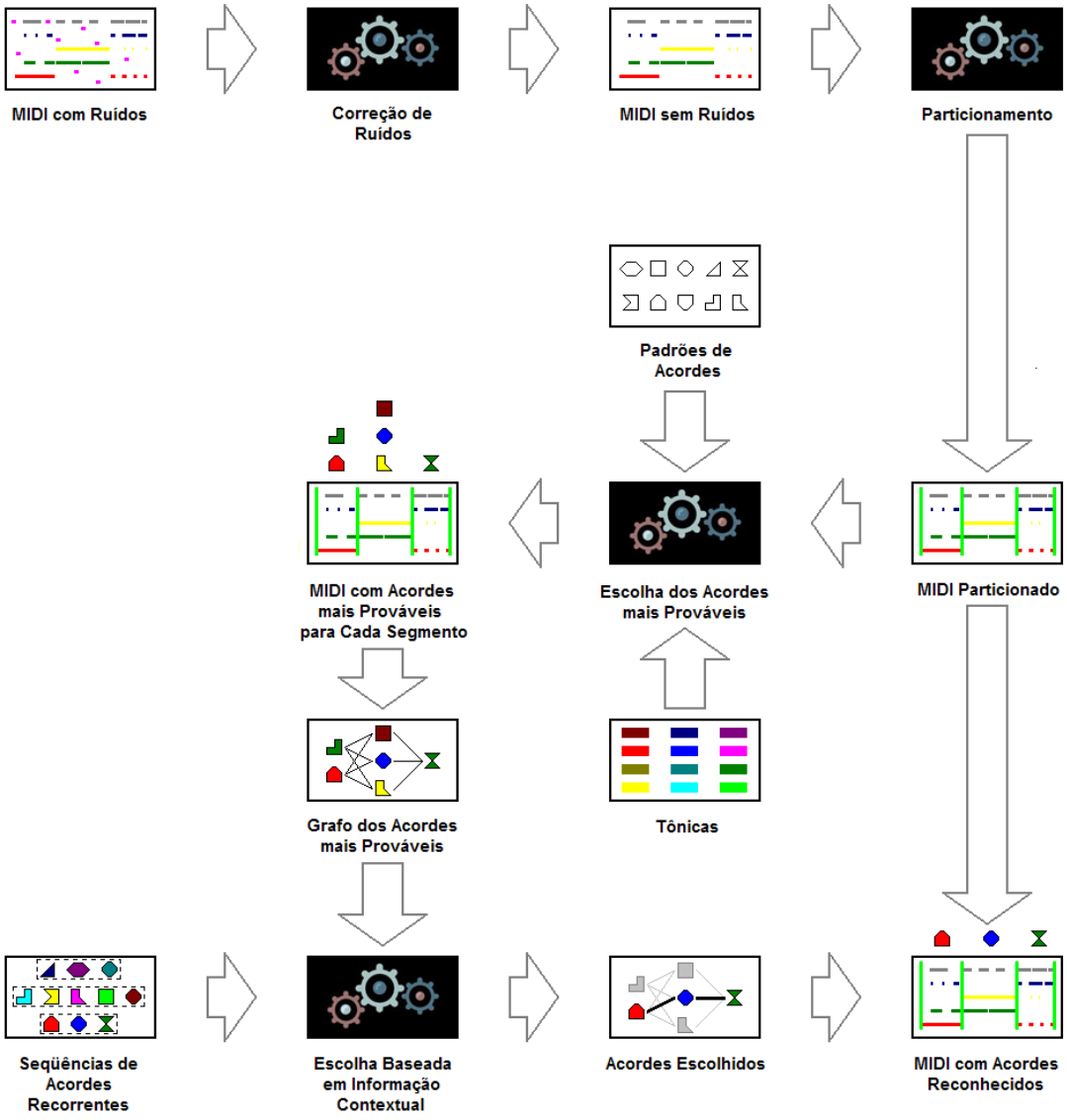


Figura 4.2 – Etapas do processo proposto pelo COCHONUT.

A seguir, cada etapa do processo será detalhada, iniciando-se pelo pré-processamento para correção de erros nas seqüências MIDI, seguido das técnicas utilizadas para particionamento e segmentação, a definição dos padrões de acordes e da função de utilidade e sua aplicação, e por fim, a

busca do caminho mais adequado no grafo, considerando as informações contextuais.

## 4.1. Pré-processamento para Correção de Ruídos

A abordagem utilizada para corrigir os ruídos nas seqüências geradas pelo violão MIDI utiliza o motor de inferência JEOPS [14] e dois conjuntos de regras, que são aplicados em seqüência. A primeira base de regras contém regras simples, que marcam notas não plausíveis ou possíveis ruídos, ou removem notas claramente ruidosas. Já a segunda base de regras, contém as regras contextuais, que são aplicadas após um prévio particionamento da seqüência, onde os blocos nos quais potencialmente não há mudança de harmonia são identificados. As regras desta base comparam os segmentos onde se encontram as notas marcadas como não plausíveis ou ruidosas com seus segmentos vizinhos, e modificam a freqüência de tais notas, ou removem-nas, adequadamente.

A primeira base contém sete regras, como mostrado na tabela 4.1. O motor de inferência dispara as regras de forma prioritária, ou seja, as regras posicionadas acima na tabela têm prioridade sobre as posicionadas abaixo delas.

Tabela 4.1 – Regras simples para minimização de ruídos.

| Nome da Regra | Função da Regra |
|---|---|
| **DeleteShortNoteEvents** | **Remove notas com duração inferior a 100 *ticks, ou 125 milésimos de segundo*** |
| **DeleteLowVelocityEvents** | **Remove notas com *velocity* inferior a 15** |
| **MarkOutOfRangeEvents** | **Marca como não plausíveis eventos fora dos limites da corda do violão** |
| **MarkPotentialNoisyEvent_SixStrings** | **Marca como possíveis ruídos eventos muito distantes fisicamente, no braço do violão, de outros eventos ocorrendo simultaneamente em outras cordas do violão** |
| **MarkPotentialNoisyEvent_FiveStrings** | |
| **MarkPotentialNoisyEvent_FourStrings** | |
| **MarkSimultaneousEvents** | **Marca como possíveis ruídos notas simultâneas numa mesma corda do violão** |

As duas primeiras regras da base são responsáveis por remover notas cuja duração ou *velocity*[1] são desprezíveis. A terceira regra marca como não plausíveis as notas cujas freqüências estão fora dos limites físicos de suas respectivas cordas, ou seja, freqüências abaixo da freqüência da corda solta, ou acima da freqüência da vigésima quarta casa. As três regras seguintes marcam como possíveis ruídos notas cuja distância no braço do violão para outras notas simultâneas é maior que cinco casas, uma vez que a execução simultânea de notas tão distantes entre si é anatomicamente impossível para a grande maioria dos seres humanos. Por fim, a última regra marca como possíveis ruídos notas que ocorrem simultaneamente numa mesma corda, dado que no violão tal comportamento é impossível. Assim, sabendo-se que pelo menos uma das duas notas é ruidosa, a marcação de ambas, nesta etapa, permite remover a nota mais adequada na segunda etapa de aplicação de regras, onde há mais informações disponíveis para a tomada de decisão.

A segunda base de regras contém cinco regras, como mostrado na tabela 4.2. Como na primeira base, o motor de inferência dispara as regras de forma prioritária. Entretanto, antes da execução desta base de regras, realiza-se o particionamento e segmentação iniciais da seqüência, sem classificação de acordes, utilizando a técnica descrita na sessão 4.2.

Tabela 4.2 – Regras contextuais para minimização de ruídos.

| Nome da Regra | Função da Regra |
|---|---|
| **ComplyWithPreviousChordBySimilarity** | **Tenta corrigir a freqüência de notas ruidosas ou não plausíveis considerando a similaridade com o acorde vizinho** |
| **ComplyWithNextChordBySimilarity** | |
| **ComplyWithPreviousChord** | **Tenta corrigir a freqüência de notas ruidosas ou não plausíveis considerando apenas os acordes anteriores ou posteriores** |
| **ComplyWithNextChord** | |
| **DeleteRemainingNonPlausibleEvents** | **Após todas as tentativas de correção, remove as notas marcadas como não plausíveis ainda existentes** |

[1] ***Velocity*** é um atributo dos eventos MIDI que indica a intensidade com que um evento deve ser executado pelo sintetizador. Nos arquivos capturados por violões MIDI, equivale à força com que a nota foi tocada pelo intérprete.

As duas primeiras regras da base tentam corrigir a freqüência das notas ruidosas e não plausíveis considerando a similaridade com os segmentos anterior e posterior. Considerando que um segmento pode ser simbolizado como um vetor de doze *bits* representando as notas da escala cromática, onde cada *bit* é ativado quando a nota que representa está presente no segmento, e desativado caso contrário, a similaridade entre dois segmentos é dada pelo número de bits com valor coincidente, subtraído do número de bits com valor não coincidente. Ou seja, acrescenta-se um ponto a cada nota que existe em ambos os segmentos, e a cada nota que não existe em ambos os segmentos, e diminui-se um ponto a cada nota que existe em apenas um dos dois segmentos. A seguir, um exemplo da representação do acorde $C^{7(b9)}$ em vetor de bits.

$C^{7(b9)}$ → [ **C** , C#/**Db**, D , D#/Eb, **E** , F , F#/Gb, **G** , G#/Ab, A , A#/**Bb**, B ]

$C^{7(b9)}$ → [ **1** , **1** , 0 , 0 , **1** , 0 , 0 , **1** , 0 , 0 , **1** , 0 ]

Logo, uma vez encontrada uma nota não plausível ou ruidosa, e calculada a similaridade do segmento no qual a nota se encontra com os segmentos imediatamente anteriores e posteriores, modifica-se a freqüência da nota não plausível ou ruidosa para a freqüência da nota que se encontra na mesma corda, no segmento mais similar. No exemplo da figura 4.3 a seguir, o similaridade do segmento $S_1$ é maior que a similaridade do segmento $S_3$, quando comparados ao segmento $S_2$. Assim, a freqüência da nota ruidosa na

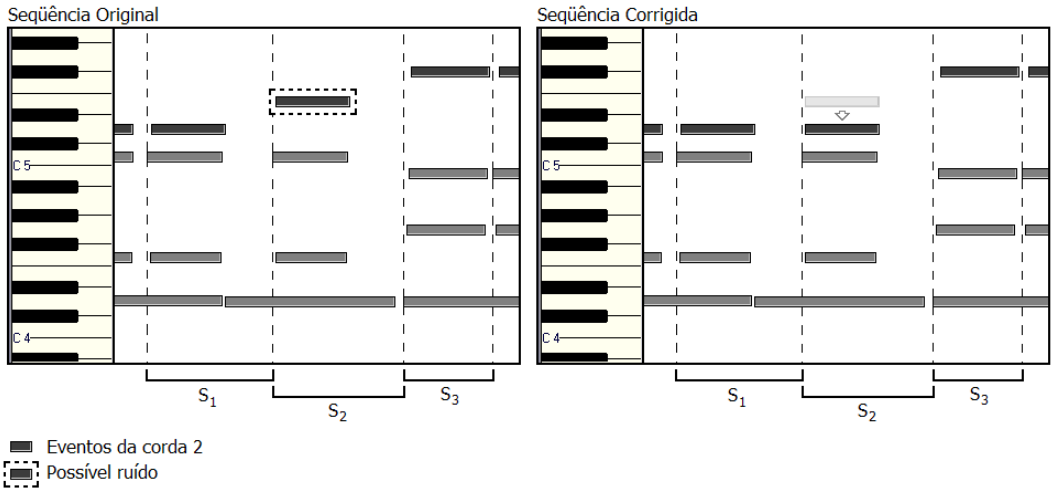


Figura 4.3 – Correção de evento com freqüência mal identificada por similaridade com segmento vizinho.

corda 2, no segmento $s_2$, é modificada de E para D, que é a nota tocada pela corda 2 no segmento $S_1$.

Entretanto, quando uma possível nota ruidosa ou não plausível encontra-se no primeiro ou no último segmento da seqüência, as duas primeiras regras não podem ser aplicadas, já que, para disparar, necessitam dos segmentos vizinhos anteriores e posteriores ao segmento da nota ruidosa ou não plausível. Logo, as duas próximas regras simplesmente tentam corrigir as notas não plausíveis ou ruidosas da mesma maneira que as duas primeiras regras, porém independentemente da similaridade com os segmentos vizinhos. Na prática, estas regras só se aplicam quando há notas ruidosas ou não plausíveis no primeiro ou último segmentos da seqüência.

E por fim, os eventos marcados como não plausíveis que não foram corrigidos nem removidos anteriormente pela aplicação de outras regras são excluídos, já que pode-se assumir que são ruídos devido à sua não plausibilidade. Por outro lado, os eventos marcados como possíveis ruídos e que igualmente não foram removidos ou corrigidos pela aplicação das demais regras permanecem na seqüência, já que não houve evidência suficiente de que de fato se tratava de ruídos. Tal comportamento assume que eventos não plausíveis são de fato ruídos, mesmo quando não foram tratados por outras regras, enquanto eventos marcados como possíveis ruídos podem não o ser quando não existirem evidências suficientes.

As seqüências corrigidas automaticamente foram confrontadas com as seqüências corrigidas manualmente por Lima [24], através de um algoritmo para realizar operações de subtração de eventos entre seqüências MIDI (ou seja, dadas duas seqüências MIDI, o algoritmo remove todos os eventos idênticos em ambas as seqüências). Assim, a visualização, e a contagem automática dos falsos positivos, falsos negativos e verdadeiros positivos se torna possível.

Algumas regras inicialmente propostas, e não mencionadas acima, tiveram que ser abandonadas, para que se obtivesse a taxa nula de falso-positivos, mesmo acarretando num pior desempenho na correção dos ruídos, por ser um requisito essencial da solução. A aplicação desta técnica chegou a bons resultados, como será mostrado no capitulo 7.

## 4.2. Segmentação das Seqüências MIDI

O processo descrito a seguir visa definir pontos de partição que marcam potenciais mudanças de acorde na seqüência MIDI. Entre cada par de pontos de partição consecutivos, define-se um segmento. O estado dos segmentos é dado pela quantidade de notas que estão soando entre os dois pontos de partição que o definem, para cada classe de freqüências da escala bem temperada, e é notado como um vetor de doze elementos, de maneira similar à representação em vetor de bits. No exemplo da figura 4.4, percebe-se que no segmento zero ($S_0$) há duas ocorrências da nota C, quatro da nota Db, três da nota E, quatro da nota Bb e uma da nota B. Na representação gráfica, pode-se perceber que o vetor de estado se parece bastante com um PCP.

$S_0$ → [ **C** , C#/**Db**, D , D#/Eb, **E** , F , F#/Gb, G , G#/Ab, A , A#/**Bb**, B ]

$S_0$ → [ **2** , **4** , 0 , 0 , **3** , 0 , 0 , **0** , 0 , 0 , **4** , 1 ]

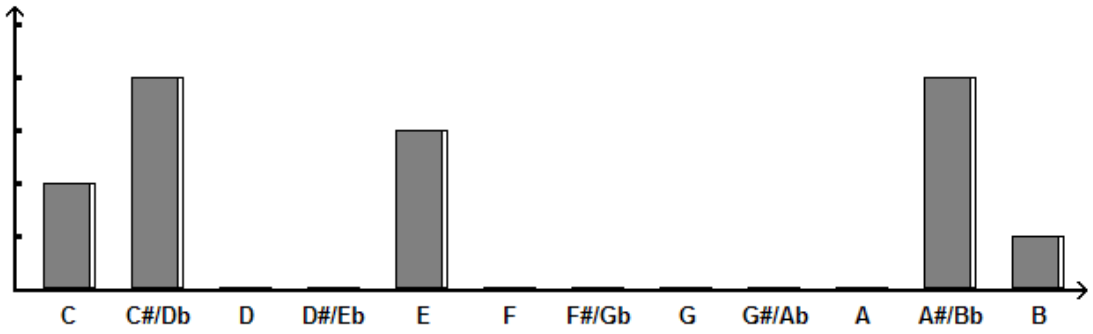


Figura 4.4 – Representação gráfica do vetor de estado $S_0$.

O resultado da aplicação da abordagem original de Birmingham e Pardo [30] sobre os dados não quantizados causa um superparticionamento, como pode ser visualizado na figura 4.5 a seguir. Assim, o algoritmo inicial sofreu diversas modificações, visando melhorar seu desempenho em seqüências capturadas por violões MIDI, e portanto não quantizadas.

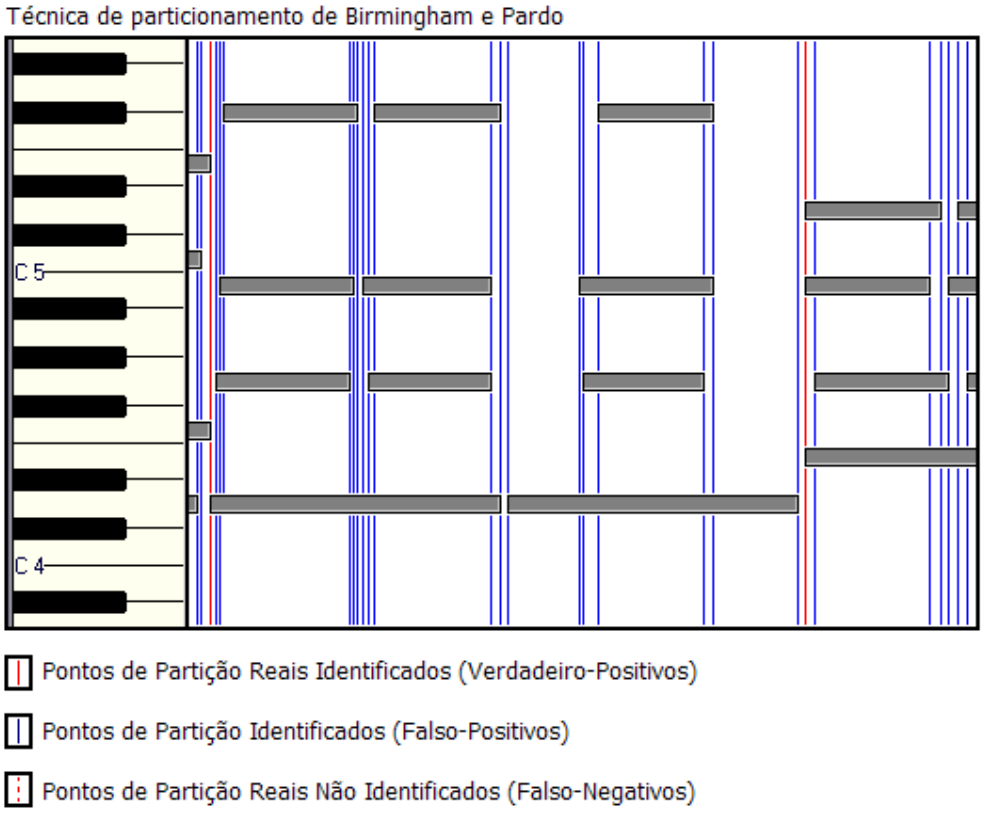


Figura 4.5 – Técnica de particionamento original de Birmingham e Pardo [30].

Inicialmente, percebeu-se que a política utilizada por Birmingham e Pardo, que define como pontos de partição cada evento NOTE ON ou NOTE OFF, poderia ser melhorada pela simples eliminação dos pontos de partição gerados por eventos NOTE OFF, já que as mudanças de acorde acontecem somente quando um novo acorde é tocado, e portanto sempre estão relacionadas a eventos do tipo NOTE ON. O resultado desta modificação pode ser visto na figura 4.6 a seguir.

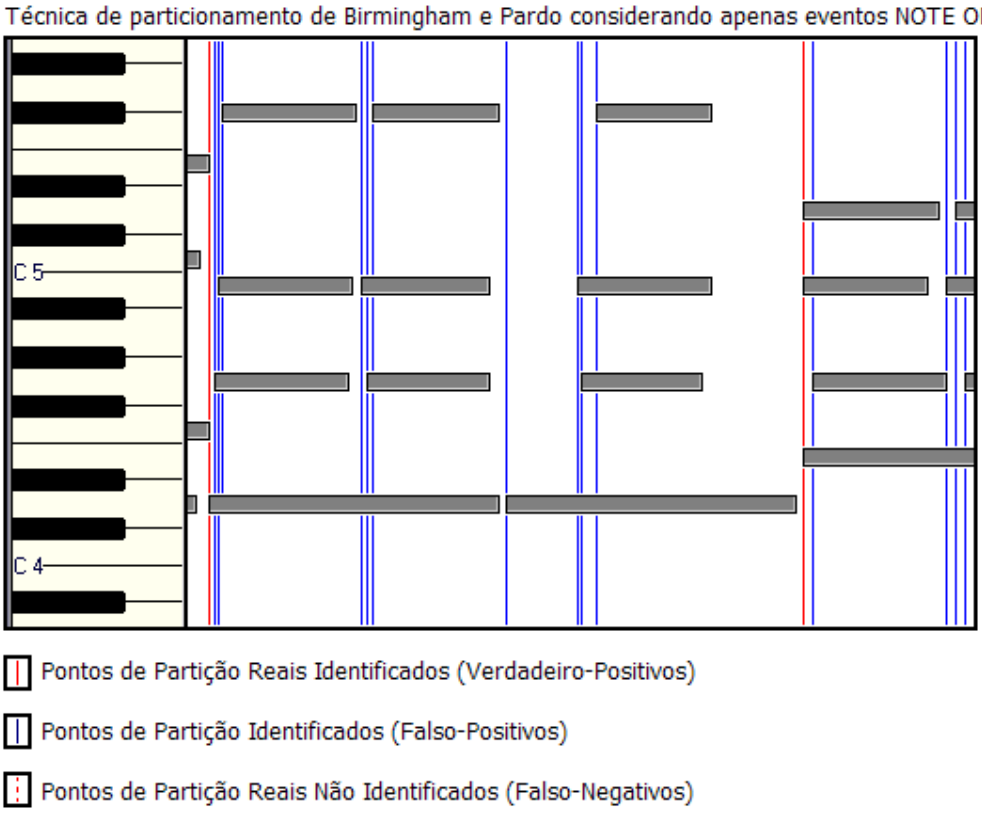


Figura 4.6 – Técnica de particionamento original de Birmingham e Pardo [30] desconsiderando eventos do tipo NOTE OFF.

Ainda assim, percebe-se uma quantidade grande de falso-positivos. A primeira tentativa de fato de melhoria do particionamento consistiu na realização de um pré-processamento das seqüências MIDI para tentar identificar blocos potenciais. Sabendo que para haver mudança de acorde ao menos uma nota do conjunto de notas que está sendo executado deve se modificar, definiu-se o pré-processamento das seqüências de forma a unir as notas com mesma freqüência que estivessem sendo tocadas em um intervalo de tempo muito curto, isto é, se a diferença entre o evento NOTE OFF da primeira nota e o evento NOTE ON da segunda nota fosse inferior a um determinado limiar, as notas eram substituídas por uma nova nota, iniciada no começo da primeira nota, e com duração até o final da segunda nota. A figura 4.7 mostra o resultado do pré-processamento sobre o mesmo trecho de música das figuras 4.5 e 4.6, e o resultado do particionamento.

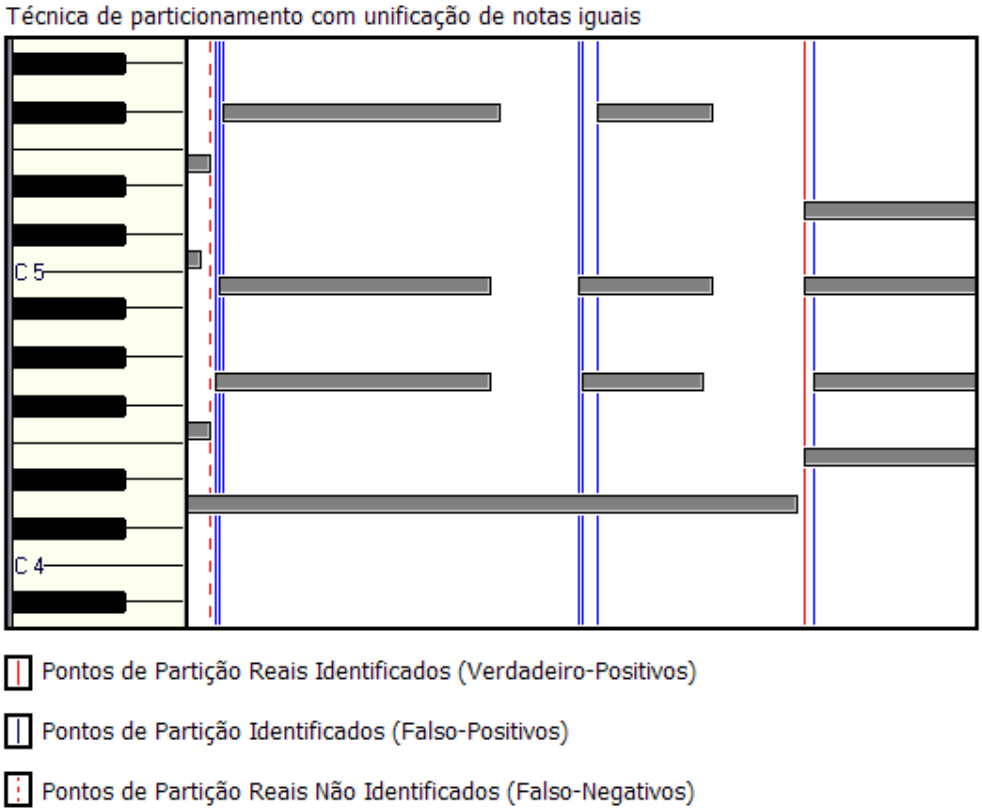


Figura 4.7 – Técnica de particionamento utilizando pré-processamento da seqüência para unificar notas idênticas muito próximas.

Esta abordagem mostrou-se inadequada devido à perda da informação quantitativa dos ataques, ou seja, ao substituir os eventos consecutivos por um único evento, não era mais possível saber quantas vezes uma determinada nota foi atacada dentro de uma partição, o que influenciou negativamente a função de utilidade que realiza a comparação do estado dos segmentos com os padrões de acordes previamente definidos. Além disso, em alguns casos, a técnica deixava de identificar pontos de partição reais, gerando falso-negativos, como pode-se perceber na figura 4.7.

Uma nova técnica de particionamento foi definida, adotando uma política mais restritiva na detecção dos pontos de particionamento, onde considerou-se como pontos de particionamento válidos apenas os pontos em que havia três ou mais ataques simultâneos, ou dentro de uma janela de tempo pré-definida. A justificativa para escolha de tal abordagem leva em consideração os dados de entrada serem gravações de acompanhamento de bossa nova ao violão, onde as mudanças de acorde acontecem normalmente nas puxadas,

que causam três ou mais eventos de nota simultâneos ou com intervalos muito curtos entre si. Experimentos foram realizados considerando-se duas, três e quatro notas simultâneas como limiar para definição de pontos de partição potenciais. A janela foi definida como a duração de uma semi-colcheia numa execução a 100 BPM. Entretanto, como as seqüências não foram executas a tal andamento, mas é sabido que seus andamentos não ultrapassam 200 BPM, a janela utilizada, na prática, garante uma duração entre uma semi-colcheia (nas execuções a 100 BPM) e uma colcheia (nas execuções a 200 BPM), dependendo do andamento de fato utilizado na gravação em questão. Acredita-se ser este um valor bastante razoável, considerando que as mudanças de acorde acontecem, na grande maioria das vezes, nos tempos fortes, cuja granularidade não é maior que uma semínima.

Os melhores resultados foram obtidos quando a política considerava o limiar de três notas, o que já era de se esperar, dada a particularidade das puxadas no violão serem realizadas, na grande maioria das vezes, em três cordas simultaneamente, pelos os dedos indicador, médio e anular. A figura 4.8 a seguir mostra o resultado da aplicação desta técnica no mesmo trecho de música dos exemplos anteriores.

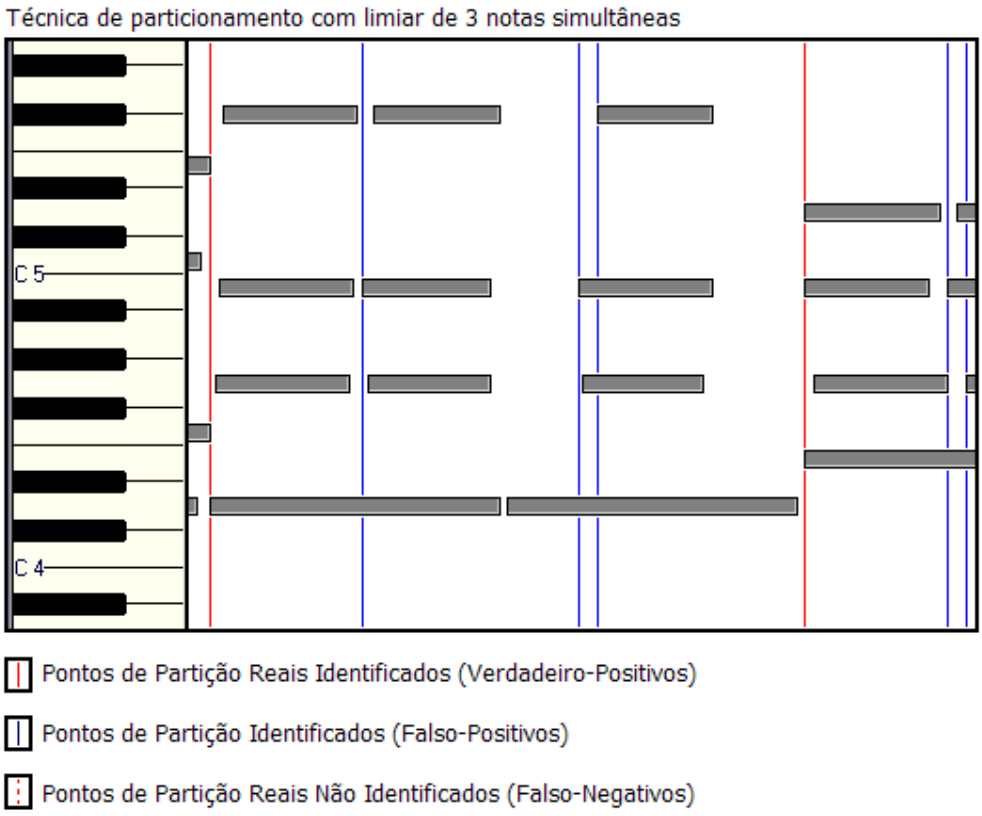


Figura 4.8 – Técnica de particionamento considerando limiar de 3 notas simultâneas, dentro de uma janela pré-definida.

Tal abordagem teve êxito na identificação dos pontos de particionamento corretos, mas continuou apresentando uma quantidade maior que a desejada de falso-positivos. Entretanto, os pontos de partição incorretamente identificados tinham a particularidade de resultar em segmentos consecutivos com estado idêntico, o que de certa forma, pela definição do algoritmo de particionamento, já era um comportamento esperado. Definiu-se então uma fase de pós-processamento dos segmentos encontrados, a fim de unificar os segmentos consecutivos idênticos.

O pós-processamento percorre a lista de segmentos, considerando cada par de segmentos consecutivos, e unificando-os quando seus estados são equivalentes, como exemplificado na figura 4.9. Tal abordagem melhorou significativamente a qualidade dos resultados.

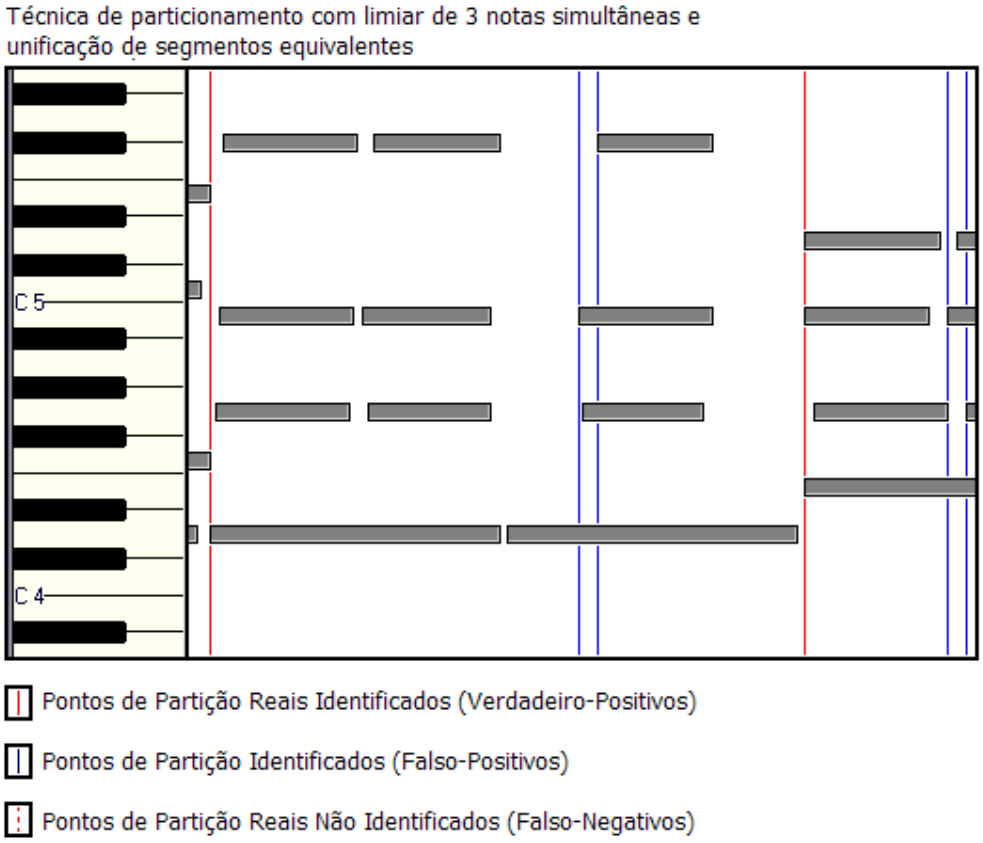


Figura 4.9 – Técnica de particionamento considerando limiar de 3 notas simultâneas, dentro de uma janela pré-definida e realizando unificação de segmentos equivalentes, *a posteriori*.

Após o final do processo de reconhecimento, segmentos adjacentes que foram identificados como acordes iguais são unificados. Isso permite uma diminuição ainda maior do número de falso-positivos identificados.

## 4.3. Escolha dos Acordes Mais Prováveis

A escolha dos acordes mais prováveis é baseada em uma lista de padrões de acordes que são confrontados com o estado dos segmentos, usando como tônica cada uma das doze notas da escala cromática. Chega-se então a um valor numérico de adaptação de cada acorde (i.e., par tônica/padrão) a cada segmento. Abordagens utilizando casamento de padrões já foram utilizadas anteriormente [22, 30], entretanto, os padrões de acordes definidos por tais abordagens não se adaptam bem às músicas com harmonia jazzística, devido à falta de padrões de acordes mais complexos, contendo acidentes como sétimas, nonas, décimas primeiras e décimas terceiras, ou acordes suspensos, diminutos, meio diminutos ou aumentados.

Optou-se então por utilizar uma lista de padrões de acordes que incluísse os padrões básicos e os padrões mais recorrentes na harmonia jazzística, além de padrões de quatro notas representando as posições mais comuns na execução de harmonia jazzística no braço do violão [1, 11, 13]. Vários experimentos foram realizados, a partir de uma lista inicial de padrões, para remover os padrões inadequados, deixando apenas os padrões que obtiveram os melhores resultados. Partindo de uma lista com os padrões inicialmente enumerados, realizou-se várias vezes o ciclo "experimento → observação dos resultados → retirada/inserção de padrões", até que se chegasse a resultados satisfatórios. A tabela 4.3 mostra os padrões finais utilizados no processo proposto.

Tabela 4.3 – Padrões de acordes recorrentes em harmonia jazzística ao violão.

| Padrão | Descrição | Notação (exemplo em C) |
|---|---|---|
| [0, 3, 7] | Acorde menor | Cm |
| [0, 3 , 6] | Acorde menor com quinta diminuta | $Cm^{(b5)}$ |
| [0, 3, 6, 10] | Acorde menor com sétima e quinta diminuta (meio-diminuto) | $Cm^{7(b5)}$ |
| [0, 3, 7, 10] | Acorde menor com sétima | $Cm^{7}$ |
| [0, 1, 3, 10] | Acorde menor com sétima e nona menor | $Cm^{7(b9)}$ |
| [0, 2, 3, 10] | Acorde menor com sétima e nona | $Cm^{7(9)}$ |
| [0, 3, 7, 11] | Acorde menor com sétima maior | $Cm^{7M}$ |
| [0, 2, 3, 11] | Acorde menor com sétima maior e nona | $Cm^{7M(9)}$ |
| [0, 4, 7] | Acorde maior | C |
| [0, 4, 7, 10] | Acorde maior com sétima | $C^{7}$ |
| [0, 2, 4, 10] | Acorde maior com sétima e nona | $C^{7(9)}$ |
| [0, 2, 5, 10] | Acorde suspenso com sétima e nona | $C^{sus4(7)(9)}$ |
| [0, 4, 8] | Acorde maior com décima terceira menor | $C^{(b13)}$ |
| [0, 1, 4, 10] | Acorde maior com sétima e nona menor | $C^{7(b9)}$ |
| [0, 1, 5, 10] | Acorde suspenso com sétima e nona menor | $C^{sus4(7)(b9)}$ |
| [0, 4, 7, 11] | Acorde maior com sétima maior | $C^{7M}$ |
| [0, 2, 4, 11] | Acorde maior com sétima maior e nona | $C^{7M(9)}$ |
| [0, 3, 6, 9] | Acorde diminuto | $C^{o}$ |
| [0, 4, 8, 10] | Acorde maior com sétima e décima terceira menor | $C^{7(b13)}$ |

A notação dos padrões segue a notação utilizada por Birmingham e Pardo [30], onde cada elemento do vetor que representa o padrão, equivale à

distância em semitons da tônica considerada. Por exemplo, o terceiro padrão da tabela 4.3 sobre a tônica D, teria suas notas calculadas de acordo com o quadro 4.1 a seguir.

Considere o padrão $p$, a tônica $t$, as notas $n_0$ a $n_3$, que formam o acorde $c_{pt}$:

$p = [p_0, p_1, p_2, p_3] = [0, 3, 6, 10]$
$t = \text{D}$
$c_{pt} = [n_0, n_1, n_2, n_3] = ?$

Dado que $n_i = t + p_i$, temos:

$n_0 = t + p_0 = \text{D} + 0 = \text{D}$
$n_1 = t + p_1 = \text{D} + 3 = \text{F}$
$n_2 = t + p_2 = \text{D} + 6 = \text{Ab}$
$n_3 = t + p_3 = \text{D} + 10 = \text{C}$

Logo :

$c_{pt} = [n_0, n_1, n_2, n_3] = [\text{D, F, Ab, C}] = \text{Dm}^{7(\text{b5})}$

Quadro 4.1 – Cálculo de acorde, a partir de par ordenado {padrão, tônica}.

A função de utilidade proposta por Birmingham e Pardo [30], e mostrada no Quadro 4.2, foi utilizada sem alterações para calcular a adequação de cada segmento aos padrões.

Considere o estado $e$ e o acorde $c_{pt}$, calculado a partir da tônica $t$ aplicada ao padrão $p$. Além disso, considere E como a evidência positiva, N como a evidência negativa e F como o número de falhas. A função de utilidade é calculada como:

$$f_u(e, p, t) = \text{E} - (\text{N} + \text{F})$$

Quadro 4.2 – Função de utilidade para cálculo do valor de adaptação de um padrão *p*, sobre uma tônica *t*, a um estado *e* de um segmento.

A função define uma pontuação positiva para cada nota coincidente entre o padrão e o estado (evidência positiva), além pontuações negativas para cada nota do padrão que não tem correspondente no estado (evidência negativa), e vice-versa (falhas). O quadro 4.3 mostra o pseudocódigo utilizado para calcular tais valores.

```
E <= 0, N <= 0, F <= 0
e = [e0, ..., e11]
cpt = [n0, ... , nn]
para cada elemento ei de e
      encontrado <= falso
      para cada elemento nj de cpt
            se ei = nj então
                  encontrado <= verdadeiro
                  pare laço
            fim do se
      fim do para
      se encontrado = verdadeiro então
            E <= E + 1
      senão
            N <= N + 1
      fim do se
fim do para

para cada elemento ni de cpt
      encontrado <= falso
      para cada elemento ej de e
            se ni = ej então
                  encontrado <= verdadeiro
                  pare laço
            fim do se
      fim do para
      se encontrado = falso então
            F <= F + 1
      fim do se
fim do para
```

Quadro 4.3 – Pseudo-código para cálculo da evidência positiva E, da evidência negativa N e das falhas F, na adaptação de um estado *e* de um segmento, a um acorde $c_{pt}$, calculado a partir de um padrão *p* e uma tônica *t*.

Os resultados obtidos com a nova lista de padrões, adaptada à harmonia jazzística[1] e às posições recorrentes no violão foram bem melhores que os resultados utilizando a lista de padrões proposta por Birmingham e Pardo [30], como será mostrado no capítulo 7 e no apêndice B.

Uma vez definidas a lista de padrões de acordes e a função de utilidade, para cada segmento, escolhe-se uma lista de acordes mais prováveis, com base na função de utilidade descrita no quadro 4.2, descartando-se todos os acordes cuja pontuação foi inferior a oitenta e cinco por cento da pontuação do vencedor. O limiar de oitenta e cinco por cento foi obtido empiricamente,

[1] A **harmonia jazzística** é a mais complexa estrutura harmônica na música ocidental. Considerando-se harmonia funcional (característica da música ocidental), a harmonia utilizada no Jazz (e que a bossa-nova incorporou) utiliza as mais complexas relações entre os acordes, sendo assim um super-conjunto das harmonias utilizadas em outros estilos ocidentais populares, como rock, balada, bolero, samba, pop, entre outros.

realizando-se experimentos com limiares variando entre setenta e cinco por cento e noventa por cento.

Observou-se que nos experimentos com limiar muito baixo havia uma quantidade muito grande de acordes possíveis para cada segmento, conseguindo-se chegar a um número muito baixo de segmentos de fato classificados, ou seja, segmentos onde nenhum acorde atingiu o limiar percentual em relação ao vencedor. Por outro lado, nas experiências com o limiar mais alto, de noventa por cento, percebeu-se que em alguns casos, os acordes que deveriam vencer acabavam por ficar de fora da lista de acordes prováveis, o que levaria fatalmente a um erro na próxima etapa do reconhecimento. O limiar de oitenta e cinco por cento apresentou os melhores resultados, mantendo os acordes corretos dentro do conjunto de acordes possíveis na grande maioria dos casos, e evitando uma superpopulação de acordes prováveis em cada segmento.

Constrói-se o grafo G(V, E), contendo k grupos disjuntos de vértices, constituídos pelos acordes candidatos para cada segmento:

$$V_i = \{\text{acordes candidatos para o segmento } i\}, \text{ para } i = 1, ..., k$$

Há k-1 grupos disjuntos de arestas, ligando os vértices referentes aos segmentos adjacentes:

$$E_i = \{(u, v) : u \in V_i, v \in V_{(i+1)}\}, \text{ para } i = 1, ..., (k-1)$$

Sendo assim, tem-se para o grafo G(V, E):

$$V = \bigcup V_i, \text{ para } i = 1, ..., k$$

$$E = \bigcup E_i, \text{ para } i = 1, ..., (k-1)$$

Desta maneira, G(V, E) é um grafo k-partido. Em outras palavras, G(V, E) é uma grafo direcionado e acíclico, onde cada camada contém a lista de acordes candidatos para um dado segmento, e é totalmente conectada à camada imediatamente posterior, não havendo conexões com camadas anteriores ou outras camadas posteriores, conforme o exemplo da figura 4.10.

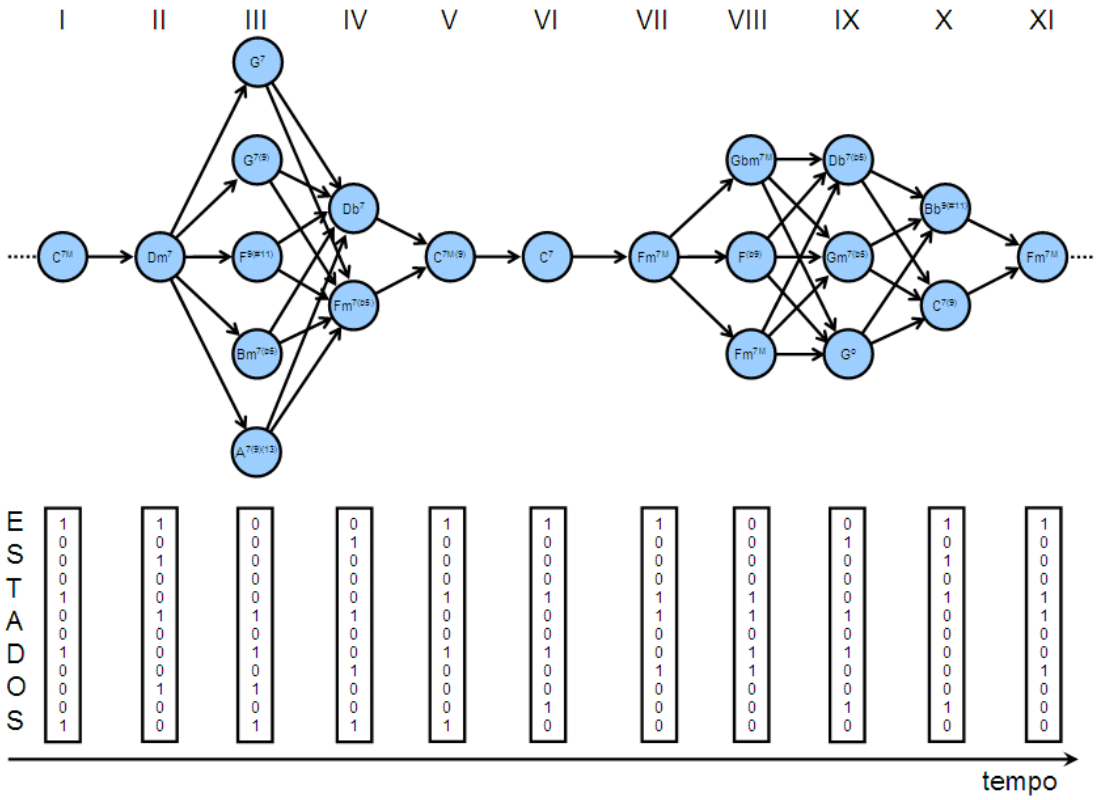


Figura 4.10 – Grafo construído a partir da segmentação inicial e escolha dos acordes com maior pontuação após a aplicação da função de utilidade sobre o estado de cada segmento e descarte dos acordes que não atingiram o limiar em cada segmento.

O problema passa então a ser a escolha do caminho no grafo *G* que represente mais fielmente a informação harmônica da peça musical em questão. Entretanto, desta forma torna-se possível fazer uso de informação contextual para escolha do acorde vencedor em cada camada, uma vez que é possível utilizar a informações dos prováveis acordes nas camadas vizinhas, ou mesmo calcular soluções considerando várias camadas consecutivas simultaneamente.

## 4.4. Análise Contextual para Escolha do Acorde Vencedor

Uma vez que a análise local mostrou-se ineficiente em casos onde havia ambigüidade, ou seja, mais de um acorde potencial para o conjunto de notas do segmento, chegou-se à conclusão que seria necessário uma abordagem que considerasse não somente a adaptação do estado do segmento a um dado padrão de acorde, mas também o contexto no qual o acorde estava inserido.

Antes de chegar à proposta final, considerada em mais detalhes nesta sessão, é importante discorrer, ainda que superficialmente, sobre as propostas iniciais, bem como suas justificativas, vantagens e os porquês de seus insucessos.

Todas as propostas partiram do grafo G, definido na sessão anterior, que apresenta características importantes:

- o grafo é acíclico e cada camada é totalmente conectada à camada posterior, e somente a ela;
- empiricamente, percebeu-se que G exibe uma quantidade grande de camadas contendo apenas um nó; tais camadas podem ser consideradas como camadas de certeza; e
- conseqüentemente, as demais camadas, contendo mais de um nó, podem ser consideradas como camadas de incerteza.

Duas técnicas poderiam ser utilizadas para busca do caminho mais adequado no grafo gerado: programação dinâmica e otimização local. Entretanto, a utilização de programação dinâmica pressupõe a definição das restrições da solução final, e das restrições entre camadas vizinhas, e embora fosse possível definir restrições entre camadas vizinhas considerando tonalidade, empréstimos modais, ou outros fatores musicais, o número de restrições abrangendo mais de duas camadas, afim de mapear relações harmônicas mais complexas, seria muito grande. Por outro lado, a busca local exige a definição de uma relação de vizinhança entre soluções, o que não é apropriado à maneira como o problema foi modelado até então. Assim, outras abordagens foram testadas até que se chegasse à abordagem final utilizada no processo proposto.

A abordagem inicial consistia na utilização de um algoritmo de busca do caminho ótimo no grafo, a partir da definição de pesos nos nós e nas arestas. O algoritmo de busca escolhido foi o *Dijikstra*, por ser um algoritmo ótimo, ou seja, garante que o melhor caminho será encontrado, e por apresentar boa performance. Imaginou-se inicialmente, que, uma vez definidos os pesos, o algoritmo de busca do caminho ótimo chegaria naturalmente ao resultado ótimo.

Entretanto, tal abordagem mostrou-se inadequada, devido a duas dificuldades principais: a primeira era a impossibilidade de definir relações contextuais mais amplas que as relações diretas entre vizinhos em camadas consecutivas, o que, por definição não faz sentido quando se deseja utilizar ao máximo a informação contextual dos dados como um todo. E a segunda dificuldade era a própria definição dos pesos das arestas, ainda que não utilizando toda a informação contextual desejada. Embora o valor da função de utilidade aplicada ao acorde representado pelo nó em questão pudesse ser utilizado em todas as arestas partindo dele, sem apresentar pontos negativos consideráveis, os pesos das arestas necessitariam ser calculados por uma função mais complexa, que incluísse também a informação contextual disponível. De outro modo, o algoritmo chegaria ao caminho definido pelos nós vencedores em cada camada. Uma das funções possíveis seria definir o peso de cada aresta em função da distância entre as tônicas dos nós associados a ela, num ciclo de quintas. Entretanto, apesar do ciclo de quintas medir razoavelmente bem o quão próximas estão duas notas, do ponto de vista tonal, tal mapeamento quase não faz uso das informações contextuais potenciais.

Sendo assim, a proposta seguinte considerou a definição de uma função para comparar a coerência de seqüências harmônicas, e a aplicação de tal função às seqüências harmônicas geradas por todos os caminhos possíveis do grafo. Entretanto, percebeu-se, através de experimentos preliminares, que seriam necessárias várias horas para analisar um trecho de menos de um minuto de música, considerando uma função de cálculo de coerência da seqüência harmônica com custo constante e muito próximo de zero. Isto tornou a abordagem computacionalmente inviável para utilização em larga escala, em especial se o custo da função para o cálculo da coerência harmônica não for constante, nem próximo de zero.

Tendo em mente que o tratamento do grafo inteiro é uma tarefa cara, uma terceira abordagem foi sugerida: a divisão do grafo G em vários sub-grafos $G_k$, contendo as camadas de incerteza adjacentes, e a resolução do problema separadamente para cada sub-grafo, como mostrado na figura 4.11. O algoritmo de divisão do grafo original considera que os sub-grafos serão definidos pelas camadas com mais de um nó, cercados pelo maior número de

camadas com um único nó consecutivas, o que, embora cause a replicação das camadas com um único nó em até dois sub-grafos, não aumenta a quantidade de processamento, porque a adição de camadas com apenas um nó não aumenta a complexidade do grafo, ou seja, não há incremento no número de caminhos total do grafo, dado que o fator de ramificação dos mesmos é 1.

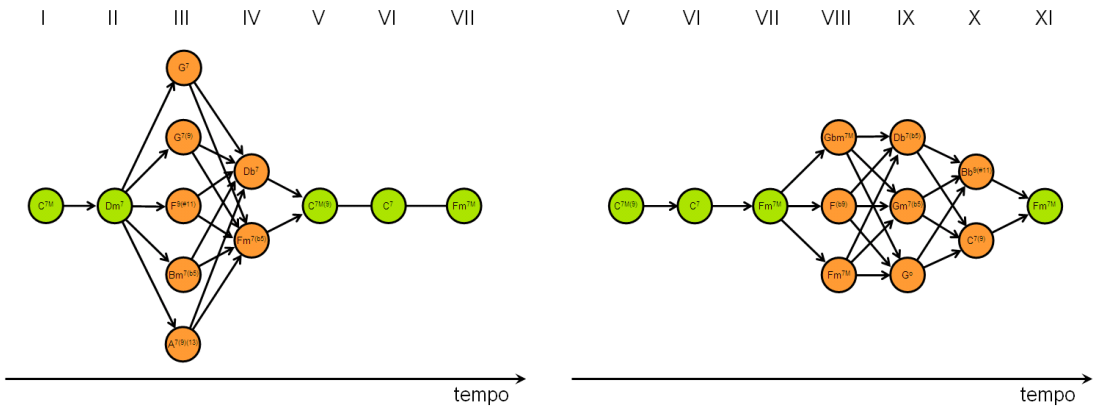


Figura 4.11 – Divisão do grafo da figura 4.9 em sub-grafos, isolando as camadas de incerteza (em laranja). Os nós verdes representam as camadas de certeza, que são replicadas nos sub-grafos anteriores e posteriores a elas (como as camadas V, VI e VII).

Formalmente, a divisão do grafo *G*, implica na construção de todos os grafos $G_k$ possíveis, através da função 'sub-grafo', que calcula o sub-grafo de um grafo *G*, da camada *p* até a camada *s*:

$$G_k = sub\text{-}grafo(G, p, s)$$

Os sub-grafos são construídos para todos os pares ordenados *{p, s}* que satisfazem as restrições a seguir, sempre que existirem também *q* e *r* tais que:

$$p \leq q \leq r \leq s$$

$$p \geq 1$$

$$s \leq n$$

$$\text{se } p > 1, \text{ então } | V_{p-1} | > 1$$

$$\text{se } s < n, \text{ então } | V_{s+1} | > 1$$

$$\text{para todo } V_j \text{ tal que } p \leq j < q, \text{ então } | V_j | = 1$$

$$\text{para todo } V_j \text{ tal que } q \leq j \leq r, \text{ então } | V_j | > 1$$

$$\text{para todo } V_j \text{ tal que } r < j \leq s, \text{ então } | V_j | = 1$$

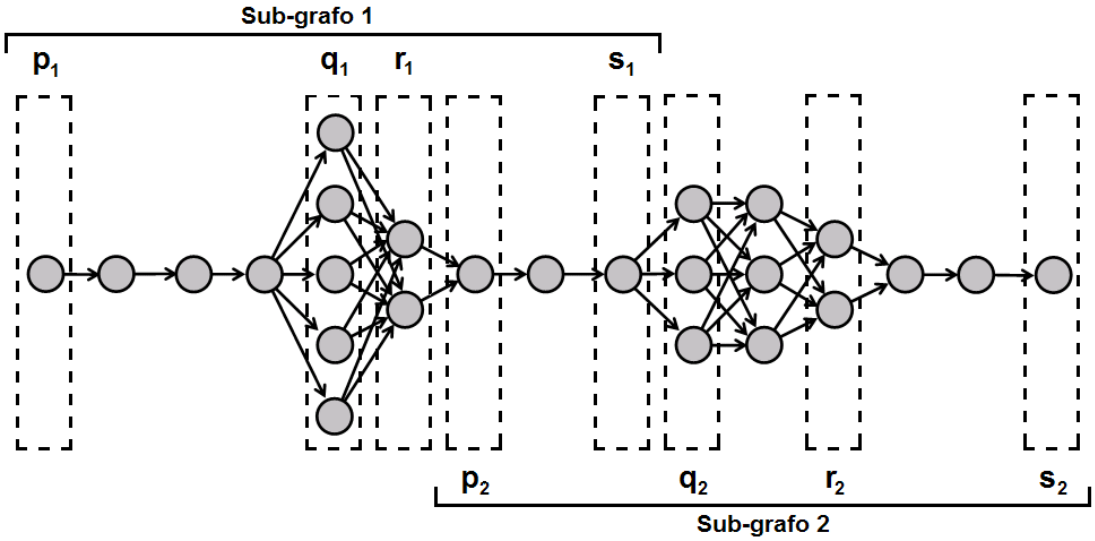


Figura 4.12 – Sub-grafos contendo ilhas de incerteza.

Desta forma, os sub-grafos construídos possuem no mínimo uma camada de incerteza (quando *q* = *r*). As camadas de incerteza (q a r) são cercadas pelas maiores camadas de certeza possíveis (*p* a *q-1* e *r+1* a *s*).

Embora a idéia de resolver os pontos de incerteza a partir das informações dos pontos vizinhos consideradas confiáveis seja musicalmente justificável, percebeu-se que a definição de uma boa função para o cálculo da coerência de seqüências harmônicas era um problema complicado e de difícil resolução.

Assim, decidiu-se por definir um conjunto de padrões de seqüências de acordes independentes de tônica, utilizando-se as funções harmônicas, em notação romana, e não os acordes propriamente ditos. Os padrões são aplicados sobre seqüências curtas, seguindo uma prioridade e classificando os acordes onde há incerteza. Para isso, utilizou-se o motor de inferência JEOPS [14], com uma base de regras prioritárias.

Os padrões definem seqüências de acordes recorrentes na harmonia jazzística, e o motor de inferência tenta aplicá-las a cada combinação de acordes que satisfaz as condições da regra em questão. No caso da regra ser disparada, marca-se os nós envolvidos como selecionados e marca-se a camada de cada nó envolvido como resolvida, como exemplificado na figura 4.13. Assim, evita-se a aplicação de novas regras nas camadas já envolvidas, e permite-se a exclusão, no final do processo, dos nós das camadas resolvidas que não foram selecionados. Desta maneira, no final da aplicação

das regras, reconstrói-se o grafo somente incluindo os nós marcados como selecionados, como mostrado posteriormente na figura 4.13.

De maneira formal, cada padrão de seqüência define um conjunto de caminhos no grafo que são recorrentes na harmonia jazzística, transformando o problema da busca do caminho mais adequado num problema de casamento de caminhos recorrentes dentro do grafo. Assim, não se faz necessária a definição de pesos nas arestas.

Tabela 4.4 – Regras para identificar padrões harmônicos recorrentes na harmonia jazzística.

| Nome da Regra | Função da Regra |
|---|---|
| **Regras que abrangem 6 camadas** | |
| **MajorChristophe** | **Identifica uma cadência *Christophe*, na tonalidade maior: $IV^{7M}$ $IIIm^{7}$ $VIm^{7}$ $IIm^{7}$ $V^{7}$ I** |
| **MinorHarmonicChristophe** | **Identifica uma cadência *Christophe*, na tonalidade menor harmônica: $IVm^{7}$ $IIIb^{7M(\#5)}$ $VIb^{7M}$ $IIm^{7(b5)}$ $V^{7}$ Im** |
| **Regras que abrangem 4 camadas** | |
| **MajorSixTwoFiveOne** | **Identifica uma cadência maior $VIm^{7}$ $IIm^{7}$ $V^{7}$ I** |
| **MinorHarmonicSixTwoFiveOne** | **Identifica uma cadência menor harmônica $VIb^{7M}$ $IIm^{7(b5)}$ $V^{7}$ Im** |
| **Regras que abrangem 3 camadas** | |
| **MajorTwoFiveOne** | **Identifica uma cadência maior $IIm^{7}$ $V^{7}$ I** |
| **MinorHarmonicTwoFiveOne** | **Identifica uma cadência menor harmônica $IIm^{7(b5)}$ $V^{7}$ Im** |
| **Regras que abrangem 2 camadas** | |
| **MajorFiveOne** | **Identifica uma cadência maior $V^{7}$ I** |
| **MajorTwoFive** | **Identifica uma cadência maior $IIm^{7}$ $V^{7}$** |
| **MinorHarmonicFiveOne** | **Identifica uma cadência menor harmônica $V^{7}$ Im** |
| **MinorHarmonicTwoFive** | **Identifica uma cadência menor harmônica $IIm^{7(b5)}$ $V^{7}$** |
| **CycleOfFifths** | **Identifica uma dominante secundária $V^{7}/V$ $V^{7}$** |

| Nome da Regra | Função da Regra |
|---|---|
| **Regras para identificar padrões menos comuns** | |
| MajorTwoFlatTwoOne | Identifica uma cadência maior<br>$IIm^7$ $IIb^7$ I |
| MinorTwoFlatTwoOne | Identifica uma cadência menor<br>$IIm^{7(b5)}$ $IIb^7$ Im |
| SecondaryDominant | Identifica outras dominantes secundárias<br>$V^7$/Y Y |
| MajorFourOne | Identifica uma cadência maior<br>$IV^{7M}$ I |
| MinorFourOne | Identifica uma cadência menor<br>$IVm^7$ Im |
| MinorHarmonicSevenOne | Identifica uma cadência menor harmônica<br>VII° Im |
| MajorFlatTwoOne | Identifica uma cadência maior<br>$IIb^7$ I |
| MinorFlatTwoOne | Identifica uma cadência menor<br>$IIb^7$ Im |
| **Regras adicionais de controle** | |
| MarkRemainingVertex | Marca os vértices das camadas ainda não resolvidas. |
| DeletedNonMarkedVertex | Remove os vértices não marcados das camadas já resolvidas, deixando apenas o nó escolhido, já que este é marcado quando da aplicação da regra que o escolheu. |
| ResolveRemainingVertex | Para cada par de nós marcados e que se encontram na mesma camada, remove o nó cuja pontuação é menor. |

Os padrões de seqüências de acordes utilizados são um sub-conjunto dos padrões utilizados pelo autor em trabalhos anteriores relacionados a análise harmônica funcional automática [33, 34, 35], e são mostrados na tabela 4.4, por ordem de prioridade. Além disso, algumas regras adicionais, não relacionadas aos padrões de seqüências recorrentes se fizeram necessárias para permitir o bom funcionamento do algoritmo. Percebe-se que, em grande parte dos casos, as regras são organizadas por ordem decrescente de tamanho da seqüência, isto é, quanto maior a seqüência, maior a sua prioridade.

A figura 4.13 exemplifica a aplicação da base de regras sobre um sub-grafo. Para cada sub-grafo encontrado (A) de acordo com o algoritmo

mostrado no Quadro 4.4, insere-se todos os nós na base de conhecimento e em seguida, executa-se o motor de inferência sobre os nós inseridos na base. Em (B), observa-se que a regra *MajorTwoFive* disparou para os nós em vermelho, logo, tais nós são marcados como selecionados e suas camadas são marcadas como resolvidas, em verde (C). Em seguida, a regra *MajorFlatTwoOne* é disparada para os nós em vermelho (C), e novamente tais nós e suas respectivas camadas são marcados como selecionados e resolvidos, respectivamente (D). Após o disparo de todas as regras de seqüências harmônicas possíveis, a regra *MarkRemainingVertex* é disparada para o nó em vermelho (D) e tal nó é então marcado como selecionado. A seguir, a mesma regra é disparada para os nós das camadas VI e VII, percebendo-se o resultado em (E). Quando a regra *MarkRemainingVertex* não pode ser mais aplicada a nenhum nó, o motor de inferência tenta aplicar a regra *DeleteNonMarkedVertex* aos nós não marcados como selecionados. No exemplo, esta regra é aplicada ao nó em vermelho (F), e o resultado é a remoção deste nó da base de conhecimento (G). Após a aplicação desta regra a todos os nós ainda não marcados, o resultado da execução do motor de inferência pode ser visto em (H).

Os resultados obtidos utilizando esta abordagem foram bastante satisfatórios e são discutidos no capítulo 7.

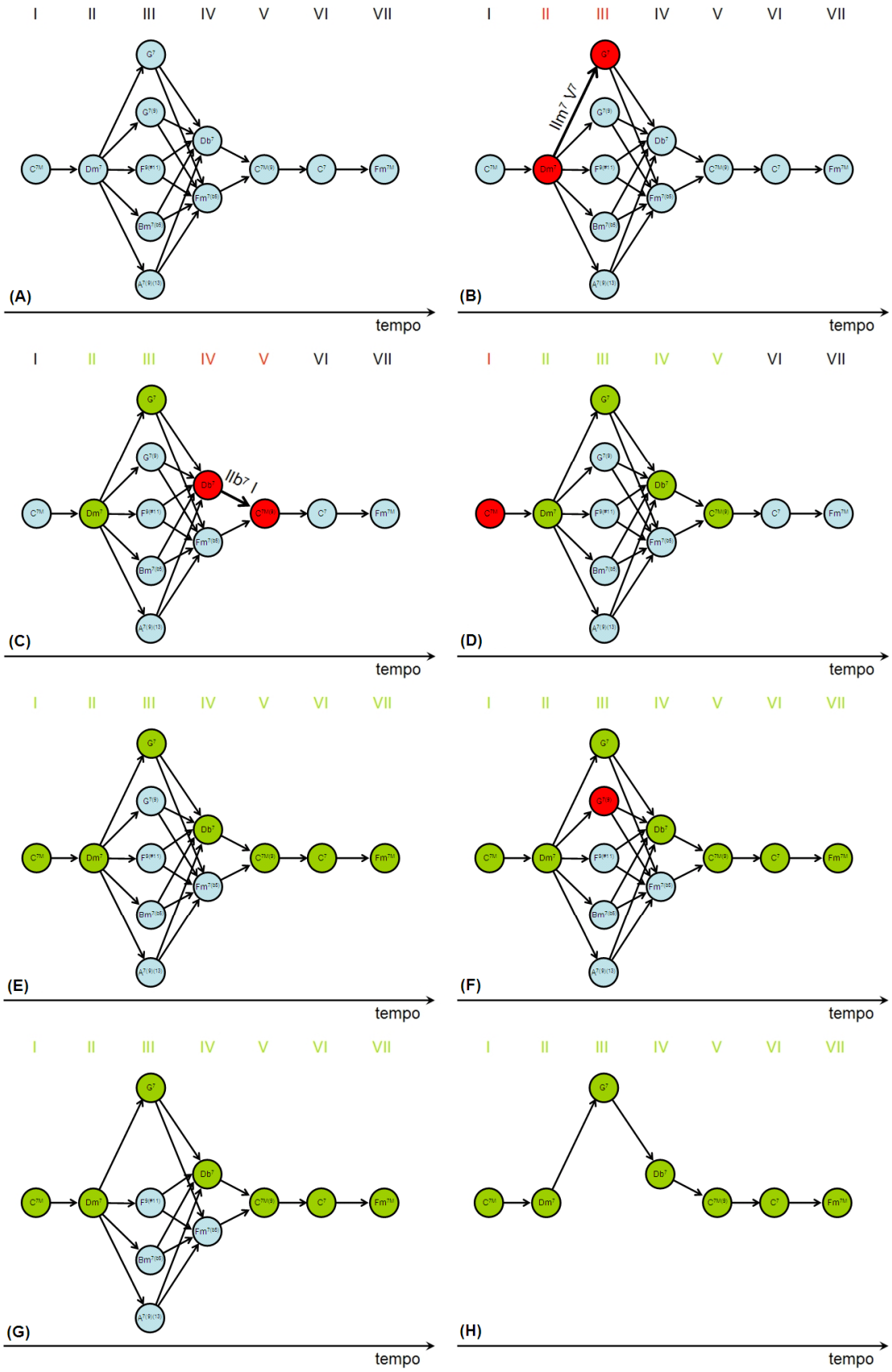


Figura 4.13 – Aplicação de regras sobre sub-grafos de ilhas de incerteza.

# 5. Precisão de Ataques em Seqüências Capturadas por Violões MIDI

Os processos de captura em violões MIDI são baseados em *hardware*, e naturalmente exigem algum tempo até que uma determinada nota executada no instrumento seja processada e o evento correspondente seja escrito na saída MIDI. O fato de o processo ocorrer em *hardware* sugere sua eficiência com relação ao tempo necessário para conversão do sinal de áudio em um evento MIDI. Entretanto, na literatura científica atual não foi encontrado nenhum trabalho que garantisse, ou ao menos indicasse, que os processos atuais de captura MIDI tinham precisão temporal razoável.

O experimento realizado consistiu na comparação entre a precisão temporal dos ataques nas seqüências capturadas pelo violão MIDI e dos ataques detectados nos arquivos de áudio pelo algoritmo de detecção de ataques do *BeatRoot*[1] [12], algoritmo vencedor da categoria *beat tracking* no MIREX 2006 (*Music Information Retrieval eXchange*) [27], competição de algoritmos de tratamento de áudio promovida pelo *International Music Information Retrieval Systems Evaluation Laboratory – ISMIRSEL*. Os resultados foram comparados com uma base rotulada a partir do áudio original, onde foram anotados os tempos de cada ataque. O algoritmo vencedor da categoria de detecção de ataques não possuía código aberto, impossibilitando a sua utilização nos experimentos.

A partir das comparações, foram medidos os seguintes aspectos: precisão, *recall* e medida-f (*f-measure*), cujos conceitos serão explicados posteriormente, além da matriz de confusão. Após os experimentos, chegou-se à conclusão de que o aspecto temporal do processo de captura MIDI é tão preciso quanto os algoritmos atuais de detecção de ataques, saindo-se ligeiramente melhor em alguns casos. Os experimentos foram realizados em conjunto com Cássio [6], e foram utilizados em parte do seu Trabalho de Graduação.

[1] De fato, o algoritmo *BeatRoot* foi vencedor da competição entre algoritmos de *beat tracking* do MIREX. Entretanto, dado que o *beat tracking* exige a realização preliminar de detecção de ataques, e que os vencedores nesta categoria não tinham código aberto, o *Beat Root*, que tem parte do código sob licença GPL, pode ser considerado como uma opção razoável para os experimentos.

## 5.1. Dados Rotulados

A base de dados rotulada consiste de trechos de 12 músicas, aleatoriamente coletados da base de dados do projeto “Um País, Um Violão”, com duração variando entre 8 e 12 segundos cada um, totalizando cerca de 121 segundos de música e 376 ataques.

Para rotulação dos dados, utilizou-se o SOL [23], um *toolbox* para Matlab que permite a um músico, além de escutar o trecho de música sendo rotulado, também visualizar o sinal e o espectrograma do mesmo e marcar o ponto exato em que ocorreu o ataque, como mostrado na figura 5.1.

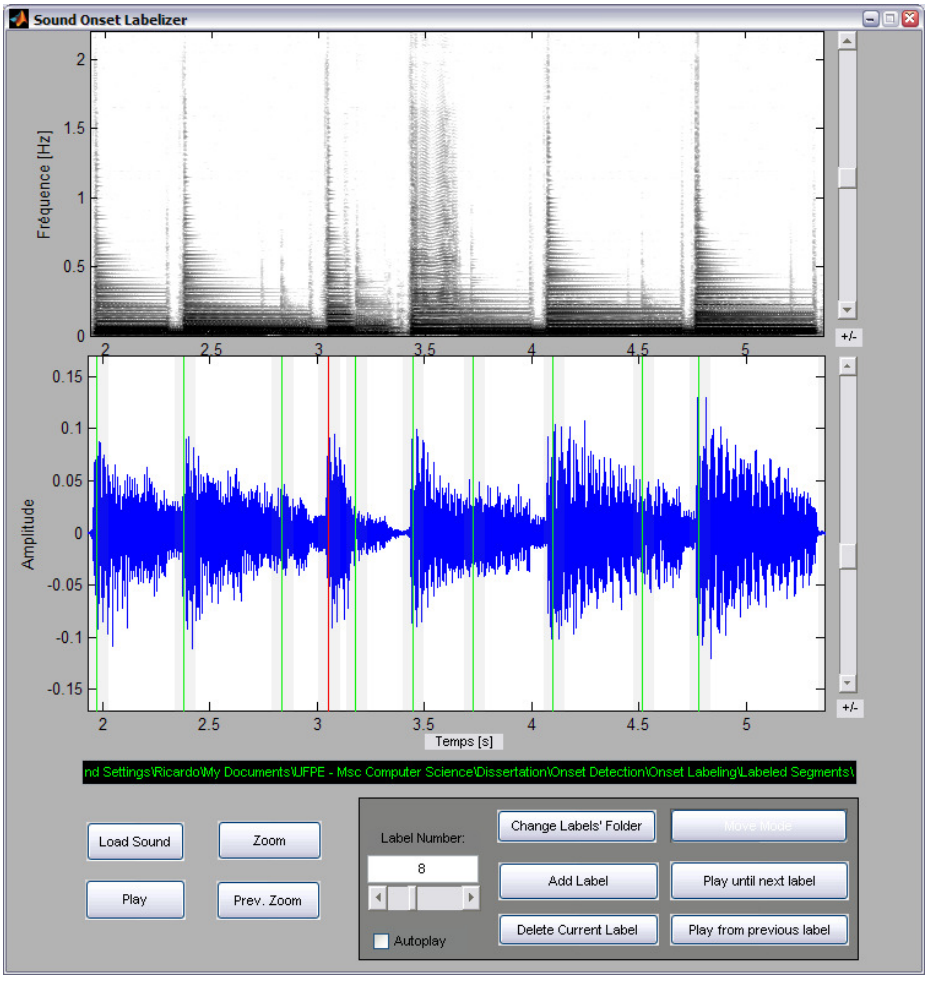


Figura 5.1 – O *Toolbox* SOL, utilizado para etiquetagem dos ataques em áudio, rodando sobre o Matlab: linhas verticais verdes marcam pontos de ataque.

Entretanto, uma pequena alteração foi realizada no código fonte do *toolbox*, para que fosse gerado um arquivo de *log* de ataques, i.e., uma saída em formato texto, contendo os pontos em que ocorreram os ataques

rotulados, em segundos e notação científica, separados por quebras de linha, como no quadro 5.1.

```
1.9715153e+000
2.3803355e+000
2.8359545e+000
3.0511824e+000
3.1805026e+000
3.4438581e+000
3.7282848e+000
4.0998540e+000
4.5171873e+000
4.7800607e+000
```

Quadro 5.1 – Exemplo de arquivo texto de rótulo. Entrada em vermelho corresponde à linha vertical vermelha na figura 5.1, enquanto entradas em verde correspondem às demais linhas verticais da mesma figura.

Desta forma, o resultado de qualquer técnica de detecção de ataques, em qualquer formato, pode ser comparado com a base rotulada, sem necessidade de ajustes no módulo de comparação, desde que a técnica utilizada forneça como saída um arquivo de texto no mesmo formato dos arquivos de *log* de ataques da base rotulada.

Por fim, é importante ressaltar que, embora as puxadas ao violão sejam constituídas, formalmente, por vários ataques quase simultâneos, é impossível para o ouvido humano diferenciá-los, mesmo com a ajuda de ferramentas como o SOL. Logo, as puxadas foram consideradas como um único ataque nos arquivos rotulados.

## 5.2. Experimento

Como anteriormente explicitado, o algoritmo de detecção de ataques em arquivos de áudio escolhido foi o utilizado pelo *BeatRoot*, versão 0.5.3 [12]. Adicionalmente, realizou-se uma pequena alteração no código fornecido pelo autor para permitir a geração de um arquivo de *log* de ataques, o que de fato foi utilizado nos experimentos, descartando-se as outras saídas do algoritmo.

Paralelamente, os trechos MIDI equivalentes aos trechos de áudio rotulados passaram por um processamento para gerar os arquivos de *log* de ataques, em formato texto e seguindo a mesma sintaxe dos rótulos dos arquivos de áudio, contendo as informações de cada ataque, também em segundos.

Entretanto, dado que os arquivos de rótulo consideram as puxadas como um único evento de ataque, é necessário adaptar o algoritmo de geração do arquivo de *log* de ataques utilizado nas seqüências MIDI, para garantir que, nas puxadas, apenas um evento de ataque será incluído no *log*, dentre todos os eventos NOTE ON gerados por cada corda do violão tocada.

Para tal, passou-se a utilizar a política de considerar apenas o primeiro evento NOTE ON nas seqüências de eventos de ataque dentro de uma janela de 20 milésimos de segundos.

O processamento das seqüências MIDI utilizou o algoritmo definido no quadro 5.2 a seguir.

```
função double[] extrairAtaquesMidi(Midi midi, long tamanhoDoTick){

      MidiEvent[] eventos <= midi.events

      Set conjDeOnsets <= [ ]

      //Cálculo do tempo de cada evento de ataque
      para cada evento em eventos
            se event.type = EVENT_NOTE_ON então
                  se event.velocity > 0 então
                        conjDeOnsets.add(tamanhoDoTick * event.tick)
                  fim do se
            fim do se
      fim do para

      double[] ataques <= conjDeAtaques.toArray();

      //Eliminação dos ataques muito próximos
      double JANELA <= 0.02; //20 milésimos de segundo
      List resultado;

      double ataqueAnterior <= Double.MIN_VALUE;

      para cada ataque em ataques
            se (ataque - ataqueAnterior) > JANELA então
                  resultado.add(ataque)
                  ataqueAnterior <= ataque
            final do se
      final do para

      retorne result.toArray()
}
```

Quadro 5.2 – Algoritmo de extração de ataques em seqüências MIDI.

Dado que as informações de tempo no formato MIDI são todas discretas e medidas em *ticks*, são necessários alguns cálculos matemáticos para descobrir, em milésimos de segundo, o instante de um dado *tick*.

Seja *t* o *tick* em questão, e *R* a resolução do MIDI, ou seja, a quantidade de *ticks* por unidade de tempo da música (*beat*). Considere também *B* como sendo a quantidade de unidades de tempo por minuto, em BPM (*beats per minute*) e *d* como sendo a duração de um *tick*, em milésimos de segundo. Assim, os cálculos são realizados como mostrado no quadro 5.3.

$$\frac{1}{d} = \frac{R \ \ ticks}{beat} \times \frac{B \ \ beats}{\min}$$

$$\frac{1}{d} = \frac{R \ \ ticks}{beat} \times \frac{B \ \ beats}{60 \ \ s}$$

$$\frac{1}{d} = \frac{R \times B \ \ ticks}{60 \ \ s}$$

$$\frac{1}{d} = \frac{R \times B \ \ ticks}{60 \times 10^{3} \ ms}$$

$$d = \frac{60 \times 10^{3} \ ms}{R \times B \ \ ticks}$$

Quadro 5.3 – Fórmula do cálculo da duração de um *tick* MIDI, em milésimos de segundo.

Considerando que a resolução das seqüências MIDI do projeto “Um País, Um Violão” é de *480 ticks/beat*, e todos foram gravados a *100 BPM*, cada *tick* tem duração de *1,25 milésimos de segundo*.

Embora os arquivos de áudio e as seqüências MIDI do projeto “Um País, Um Violão” tenham sido gravados simultaneamente, não há um mecanismo seguro para garantir que as partes de ambos escolhidas para compor o conjunto de testes estariam sincronizadas entre si, considerando a precisão de milésimos de segundo. A ferramenta utilizada para corte dos arquivos somente possibilita garantir, por meios visuais e auditivos, que ambos os arquivos representam um mesmo trecho de música, sem maiores garantias de micro-sincronização. Portanto, fez-se necessária a adoção de uma política de sincronização dos arquivos, após a detecção dos ataques, para que as métricas não fossem distorcidas por erros de sincronização entre o áudio, as seqüências MIDI e os arquivos rotulados.

É bastante razoável assumir que os algoritmos de detecção e o processo de captura MIDI apresentam uma latência de pouca variância, ou seja, o atraso da detecção não apresenta variações consideráveis. Assumindo-se que o que se deseja calcular é a precisão da detecção de ataques com relação a uma base etiquetada, optou-se por realizar o alinhamento do primeiro evento

de ataque de cada arquivo de *log* e o cálculo das métricas sobre os demais eventos de ataque. Ou seja, para cada arquivo de *log*, subtraiu-se de cada evento posterior ao primeiro, a quantidade de milésimos de segundo do primeiro evento, e calcularam-se as métricas em torno dos eventos posteriores ao primeiro, como mostrado na figura 5.2 (A).

É importante ressaltar que durante a comparação entre dois arquivos de *log* de ataques, cada ataque no arquivo de *log* da base rotulada só pode corresponder, no máximo, a um ataque no arquivo de *log* de entrada, e vice-versa.

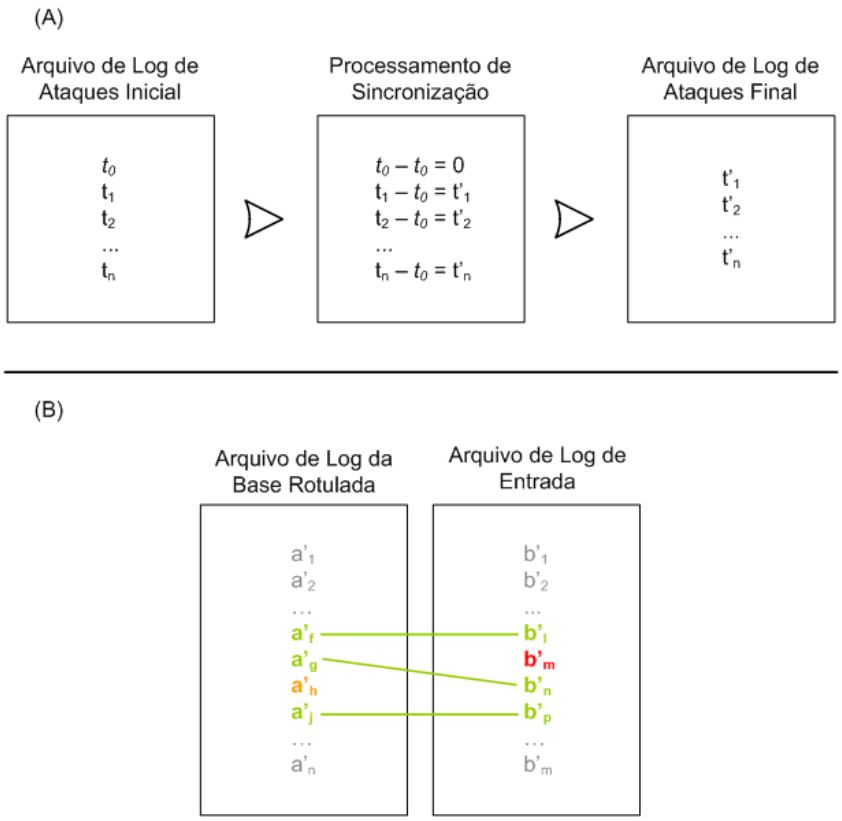


Figura 5.2 – Sincronização dos *logs* de ataque. Processamento inicial (A), onde todos os ataques são re-calculados subtraindo-se o valor do primeiro ataque. E exemplo de comparação (B): positivos-verdadeiros em verde, falso-negativos em laranja e falso-positivos em vermelho.

É evidente que as métricas por si só não garantem a precisão das técnicas envolvidas, somente sendo suficientes para mensurar o quão melhor ou pior é uma técnica em relação à outra técnica com a qual está sendo comparada. Entretanto, a utilização da base rotulada como representando o resultado de

uma técnica de detecção de ataques ideal, i.e., um *benchmark*, permite definir qual das técnicas pesquisadas é, de fato, mais precisa.

As métricas utilizadas para comparação dos arquivos de *log* são baseadas nos conceitos a seguir:

- **Verdadeiro-Positivo (ou acerto)**: são considerados como acertos, os ataques do arquivo de *log* que tenham ataques correspondentes no arquivo de *log* do rótulo, considerando-se uma janela de 15 milésimos de segundo, para mais ou para menos. Isto é, se o arquivo de *log* contém um ataque no tempo *t*, tal ataque é considerado como um acerto se, e somente se, o arquivo de rótulo contiver um ataque com tempo $t_R$, tal que $t - 15 < t_R < t + 15$, e o ataque $t_R$ não houver sido associado a nenhum outro ataque do arquivo de log anteriormente.
- **Falso-Negativo (ou *miss*)**: de maneira análoga, são considerados falso-negativos os ataques do arquivo de *log* do rótulo que não tenham ataques correspondentes no arquivo de *log* de entrada, considerando-se a mesma janela de 15 milésimos de segundo, para mais ou para menos.
- **Falso-Positivo**: de maneira contrária, são considerados falso-positivos os ataques do arquivo de *log* que não tenham ataques correspondentes no arquivo de *log* do rótulo, considerando-se novamente a janela de 15 milésimos de segundo, em ambas as direções.

Tais medidas podem fornecer informações importantes quanto à qualidade das abordagens analisadas, entretanto, utilizando fórmulas apropriadas pode-se extrair informações mais significativas para o que se deseja analisar. Com base nas métricas utilizadas no MIREX 2006 (*Music Information Retrieval eXchange*) [27], decidiu-se por utilizar as métricas definidas a seguir:

- **Precisão (P)**: $P = VP / (VP + FP)$
- ***Recall* (R)**: $R = VP / (VP + FN)$
- **Medida-F (F)**: $F = 2 * P * R / (P + R)$

Onde VP são os verdadeiro-positivos, FP os falso-positivos, e FN os falso-negativos. Numa situação ideal, FP e FN tem valor nulo, e VP tem valor 1.

Logo, os valores de precisão, *recall* e medida-f, em situações ideais, devem ser 1.

A sessão 7.2 do sétimo capítulo trata da análise dos resultados obtidos na comparação dos arquivos de *log* de ataques gerados pela captura MIDI e pelo algoritmo de detecção de ataques com os arquivos de *log* da base rotulada.

# 6. Implementação

Neste capítulo, os pormenores da implementação e da modelagem, bem como os artefatos produzidos, serão detalhados. Na sessão seguinte, a organização dos módulos desenvolvidos será exposta, bem como as principais responsabilidades de cada módulo, e seus sub-módulos. Em seguida, a arquitetura do projeto será detalhada.

## 6.1 Organização dos Módulos Desenvolvidos

A implementação dos algoritmos foi realizada em linguagem Java, utilizando-se as APIs *Rittornello* [36] e *Java Sound MIDI* [40], e estendendo a primeira quando necessário. Assim, as implementações resultantes desta pesquisa acabaram por gerar uma série de novos módulos, que podem ser usados juntamente com a API *Rittornello*, e provêem funcionalidades específicas. Os novos módulos desenvolvidos são descritos a seguir.

- ***Parser:*** permite utilizar entradas de texto representando acordes, no formato descrito no apêndice A desta dissertação. O *parsing* de cifras em formato texto foi utilizado para ler as informações dos arquivos de rótulo e gerar as estruturas Java adequadas, representando cada acorde. Sua implementação foi adaptada de um trabalho anterior do autor [33].
- ***High Level MIDI Handler:*** é um módulo para manipulação em alto nível de seqüências MIDI, que utiliza, de maneira transparente para o programador, a API *Java Sound MIDI* (`javax.soud.midi`) [40]. Neste módulo, foram definidos eventos de alto nível, com conceitos mais próximos dos conceitos utilizados nas partituras, como eventos de nota (formados por dois eventos MIDI: NOTE ON e NOTE OFF), além de um sub-módulo específico que considera as particularidades da execução ao violão, contendo conceitos como *Guitar Note Event*, que são eventos de nota executados ao violão e, portanto associados a uma corda e uma casa no braço do violão; Através da definição de sub-módulos específicos, é possível estender com facilidade o módulo original para adaptar-se a execuções em outros instrumentos, se necessário.
- ***MIDI Noise Cleaner:*** é um módulo que realiza a correção de ruídos em seqüências capturadas por violões MIDI. Contém também um sub-módulo

para cálculo de estatísticas de acertos sobre uma base rotulada, permitindo a validação de novas bases de regras, visando melhorar os resultados, ou adapta-los a outros tipos de ruídos que possam ocorrer em gravações específicas ou mesmo capturas a partir de outros instrumentos. Além disso, dado que a abordagem de correção de ruídos é baseada em regras, este módulo possui um sub-módulo de re-compilação, facilitando a integração com o motor de inferência JEOPS [14] quando os arquivos de regras são modificados.

- ***MIDI Onsets Detector:*** é o módulo responsável pela extração de ataques em seqüências MIDI. Permite ao programador configurar os parâmetros da extração, como o tamanho da janela na unificação de eventos de ataque muito próximos, além de outros parâmetros na política de extração, de acordo com suas necessidades específicas. Possui um sub-módulo para comparação de precisão de ataques entre diferentes técnicas de extração. O sub-módulo compara arquivos de texto contendo uma lista de tempos de ataques, permitindo a comparação de técnicas diferentes aplicadas a uma mesma base de dados, ou ainda aplicadas a dados em formatos diferentes (como áudio *versus* MIDI), desde que as técnicas sendo comparadas forneçam arquivos de *log* bem formatados.
- ***MIDI Chords Recognizer:*** é o módulo que realiza a extração de harmonia em seqüências MIDI utilizando a técnica proposta nesta dissertação. Dado que parte da abordagem para reconhecimento de acordes é baseada em regras, possui um sub-módulo de recompilação, facilitando a integração com o motor de inferência JEOPS [14] quando os arquivos de regras são modificados. Além disso, possui um sub-módulo para leitura de bases rotuladas (seguindo o formato de rótulo definido nesta dissertação), e um sub-módulo para comparação de resultados com as bases rotuladas ou outras técnicas, quando por exemplo, desejar-se modificar as bases de regras.

## 6.2. Arquitetura

O projeto seguiu as premissas de desenvolvimento orientado a objetos, utilizando alguns padrões de projeto já estabelecidos no mercado, sempre que aplicáveis, visando aumentar sua facilidade de extensão, reuso e manutenção.

A arquitetura apresenta apenas duas camadas, negócio e dados, com uma interface de dados entre elas. Não foi implementada qualquer interface gráfica para os experimentos, já que todos foram realizados através de leitura de arquivos em *batch* e geração de arquivos de *log* de resultados, via comandos no *prompt* de comandos. A justificativa para tal é a quantidade de dados a ser analisada, e principalmente a quantidade de informações apresentada a cada relatório de estatísticas.

Assim, a geração de arquivos de *log* de estatísticas permite mais conforto na utilização dos módulos, inclusive pela facilidade de imprimir os resultados quando necessário. Portanto, não há camada de interface gráfica na arquitetura, embora os padrões de desenvolvimento utilizados facilitem bastante uma futura implementação de interface gráfica, se necessário. Uma versão reduzida do diagrama de classes, contendo apenas as classes mais importantes da arquitetura, é mostrada no apêndice C.

Priorizou-se pela alta coesão e acoplamento fraco durante o desenvolvimento de cada módulo, representado por um pacote específico. O conteúdo de cada pacote é descrito a seguir.

- `scholz.basic`: contém classes básicas da modelagem, como o objeto que representa uma seqüência MIDI e fornece métodos de alto nível para acessar atributos e informações, e realizar comparações entre seqüências MIDI.
- `scholz.basic.handler`: contém classes para manipulação de sequências MIDI em baixo nível, encapsulando as constantes definidas no protocolo MIDI e o processamento em baixo nível das mensagens, *byte* a *byte*.
- `scholz.basic.highlevel`: define objetos que representam conceitos musicais em alto nível, como nota, nota de guitarra e acordes.
- `scholz.cleaner`: contém a fachada para os serviços de correção de ruídos, bem como o controlador e as demais classes de negócio relacionadas à correção de ruídos nas seqüências capturadas por violões MIDI.

- `scholz.cleaner.knowledgebase`: contém os arquivos das bases de dados gerados pelo JEOPS [14], além de uma classe utilitária para realizar a compilação e cópia das bases de conhecimento geradas para a pasta e pacotes corretos, sempre que alguma mudanças nos arquivos de regras for realizada.
- `scholz.cleaner.statistics`: fornece a implementação do cálculo das estatísticas sobre o algoritmo de correção de ruídos, a partir de uma base rotulada.
- `scholz.harmonyextractor`: contém a classe de fachada da extração de harmonia, além das classes básicas de negócio relativas à tarefa de extração harmônica e classes utilitárias diversas.
- `scholz.harmonyextractor.graph`: contém classes básicas para implementação genérica de grafos, com conceitos como grafo, vértice e aresta.
- `scholz.harmonyextractor.graph.dijikstra`: contém as classes específicas que implementam as particularidades do grafo para realização da busca do melhor caminho utilizando o algoritmo Dijikstra.
- `scholz.harmonyextractor.graph.global`: contém as classes específicas que implementam as particularidades do grafo para realização da busca do melhor caminho utilizando a abordagem de busca global (percorre cada caminho possível, atribuindo uma pontuação à análise harmônica realizada, e escolhe o caminho de melhor pontuação). Este pacote não se encontra completamente implementado, já que os experimentos iniciais mostraram a inviabilidade computacional desta abordagem, conforme descrito no quarto capítulo.
- `scholz.harmonyextractor.graph.hybrid`: contém as classes específicas que implementam as particularidades do grafo para realização da busca do melhor caminho utilizando a abordagem híbrida (divisão do grafo em sub-grafos e análise global em cada sub-grafo). Entretanto, a função de utilidade para uma dada seqüência harmônica não chegou a ser definida, pelos motivos explicitados anteriormente no capítulo 4.

- `scholz.harmonyextractor.graph.islands`: contém as classes específicas que implementam as particularidades do grafo para realização da busca do caminho mais adequado utilizando a abordagem final desta proposta, que consiste na divisão do grafo em vários sub-grafos, e na utilização de regras sobre o motor de inferência JEOPS [14] para escolher os melhores vértices em cada camada dos sub-grafos.
- `scholz.harmonyextractor.graph.islands.knowledgebase`: contém a classe que representa a base de regras de padrões harmônicos recorrentes, além da classe de controle utilizada para guardar o estado das camadas durante a execução do motor de inferência e classes auxiliares para compilação do arquivo de regras.
- `scholz.harmonyextractor.graph.bpardo`: implementação fiel do algoritmo de Birmingham e Pardo [30], como *benchmark* para comparação de resultados.
- `scholz.harmonyextractor.statistics`: módulo contendo implementação do cálculo das métricas definidas para comparação dos resultados em extração harmônica, utilizando uma base rotulada.
- `scholz.onset`: fornece a classe de fachada para as tarefas relacionadas aos experimentos em detecção de ataques.
- `scholz.onset.statistics`: fornece a implementação do cálculo das estatísticas dos experimentos em detecção de ataques, a partir de entradas no formato definido no capítulo 4.
- `scholz.parser`: implementação do *parser* para reconhecimento de cifras em formato texto, no padrão definido pelo Apêndice A desta dissertação.
- `scholz.storage`: contém as interfaces com a camada de dados e as classes para leitura de seqüências MIDI do sistema de arquivos, retornando objetos Java definidos no módulo *High Level MIDI Handler*.

# 7. Experimentos e Resultados

Os experimentos realizados tiveram como objetivo mensurar a qualidade da abordagem proposta com relação aos requisitos definidos no segundo capítulo, para os dados não quantizados e ruidosos do projeto “Um País, Um Violão”, que contêm gravações de intérpretes executando acompanhamentos em bossa-nova em um violão MIDI.

Neste capítulo, serão detalhados os experimentos realizados e os resultados obtidos.

## 7.1. Minimização de Ruídos em Seqüências Capturadas por Violões MIDI

Os experimentos em minimização de ruídos foram realizados com um conjunto de 10 músicas do projeto “Um País, Um Violão”, cujos ruídos foram manualmente corrigidos por Lima [24]. O conjunto continha um total de 1264 eventos ruidosos, dos quais aproximadamente 72,15% eram eventos indevidamente adicionados, e 27,85% eram de eventos com freqüência mal reconhecida.

A abordagem atingiu os seguintes valores percentuais de falso-positivos, falso-negativos e verdadeiro-positivos na identificação e correção de ruídos, medidos através da subtração de eventos idênticos em seqüências MIDI, utilizando o pacote básico para manipulação de seqüências MIDI descrito no capítulo 6.

**Falso-positivos (FP):** 0%

**Falso-negativos (FN):** 47.62%

**Verdadeiro-positivos (VP):** 52.38%

Embora os resultados não pareçam tão bons, é importante ressaltar que cada minuto de música levou cerca de 8 horas para ser manualmente corrigido por Lima [24]. Logo, cinqüenta por cento de redução nos ruídos significa a redução de cerca de 4 horas de trabalho, para cada minuto de música. Pode-se então considerar um resultado razoável.

Além disso, uma das razões pelas quais os resultados não foram melhores é a exigência de uma taxa nula de falso-positivos. Visando garantir tal característica na abordagem, foi necessário remover diversas regras que, em alguns casos, causavam falso-positivos. Agora, o desafio passa a ser o

aumento da taxa de verdadeiro-positivos, mantendo a taxa nula de falso-positivos.

## 7.2. Extração de Acordes em Seqüências Capturadas por Violões MIDI

Nesta sessão, serão explicitados os detalhes dos experimentos realizados. A sub-sessão 7.2.1 trata da etiquetagem dos dados do conjunto de testes, enquanto a sub-sessão seguinte detalha as políticas e métricas utilizadas no cálculo dos resultados. A sub-sessão 7.2.3 mostra os resultados obtidos, e por fim, a sub-sessão 7.2.4 realiza uma análise objetiva sobre os mesmos.

### *7.2.1. Etiquetagem do Conjunto de Testes*

O conjunto de dados rotulados consistiu de quatro gravações do projeto "Um País, Um Violão" realizadas por dois violonistas, totalizando 270 acordes. Todas as gravações tiveram os ruídos eliminados por Lima [24].

Os dados do conjunto de testes para reconhecimento de harmonia foram rotulados inserindo-se uma nova trilha nas seqüências MIDI, contendo um evento de nota (de fato, dois eventos MIDI: NOTE ON e NOTE OFF), com duração de um *tick*, a cada mudança de acorde. Assumiu-se que há mudança de acorde quando a primeira nota de um novo acorde é tocada.

Entretanto, a nova seqüência MIDI rotulada somente contém informações sobre os pontos de partição, ou seja, os pontos em que há mudança de acordes, mas não contém as informações de que acordes são tocados em cada segmento. Logo, um arquivo em formato texto é utilizado para listar os acordes identificados na seqüência MIDI, utilizando a convenção sobre notação de acordes descrita no apêndice A desta dissertação. Neste arquivo, os acordes devem ser listados na ordem em que ocorrem na seqüência MIDI sendo um acorde por linha. Assim, a trilha adicionada à seqüência MIDI deve conter tantos eventos de nota quantas forem as linhas do arquivo texto.

O par de arquivos de rótulo é lido e uma estrutura contendo toda a informação rotulada é gerada: uma lista de acordes, onde cada acorde tem a informação do *tick* em que é executado. A figura 7.1 mostra um par de arquivos de rótulo para um trecho de música.

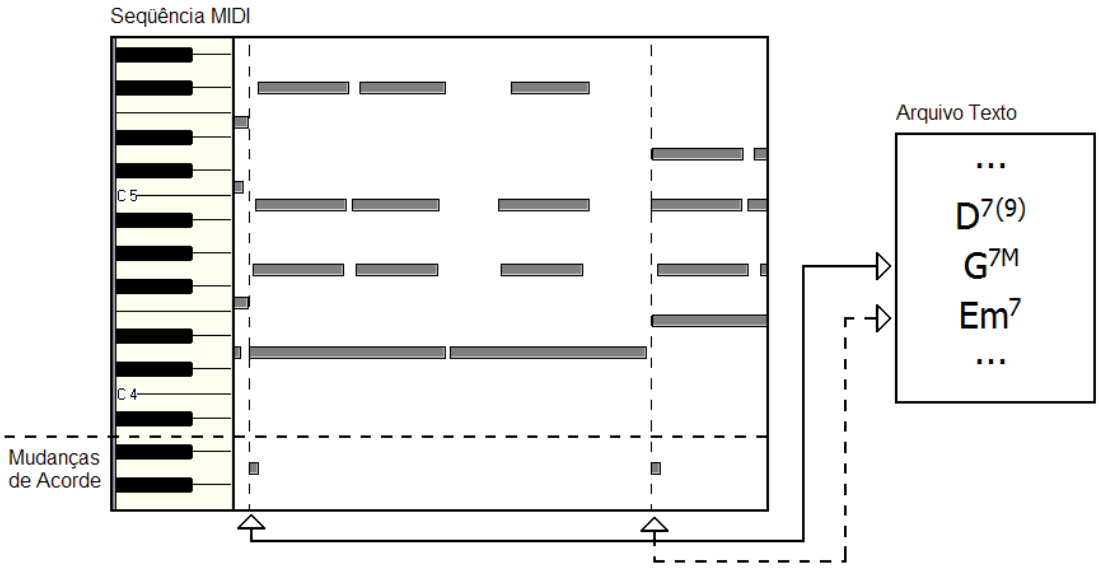


Figura 7.1 – Par de arquivos de rótulo para experimentos em reconhecimento de acordes: eventos são adicionados nas seqüências MIDI para marcar mudanças de acorde, enquanto arquivo texto provê informação dos acordes utilizados.

### *7.2.2. Políticas de Comparação de Acordes e Métricas Utilizadas*

Os experimentos consistiram na extração automática da harmonia das seqüências da base de testes, utilizando o processo proposto, e comparando os resultados obtidos com a base etiquetada. Diversos experimentos foram realizados durante o desenvolvimento das diversas técnicas testadas antes que se chegasse à proposta final. Entretanto, somente o experimento final, realizado com a abordagem de fato proposta por esta dissertação, terá os resultados mostrados nesta sessão, por acreditar-se que é fora do escopo e sem maior utilidade a discussão sobre os resultados das demais abordagens testadas, uma vez que não foram satisfatórios e tais abordagens tiveram suas deficiências sanadas pela última abordagem de fato proposta pelo COCHONUT.

A definição das métricas e políticas envolvidas na comparação dos resultados com a base etiquetada mereceu atenção especial, para melhor mensurar quão boa era a solução proposta, de acordo com os requisitos do problema. Assim, a primeira questão foi decidir quando uma classificação deveria ser considerada correta. A importância de tal decisão se deve ao fato de algumas pequenas variações nos acordes não implicarem em grandes diferenças harmônicas, entretanto, de acordo com os requisitos definidos no

segundo capítulo, era preciso mensurar o quanto os acidentes e dissonâncias estavam sendo corretamente reconhecidos.

Portanto, foram definidas duas políticas de comparação de acordes: a primeira visa mensurar o acerto das informações principais dos acordes reconhecidos (a saber, tônica, terça e quinta), ou seja, a corretude harmônica. Já a segunda política visa mensurar a completude harmônica através da comparação exata com os acordes da base etiquetada, incluindo sétimas, nonas, décimas primeiras, décimas terceiras e suspensões. As diferenças entre acordes enarmônicos não foram consideradas como erros, já que há abordagens que realizam análise harmônica capazes de corrigir tais informações com certa facilidade [33], e não era o foco da abordagem para descoberta de harmonia a realização de um pós-processamento para realizar análise harmônica, visando a correção deste tipo de informação. Para melhor ilustrar as duas políticas de comparação de acordes, a tabela 7.1 mostra alguns exemplos e o resultado da comparação, considerando cada política.

Tabela 7.1 – Exemplos de comparação de acordes considerando ambas as políticas definidas.

| Acorde Real (Base Rotulada) | Acorde Reconhecido (Algoritmo de Reconhecimento) | Política Básica (Tônica, 3ª e 5ª) | Política Completa (Todas as notas do acorde) |
|---|---|---|---|
| $C^{7M(9)}$ | $Cm^{7M}$ | Diferentes | Diferentes |
| $C^{7M(9)}$ | $C^{7M(b9)(13)}$ | Iguais | Diferentes |
| $C^{7M(9)}$ | $C^{7M(9)}$ | Iguais | Iguais |
| $Cm^{7(b9)(b13)}$ | $Cm^{7(b5)}$ | Diferentes | Diferentes |
| $Cm^{7(b9)(b13)}$ | $Cm^{7(11)}$ | Iguais | Diferentes |
| $Cm^{7(b9)(b13)}$ | $Cm^{7(b9)(b13)}$ | Iguais | Iguais |

Na primeira linha, os acordes são considerados diferentes porque o acorde da base rotulada tem terça maior, e o acorde reconhecido, terça menor. Na segunda linha, a política básica identifica que ambos os acordes são formados pelas mesmas tônicas, e apresentam uma terça maior e uma quinta justa, considerando-os iguais. Entretanto, a política completa identifica a diferença entre as nonas (no acorde da base rotulada a nona é maior, enquanto no acorde reconhecido pelo algoritmo a nona é menor) e o reconhecimento enganado de uma décima terceira, considerando os acordes diferentes. Por

fim, na terceira linha, ambas as políticas consideram os acordes iguais, porque todas as suas notas coincidem, incluindo-se as dissonâncias.

Uma vez escolhidas as políticas de comparação de acordes, foram definidas algumas métricas, visando obter informações quantitativas mais precisas sobre a qualidade das classificações realizadas, permitindo a comparação com uma base etiquetada e com outras abordagens de maneira objetiva. Todas as métricas foram calculadas considerando ambas as políticas de comparação de acordes, definidas anteriormente. As métricas definidas são detalhadas a seguir:

- **Número total de acordes do rótulo (A):** número de acordes contido no arquivo de rótulo;
- **Número total de acordes reconhecidos (R):** número de acordes reconhecidos pela abordagem sendo testada;
- **Precisão quantitativa (PQ):** é definida como o quociente entre o número de acordes do rótulo e o número de acordes reconhecidos (ou seja, PQ = R ÷ A); logo, numa situação ideal, onde o dividendo e o divisor têm o mesmo valor, o resultado é 1. Se o quociente for maior que 1, significa que foram reconhecidos mais acordes do que os de fato existentes (R > A → PQ > 1). De maneira oposta, se o quociente for menor que 1, significa que foram reconhecidos menos acordes do que os de fato existentes (R > A → PQ > 1). Esta métrica serve para que se tenha uma idéia da sensibilidade do algoritmo de particionamento;
- **Percentual de corretude (PC):** é definido como o percentual de tempo total da música em que o acorde reconhecido está correto. Ou seja, é o quociente, para cada música, entre a duração total dos trechos em que o acorde reconhecido está correto e a duração total da música, multiplicado por cem;
- **Percentual de falha (PF)**: analogamente, é definido pelo percentual de tempo total da música em que o acorde reconhecido está incorreto. Obviamente, somando-se os percentuais de corretude e falha, teremos sempre 100%;

- **Total de classificações corretas (O):** total de acordes corretamente identificados, considerando-se uma janela de 250 milésimos de segundo para mais ou para menos, do ataque correspondente na base rotulada;
- **Total de classificações incorretas (X):** total de acordes para os quais nenhum acorde correspondente foi identificado na base rotulada, dentro da janela considerada ou cujo acorde correspondente na base rotulada, dentro da janela considerada, difere do acorde reconhecido;
- **Total de acordes não detectados (M):** total de acordes da base rotulada para os quais nenhum acorde foi identificado dentro da janela considerada;
- **Total de classificações adiantadas (O←):** total de acordes identificados corretamente fora da janela considerada e imediatamente antes da execução do acorde real na base rotulada;
- **Total de classificações atrasadas (→O):** total de acordes identificados corretamente fora da janela considerada e imediatamente após a execução do acorde real na base rotulada;
- **Adiantamento médio ($O\leftarrow_M$):** adiantamento médio, em milésimos de segundo, das classificações adiantadas, ou seja, o quociente entre a soma dos adiantamentos, em milésimos de segundo, e a quantidade de adiantamentos existente. Nesta métrica as classificações cujo adiantamento é menor que a janela estabelecida não são computadas;
- **Atraso médio ($\rightarrow O_M$):** analogamente, é definido como o atraso médio, também em milésimos de segundo, das classificações atrasadas. Também não inclui as classificações cujo atraso é menor que a janela estabelecida;
- **Adiantamento total médio ($O\leftarrow_T$):** definido analogamente ao adiantamento médio, porém incluindo todos os acordes reconhecidos corretamente, mesmo quando o adiantamento é menor que a janela estabelecida. Evidentemente, tem valor menor ou igual ao adiantamento médio e serve como bom parâmetro para identificar a qualidade do particionamento;

- **Atraso total médio ($\rightarrow O_T$):** definido analogamente ao adiantamento total médio;
- **Adiantamento máximo ($O \leftarrow_{MAX}$):** maior diferença, em milésimos de segundo, entre os tempos de ataque dos acordes considerados como adiantamento e seus respectivos correspondentes na base rotulada;
- **Atraso máximo ($\rightarrow O_{MAX}$):** analogamente, é a maior diferença, também em milésimos de segundo, entre os tempos de ataque dos acordes do rótulo e seus respectivos correspondentes considerados como atrasos.

Para que se torne mais clara a compreensão dos conceitos de acordes não detectados, classificações corretas, incorretas, atrasadas e adiantadas, a figura 7.2 a seguir mostra com mais detalhes os casos considerados.

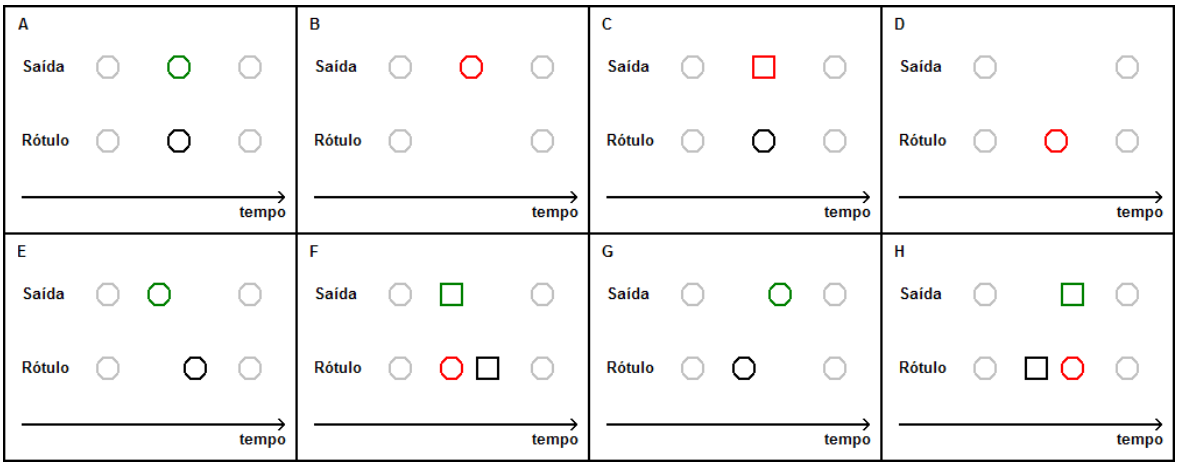


Figura 7.2 – Exemplos das métricas utilizadas na classificação de acordes.

Na figura 7.2, o quadro A é um exemplo de classificação correta, onde o acorde identificado é o mesmo acorde presente na base rotulada. Já os quadros B e C são casos de classificação incorreta: no primeiro caso, a identificação de um acorde inexistente na base rotulada, e no segundo caso, a má classificação de um acorde da base rotulada. Em D observa-se um caso de acorde não detectado, já que a base rotulada contém um acorde, mas a saída não o identificou. Percebem-se tanto em E como em F casos de adiantamento, onde o acorde foi corretamente identificado, porém fora do intervalo de tolerância e antes do ataque real do acorde na base rotulada. Analogamente, os casos G e H mostram atrasos, onde o acorde foi corretamente identificado, entretanto fora do intervalo de tolerância e após o

ataque real do acorde na base rotulada. Atrasos e adiantamentos necessariamente geram uma classificação incorreta, seja pela identificação de um acorde inexistente na base rotulada (casos em E e G) ou pela má classificação de um acorde da base rotulada (casos em F e H).

Há ainda duas métricas importantes que podem ser adquiridas a partir das métricas anteriormente citadas, que são:

- **Percentual de acerto dentro da janela ($PA_j$):** definido como o quociente entre número de acordes corretamente identificados dentro da janela temporal estabelecida e o número de acordes da base rotulada:

$$PA_j = O \div A$$

- **Percentual de acerto total ($PA_t$):** definido como o quociente entre o total de acordes identificados corretamente, incluindo os atrasos e adiantamentos, e o número de acordes da base rotulada:

$$PA_t = (O + O\leftarrow + \rightarrow O) \div A$$

### *7.2.3. Resultados dos Experimentos*

Os experimentos foram realizados considerando as combinações de técnicas descritas na tabela 7.2 a seguir, com ambas as políticas de comparação de acordes definidas na sub-sessão anterior.

Tabela 7.2 – Experimentos realizados em reconhecimento de acordes

| Experimento | Particionamento | Conjunto de Padrões | Grafo – Criação e Busca |
|---|---|---|---|
| **Experimento I** | **Birmingham & Pardo** | **Birmingham & Pardo** | **Birmingham & Pardo** |
| **Experimento II** | Birmingham & Pardo | COCHONUT | Birmingham & Pardo |
| **Experimento III** | COCHONUT | Birmingham & Pardo | Birmingham & Pardo |
| **Experimento IV** | COCHONUT | COCHONUT | Birmingham & Pardo |
| **Experimento V** | **COCHONUT** | **COCHONUT** | **COCHONUT** |

O experimento I foi realizado para servir como *benchmark* para os demais experimentos. Os experimentos II e III foram realizados visando mensurar separadamente a influência exata do conjunto de padrões de acordes e das modificações na técnica de particionamento. O experimento IV foi realizado a fim de medir o incremento de qualidade proporcionado pela técnica de

construção e busca no grafo utilizada no processo proposto. Por fim, o experimento V permitiu a comparação entre o processo proposto pelo COCHONUT e o algoritmo usado como *benchmark*, proposto por Birmingham e Pardo [30].

A seguir, serão mostradas as tabelas que detalham os resultados dos experimentos I e V. As tabelas com os resultados detalhados dos demais experimentos podem ser encontrados no Apêndice B desta dissertação.

## Experimento I – Abordagem original proposta por Birmingham & Pardo

### Comparação básica de acordes (tônica, terça e quinta)

Tabela 7.3 – Resultados médios do Experimento I, considerando a política de comparação básica entre acordes (tônica, terça e quinta)

| Experimento | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| Músicas I, II, III e IV | 64.25 | 132.25 | 2.3871 | 46.2344 | 53.6446 | 18.75 | 95.25 | 28.50 |
| **Experimento** | **O←** | **→O** | **$O←_M$** | **$→O_M$** | **$O←_T$** | **$→O_T$** | **$O←_{MAX}$** | **$→O_{MAX}$** |
| Músicas I, II, III e IV | 13.50 | 4.00 | 670.29 | 755.32 | 552.54 | 755.32 | 3555.00 | 1680.00 |

Tabela 7.4 – Resultados do Experimento I, considerando a política de comparação básica entre acordes (tônica, terça e quinta).

| Experimento | A | O | O← | →O | $PA_j = O \div A$ | $PA_t = (O + O← + →O) \div A$ |
|---|---|---|---|---|---|---|
| Música I | 36 | 8 | 16 | 4 | 22.22% | 77.77% |
| Música II | 33 | 13 | 6 | 5 | 39.39% | 72.72% |
| Música III | 63 | 12 | 12 | 2 | 19.04% | 41.26% |
| Música IV | 125 | 42 | 20 | 5 | 33.60% | 53.60% |
| **Total** | **257** | **75** | **54** | **16** | **29.18%** | **56.42%** |

### Comparação completa de acordes (todas as notas)

Tabela 7.5 – Resultados médios do Experimento I, considerando a política de comparação completa entre acordes (todas as notas)

| Músicas | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| Músicas I, II, III e IV | 67.50 | 208.5 | 3.7794 | 33.7084 | 66.2914 | 14.25 | 178.75 | 42.25 |
| **Músicas** | **O←** | **→O** | **$O←_M$** | **$→O_M$** | **$O←_T$** | **$→O_T$** | **$O←_{MAX}$** | **$→O_{MAX}$** |
| Músicas I, II, III e IV | 9.75 | 3.75 | 571.15 | 653.34 | 489.26 | 653.34 | 1255.00 | 3252.50 |

Tabela 7.6 – Resultados do Experimento I, considerando a política de comparação completa entre acordes (todas as notas).

| Experimento | A | O | O← | →O | $PA_j = O \div A$ | $PA_t = (O + O← + →O) \div A$ |
|---|---|---|---|---|---|---|
| Música I | 38 | 10 | 13 | 7 | 26.31% | 78.94% |
| Música II | 34 | 6 | 4 | 3 | 17.64% | 38.23% |
| Música III | 69 | 8 | 9 | 0 | 11.59% | 24.63% |
| Música IV | 129 | 33 | 13 | 5 | 25.58% | 39.53% |
| **Total** | **270** | **57** | **39** | **15** | **21.11%** | **41.11%** |

## Experimento V – SCHOLZ

### Comparação básica de acordes (tônica, terça e quinta)

Tabela 7.7 – Resultados médios do Experimento V, considerando a política de comparação básica entre acordes (tônica, terça e quinta)

| Experimento | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Músicas I, II, III e IV** | 64,25 | 69.25 | 1.1277 | 78.3240 | 21.6758 | 45.50 | 17.25 | 12.75 |
| **Experimento** | **O←** | **→O** | **$O←_M$** | **$→O_M$** | **$O←_T$** | **$→O_T$** | **$O←_{MAX}$** | **$→O_{MAX}$** |
| **Músicas I, II, III e IV** | 3.75 | 2.25 | 276.15 | 482.05 | 228.50 | 542.36 | 388.75 | 1017.50 |

Tabela 7.8 – Resultados do Experimento V, considerando a política de comparação básica entre acordes (tônica, terça e quinta).

| Experimento | A | O | O← | →O | $PA_j = O \div A$ | $PA_t = (O + O← + →O) \div A$ |
|---|---|---|---|---|---|---|
| **Música I** | 36 | 17 | 10 | 7 | 47.22% | 94.44% |
| **Música II** | 33 | 22 | 1 | 1 | 66.66% | 72.72% |
| **Música III** | 63 | 50 | 2 | 0 | 79.36% | 82.53% |
| **Música IV** | 125 | 93 | 2 | 1 | 74.40% | 76.80% |
| **Total** | **257** | **182** | **15** | **9** | **70.81%** | **80.15%** |

### Comparação completa de acordes (todas as notas)

Tabela 7.9 – Resultados médios do Experimento V, considerando a política de comparação completa entre acordes (todas as notas)

| Experimento | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Músicas I, II, III e IV** | 67.50 | 72.75 | 1.1454 | 74.7762 | 25.2237 | 44 | 21.50 | 16.75 |
| **Experimento** | **O←** | **→O** | **$O←_M$** | **$→O_M$** | **$O←_T$** | **$→O_T$** | **$O←_{MAX}$** | **$→O_{MAX}$** |
| **Músicas I, II, III e IV** | 4.25 | 2.50 | 293.34 | 490.11 | 243.97 | 550.42 | 397.50 | 1017.50 |

Tabela 7.10 – Resultados do Experimento V, considerando a política de comparação completa entre acordes (todas as notas).

| Experimento | A | O | O← | →O | $PA_j = O \div A$ | $PA_t = (O + O← + →O) \div A$ |
|---|---|---|---|---|---|---|
| **Música I** | 38 | 18 | 10 | 8 | 47.36% | 94.73% |
| **Música II** | 34 | 23 | 1 | 1 | 67.64% | 73.52% |
| **Música III** | 69 | 49 | 3 | 0 | 71.01% | 75.36% |
| **Música IV** | 129 | 86 | 3 | 1 | 66.66% | 69.76% |
| **Total** | **270** | **175** | **17** | **9** | **64.81%** | **74.44%** |

## Resumo dos Experimentos

Tabela 7.11 – Resumo dos experimentos realizados

| Experimento | Particionamento | Conjunto de Padrões | Grafo – Criação e Busca |
|---|---|---|---|
| **Experimento I** | **Birmingham & Pardo** | **Birmingham & Pardo** | **Birmingham & Pardo** |
| **Experimento II** | Birmingham & Pardo | COCHONUT | Birmingham & Pardo |
| **Experimento III** | COCHONUT | Birmingham & Pardo | Birmingham & Pardo |
| **Experimento IV** | COCHONUT | COCHONUT | Birmingham & Pardo |
| **Experimento V** | **COCHONUT** | **COCHONUT** | **COCHONUT** |

## Resumo dos Resultados

### Comparação básica de acordes (tônica, terça e quinta)

Tabela 7.12 – Resultados do Experimento I, considerando a política de comparação básica entre acordes (tônica, terça e quinta).

| Experimento | A | O | O← | →O | $PA_j = O \div A$ | $PA_t = (O + O\leftarrow + \rightarrow O) \div A$ |
|---|---|---|---|---|---|---|
| **Experimento I** | **257** | **75** | **54** | **16** | **29.18%** | **56.42%** |
| **Experimento II** | 257 | 62 | 65 | 17 | 24.12% | 56.03% |
| **Experimento III** | 257 | 107 | 21 | 8 | 41.63% | 52.91% |
| **Experimento IV** | 257 | 138 | 19 | 18 | 53.69% | 68.09% |
| **Experimento V** | **257** | **182** | **15** | **9** | **70.81%** | **80.15%** |

### Comparação completa de acordes (todas as notas)

Tabela 7.13 – Resultados do Experimento I, considerando a política de comparação completa entre acordes (todas as notas).

| Experimento | A | O | O← | →O | $PA_j = O \div A$ | $PA_t = (O + O\leftarrow + \rightarrow O) \div A$ |
|---|---|---|---|---|---|---|
| **Experimento I** | **270** | **57** | **39** | **15** | **21.11%** | **41.11%** |
| **Experimento II** | 270 | 60 | 57 | 19 | 22.22% | 50.37% |
| **Experimento III** | 270 | 95 | 21 | 5 | 35.18% | 44.81% |
| **Experimento IV** | 270 | 132 | 20 | 16 | 48.88% | 62.22% |
| **Experimento V** | **270** | **175** | **17** | **9** | **64.81%** | **74.44%** |

### *7.2.4. Análise dos Resultados*

Apesar do espaço amostral reduzido, o que em parte é devido ao enorme esforço necessário para rotular manualmente as bases, pode-se considerar suficiente, dentro do escopo desta dissertação, a quantidade e diversidade dos acordes existente nos dados rotulados.

Observando os resultados obtidos pelo COCHONUT, nas tabelas 7.8 e 7.10, percebe-se que, embora a política de comparação de acordes básica tenha chegado a melhores resultados (80.15%), a diferença para os resultados atingidos considerando-se a política de comparação completa de acordes (74.44%) foi muito pequena, provando que na maior parte dos casos o reconhecimento dos acordes inclui as dissonâncias corretamente. De maneira contrária, a diferença entre os resultados obtidos pela abordagem proposta por Birmingham & Pardo, nas tabelas 7.4 e 7.6 (56.42% e 41.11%, respectivamente), com respeito às duas políticas de comparação de acordes, é bem maior. Além disso, a diferença para os resultados do COCHONUT chega a mais de 30% considerando a política de comparação completa de acordes. Isto fornece indícios claros da inadequação da abordagem às exigências de completude harmônica envolvidas no problema, bem como às particularidades dos dados capturados por violões MIDI.

Percebe-se ainda, nas tabelas 7.12 e 7.13, que a utilização da técnica de particionamento, do conjunto de padrões de acordes e da técnica de criação e busca no grafo, separadamente, não chegam a causar incrementos significativos na performance final.

Considerando o algoritmo de particionamento, percebe-se uma melhora bastante significativa, comparando-se ao algoritmo utilizado por Birmingham e Pardo. Tal melhora pode ser facilmente percebida pela precisão quantitativa média, que na abordagem de Birmingham e Pardo tem valor 3.7794 (tabela 7.5), enquanto no COCHONUT fica muito próxima de 1, apresentando o valor 1.1454 (tabela 7.9). Juntando-se a isso os valores de detecções corretas dentro da janela de tempo estabelecida (64.81% do COCHONUT, contra 21.11% da abordagem de Birmingham e Pardo, de acordo com a tabela 7.13), pode-se concluir que o processo proposto chega valores razoáveis de **completude temporal**.

Percebe-se que os atrasos normalmente apresentam discrepâncias temporais maiores com os dados rotulados (em média, próximas a meio segundo, mas chegando a pouco mais de um segundo no pior caso), mas acontecem com menor freqüência; enquanto que os adiantamentos acontecem em maior freqüência, porém apresentam discrepâncias consideravelmente menores (com média não ultrapassando meio segundo), conforme os dados das tabelas 7.7 e 7.9. Sabendo-se que, pela precisão quantitativa média encontrada nestas tabelas, a quantidade de pontos de partição reconhecida é quase sempre muito próxima da quantidade real de tais pontos, e considerando a baixa quantidade dos atrasos e adiantamentos (tabelas 7.7 e 7.9, colunas O← e →O), além da baixa discrepância temporal média dos mesmos (tabelas 7.7 e 7.9, colunas $O\leftarrow_M$, $\rightarrow O_M$, $O\leftarrow_T$ e $\rightarrow O_T$) e os altos percentuais de corretude (tabelas 7.7 e 7.9, coluna PC), pode-se considerar que os resultados foram razoáveis com respeito à **corretude temporal**.

A **corretude harmônica** pode ser melhor verificada pelos percentuais de corretude e falha, e os percentuais de acerto dentro da janela e totais. O percentual médio de corretude, quando considerando-se a política básica de comparação de acordes, chega a 78.32% (tabela 7.7, coluna PC), atingindo 85.33% no melhor caso (apêndice B, tabela B.26, música 1, coluna PC). A quantidade média de acordes corretamente classificados ($PA_j$) é maior que 70%, e quando consideram-se os atrasos e adiantamentos ($PA_t$), ultrapassa 80% (tabela 7.8). Comparando-se estes resultados aos demais experimentos (tabelas 7.12), percebe-se um incremento bastante satisfatório nos resultados.

Embora os resultados sejam afetados negativamente quando a política de comparação completa de acordes é utilizada, não há diferença significativa nos resultados, já que o percentual médio de corretude (tabela 7.7, coluna PC), cai apenas 3.54%, para 74.77%, atingindo ainda 84.20% no melhor caso (apêndice B, tabela B.29, música 1, coluna PC). Além disso, a quantidade média de acordes corretamente classificados cai apenas 6%, para 64.81%, quando não são considerados os atrasos e adiantamentos, chegando a 74.44% quando os mesmos são considerados, com uma queda de apenas 5.71% (tabelas 7.8 e 7.10). Em comparação com os outros experimentos, percebe-se clara melhoria na performance, já que a abordagem inicial de

Birmingham e Pardo apresenta média de acordes corretamente classificados dentro da janela de apenas 21.11%, chegando a 41.11% quando consideram-se os atrasos e adiantamentos. Assim, pode-se considerar que a solução chega a bons resultados mesmo quando os requisitos de **completude harmônica** são mais rigorosos.

Os resultados obtidos dão claras indicações que o uso de uma abordagem híbrida, utilizando casamento de padrões de acordes para resolver os casos mais óbvios, e fazendo uso de informações contextuais harmônicas para identificação dos acordes mais ambíguos, aproveitando os acordes identificados com mais alto grau de certeza, pode chegar a resultados bastante razoáveis nos problemas de reconhecimento de harmonia, particularmente quando tratando-se de dados simbólicos, como seqüências MIDI.

## 7.3. Precisão Temporal em Seqüências Capturadas por Violões MIDI

A abordagem de sincronização dos arquivos de ataque baseada no primeiro ataque invalida o experimento quando o primeiro ataque é um falso positivo ou falso negativo, porque não consegue sincronizar devidamente o restante dos ataques. Embora tal comportamento já fosse esperado desde a definição da política de sincronização dos arquivos de ataques, optou-se por manter tal política e descartar os experimentos que apresentassem este tipo de problema, pela facilidade de implementação e pela inexistência da necessidade de definir políticas mais complexas. Assim, foram descartados os experimentos 6, 8 e 11, todos por existência de falsos positivos ou falsos negativos no início do arquivo de *log* de ataques gerado pelo *BeatRoot*, com exceção do experimento 11, que apresentou falsos positivos no início de ambas as entradas.

A tabela 7.14 mostra a comparação entre o MIDI capturado e a base rotulada, enquanto que a tabela 7.15 mostra a comparação entre o algoritmo de detecção de ataques *BeatRoot* [12] e a base rotulada. Por fim, a tabela 7.16 mostra os resultados médios de cada abordagem.

Tabela 7.14 – Resultados: Midi vs Base Rotulada.

| Experimento | Duração Aprox. (seg) | Positivos Verdadeiros | Falsos Positivos | Falsos Negativos | Distância Máxima (ms) | Distância Média (ms) | Precisão | Recall | Medida-F (*f-measure*) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 8 | 18 | 5 | 5 | 13.38 | 5.01 | 0.7826 | 0.7826 | 0.7826 |
| 2 | 8 | 16 | 5 | 4 | 6.96 | 4.06 | 0.7619 | 0.8000 | 0.7804 |
| 3 | 11 | 43 | 9 | 3 | 13.60 | 5.74 | 0.8269 | 0.9347 | 0.8775 |
| 4 | 10 | 27 | 2 | 1 | 7.37 | 3.52 | 0.9310 | 0.9642 | 0.9473 |
| 5 | 12 | 26 | 6 | 3 | 14.82 | 6.76 | 0.8125 | 0.8965 | 0.8524 |
| 6 | ***EXPERIMENTO DESCARTADO*** | | | | | | | | |
| 7 | 10 | 24 | 10 | 12 | 14.83 | 6.07 | 0.7058 | 0.6666 | 0.6857 |
| 8 | ***EXPERIMENTO DESCARTADO*** | | | | | | | | |
| 9 | 12 | 33 | 7 | 5 | 13.19 | 4.49 | 0.8250 | 0.8684 | 0.8461 |
| 10 | 10 | 31 | 3 | 2 | 13.19 | 4.51 | 0.9117 | 0.9393 | 0.9253 |
| 11 | ***EXPERIMENTO DESCARTADO*** | | | | | | | | |
| 12 | 10 | 23 | 6 | 4 | 13.43 | 6.27 | 0.7931 | 0.8518 | 0.8214 |

Tabela 7.15– Resultados: *BeatRoot* vs Base Rotulada.

| Experimento | Duração Aprox. (seg) | Positivos Verdadeiros | Falsos Positivos | Falsos Negativos | Distância Máxima (ms) | Distância Média (ms) | Precisão | Recall | Medida-F (*f-measure*) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 8 | 19 | 11 | 4 | 14.02 | 7.39 | 0.6333 | 0.8260 | 0.7169 |
| 2 | 8 | 19 | 7 | 1 | 10.11 | 4.40 | 0.7307 | 0.9500 | 0.8260 |
| 3 | 11 | 38 | 6 | 8 | 14.60 | 4.89 | 0.8636 | 0.8260 | 0.8444 |
| 4 | 10 | 24 | 4 | 3 | 11.90 | 4.48 | 0.8620 | 0.8928 | 0.8771 |
| 5 | 12 | 27 | 10 | 2 | 14.85 | 3.98 | 0.7297 | 0.9310 | 0.8181 |
| 6 | ***EXPERIMENTO DESCARTADO*** | | | | | | | | |
| 7 | 10 | 28 | 11 | 8 | 12.68 | 6.03 | 0.7179 | 0.7777 | 0.7466 |
| 8 | ***EXPERIMENTO DESCARTADO*** | | | | | | | | |
| 9 | 12 | 29 | 19 | 9 | 12.51 | 4.71 | 0.6041 | 0.7631 | 0.6744 |
| 10 | 10 | 23 | 4 | 10 | 13.15 | 4.54 | 0.8518 | 0.6969 | 0.7666 |
| 11 | ***EXPERIMENTO DESCARTADO*** | | | | | | | | |
| 12 | 10 | 23 | 10 | 4 | 13.77 | 5.20 | 0.6969 | 0.8518 | 0.7666 |

Tabela 7.16 – Médias dos resultados de cada técnica.

| Técnica | Duração Aprox. (seg) | Positivos Verdadeiros | Falsos Positivos | Falsos Negativos | Distância Máxima (ms) | Distância Média (ms) | Precisão | Recall | Medida-F (*f-measure*) |
|---|---|---|---|---|---|---|---|---|---|
| MIDI | 10.11 | 26.77 | 5.88 | 4.33 | 12.30 | 5.15 | 0.8167 | 0.8560 | 0.8354 |
| BeatRoot | 10.11 | 25.55 | 9.11 | 5.44 | 13.06 | 5.06 | 0.7433 | 0.8350 | 0.7818 |

Percebe-se que a detecção de ataques a partir do MIDI leva vantagem na grande maioria dos experimentos, principalmente nos experimentos 9 e 10, onde, embora as distâncias máxima e média não apresentem grandes

diferenças de precisão entre as duas abordagens, a abordagem MIDI leva larga vantagem em precisão, *recall* e medida-f. O experimento 4 também é um bom exemplo da melhor performance da abordagem MIDI, pelos melhores valores de precisão, *recall* e medida-f, embora apresente uma distância média ligeiramente inferior ao *BeatRoot*. Além disso, de maneira geral, a abordagem MIDI apresenta resultados melhores que o *BeatRoot* nos experimentos 1, 3, 5 e 12.

É possível perceber também que o *BeatRoot* obteve melhores resultados nos experimentos 2 e 7. Entretanto, nestes casos, embora a diferença na distância máxima seja razoável, em torno de 3 milésimos de segundo, não há diferença significativa na distância média. Além disso, a disparidade entre a precisão do *BeatRoot* e do MIDI, nestes casos, é muito pequena; e por fim, nota-se o *BeatRoot* leva uma boa vantagem apenas no *recall*, enquanto a medida-f apresenta uma vantagem não mais que razoável.

Sendo assim, pode-se concluir que a detecção de ataques a partir das seqüências MIDI tem resultados, em média, melhores ou tão confiáveis quanto o algoritmo de detecção de ataques escolhido, sendo suficiente para garantir a precisão temporal necessária nas tarefas de recuperação de informação simbólica, como particularmente, no caso desta dissertação, o reconhecimento de acordes.

# 8. Conclusões e Trabalhos Futuros

O reconhecimento de harmonia é uma tarefa difícil, devido à ambigüidade e forte dependência contextual inerentes ao problema. Além disso, quanto maior a completude harmônica considerada, maior a existência de casos ambíguos, aumentando assim a dificuldade de se chegar a bons resultados.

Os dados capturados a partir de violões MIDI oferecem desafios particulares, como a enorme quantidade de ruídos inerentes à tecnologia atual de captura MIDI em violões, e o fato dos dados capturados não serem quantizados.

Como pré-requisito para o reconhecimento de acordes sobre seqüências adquiridas a partir de violões MIDI, neste trabalho realizou-se um experimento visando garantir a precisão temporal de tal processo de captação, e chegou-se à conclusão que a tecnologia atual já provê precisão temporal comparável à precisão obtida pela execução de algoritmos de detecção de ataques sobre sinais de áudio.

Além disso, dado que a correção manual dos ruídos adquiridos durante a captação é uma tarefa morosa que requer grande esforço e tempo, uma etapa de pré-processamento foi proposta visando corrigir os ruídos existentes. Os resultados chegaram a corrigir cerca de metade dos ruídos, sem produzir falso-positivos. Considerando que cada minuto de seqüência capturado pelo violão MIDI leva cerca de 8 horas para ter seus ruídos manualmente corrigidos por um especialista [24], pode-se considerar que os resultados foram bons. O requisito de inexistência de falso-positivos torna-se o grande desafio para a melhoria dos resultados atingidos.

O processo proposto chegou a resultados bastante satisfatórios em corretude e completude temporal, já que conseguiu identificar com razoável precisão a maioria dos pontos de transição de acordes. Além disso, a corretude harmônica, ou seja, a quantidade de acordes corretamente identificados, e a completude harmônica, medida pela correta e completa identificação das dissonâncias nos acordes identificados, chegaram a resultados igualmente satisfatórios.

Embora ainda seja possível melhorar tanto o particionamento quanto a classificação, os resultados obtidos dão claros indícios que uma abordagem híbrida, utilizando padrões de acordes completos, porém ambíguos, e

informações sobre a estrutura harmônica para resolver as ambigüidades, podem chegar a bons resultados.

Pode-se enumerar como principais contribuições do processo proposto a abordagem utilizando casamento de padrões harmônicos recorrentes para achar o melhor caminho nos grafos gerados e os requisitos de completude harmônica considerados serem mais exigentes que os considerados na literatura atual. Além disso, realizou-se um experimento inédito para garantir a precisão temporal da captura realizada por violões MIDI, o que acredita-se ser uma contribuição também importante para futuras pesquisas considerando tais dados, não só em reconhecimento de acordes, mas em outras áreas, como estudo de micro-variações temporais, ou recuperação de outras informações musicais.

Como sugestão de trabalho futuro, pode-se considerar a inclusão de regras contextuais mais complexas, que utilizem informações como campo harmônico ou considerem empréstimos modais. Tais modificações podem permitir a utilização de mais padrões de acordes na primeira etapa, aumentando a completude harmônica, por conseguirem lidar com o aumento de ambigüidade inerente ao uso de mais padrões.

A utilização de informações sobre a estrutura musical, através da identificação de seqüências de acordes recorrentes dentro de uma mesma peça, antes da execução do motor de inferência contendo as regras contextuais, pode também ajudar a melhorar os resultados obtidos.

Além disso, a modificação da busca do melhor caminho no grafo, através de uma modelagem matemática adequada e uso de técnicas de otimização [10] ao invés do casamento de padrões por ordem de prioridade pode ser uma boa alternativa.

Seria interessante também realizar uma adaptação da abordagem para o reconhecimento de acordes em sinais de áudio. Embora a identificação dos pontos de partição possa ser considerada uma das partes mais difíceis de adaptar na abordagem proposta, as demais etapas podem ser adaptadas comparando os *Pitch Class Profiles* de cada segmento com os padrões definidos, e gerando um grafo contendo os acordes mais prováveis, no qual a mesma técnica de busca através da execução de regras contextuais pode ser aplicada. No caso particular do projeto “Um País, Um Violão”, onde o áudio é gravado simultaneamente à captura MIDI, pode-se ainda utilizar as informações de

particionamento adquiridas das seqüências MIDI para particionar o sinal de áudio, e combinar a análise dos segmentos de áudio e dos segmentos MIDI para melhorar os resultados finais.

# Apêndice A - Convenções sobre Notação de Acordes

A convenção para notação de acordes utilizada neste trabalho segue o utilizado nos *Songbooks* de Almir Chediak [7] , referência indiscutível no meio musical no que diz respeito às transcrições de música brasileira, em particular a bossa-nova, devido à sua larga utilização no meio musical, principalmente na música popular de harmonia jazzística. A Tabela A.1 mostra as convenções sobre a notação dos tipos de acordes: cada acorde deve exibir uma, e somente uma, das notações contidas nesta tabela. Já a Tabela A.2 mostra as convenções sobre a notação das alterações; neste caso, um acorde pode conter mais de uma alteração, embora no máximo uma alteração de cada grupo.

Tabela A.1 – Convenção de notação sobre tipos de acorde

| Notação (em C) | Tipo do Acorde | Intervalos que Compõem o Acorde | Notas do Acorde (em C) |
|---|---|---|---|
| C | Maior | T – 3ªM – 5ªJ | C-E-G |
| Cm | Menor | T – 3ªm – 5ªJ | C-Eb-G |
| $C^{o}$ ou $C^{dim}$ | Diminuto | T – 3ªm – 5ªb – 7ªb | C-Eb-Gb-Bbb |
| $C^{ø}$ ou $Cm^{7(b5)}$ | Meio Diminuto | T – 3ªm – 5ªb – 7ªm | C-Eb-Gb-Bb |
| $C^{(\#5)}$ | Aumentado | T – 3ªM – 5ª# | C-E-G# |
| $C^{sus4}$ | Suspenso | T – 4ªJ – 5ªJ | C-F-G |

Tabela A.2 – Convenção de notação sobre alterações nos acordes

| Notação | Dissonância (Intervalo para a Tônica) | Exemplo de Nota Adicionada em C |
|---|---|---|
| b5 | Quinta diminuta | Gb |
| #5 | Quinta aumentada | G# |
| 7 | Sétima menor | Bb |
| 7M | Sétima maior | B |
| 9 | Nona maior | D |
| b9 | Nona bemol | Db |
| #9 | Nona aumentada | D# |
| 11 | Décima primeira justa | F |
| #11 | Décima primeira aumentada | F# |
| 13 | Décima terceira menor | A |
| b13 | Décima terceira diminuta | Ab |

# Apêndice B – Resultados Detalhados dos Experimentos em Reconhecimento de Acordes

Os resultados detalhados dos experimentos realizados em reconhecimento de harmonia, descritos na sessão 7.1 são exibidos a seguir. A opção por inseri-los num apêndice se deu pela grande quantidade de tabelas necessárias, facilitando assim a leitura e identificação das informações mais importantes no texto da dissertação, e deixando a consulta a este apêndice a critério do leitor, caso deseje informações mais detalhadas sobre as métricas calculadas.

A tabela 7.2, que descreve os experimentos realizados, é repetida a seguir (tabela B.1) visando facilitar a leitura e evitar que o leitor necessite voltar a tal capítulo para obter as informações.

Tabela B.1 – Experimentos realizados

| Experimento | Particionamento | Conjunto de Padrões | Grafo – Criação e Busca |
|---|---|---|---|
| **Experimento I** | **Birmingham & Pardo** | **Birmingham & Pardo** | **Birmingham & Pardo** |
| **Experimento II** | Birmingham & Pardo | COCHONUT | Birmingham & Pardo |
| **Experimento III** | COCHONUT | Birmingham & Pardo | Birmingham & Pardo |
| **Experimento IV** | COCHONUT | COCHONUT | Birmingham & Pardo |
| **Experimento V** | **COCHONUT** | **COCHONUT** | **COCHONUT** |

A seguir, os resultados dos experimentos I a V, considerando ambas as políticas para comparação de acordes (básica e completa) serão expostos.

## Experimento I

## [Particionamento: B&P – Padrões: B&P – Grafo: B&P]

### Comparação básica de acordes (tônica, terça e quinta)

Tabela B.2 – Resultados do Experimento I, considerando a comparação básica de acordes.

| Música | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Música I** | 36 | 127 | 3.5277 | 56.3428 | 43.6571 | 8 | 98 | 9 |
| **Música II** | 33 | 94 | 2.8484 | 52.8858 | 47.1141 | 13 | 69 | 10 |
| **Música III** | 63 | 90 | 1.4285 | 33.5445 | 66.4554 | 12 | 64 | 37 |
| **Música IV** | 125 | 218 | 1.7440 | 42.1647 | 57.8352 | 42 | 150 | 58 |
| **Música** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| **Música I** | 16 | 4 | 951.40 | 700.31 | 863.19 | 700.31 | 3555.00 | 1003.75 |
| **Música II** | 6 | 5 | 582.08 | 783.00 | 404.20 | 783.00 | 948.75 | 1343.75 |
| **Música III** | 12 | 2 | 592.70 | 998.75 | 470.00 | 998.75 | 895.00 | 1680.00 |
| **Música IV** | 20 | 5 | 555.00 | 539.25 | 472.80 | 539.25 | 933.75 | 746.25 |

Tabela B.3 – Valores médios das métricas calculadas na tabela B.2, com exceção de $O\leftarrow_{MAX}$ e $\rightarrow O_{MAX}$, que exibem os valores máximos das respectivas colunas da tabela B.2.

| Músicas | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Médias de I, II, III e IV** | 64.25 | 132.25 | 2.3871 | 46.2344 | 53.6446 | 18.75 | 95.25 | 28.50 |
| **Músicas** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| **Médias de I, II, III e IV** | 13.50 | 4.00 | 670.29 | 755.32 | 552.54 | 755.32 | 3555.00 | 1680.00 |

Tabela B.4 – Resultados do Experimento I, considerando a comparação básica de acordes.

| Experimento | A | O | O← | →O | $PA_j = O \div A$ | $PA_t = (O + O\leftarrow + \rightarrow O) \div A$ |
|---|---|---|---|---|---|---|
| **Música I** | 36 | 8 | 16 | 4 | 22.22% | 77.77% |
| **Música II** | 33 | 13 | 6 | 5 | 39.39% | 72.72% |
| **Música III** | 63 | 12 | 12 | 2 | 19.04% | 41.26% |
| **Música IV** | 125 | 42 | 20 | 5 | 33.60% | 53.60% |
| **Total** | **257** | **75** | **54** | **16** | **29.18%** | **56.42%** |

## Experimento I

## [Particionamento: B&P – Padrões: B&P – Grafo: B&P]

### Comparação completa de acordes (todas as notas)

Tabela B.5 – Resultados do Experimento I, considerando a comparação completa de acordes.

| Experimento | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Música I** | 38 | 236 | 6.2105 | 51.9570 | 48.0429 | 10 | 204 | 14 |
| **Música II** | 34 | 158 | 4.6470 | 31.3008 | 68.6991 | 6 | 143 | 23 |
| **Música III** | 69 | 126 | 1.8260 | 17.5451 | 82.4548 | 8 | 107 | 53 |
| **Música IV** | 129 | 314 | 2.4341 | 34.0308 | 65.9691 | 33 | 261 | 79 |
| **Experimento** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| **Música I** | 13 | 7 | 573.17 | 1333.03 | 464.77 | 1333.03 | 1255.00 | 3252.50 |
| **Música II** | 4 | 3 | 572.50 | 740.83 | 454.79 | 740.83 | 626.25 | 1343.75 |
| **Música III** | 9 | 0 | 548.75 | 0.00 | 447.29 | 0.00 | 895.00 | 0.00 |
| **Música IV** | 13 | 5 | 590.19 | 539.50 | 590.19 | 539.50 | 933.75 | 746.25 |

Tabela B.6 – Valores médios das métricas calculadas na tabela B.5, com exceção de $O\leftarrow_{MAX}$ e $\rightarrow O_{MAX}$, que exibem os valores máximos das respectivas colunas da tabela B.5.

| Experimento | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Músicas I, II, III e IV** | 67.50 | 208.5 | 3.7794 | 33.7084 | 66.2914 | 14.25 | 178.75 | 42.25 |
| **Experimento** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| **Músicas I, II, III e IV** | 9.75 | 3.75 | 571.15 | 653.34 | 489.26 | 653.34 | 1255.00 | 3252.50 |

Tabela B.7 – Resultados do Experimento I, considerando a comparação completa de acordes.

| Experimento | A | O | O← | →O | $PA_j = O \div A$ | $PA_t = (O + O\leftarrow + \rightarrow O) \div A$ |
|---|---|---|---|---|---|---|
| **Música I** | 38 | 10 | 13 | 7 | 26.31% | 78.94% |
| **Música II** | 34 | 6 | 4 | 3 | 17.64% | 38.23% |
| **Música III** | 69 | 8 | 9 | 0 | 11.59% | 24.63% |
| **Música IV** | 129 | 33 | 13 | 5 | 25.58% | 39.53% |
| **Total** | **270** | **57** | **39** | **15** | **21.11%** | **41.11%** |

## Experimento II

## [Particionamento: B&P – Padrões: COCHONUT – Grafo: B&P]

### Comparação básica de acordes (tônica, terça e quinta)

Tabela B.8 – Resultados do Experimento II, considerando a comparação básica de acordes.

| Música | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Música I** | 36 | 127 | 3.5277 | 57.5309 | 42.4690 | 7 | 98 | 9 |
| **Música II** | 33 | 83 | 2.5151 | 58.1037 | 41.8962 | 10 | 57 | 7 |
| **Música III** | 63 | 85 | 1.3492 | 29.0269 | 70.9730 | 11 | 62 | 42 |
| **Música IV** | 125 | 223 | 1.7840 | 47.6251 | 52.3748 | 34 | 153 | 58 |
| **Música** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| **Música I** | 18 | 3 | 784.37 | 533.75 | 661.53 | 533.75 | 3420.00 | 642.50 |
| **Música II** | 8 | 8 | 903.59 | 1137.65 | 662.81 | 1137.65 | 2005.00 | 5205.00 |
| **Música III** | 10 | 1 | 881.37 | 1680.00 | 690.71 | 1680.00 | 1683.75 | 1690.00 |
| **Música IV** | 29 | 5 | 562.75 | 403.75 | 485.75 | 403.75 | 870.00 | 533.75 |

Tabela B.9 – Valores médios das métricas calculadas na tabela B.8, com exceção de $O\leftarrow_{MAX}$ e $\rightarrow O_{MAX}$, que exibem os valores máximos das respectivas colunas da tabela B.8.

| Músicas | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Médias de I, II, III e IV** | 64.25 | 129.50 | 2.2940 | 48.0716 | 51.9282 | 15.50 | 92.50 | 29 |
| **Músicas** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| **Médias de I, II, III e IV** | 16.25 | 4.25 | 783.02 | 938.78 | 625.20 | 938.78 | 3420.00 | 5205.00 |

Tabela B.10 – Resultados do Experimento II, considerando a comparação básica de acordes.

| Experimento | A | O | O← | →O | $PA_j = O \div A$ | $PA_t = (O + O\leftarrow + \rightarrow O) \div A$ |
|---|---|---|---|---|---|---|
| **Música I** | 36 | 7 | 18 | 3 | 19.44% | 77.77% |
| **Música II** | 33 | 10 | 8 | 8 | 30.30% | 78.78% |
| **Música III** | 63 | 11 | 10 | 1 | 17.46% | 34.92% |
| **Música IV** | 125 | 34 | 29 | 5 | 27.20% | 54.40% |
| **Total** | **257** | **62** | **65** | **17** | **24.12%** | **56.03%** |

## Experimento II

## [Particionamento: B&P – Padrões: COCHONUT – Grafo: B&P]

### Comparação completa de acordes (todas as notas)

Tabela B.11 – Resultados do Experimento II, considerando a comparação completa de acordes.

| Experimento | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Música I** | 38 | 236 | 6.2105 | 53.5588 | 46.4411 | 9 | 201 | 11 |
| **Música II** | 34 | 155 | 4.5588 | 47.0199 | 52.9800 | 8 | 133 | 15 |
| **Música III** | 69 | 125 | 1.8115 | 21.5114 | 78.4885 | 11 | 103 | 50 |
| **Música IV** | 129 | 313 | 2.4263 | 44.3869 | 55.6130 | 32 | 249 | 68 |
| **Experimento** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| **Música I** | 16 | 8 | 521.32 | 1212.34 | 421.84 | 1212.34 | 1255.00 | 3252.50 |
| **Música II** | 9 | 4 | 792.91 | 1998.12 | 651.35 | 1998.12 | 1681.25 | 5205.00 |
| **Música III** | 9 | 0 | 792.22 | 0.00 | 614.32 | 0.00 | 1080.00 | 0.00 |
| **Música IV** | 23 | 7 | 536.79 | 415.17 | 474.72 | 415.17 | 828.75 | 562.50 |

Tabela B.12 – Valores médios das métricas calculadas na tabela B.11, com exceção de $O\leftarrow_{MAX}$ e $\rightarrow O_{MAX}$, que exibem os valores máximos das respectivas colunas da tabela B.11.

| Experimento | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Músicas I, II, III e IV** | 67.50 | 207.25 | 3.7517 | 41.6192 | 58.3806 | 15 | 171.50 | 36 |
| **Experimento** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| **Músicas I, II, III e IV** | 14.25 | 4.75 | 660.81 | 906.40 | 540.55 | 906.40 | 1681.25 | 5205.00 |

Tabela B.13 – Resultados do Experimento II, considerando a comparação completa de acordes.

| Experimento | A | O | O← | →O | $PA_j = O \div A$ | $PA_t = (O + O\leftarrow + \rightarrow O) \div A$ |
|---|---|---|---|---|---|---|
| **Música I** | 38 | 9 | 16 | 8 | 23.68% | 86.84% |
| **Música II** | 34 | 8 | 9 | 4 | 23.52% | 61.76% |
| **Música III** | 69 | 11 | 9 | 0 | 15.94% | 28.98% |
| **Música IV** | 129 | 32 | 23 | 7 | 24.80% | 48.06% |
| **Total** | **270** | **60** | **57** | **19** | **22.22%** | **50.37%** |

## Experimento III

## [Particionamento: COCHONUT – Padrões: B&P – Grafo: B&P]

### Comparação básica de acordes (tônica, terça e quinta)

Tabela B.14 – Resultados do Experimento III, considerando a comparação básica de acordes.

| Música | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| Música I | 36 | 37 | 1.0277 | 74.4042 | 25.5957 | 11 | 9 | 8 |
| Música II | 33 | 28 | 0.8484 | 52.4485 | 47.5514 | 13 | 13 | 15 |
| Música III | 63 | 50 | 0.7936 | 47.2201 | 52.7798 | 21 | 24 | 35 |
| Música IV | 125 | 97 | 0.7760 | 58.2757 | 41.7242 | 62 | 32 | 61 |
| **Música** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| Música I | 11 | 6 | 604.20 | 900.41 | 477.08 | 900.41 | 1045.00 | 1223.75 |
| Música II | 4 | 0 | 1430.0 | 0.00 | 908.92 | 0.00 | 3428.75 | 0.00 |
| Música III | 5 | 1 | 546.75 | 1515.00 | 391.12 | 1515.00 | 1090.00 | 1515.00 |
| Música IV | 1 | 1 | 253.75 | 911.25 | 172.91 | 911.25 | 253.75 | 911.25 |

Tabela B.15 – Valores médios das métricas calculadas na tabela B.14, com exceção de $O\leftarrow_{MAX}$ e $\rightarrow O_{MAX}$, que exibem os valores máximos das respectivas colunas da tabela B.14.

| Músicas | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| Médias de I, II, III e IV | 64.25 | 53 | 0.8614 | 58.0871 | 41.9127 | 26.75 | 19.50 | 29.75 |
| **Músicas** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| Médias de I, II, III e IV | 5.25 | 2.00 | 708.67 | 831.66 | 487.50 | 831.66 | 3428.75 | 1515.00 |

Tabela B.16 – Resultados do Experimento III, considerando a comparação básica de acordes.

| Experimento | A | O | O← | →O | $PA_j = O \div A$ | $PA_t = (O + O\leftarrow + \rightarrow O) \div A$ |
|---|---|---|---|---|---|---|
| Música I | 36 | 11 | 11 | 6 | 30.55% | 77.77% |
| Música II | 33 | 13 | 4 | 0 | 39.39% | 51.51% |
| Música III | 63 | 21 | 5 | 1 | 33.33% | 42.85% |
| Música IV | 125 | 62 | 1 | 1 | 49.60% | 51.20% |
| **Total** | **257** | **107** | **21** | **8** | **41.63%** | **52.91%** |

## Experimento III

## [Particionamento: COCHONUT – Padrões: B&P – Grafo: B&P]

### Comparação completa de acordes (todas as notas)

Tabela B.17 – Resultados do Experimento III, considerando a comparação completa de acordes.

| Experimento | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Música I** | 38 | 38 | 1.0000 | 72.9378 | 27.0621 | 11 | 10 | 10 |
| **Música II** | 34 | 28 | 0.8235 | 50.3048 | 49.6951 | 12 | 14 | 17 |
| **Música III** | 69 | 50 | 0.7246 | 35.1148 | 64.8851 | 14 | 33 | 51 |
| **Música IV** | 129 | 97 | 0.7519 | 54.4461 | 45.5538 | 58 | 36 | 69 |
| **Experimento** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| **Música I** | 13 | 4 | 603.84 | 934.06 | 491.76 | 934.06 | 1045.00 | 1223.75 |
| **Música II** | 4 | 0 | 1430.00 | 0.00 | 908.92 | 0.00 | 3428.75 | 0.00 |
| **Música III** | 3 | 0 | 673.33 | 0.00 | 457.91 | 0.00 | 1090.00 | 0.00 |
| **Música IV** | 1 | 1 | 377.50 | 911.25 | 175.00 | 911.25 | 377.50 | 911.25 |

Tabela B.18 – Valores médios das métricas calculadas na tabela B.17, com exceção de $O\leftarrow_{MAX}$ e $\rightarrow O_{MAX}$, que exibem os valores máximos das respectivas colunas da tabela B.17.

| Experimento | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Músicas I, II, III e IV** | 67.5 | 53.25 | 0.8250 | 53.2008 | 46.7990 | 23.75 | 23.25 | 36.75 |
| **Experimento** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| **Músicas I, II, III e IV** | 5.25 | 1.25 | 771.16 | 461.32 | 508.39 | 461.32 | 3428.75 | 1223.75 |

Tabela B.19 – Resultados do Experimento III, considerando a comparação completa de acordes.

| Experimento | A | O | O← | →O | $PA_j = O \div A$ | $PA_t = (O + O\leftarrow + \rightarrow O) \div A$ |
|---|---|---|---|---|---|---|
| **Música I** | 38 | 11 | 13 | 4 | 28.94% | 73.68% |
| **Música II** | 34 | 12 | 4 | 0 | 35.29% | 47.05% |
| **Música III** | 69 | 14 | 3 | 0 | 20.28% | 24.63% |
| **Música IV** | 129 | 58 | 1 | 1 | 44.96% | 46.51% |
| **Total** | **270** | **95** | **21** | **5** | **35.18%** | **44.81%** |

## Experimento IV

## [Particionamento: COCHONUT – Padrões: COCHONUT – Grafo: B&P]

### Comparação básica de acordes (tônica, terça e quinta)

Tabela B.20 – Resultados do Experimento IV, considerando a comparação básica de acordes.

| Música | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Música I** | 36 | 38 | 1.0555 | 79.9607 | 20.0392 | 12 | 4 | 2 |
| **Música II** | 33 | 30 | 0.9090 | 63.9393 | 36.0606 | 15 | 9 | 12 |
| **Música III** | 63 | 49 | 0.7777 | 68.0878 | 31.9121 | 34 | 9 | 21 |
| **Música IV** | 125 | 97 | 0.7760 | 71.8388 | 28.1611 | 77 | 17 | 46 |
| **Música** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| **Música I** | 10 | 12 | 461.37 | 897.18 | 364.10 | 897.18 | 1045.00 | 1256.25 |
| **Música II** | 3 | 3 | 637.08 | 582.08 | 398.75 | 582.08 | 1050.00 | 667.50 |
| **Música III** | 5 | 2 | 628.75 | 1595.00 | 363.25 | 1595.00 | 1090.00 | 1675.00 |
| **Música IV** | 1 | 1 | 253.75 | 911.25 | 172.91 | 911.25 | 253.75 | 911.25 |

Tabela B.21 – Valores médios das métricas calculadas na tabela B.20, com exceção de $O\leftarrow_{MAX}$ e $\rightarrow O_{MAX}$, que exibem os valores máximos das respectivas colunas da tabela B.20.

| Músicas | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Médias de I, II, III e IV** | 64.25 | 53.50 | 0.8795 | 70.9566 | 29.0432 | 34.50 | 9.75 | 20.25 |
| **Músicas** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| **Médias de I, II, III e IV** | 4.75 | 4.50 | 495.23 | 996.37 | 324.75 | 996.37 | 1090.00 | 1675.00 |

Tabela B.22 – Resultados do Experimento IV, considerando a comparação básica de acordes.

| Experimento | A | O | O← | →O | $PA_j = O \div A$ | $PA_t = (O + O\leftarrow + \rightarrow O) \div A$ |
|---|---|---|---|---|---|---|
| **Música I** | 36 | 12 | 10 | 12 | 33.33% | 94.44% |
| **Música II** | 33 | 15 | 3 | 3 | 45.45% | 63.63% |
| **Música III** | 63 | 34 | 5 | 2 | 53.96% | 65.07% |
| **Música IV** | 125 | 77 | 1 | 1 | 61.60% | 63.20% |
| **Total** | **257** | **138** | **19** | **18** | **53.69%** | **68.09%** |

## Experimento IV

## [Particionamento: COCHONUT – Padrões: COCHONUT – Grafo: B&P]

### Comparação completa de acordes (todas as notas)

Tabela B.23 – Resultados do Experimento IV, considerando a comparação completa de acordes.

| Experimento | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Música I** | 38 | 41 | 1.0789 | 78.6596 | 21.3403 | 13 | 5 | 2 |
| **Música II** | 34 | 31 | 0.9117 | 62.6512 | 37.3487 | 15 | 10 | 13 |
| **Música III** | 69 | 50 | 0.7246 | 58.4463 | 41.5536 | 31 | 14 | 32 |
| **Música IV** | 129 | 97 | 0.7519 | 68.0093 | 31.9906 | 73 | 21 | 54 |
| **Experimento** | **O←** | **→O** | **$O←_M$** | **$→O_M$** | **$O←_T$** | **$→O_T$** | **$O←_{MAX}$** | **$→O_{MAX}$** |
| **Música I** | 11 | 12 | 460.68 | 897.18 | 370.08 | 897.18 | 1045.00 | 1256.25 |
| **Música II** | 3 | 3 | 637.08 | 582.08 | 398.75 | 582.08 | 1050.00 | 667.50 |
| **Música III** | 5 | 0 | 702.25 | 0.00 | 399.37 | 0.00 | 1090.00 | 0.00 |
| **Música IV** | 1 | 1 | 377.50 | 911.25 | 175.00 | 911.25 | 377.50 | 911.25 |

Tabela B.24 – Valores médios das métricas calculadas na tabela B.23, com exceção de $O←_{MAX}$ e $→O_{MAX}$, que exibem os valores máximos das respectivas colunas da tabela B.23.

| Experimento | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Músicas I, II, III e IV** | 67.50 | 54.75 | 0.8667 | 66.9416 | 33.0583 | 33 | 12.50 | 25.25 |
| **Experimento** | **O←** | **→O** | **$O←_M$** | **$→O_M$** | **$O←_T$** | **$→O_T$** | **$O←_{MAX}$** | **$→O_{MAX}$** |
| **Músicas I, II, III e IV** | 5.00 | 4.00 | 544.37 | 597.62 | 335.80 | 597.62 | 1090.00 | 1256.25 |

Tabela B.25 – Resultados do Experimento IV, considerando a comparação completa de acordes.

| Experimento | A | O | O← | →O | $PA_j = O \div A$ | $PA_t = (O + O← + →O) \div A$ |
|---|---|---|---|---|---|---|
| **Música I** | 38 | 13 | 11 | 12 | 34.21% | 94.73% |
| **Música II** | 34 | 15 | 3 | 3 | 44.11% | 61.76% |
| **Música III** | 69 | 31 | 5 | 0 | 44.92% | 52.17% |
| **Música IV** | 129 | 73 | 1 | 1 | 56.68% | 58.13% |
| **Total** | **270** | **132** | **20** | **16** | **48.88%** | **62.22%** |

## Experimento V

## [Particionamento: COCHONUT – Padrões: COCHONUT – Grafo: COCHONUT]

### Comparação básica de acordes (tônica, terça e quinta)

Tabela B.26 – Resultados do Experimento V, considerando a comparação básica de acordes.

| Música | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Música I** | 36 | 45 | 1.2500 | 85.3374 | 14.6625 | 17 | 11 | 2 |
| **Música II** | 33 | 39 | 1.1818 | 71.7877 | 28.2122 | 22 | 15 | 9 |
| **Música III** | 63 | 68 | 1.0793 | 78.4380 | 21.5619 | 50 | 15 | 11 |
| **Música IV** | 125 | 125 | 1.0000 | 77.7332 | 22.2667 | 93 | 28 | 29 |
| **Música** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| **Música I** | 10 | 7 | 269.62 | 749.46 | 242.21 | 749.46 | 292.50 | 1017.50 |
| **Música II** | 1 | 1 | 253.75 | 267.50 | 225.35 | 267.50 | 253.75 | 267.50 |
| **Música III** | 2 | 0 | 260.00 | 0.00 | 228.33 | 241.25 | 261.25 | ≤ 250 |
| **Música IV** | 2 | 1 | 321.25 | 911.25 | 218.12 | 911.25 | 388.75 | 911.25 |

Tabela B.27 – Valores médios das métricas calculadas na tabela B.26, com exceção de $O\leftarrow_{MAX}$ e $\rightarrow O_{MAX}$, que exibem os valores máximos das respectivas colunas da tabela B.26.

| Músicas | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Médias de I, II, III e IV** | 64,25 | 69.25 | 1.1277 | 78.3240 | 21.6758 | 45.50 | 17.25 | 12.75 |
| **Músicas** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| **Médias de I, II, III e IV** | 3.75 | 2.25 | 276.15 | 482.05 | 228.50 | 542.36 | 388.75 | 1017.50 |

Tabela B.28 – Resultados do Experimento V, considerando a comparação básica de acordes.

| Experimento | A | O | O← | →O | $PA_j = O \div A$ | $PA_t = (O + O\leftarrow + \rightarrow O) \div A$ |
|---|---|---|---|---|---|---|
| **Música I** | 36 | 17 | 10 | 7 | 47.22% | 94.44% |
| **Música II** | 33 | 22 | 1 | 1 | 66.66% | 72.72% |
| **Música III** | 63 | 50 | 2 | 0 | 79.36% | 82.53% |
| **Música IV** | 125 | 93 | 2 | 1 | 74.40% | 76.80% |
| **Total** | **257** | **182** | **15** | **9** | **70.81%** | **80.15%** |

## Experimento V

## [Particionamento: COCHONUT – Padrões: COCHONUT – Grafo: COCHONUT]

### Comparação completa de acordes (todas as notas)

Tabela B.29 – Resultados do Experimento V, considerando a comparação completa de acordes.

| Experimento | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Música I** | 38 | 49 | 1.2894 | 84.2091 | 15.7908 | 18 | 13 | 2 |
| **Música II** | 34 | 43 | 1.2647 | 71.0023 | 28.9976 | 23 | 18 | 9 |
| **Música III** | 69 | 72 | 1.0434 | 71.9459 | 28.0540 | 49 | 19 | 17 |
| **Música IV** | 129 | 127 | 0.9844 | 71.9475 | 28.0524 | 86 | 36 | 39 |
| **Experimento** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| **Música I** | 10 | 8 | 269.62 | 781.71 | 242.70 | 781.71 | 292.50 | 1017.50 |
| **Música II** | 1 | 1 | 253.75 | 267.50 | 225.35 | 267.50 | 253.75 | 267.50 |
| **Música III** | 3 | 0 | 262.08 | 0.00 | 230.08 | 241.25 | 266.25 | ≤ 250 |
| **Música IV** | 3 | 1 | 387.91 | 911.25 | 277.75 | 911.25 | 397.50 | 911.25 |

Tabela B.30 – Valores médios das métricas calculadas na tabela B.29, com exceção de $O\leftarrow_{MAX}$ e $\rightarrow O_{MAX}$, que exibem os valores máximos das respectivas colunas da tabela B.29.

| Experimento | A | R | PQ | PC | PF | O | X | M |
|---|---|---|---|---|---|---|---|---|
| **Músicas I, II, III e IV** | 67.50 | 72.75 | 1.1454 | 74.7762 | 25.2237 | 44 | 21.50 | 16.75 |
| **Experimento** | **O←** | **→O** | **$O\leftarrow_M$** | **$\rightarrow O_M$** | **$O\leftarrow_T$** | **$\rightarrow O_T$** | **$O\leftarrow_{MAX}$** | **$\rightarrow O_{MAX}$** |
| **Músicas I, II, III e IV** | 4.25 | 2.50 | 293.34 | 490.11 | 243.97 | 550.42 | 397.50 | 1017.50 |

Tabela B.31 – Resultados do Experimento V, considerando a comparação completa de acordes.

| Experimento | A | O | O← | →O | $PA_j = O \div A$ | $PA_t = (O + O\leftarrow + \rightarrow O) \div A$ |
|---|---|---|---|---|---|---|
| **Música I** | 38 | 18 | 10 | 8 | 47.36% | 94.73% |
| **Música II** | 34 | 23 | 1 | 1 | 67.64% | 73.52% |
| **Música III** | 69 | 49 | 3 | 0 | 71.01% | 75.36% |
| **Música IV** | 129 | 86 | 3 | 1 | 66.66% | 69.76% |
| **Total** | **270** | **175** | **17** | **9** | **64.81%** | **74.44%** |

## Apêndice C – Diagrama de Classes Reduzido

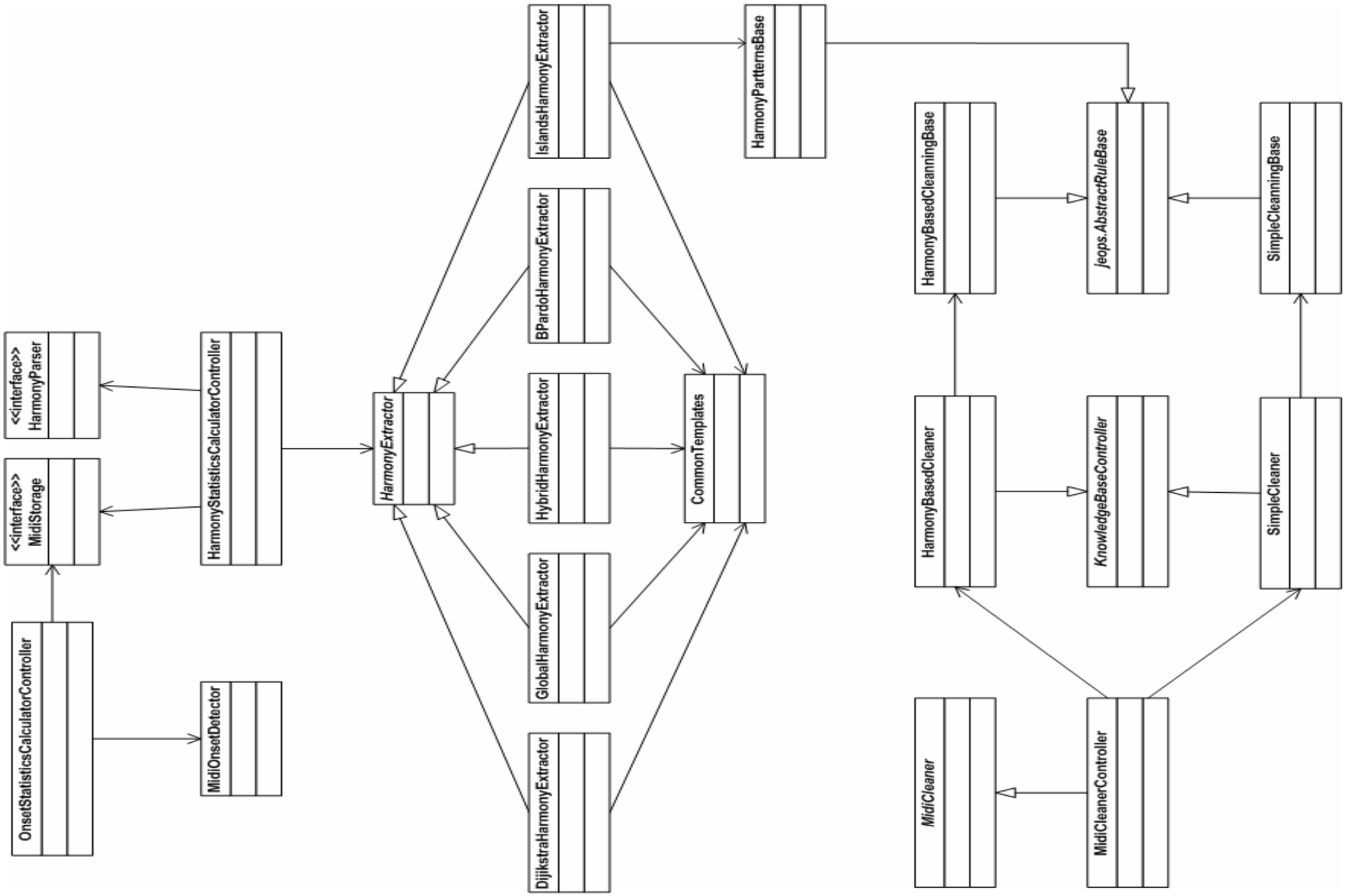


Figura C.1 - Diagrama de classes reduzido. Somente as principais classes de cada módulo são mostradas, possibilitando um melhor entendimento em alto nível da arquitetura dos módulos desenvolvidos.

# Apêndice D – Siglas e Abreviações

**API –** *Application Programming Interface* é um conjunto de classes, rotinas, padrões e bibliotecas fornecidos por um software para utilização de suas funcionalidades;

**BPM –** *Beats per Minute* é uma medida usada para indicar o andamento de peças musicais, em quantidade de pulsos por minuto;

**DFT –** *Discrete Fourier Transform* é uma das formas específicas da Análise de Fourier. Transforma uma função em outra, que é chamada de domínio de freqüência, ou DFT, da função original;

**EDS –** *Extractor Discovery System, sistema desenvolvido pela SONY Labs Paris*;

**EM –** *Expectation Maximization* é uma técnica de treinamento supervisionado usada em modelos escondidos de Markov (HMMs);

**EPCP –** *Enhanced Pitch Class Profile* é uma versão modificada do *Pitch Class Profile* (PCP);

**HMM –** *Hidden Markov Models*, ou Modelos Escondidos de Markov, é uma técnica de aprendizagem de máquina supervisionada;

**JEOPS –** *Java Embedded Object Production System* é um motor de inferência desenvolvido em Java com API livre;

**KNN –** *K-Nearest Neighbors*, é uma técnica de aprendizagem de máquina para tarefas de classificação;

**MIDI** – *Musical Instrument Interface* é uma tecnologia padronizada de comunicação entre instrumentos musicais e equipamentos eletrônicos. Tecnicamente, é um protocolo. Um arquivo MIDI não contém o áudio propriamente dito, e sim as instruções para produzi-lo, ou seja, é uma espécie de partitura digitalizada;

**PCP –** *Pitch Class Profile* é uma espécie de histograma calculado a partir de um sinal de áudio, que mapeia a presença de cada classe de freqüências no mesmo;

**SOL –** *Sound Onset Labellizer* é um conjunto de ferramentas (*toolbox*), desenvolvido sobre o MATLAB, que permite a um músico rotular ataques em trechos de áudio;

**WAV –** Abreviação para *Waveform audio format*. É um formato de áudio padrão para armazenamento de áudio em PCs.

# Referências Bibliográficas


**[1]** A. Adolfo. O Livro do Músico – Harmonia e Improvisação para Piano, Teclados e Outros Instrumentos. Lumiar Editora, 4ª edição. 1989.

**[2]** L. Alves. Escalas para Improvisação. Ed. Irmãos Vitale. 2ª edição. 1997.

**[3]** J. Bello e J. Pickens. A Robust Mid-level Representation for Harmonic Content in Music Signals. *Proceedings of the International Symposium on Music Information Retrieval – ISMIR*. Londres, p304-311. 2005.

**[4]** B. Braga. Curso Fundamental de Harmonia. Ed. Ricordi Brasileira. 1ª edição.

**[5]** G. Cabral, J. Briot e F. Pachet. Impact of Distance in Pitch Class Profile Computation. *Proceedings of the 10th Brazilian Symposium on Computer Music (SBCM'2005)*. Belo Horizonte, p319-324. 2005.

**[6]** R. Cássio. Extração de Informação Simbólica de Áudio: Algoritmos de Reconhecimento de Onsets. Trabalho de Graduação. Universidade Federal de Pernambuco. 2007.

**[7a]** A. Chediak. Songbook: Bossa Nova. Editora Lumiar. Vol. 1. 5ª edição. 156 páginas. 1994.

**[7b]** A. Chediak. Songbook: Bossa Nova. Editora Lumiar. Vol. 2. 7ª edição. 160 páginas. 1990.

**[7c]** A. Chediak. Songbook: Bossa Nova. Editora Lumiar. Vol. 3. 5ª edição. 154 páginas. 1990.

**[7d]** A. Chediak. Songbook: Bossa Nova. Editora Lumiar. Vol. 4. 2ª edição. 166 páginas. 1990.

**[7e]** A. Chediak. Songbook: Bossa Nova. Editora Lumiar. Vol. 5. 5ª edição. 176 páginas. 1994.

**[8]** E. Clarke. Some Aspects of Rhythm and Expression in Performances of Erik Satie's "Gnossienne no. 5". Music Perception. v2(3). p299-328. 1985.

**[9]** E. Clarke. Timing in the Performance of Erik Satie's 'Vexations'. Acta Psychologica. v50(1). p1-19. Janeiro, 1982.

**[10]** W. Cook, W. Cunningham, W. Pulleyblank e A. Schrijver. Combinatorial Optimization. John Wiley and Sons. Nova York, 355 páginas. 1998.

**[11]** D'Accord Music Software. Dicionário de Violão 3.0 DEMO. www.daccord.com.br. Setembro de 2007.

**[12]** S. Dixon. Onset Detection Revisited. *Proceedings of the 9th International Conference on Digital Audio Effects*. Montreal. 2006.

**[13]** N. Faria. Acordes, Arpejos e Escalas Para Violão e Guitarra. Lumiar Editora. 1999.

**[14]** C. Figueira. JEOPS – Integração entre Objetos e Regras de Produção em Java. Dissertação de Mestrado. Universidade Federal de Pernambuco. 2000.

**[15]** T. Fujishima. Realtime Chord Recognition of Musical Sound: A System Using Common Lisp Music. *Proceedings of the International Computer Music Conference*. p464-467. Beijing 1999.

**[16]** E. Gomez e P. Herrera. Automatic Extraction of Tonal Metadata from Polyphonic Audio Recordings. *Proceedings of the 25th International Audio Engineering Society Conference*. Londres. 2004.

**[17]** C. Harte e M. Sandler. Automatic Chord Identification Using a Quantized Chromagram. *Proceedings of the Audio Engineering Society 118th Convention*. Barcelona. 2005.

**[18]** C. Krumhansl. Cognitive Foundations of Musical Pitch. Oxford University Press. 318 páginas. 1990.

**[19]** K. Lee e M. Slaney. A Unified System for Chord Transcription and Key Extraction Using Hidden Markov Models. *Proceedings of the International Symposium on Music Information Retrieval*. p245-250. Viena. 2007.

**[20]** K. Lee e M Slaney. Automatic Chord Recognition from Audio Using an HMM with Supervised Learning. *Proceedings of the 7th International Conference on Music Information Retrieval.* Victoria. 2006.

**[21]** K. Lee. A System for Automatic Chord Transcription from Audio Using Genre-Specific Hidden Markov Models. *Proceedings of the 5th Workshop on Adaptive Multimedia Retrieval*. Paris. 2007.

**[22]** K. Lee. Automatic Chord Recognition from Audio Using Enhanced Pitch Class Profile.

**[23]** P. Leveau, L. Daudet e G. Richard. Methodology and Tools for the Evaluation of Automatic Onset Detection Algorithms in Music. *Proceedings of the 5th International Conference on Music Information Retrieval*. Barcelona. 2004.

**[24]** E. Lima. Descoberta Automática de Conhecimento em Interpretações Musicais: O Caso do Acompanhamento Rítmico ao Violão. Tese de Doutorado. Universidade Federal de Pernambuco. 2004.

**[25]** J. Maxwell. An Expert System for Harmonizing Analysis of Tonal Music. Em *Understanding Music with AI: Perspectives on Music Cognition*. AAAI Press. v1 p335-353. 1992.

**[26]** B. Med. Teoria da Música. Ed. Musimed. 4ª edição. 2001.

**[27]** MIREX 2006. Music Information Retrieval Exchange 2006. http://www.music-ir.org/mirex2006/. Maio de 2007.

**[28]** T. Owens. Charlie Parker: Techniques of Improvisation, 2 vols. Tese de Doutorado. University of Los Angeles. 1974.

**[29]** F. Pachet. A Meta-level Architecture Applied to the Analysis of Jazz Chord Sequences. *Proceedings of the International Computer Music Conference*. Montreal. 1991.

**[30]** B. Pardo e W. Birmingham. The Chordal Analysis of Tonal Music. Technical report CSE-TR-439-01. The University of Michigan, Dept. of Electrical Engineering and Computer Science. Ann Arbor. 2001.

**[31]** B. Repp, L. Windsor e P. Desain. Effects of Tempo on the Timing of Simple Musical Rhythms. Music Perception. v19(40). p565-593. 2002.

**[32]** S. Russel e P. Norvig. Artificial Intelligence : A Modern Approach. Prentice Hall. 2ª Edição. 1132 páginas. 2002.

**[33]** R. Scholz. Análise Harmônica Funcional Automática. Trabalho de Graduação. Universidade Federal de Pernambuco. Recife. 2005.

**[34]** R. Scholz, V. Dantas e G. Ramalho. Automating Functional Harmonic Analysis: The Funchal System. *Proceedings of the Seventh IEEE International Symposium on Multimedia (ISM'05)*. v1 p759-764. Irvine, 2005.

**[35]** R. Scholz, V. Dantas e G. Ramalho. Funchal: a System for Automatic Functional Harmonic Analysis. *Proceedings of the 10th Brazilian Symposium on Computer Music*. v1. p287-290. Belo Horizonte. 2005.

**[36]** S. Serapião. Ritornello: um Framework para Representação do Conhecimento Musical. Dissertação de Mestrado. Universidade Federal de Pernambuco. 2004.

**[37]** A. Sheh e D. Ellis. Chord Segmentation and Recognition Using EM-Trained Hidden Markov Models. *Proceedings of the International Symposium on Music Information Retrieval*. Baltimore. 2003.

**[38]** SONY Computer Science Lab. Holistic Signal Processing. http://www.csl.sony.fr/items/2004/holistic-signal-processing/, Jan. 2008.

**[39]** SONY Computer Science Lab. SONY Computer Science Lab. http://www.csl.sony.fr/. Janeiro de 2008.

**[40]** SUN Microsystems. Java Sound MIDI API Package Summary. http://java.sun.com/j2se/1.5.0/docs/api/javax/sound/midi/package-summary.html. Dezembro de 2007.

**[41]** D. Temperley e D. Sleator. The Melisma Music Analyzer. http://www.link.cs.cmu.edu/music-analysis/. Janeiro de 2008.

**[42]** D. Temperley e D. Sleator. Modeling Meter and Harmony: A Preference-Rule Approach. Computer Music Journal. v23(1). p10-27. 1999.

**[43]** G. Widmer. Applications of Machine Learning to Music Research: Empirical Investigations into the Phenomenon of Musical Expression. Em *Machine Learning, Data Mining and Knowledge Discovery: Methods and Applications*. Wiley & Sons. 1998.

**[44]** G. Widmer. Discovering Strong Principles of Expressive Music Performance with the PLCG Rule Learning Strategy. *Proceedings of the 11th European Conference on Machine Learning (ECML'01)*. Berlin. 2001.

**[45]** G. Widmer. Large-scale Induction of Expressive Performance Rules: First Quantitative Results. *Proceedings of the International Computer Music Conference (ICMC'2000)*. International Computer Music Association. São Francisco. 2000.

**[46]** G. Widmer. The Synergy of Music Theory and AI: Learning Multi-level Expressive Interpretation. *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*. p114-119. AAAI Press. Seattle. 1994.

**[47]** G. Widmer. Using AI and Machine Learning to Study Expressive Music Performance: Project Survey and First Report. AI Communications. v14(3). p149-162. 2001.

**[48]** T. Winograd. Linguistics and the Computer Analysis of Tonal Harmony. The Journal of Music Theory. v12. p2-49. 1968.

**[49]** Yahoo Fórum. MIDI Guitar Yahoo Forum. http://launch.groups.yahoo.com/group/midiguitar. Maio de 2007.

**[50]** T. Yoshioka et al. Automatic Chord Transcription with Concurrent Recognition of Chord Symbols and Boundaries. *Proceedings of the 5th International Conference in Music Information Retrieval*. Barcelona. 2004.

**[51]** P. Zanon e G. Widmer. Learning to Recognize Famous Pianists with Machine Learning Techniques. *Proceedings of the Stockholm Music Acoustics Conference (SMAC'03)*. p581-584. Stockolmo. 2003.