UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

LUIZ FERNANDO CRUZ

Um Método para Recuperação de Composições Polifônicas aplicado na Busca de Tablaturas Textuais da Web

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Profa. Dra. Renata Galante
Orientadora

Prof. Dr. Marcelo Soares Pimenta
Co-orientador

Porto Alegre, novembro de 2014.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Cruz, Luiz Fernando da

Um Método para Recuperação de Composições Polifônicas aplicado na Busca de Tablaturas Textuais da Web [manuscrito] / Luiz Fernando Cruz. – 2014.

79 f.:il.

Orientador: Renata Galante; Co-orientador: Marcelo Pimenta.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2014.

1.MIR. 2.Similaridade Melódica 3.Tablatura Musical. I. Galante, Renata. II. Pimenta, Marcelo. III. Um Método para Recuperação de Composições Polifônicas aplicado na Busca de Tablaturas Textuais da Web.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Carlos Alexandre Netto
Vice-Reitor: Prof. Rui Vicente Oppermann
Pró-Reitor de Pós-Graduação: Prof. Vladimir Pinheiro do Nascimento
Diretor do Instituto de Informática: Prof. Luís da Cunha Lamb
Coordenador do PPGC: Prof. Luigi Carro
Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

"Muitas pessoas pensam que a felicidade somente será possível depois de alcançar algo, mas a verdade é que deixar para ser feliz amanhã é uma forma de ser infeliz"

Roberto Shinyashiki

AGRADECIMENTOS

Diversas pessoas merecem meus sinceros agradecimentos por terem me ajudado no desenvolvimento e definição deste trabalho. Porém, minha orientadora, professora Renata Galante, merece um agradecimento especial pelos conhecimentos transmitidos, paciência em diversos momentos, e transmissão de otimismo mesmo em momentos difíceis. Outros professores que foram incríveis, me ajudando de maneira empolgante, foram Prof. Marcelo Pimenta, Prof. Marcelo Johann, Prof. Marcus Ritt e Prof. Rodrigo Machado.

Não poderia deixar de agradecer familiares e amigos que demonstraram grande apoio, principalmente a minha esposa Kate, companheira em diversos momentos difíceis durante o período de desenvolvimento deste trabalho.

A Method for Retrieving Polyphonic Compositions Applied in the Search for Textual Tablatures on Web

ABSTRACT

Among the various operations that can be performed with musical compositions, represented through a notation, is the evaluation of melodic similarity mainly for search and retrieval of these works. This paper presents the method of evaluation of melodic similarity named Monophonic Contained Matching, focused on the search of compositions. Also describes the implementation details of a tool to search textual tabs in web using the proposed method. The analysis of the experiments demonstrated that the CMC method can locate specific compositions in two-thirds of queries (69.44%), managing to place the expected result on average in the 11th ranking position.

Keywords: MIR, Melodic Similarity, Musical Tablature.

RESUMO

Dentre as diversas operações que podem ser realizadas com composições musicais, representadas através de uma notação, está a avaliação de similaridade melódica, utilizada principalmente na busca e recuperação dessas obras. O presente trabalho apresenta o método de avaliação de similaridade melódica denominado Correspondência de Monofonia Contida (CMC), focado na operação de busca de composições. Também estão descritos os detalhes de implementação de uma ferramenta para busca de tablaturas textuais da *web* utilizando o método proposto. A análise dos experimentos realizados demonstrou que o método CMC consegue localizar composições específicas em dois terços das consultas (69,44%), conseguindo colocar o resultado esperado, em média, na 11ª posição de ranqueamento.

Palavras-Chave: MIR, Similaridade Melódica, Tablatura Musical.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AFN	Autômatos Finitos Não Determinísticos
AFNε	Autômatos Finitos Não Determinísticos com Movimentos Vazios
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Ensino Superior
C-Brahms	*Content-Based Retrieval and Analysis of Harmony and other Music Structures*
CMC	Correspondência de Monofonia Contida
DFS	*Depth First Search*
EMD	*Earth Mover's Distance*
GAD	Grafo Acíclico Dirigido
HTML	*Hypertext Markup Language*
ISMIR	*International Society for Music Information Retrieval*
LCTS	*Longest Common Transposition-Invariant Subsequence*
MAM	Métodos de Acessos Métricos
Meldex	*Melody Index*
MIDI	*Musical Instrument Digital Interface*
MIR	*Music Information Retrieval*
MRR	*Mean Reciprocal Rank*
MSM	*Maximal Subset Matching*
PDF	*Portable Document Format*
PROMS	*Procedures for Music Search*
PTD	*Proportional Transportation Distance*
QbH	*Query by Humming*
QbT	*Query by Tapping*
RI	Recuperação de Informação
RR	*Reciprocal Rank*
SMF	*Standard MIDI File*
SQL	*Structured Query Language*

TD	*Transportation Distance*
URL	*Uniform Resource Locator*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A música é admirada por proporcionar sensações e sentimentos agradáveis ao homem. Ela enquadra-se no campo artístico, mas também no campo da ciência, onde seus elementos e fenômenos relacionados são estudados. Bohumil Med (1996) define a música como sendo o produto da combinação de sons sucessivos, com ordem, equilíbrio e proporção dentro do tempo.

Informações relacionadas à música podem ser associadas a diversos contextos, sendo dois desses mais aparentes: o relativo à representação musical, através alguma notação, e aquele diretamente relacionado a fenômenos acústicos (som). Selfridge-Field (1997) aponta ainda outros contextos para informações do campo musical: o referente a parâmetros analíticos, o relacionado à percepção e entendimento da música, e ainda o relativo a movimentos gestuais de um dançarino. Este trabalho trata especificamente das informações que se enquadram no contexto de representação musical.

As notações para representação musical trabalham com domínios usuais finitos, onde os mesmos conjuntos de elementos são combinados de maneiras diferentes para gerar uma composição aparentemente original. Sendo os domínios usualmente finitos, supõe-se que quanto maior o número de composições maior a chance de repetição ou percepção de equivalência entre trechos de obras musicais, além de somente parte das combinações de elementos produzirem sonoridade agradável. Este cenário demonstra-se favorável para aplicações computacionais de avaliação de similaridade entre composições.

A similaridade musical é uma área de estudo constante no meio acadêmico, onde normalmente o objetivo é encontrar trechos de composições que combinem melodicamente. A capacidade de reconhecer semelhanças melódicas está no cerne das perguntas mais frequentes sobre música. Através da avaliação de similaridade melódica pode-se encontrar composições simplesmente pela lembrança de um trecho musical. Segundo Selfridge-Field (1998), a melodia é o que faz a música memorável, levando uma pessoa lembrar-se muito tempo após ouvi-la, mesmo tendo esquecido sua letra. Além do recurso de busca de composições, a avaliação de similaridade melódica pode auxiliar na atribuição de obras a compositores, no estudo histórico/cronológico de composições, ou ainda verificar novas obras quanto à originalidade, levando em conta bases de registros para proteção autoral.

A similaridade musical está inserida dentro de um tema de estudos denominado MIR (*Music Information Retrieval*), uma ciência interdisciplinar com o objetivo de identificar diversas palavras e propriedades contidas em seleções musicais, armazenando estes dados em bases musicais e fornecendo mecanismos para consulta (DEMOPOULOS, 2007). Ainda, o MIR inclui estudos sobre classificação de

composições, extração de características musicais, e alinhamento em tempo real entre áudio e partitura musical (TYPKE, 2007).

Um fator muito importante da busca de composições refere-se à subjetividade da percepção humana de similaridade melódica musical. É desejável que, métodos para busca de composições, utilizando similaridade melódica, permitam enfatizar e eliminar, de maneira arbitrária, padrões melódicos da análise de similaridade. Esta característica, de possibilitar enfatizar e eliminar padrões melódicos, faz com que o método de busca possa ser melhor adaptado à percepção particular de similaridade de seu utilizador.

Este trabalho tem como objetivo propor o método de avaliação de similaridade melódica Correspondência de Monofonia Contida (CMC), capaz de recuperar composições polifônicas em processos de busca. O método proposto avalia se alguma melodia monofônica, contida em uma consulta polifônica, corresponde a algum trecho contido em uma composição polifônica. Como diferencial, o método CMC permite desconsiderar padrões monofônicos, contidos na consulta, no processo de avaliação de similaridade. Para demonstrar a viabilidade do método, este trabalho também descreve uma solução capaz de buscar documentos da *web* contendo tablaturas textuais.

Foram realizados experimentos com usuários junto à ferramenta protótipo que implementa o método CMC, focada na busca de tablaturas textuais na *web*. Os resultados demonstraram que o método CMC consegue localizar composições específicas em dois terços das consultas realizadas (69,44%), conseguindo colocar o resultado esperado em uma busca, em média, na 11ª posição no ranqueamento. Além disso, foi possível identificar pontos onde o método pode ser melhorado.

O restante deste trabalho está organizado da seguinte forma: o capítulo 2 apresenta os conceitos básicos das notações musicais utilizadas no trabalho, o capítulo 3 descreve os trabalhos relacionados, o capítulo 4 apresenta em detalhes o método Correspondência de Monofonia Contida, o capítulo 5 detalha a implementação do buscador de tablaturas textuais da *web*, o capítulo 6 apresenta os experimentos realizados, e, por fim, estão apresentadas as considerações finais e referências utilizadas.

2 NOTAÇÕES MUSICAIS

Este capítulo tem como objetivo apresentar, resumidamente, os conceitos básicos das notações musicais utilizadas neste trabalho.

Uma notação musical permite representar elementos musicais através de símbolos. Todas as notações musicais utilizadas trabalham com a escala cromática temperada da música ocidental, composta por 12 notas: 7 notas naturais (C ou dó, D ou ré, E ou mi, F ou fá, G ou sol, A ou lá, e B ou si) e 5 notas acidentadas (C# ou Db, D# ou Eb, F# ou Gb, G# ou Ab, e A# ou Bb).

2.1 Partitura

A partitura musical consiste na notação padrão para representação de composições musicais, onde notas são anotadas graficamente em um pentagrama. O pentagrama consiste de um conjunto de cinco linhas horizontais paralelas. Uma nota pode ser anotada sobre as linhas ou nos espaços intermediários de um pentagrama. Uma clave, símbolo colocado no início do pentagrama, define a altura de uma determinada linha, definindo assim as alturas dos espaços e as linhas vizinhas. Os símbolos de sustenido (#) e bemol (b) são utilizados para representar as notas acidentadas. A Figura 2.1 mostra a disposição de notas de uma oitava (de um C a outro C, por exemplo) em um pentagrama anotado com uma clave de sol.

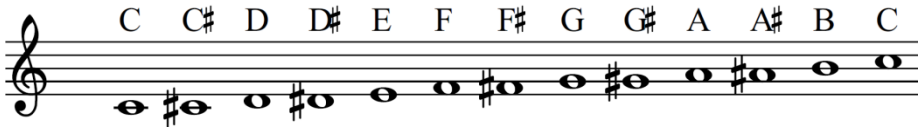


Figura 2.1: Notas em um pentagrama com clave de sol.

As notas e os silêncios são grafados por símbolos que indicam uma duração de tempo relativo. A maior duração em uso é a semibreve, seguida pela mínima, semínima, colcheia, semicolcheia, fusa e semifusa. A Figura 2.2 mostra os nomes das durações, símbolos de notas, símbolos de pausas e equivalências.

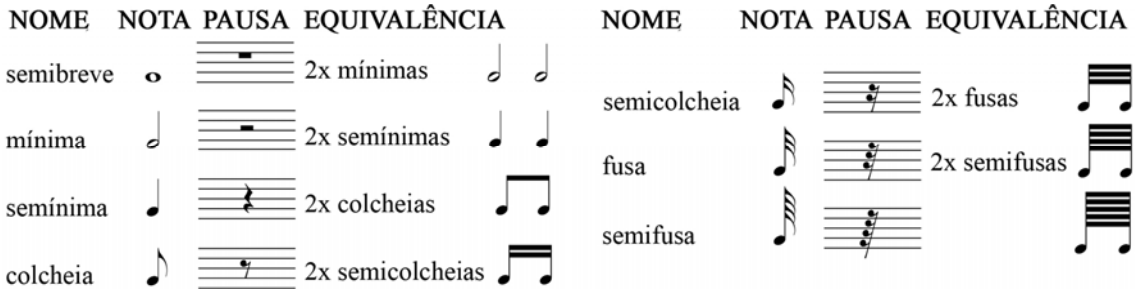


Figura 2.2: Representação de durações e suas equivalências.

Diversos outros elementos podem ser anotados em uma partitura, como pontos de duração, ligaduras, fórmulas de compasso, barras, acentos, ornamentos e vozes. Entretanto, nenhum destes outros elementos musicais é utilizado neste trabalho.

2.2 Tablatura Textual

A tablatura textual é uma notação baseada em texto que utiliza uma representação física de instrumentos trasteados, do grupo das guitarras, instrumentos de cordas dedilhadas como o violão, guitarra elétrica e o baixo elétrico.

Cada linha horizontal da tablatura textual, formada por hífens (-), representa uma corda do instrumento. Um número sobre uma representação de corda indica o traste (casa do instrumento) onde tal corda deve ser pressionada e tocada para gerar uma nota. As colunas de caracteres da tablatura textual definem a ordem cronológica de execução das notas. Números em uma mesma coluna de caracteres indicam notas simultâneas (acordes). A Figura 2.3 mostra um exemplo de um trecho da tablatura textual da música *Day Tripper*, da banda *The Beatles*.

```
e|--------------------------------|
B|--------------------------------|
G|--------------------------------|
D|--------------2-0----4-----0-2---|
A|------------2------2-----2-------|
E|---0----3-4-----------------------|
```

Figura 2.3: Trecho da tablatura textual da música Day Tripper da banda The Beatles.

Outros símbolos podem aparecer nas tablaturas textuais a fim de indicar técnicas de execução das notas, cada uma destas exigindo uma habilidade diferente do intérprete.

2.3 Notas MIDI

A especificação MIDI (*Musical Instrument Digital Interface*) define padrões de conexões físicas e comunicação entre dispositivos eletrônicos, como instrumentos musicais, sequenciadores e *mixers* (MMA, 1995). As comunicações MIDI são feitas através de mensagens, que carregam eventos representando ações de um músico tocando um instrumento, sendo `note_on` e `note_off` as mais importantes, correspondendo ao início e ao fim da emissão de uma nota (MULLER, 2007). Cada uma das 128 notas do padrão MIDI possui um número associado, tendo como referência o valor 60 para o dó central (C4), como mostra a Figura 2.4.

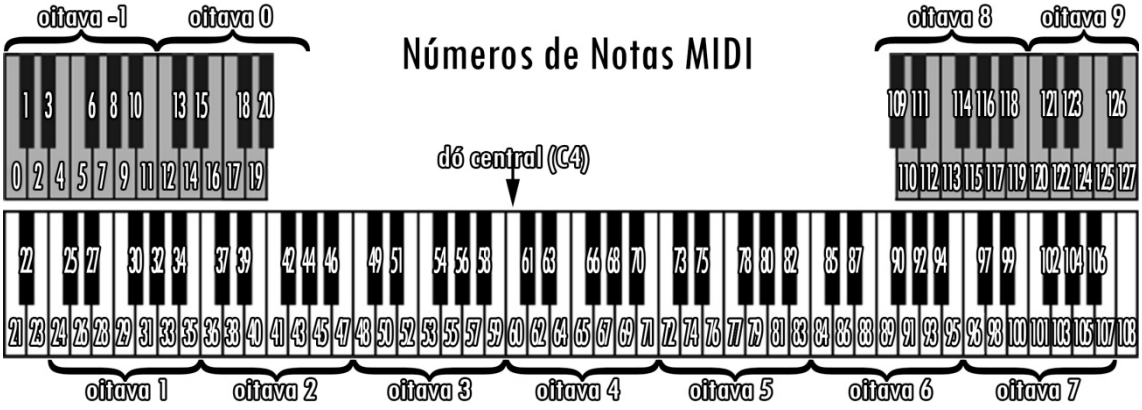


Figura 2.4: Números de notas MIDI organizadas por oitavas.

O padrão MIDI ainda define um formato de arquivo, o SMF (*Standard MIDI File*), que também utiliza os números de notas MIDI.

2.4 Considerações Finais do Capítulo

Mesmo esta dissertação estando relacionada à área musical, optou-se por apresentar neste capítulo somente os conceitos essenciais para a compreensão das notações musicais utilizadas neste trabalho.

A notação de partitura não foi utilizada diretamente na proposta deste trabalho, servindo exclusivamente como base para a apresentação de exemplos, devido sua simplicidade na representação musical. Já a tablatura textual figura como elemento central do protótipo implementado para busca de composições musicais na *web*. A codificação de notas MIDI é utilizada tanto no método proposto, quanto no protótipo implementado, devido sua simplicidade na representação de notas através de números, que permitem que informações de tom e oitava sejam derivadas.

3 TRABALHOS RELACIONADOS

Este capítulo descreve os trabalhos relacionados à recuperação de composições musicais. Na realização dos estudos buscou-se informações sobre métodos disponíveis para recuperação de composições polifônicas e ferramentas que implementam tais métodos que estejam disponíveis na *web*.

3.1 Métodos para Busca de Composições Polifônicas

Esta seção apresenta os métodos estudados que permitem a recuperação de composições musicais, simbolicamente representadas, utilizando avaliação de similaridade melódica. Duas abordagens ficam evidentes nestes estudos: métodos que se baseiam em algoritmos para comparação de *strings*, e métodos baseados em conjuntos de notas. Ao final da seção encontra-se um comparativo entre os métodos estudados.

3.1.1 Métodos Baseados em Strings

Os métodos baseados em *strings* utilizam cadeias de caracteres para representar sequências ordenadas de notas, permitindo que algoritmos já empregados na comparação de *strings* sejam utilizados para avaliar a similaridade melódica de composições. Tais métodos são usualmente utilizados para recuperar composições polifônicas, representadas através cadeias de caracteres em paralelo, a partir de consultas monofônicas.

Lemström e Tarhio (2003) apresentam quatro algoritmos baseados em *strings*, sendo que o mais simples, denominado *ShiftOrAnd*, encontra ocorrências de uma consulta em uma composição quando a altura musical de cada nota da consulta corresponder com a altura musical de alguma nota de mesmo tempo na composição (mesma coluna de caractere). Diferentemente do *ShiftOrAnd*, os outros três algoritmos, denominados *DirectCheck*, *IntervalMatching* e *MonoPoly*, não trabalham com correspondência de alturas, mas sim com correspondência de intervalos (diferenças de altura), permitindo encontrar ocorrências de uma consulta em uma composição mesmo que as melodias estejam transpostas em tons diferentes. A diferença essencial entre os algoritmos *DirectCheck*, *IntervalMatching* e *MonoPoly* está no desempenho. O *DirectCheck* é o mais lento na verificação de ocorrências de uma consulta em uma composição, porém, alternativamente, pode ser utilizado para verificar se uma determinada sequência de intervalos existe de maneira encadeada em uma composição, ou seja, se tal sequência de intervalos pode ser encontrada entre uma sequência de notas (monofonia contida na composição). A Figura 3.1 mostra a representação de uma consulta monofônica (a), juntamente com uma sequência de intervalos não encadeada (b) e outra encadeada (c).

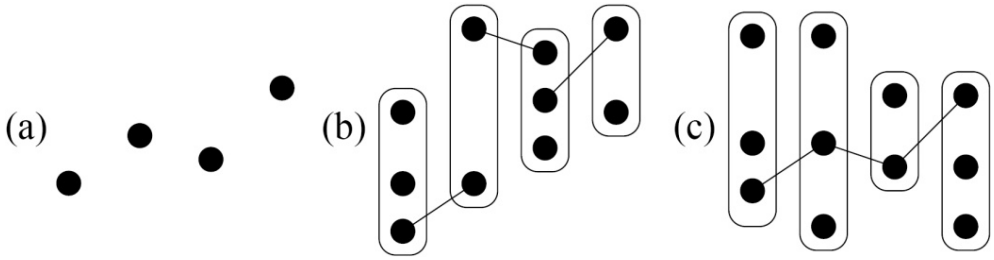


Figura 3.1: Consulta (a) e sequência de intervalos não encadeada (b) e encadeada (c).

Ainda sobre os algoritmos de Lemström e Tarhio (2003a), *IntervalMatching* e *MonoPoly* são considerados métodos de filtragem de candidatos, pois recuperam sequências de intervalos mesmo não encadeadas, necessitando apoio de outro algoritmo de checagem, como o *DirectCheck*. A vantagem do *MonoPoly* sobre o *IntervalMatching* se dá pela execução de uma etapa de pré-processamento nas composições, melhorando significativamente seu desempenho. Nenhum dos quatro algoritmos permite a correspondência parcial de notas de uma consulta na comparação com uma composição polifônica.

Lemström e Mäkinen (2003) ainda propõe um método diferente, denominado *Splitting*, que procura a menor quantidade de fragmentos de uma consulta de modo que cada um destes fragmentos corresponda em uma composição polifônica de maneira ordenada.

Em outro trabalho, Lemström, Navarro e Pinzon (2005) apresentam o método *Longest Common Transposition-Invariant Subsequence* (LCTS), que encontra a maior subsequência de uma *string* de consulta monofônica em uma composição polifônica. Como o próprio nome indica, o método permite que a melodia da consulta esteja transposta em um tom diferente da composição.

3.1.2 Métodos Baseados em Conjuntos

Métodos baseados em conjuntos utilizam pontos ponderados ou segmentos geométricos no plano euclidiano para representar notas e informações de uma melodia. O eixo horizontal (x) do plano é associado ao tempo, enquanto o eixo vertical (y) é associado à altura musical. Esta representação através de pontos ponderados ou segmentos geométricos permite incorporar outras dimensões de informações musicais de uma melodia, como a duração de notas, por exemplo. Mesmo existindo uma organização temporal, os métodos baseados em conjuntos assumem que notas de melodias não estão ordenadas.

Clausen, Engelbrecht, Meyer e Schmitz (2000) apresentam um método que utiliza a correspondência de pontos, compostos por tempos de início e altura de notas, para recuperar composições a partir de uma consulta. Tal método encontra-se implementado na ferramenta PROMS (*Procedures for Music Search*). O algoritmo de busca do método permite correspondência parcial, através da definição de um valor máximo para não correspondências de notas.

Já o trabalho de Ukkonen, Lemström e Mäkinen (2003) apresenta três algoritmos diferentes para recuperação de composições polifônicas a partir de uma consulta também polifônica. Todos os três algoritmos, denominados *P1*, *P2* e *P3*, consideram a informação de duração, representadas através de segmentos de reta indicando o início e o fim da emissão das notas. O algoritmo *P1* encontra translações de uma consulta em uma composição quando existir correspondência entre todas as

durações de notas. O algoritmo *P2* tem comportamento similar ao algoritmo *P1*, mas aceitando a correspondência parcial de durações de notas. Já o algoritmo *P3* encontra translações de uma consulta em uma composição de modo a obter a maior intersecção de durações, não exigindo a necessidade de existir correspondência de tempos de início ou fim da emissão das notas.

Lemström (2002), juntamente com Meredith e Wiggins, propõe ainda outro método baseado em conjuntos, denominado SIAMESE, que utiliza correspondência de padrões de notas, previamente indexados, na busca de composições. O trabalho descreve três algoritmos para indexação de padrões nas composições: SIA, que identifica os maiores padrões de notas que se repetem em uma melodia; SIATEC, que identifica as ocorrências dos padrões identificados pelo algoritmo SIA; e COSIATEC, que permite uma representação comprimida das composições, através de repetições sucessivas do algoritmo SIATEC, de modo a indexar as composições por completo com tamanhos diferentes de padrões de notas.

Clifford e Christodoulakis (2006) descrevem o método denominado *Maximal Subset Matching* (MSM) que obtém o maior subconjunto de uma consulta polifônica encontrado em uma composição também polifônica, utilizando informações de tempos de início e alturas de notas. O método projeta pontos, representando notas, em uma única dimensão, para posteriormente avaliar a correspondência de translações da consulta.

Outro método, proposto por Rayner Typke (2007), utilizando distâncias de transporte (*Transportation Distance* – TD) para calcular a dissimilaridade entre conjuntos de notas representadas como pontos ponderados, onde os pesos atribuídos correspondem a durações. O método consiste basicamente em calcular o menor custo para transportar os pesos de uma distribuição de pontos para outra. Duas distâncias de transporte são utilizadas no trabalho: *Proportional Transportation Distance* (PTD), que considera somente correspondências exatas, e *Earth Mover's Distance* (EMD), que permite correspondência parcial. As duas distâncias de transporte também se diferenciam pelo PTD cumprir, e EMD não cumprir, o teorema geométrico da desigualdade triangular, que permite que Métodos de Acessos Métricos (MAM) sejam utilizados na indexação e recuperação de composições em bases de dados, aumentando significativamente o desempenho em processo de busca. Devido as distâncias de transporte PTD e EMD obterem resultados relevantes somente quando comparados segmentos com tamanho e quantidade de notas similar, Rayner Typke descreve algoritmos que permitem a segmentação da consulta e das composições, possibilitando existir correspondências transladadas.

3.1.3 Comparação entre os Métodos Estudados

No estudo dos métodos para busca de composições polifônicas buscou-se apontar características relacionadas à correspondência de dados musicais. Além da abordagem e formato da consulta, a Tabela 3.1 mostra as informações de correspondência dos métodos estudados neste capítulo, sendo elas: tipo da correspondência; formato do padrão de correspondência; se o método permite impor algum tipo de restrição nos padrões a serem correspondidos em uma busca; forma de utilização da informação temporal da melodia; se o método considera a informação de duração de notas; se o método consegue identificar correspondências de tom juntamente com a oitava; se o método consegue identificar correspondências de intervalos considerando oitavas; se o método consegue identificar correspondências somente de

tom, desconsiderando oitavas; e se o método consegue identificar correspondências somente de intervalos, desconsiderando oitavas.

Tabela 3.1: Comparativo entre métodos para busca de composições polifônicas.

#	Método	Abordagem	Consulta	Correspondência								
				Tipo	Padrão	Restr. Padrões	Tempo	Duração	Tom e Oitava	Intervalo e Oitava	Somente Tom	Somente Intervalo
1	*ShiftOrAnd*	*strings*	mono	exata	mono		ordem				X	
2	*DirectCheck*	*strings*	mono	exata	mono		ordem					X
3	*IntervalMatch*	*strings*	mono	exata	mono		ordem					X
4	*MonoPoly*	*strings*	mono	exata	mono		ordem					X
5	*Splitting*	*strings*	mono	exata	mono	X	ordem			X		X
6	LCTS	*strings*	mono	parcial	mono	X	ordem			X		X
7	PROMS	conjuntos	poli	parcial	poli	X	ataque		X	X	X	X
8	*P1*	conjuntos	poli	exata	poli		ataque	X		X		X
9	*P2*	conjuntos	poli	parcial	poli		ataque	X		X		X
10	*P3*	conjuntos	poli	parcial	poli		ataque	X		X		X
11	SIAMESE	conjuntos	poli	parcial	poli		ataque			X		X
12	MSM	conjuntos	poli	parcial	poli		ataque			X		X
13	TD-EMD	conjuntos	poli	parcial	poli		ataque	X		X		X
14	TD-PTD	conjuntos	poli	exata	poli		ataque	X		X		X

Os primeiros seis métodos estudados, *ShiftOrAnd*, *DirectCheck*, *IntervalMatching*, *MonoPoly*, *Splitting* e LCTS, mostrados na Tabela 3.1, utilizam uma abordagem baseada em *strings* e, por este motivo, trabalham com consultas e correspondência de padrões monofônicos sobre uma composição. Além disso, tais métodos não consideram informações de duração e tempo de ataque das notas (início de emissão), considerando somente a ordem em que tais notas ocorrem. Também é possível observar que, os métodos *Splitting* e LCTS se destacam, permitindo identificar consultas transpostas em tons diferentes, dentro de qualquer intervalo de valores de notas, e restringir padrões de correspondência através da definição de parâmetros tais como máxima divisão do padrão e tamanho máximo de lacunas (notas não correspondidas). O LCTS também é o único método baseado em *strings* que permite correspondência parcial. Já os métodos baseados em conjuntos estudados, PROMS, *P1*, *P2*, *P3*, SIAMESE, MSM, TD-EMD e TD-PTD, caracterizam-se por permitir que a consulta e o padrão de correspondência sejam polifônicos, além de considerar o tempo de ataque de tais notas e permitir a identificação de consultas transpostas em tons diferentes, dentro de qualquer intervalo de valores de notas. Também é possível observar que grande parte dos métodos baseados em conjuntos considera a informação de duração e permite a correspondência parcial de uma consulta em uma composição. O método PROMS é o único dos métodos baseados em conjuntos que permite restrições no padrão de correspondência, através da definição de tamanho máximo de lacunas, e identificar correspondências com tom e oitavas exatas. Nenhum dos métodos estudados contempla, em sua definição ou estrutura, a possibilidade de eliminação de padrões melódicos, contidos nas consultas, do processo de avaliação de similaridade com composições.

3.2 Ferramentas para Busca de Composições Polifônicas

Esta seção descreve as ferramentas para busca de composições musicais polifônicas disponíveis na *web*, que trabalham com representação simbólica e implementam métodos de avaliação de similaridade melódica. Ao final da seção encontra-se uma análise comparativa entre as ferramentas.

3.2.1 C-Brahms

A ferramenta C-Brahms[1] (*Content-Based Retrieval and Analysis of Harmony and other Music Structures*) foi criada para testar e ilustrar métodos de busca de composições desenvolvidos em um projeto homônimo, iniciado em 2002 na Universidade de Helsinki (Finlândia) (LEMSTRÖM et al., 2003).

Estão disponíveis 9 métodos para busca de composições: *ShiftOrAnd*, *IntervalMatching*, *MonoPoly*, *Splitting*, *LCTS*, *P1*, *P2*, *P3*, e *PolyCheck*. Excluindo o método *PolyCheck*, todos outros encontram-se descritos na seção 3.1. A base de dados utilizada pela ferramenta C-Brahms possui 278 composições, todas extraídas da coleção Mutopia[2].

A interface da ferramenta C-Brahms, mostrada na Figura 3.3, é composta basicamente por controles para definição de parâmetros dos métodos, uma caixa de texto para a consulta, um teclado virtual para apoio à inserção de notas na consulta, e botões de ação. Uma consulta deve conter de duas a trinta notas, representadas em formato textual. Cada nota é composta por informações de duração, tom e oitava. A utilização de caracteres de soma (+) entre notas permite a representação de acordes em consultas polifônicas.

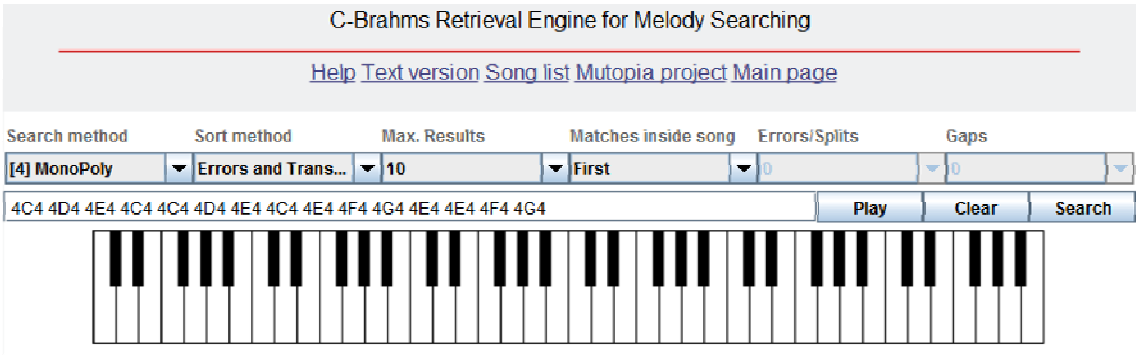


Figura 3.3: Interface da ferramenta C-Brahms.

Os resultados incluem nome do compositor, título da obra, *links* para os documentos contendo as composições recuperadas, compasso e notas correspondidas, e distância de transposição da melodia.

3.2.2 Kooplet

O Kooplet[3] é uma ferramenta gratuita para busca de documentos da *web* que contêm composições polifônicas. A empresa francesa Myriad Software, desenvolvedora da ferramenta, afirma que o Kooplet extrai informações diretamente do conteúdo dos arquivos de música disponíveis na *web*, possibilitando o acesso aos dados diretamente

[1] Disponível em http://www.cs.helsinki.fi/group/cbrahms/demoengine/

[2] Disponível em http://www.mutopiaproject.org

[3] Disponível em http://www.kooplet.com

da fonte original. Atualmente mais de 450 mil documentos estão indexadas pela ferramenta (MYRIAD, 2012).

Não estão disponibilizadas informações acerca dos métodos e algoritmos utilizados nas buscas da ferramenta.

Sua interface, mostrada na Figura 3.4, é formada basicamente por um editor de partituras com um teclado virtual, permitindo a inserção de informações de altura e duração de notas nas consultas. A ferramenta também possibilita a gravação e conversão de áudio para o editor de partituras, além de permitir a entrada de comandos especiais para determinar os sites e os tipos de documentos a serem pesquisados, e utilização de sentenças textuais a serem buscadas nas letras das composições. Somente consultas monofônicas podem ser utilizadas para buscas na ferramenta.

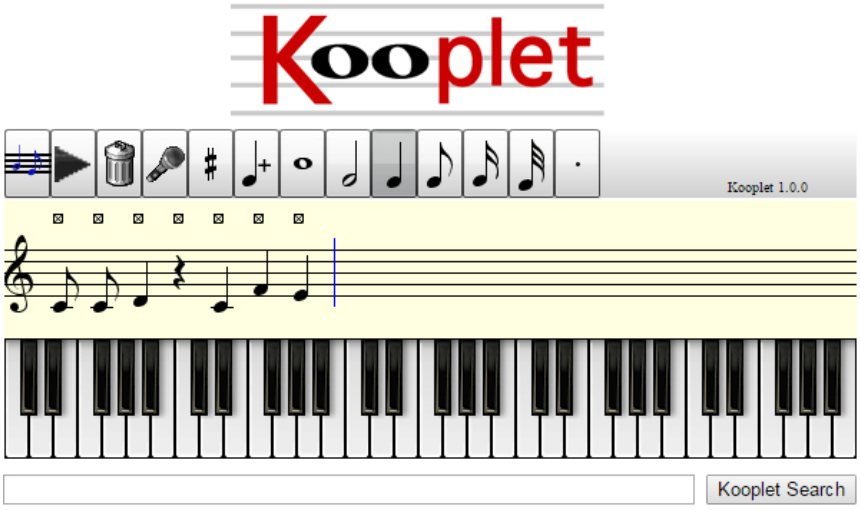


Figura 3.4: Interface da ferramenta Kooplet.

A listagem de resultados da ferramenta disponibiliza informações como autor e título da composição, tipo de arquivo e tamanho do documento, quantidade de compassos e notas da composição, duração da composição, *link* para acesso ao documento em seu repositório original, *link* para *download* do documento, e pré-visualização da partitura.

3.2.3 Meldex

O Meldex[4] (*Melody Index*) é uma ferramenta para recuperação de composições que trabalha em conjunto com o *framework* Greenstone[5], utilizado para construção e publicação de bibliotecas digitais. A ferramenta utiliza a base de dados denominada *folkfull*, da Biblioteca Digital da Nova Zelândia, contendo 9354 trechos de melodias das coleções Essen[6] e Digital Tradition[7] (MCNAB, 1997).

A ferramenta permite a representação de melodias através de intervalos ou contornos melódicos, e trabalha tanto com correspondência parcial, avaliando a similaridade de correspondências de intervalo e ritmo ou contorno e ritmo por distância

[4] Disponível em http://www.nzdl.org/fast-cgi-bin/music/musiclibrary/

[5] Disponível em http://www.greenstone.org

[6] Disponível em http://www.esac-data.org

[7] Disponível em http://mudcat.org

de edição, quanto com correspondência exata, procurando combinações de intervalo e ritmo, intervalo independente do ritmo, contorno e ritmo, ou contorno independente do ritmo.

É possível definir as consultas através de uma interface para gravação de áudio com o uso de um microfone (*Query by Humming* – QbH), ou por meio de uma interface para gravação da execução de notas em um teclado virtual, definido as durações de notas e pausas por meio de um controle específico. Em ambas as interfaces, a entrada gravada é transcrita para formato simbólico, além de não é possível a definição de consultas polifônicas. As interfaces de gravação de áudio (esquerda) e teclado virtual (direita) da ferramenta Meldex podem ser vistas na Figura 3.5.

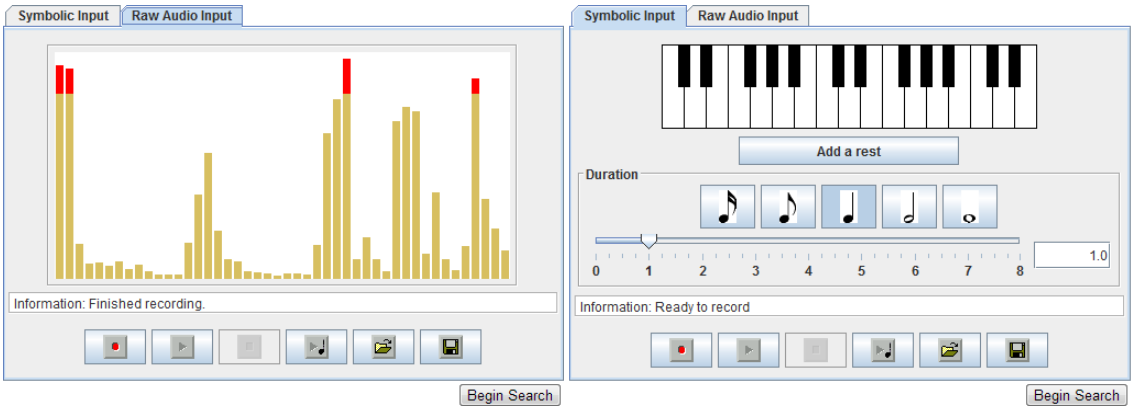


Figura 3.5: Interfaces da ferramenta Meldex.

Os resultados da ferramenta estão limitados aos primeiros 50 trechos com 90% ou mais de similaridade, nível pré-estabelecido na ferramenta para determinação de resultados relevantes. A listagem de resultados apresenta somente os títulos das composições, juntamente com a opção para reprodução do áudio.

3.2.4 Musipedia

A Musipedia[8], anteriormente chamada de TuneServer, é uma ferramenta onde usuários podem pesquisar, acrescentar e editar informações de uma base de dados de composições, composta por partituras, arquivos MIDI, textos e códigos de *Parsons* (contornos melódicos). A ferramenta também permite a busca de documentos publicados na *web* (ANGELUS, 2012).

A ferramenta trabalha com correspondência parcial e possibilita a pesquisa utilizando como entrada conjuntos de notas, contornos melódicos, ou somente informações rítmicas. Quando utilizando conjuntos de notas, a ferramenta utiliza o método TD-EMD, descrito na seção 3.1.2, que utiliza a distância de transporte *Earth Mover's Distance* para determinar a similaridade entre melodias. Pesquisas com contornos melódicos utilizam métodos de distância de edição, enquanto as que trabalham com informações rítmicas também utilizam o método TD-EMD, porém assumindo que todos os tons são iguais (TYPKE, 2004).

A Figura 3.6 mostra as diversas interfaces para busca de composições disponíveis na ferramenta, sendo elas:

[8] Disponível em http://www.musipedia.org

- Interfaces para entrada de conjunto de notas – piano virtual feito em *Flash* (a), piano virtual feito em *JavaScript* (b), *piano-roll* com interação via mouse (c), e entrada de áudio usando microfone (*Query by Humming*) (e);
- Interface para entrada de contorno (d);
- Interface para entrada de ritmo (*Query by Tapping* – QbT) (f).

As interfaces de piano também permitem a utilização de um controlador MIDI para a entrada das notas da consulta.

Todas as interfaces da ferramenta permitem incluir palavras-chave nas pesquisas juntamente com as outras informações.

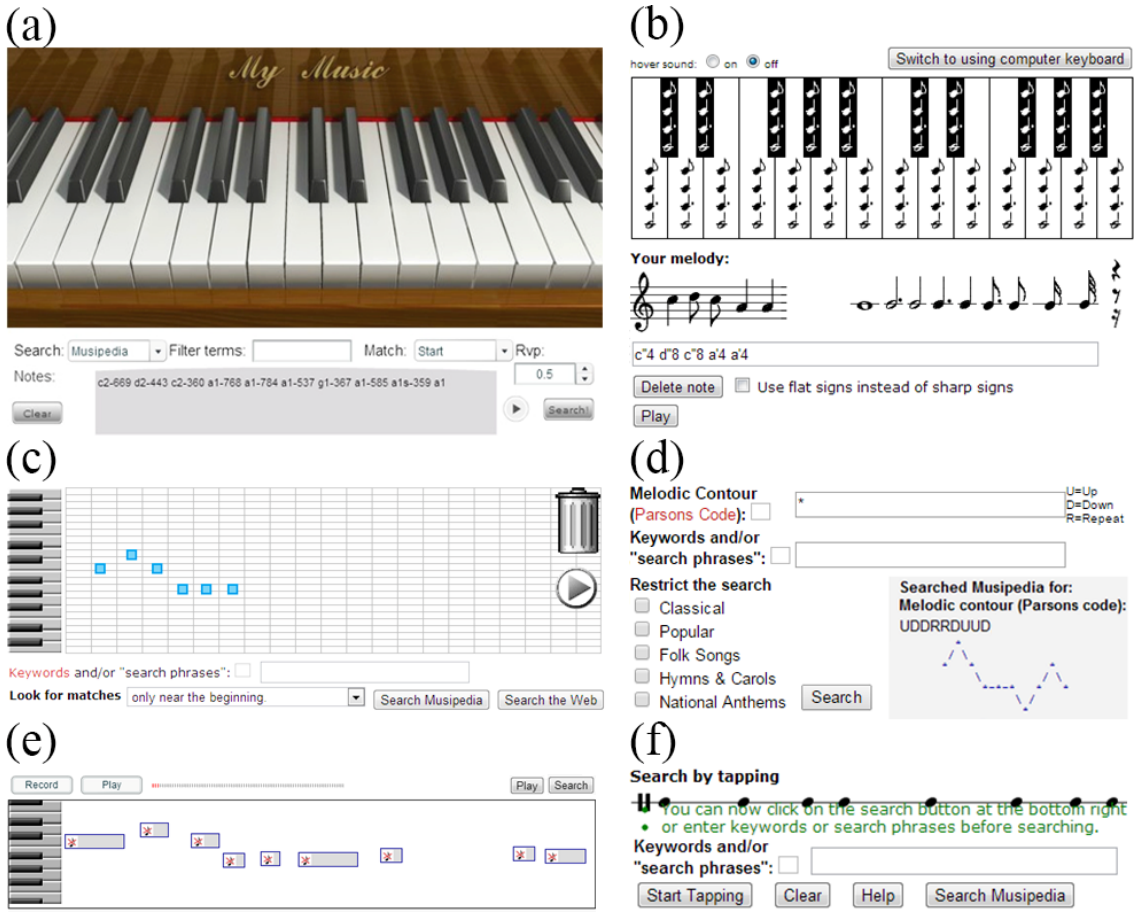


Figura 3.6: Interfaces da ferramenta Musipedia.

O formato da lista de resultados é independente da interface, porém depende do local consultado. Quando a pesquisa é feita na base de dados Musipedia, os resultados retornados apresentam informações de percentual de similaridade, tom original da composição, fonte e dados autorais, nome do compositor, título da obra, categoria, visualização gráfica da partitura, código *Parson* (contorno melódico), arquivo MIDI e *links* para vídeos da obra. Já quando a pesquisa é realizada na *web*, menos informações são disponibilizadas, sendo retornado o percentual de similaridade, nome e tamanho do arquivo, tempo do trecho onde a correspondência foi localizada, e *links* para o documento na *web*.

3.2.5 Comparação entre as Ferramentas Estudadas

Ao finalizar a busca e o estudo das ferramentas *web* para busca de composições polifônicas, foi realizado um comparativo, a fim de determinar quais características seriam relevantes para desenvolver uma ferramenta similar para demonstrar o método Correspondência de Monofonia Contida, proposto por este trabalho.

A Tabela 3.2 mostra informações das ferramentas estudadas neste capítulo, sendo elas: quantidade de interfaces disponíveis; se trabalha com uma base particular de composições; se permite realizar buscas de documentos da *web*; se permite a entrada e a busca por consultas polifônicas; se permite ajustes nos parâmetros de correspondência utilizados pelos métodos implementados; e se analisa composições contidas em documentos em formato de tablatura. Buscou-se apontar esta última característica, se a ferramenta analisa tablaturas, devido ter-se observado, durante os estudos, a dificuldade de buscar estes documentos com base em seu conteúdo musical.

Tabela 3.2: Comparativo entre ferramentas para busca de composições polifônicas.

#	Ferramenta	Interfaces	Base Particular	Busca na *Web*	Consulta Polifônica	Ajuste Params Corresp.	Analise Tablaturas
1	C-Brahms	1	X		X	X	
2	Kooplet	1		X			X
3	Meldex	2	X				
4	Musipedia	5	X	X	X		

Fica evidente na Tabela 3.2 que a ferramenta Musipedia se destaca na diversidade de interfaces disponíveis, aumentando a chance de satisfação dos usuários. A Musipedia também se destaca em relação aos locais de busca, possuindo uma base particular, construída colaborativamente por usuários, além de possibilitar a busca na *web*. As ferramentas C-Brahms e Meldex são as que analisam a menor quantidade de documentos, devido trabalharem somente com bases particulares de composições. Quanto às características relacionadas ao processo de busca, o C-Brahms se destaca, permitindo a utilização de consultas polifônicas e de ajuste de parâmetros de correspondência dos métodos. A única ferramenta que analisa o conteúdo de arquivos de tablatura é a Kooplet, porém é importante ressaltar que somente são analisados os arquivos onde a tablatura encontra-se pura (não textual), ou seja, não são analisados documentos que contenham tablaturas em meio a outros dados, como tablaturas textuais contidas em páginas HTML (*Hypertext Markup Language*).

4 CORRESPONDÊNCIA DE MONOFONIA CONTIDA

Este capítulo descreve os componentes e a forma de utilização do método proposto para busca de composições, denominado Correspondência de Monofonia Contida (CMC). Inicialmente, uma visão geral do método CMC é apresentada, seguida da definição de seus elementos básicos. Na sequência, encontram-se os detalhes do método, com a descrição de seus processos e fluxo de execução. Por fim, estão apresentadas as considerações finais, posicionando o método CMC em relação aos métodos apresentados nos trabalhos relacionados (seção 3.1).

4.1 Visão Geral

O método CMC tem como objetivo recuperar composições polifônicas, simbolicamente representadas, que possuam trechos monofônicos, contidos em sua melodia, similares aos de uma consulta, também polifônica.

Devido uma monofonia (melodia monofônica) poder ser transposta em qualquer tom, este trabalho a representa através de uma sequência de distâncias entre notas, formadas pela diferença de tempo e altura (em semitons) de duas notas. O tempo indica somente a ordenação de notas no eixo horizontal de uma melodia, não existindo relação com conceitos de duração ou andamento. A Figura 4.1 mostra uma das possíveis monofonias contidas em uma consulta, que também ocorre em uma composição.

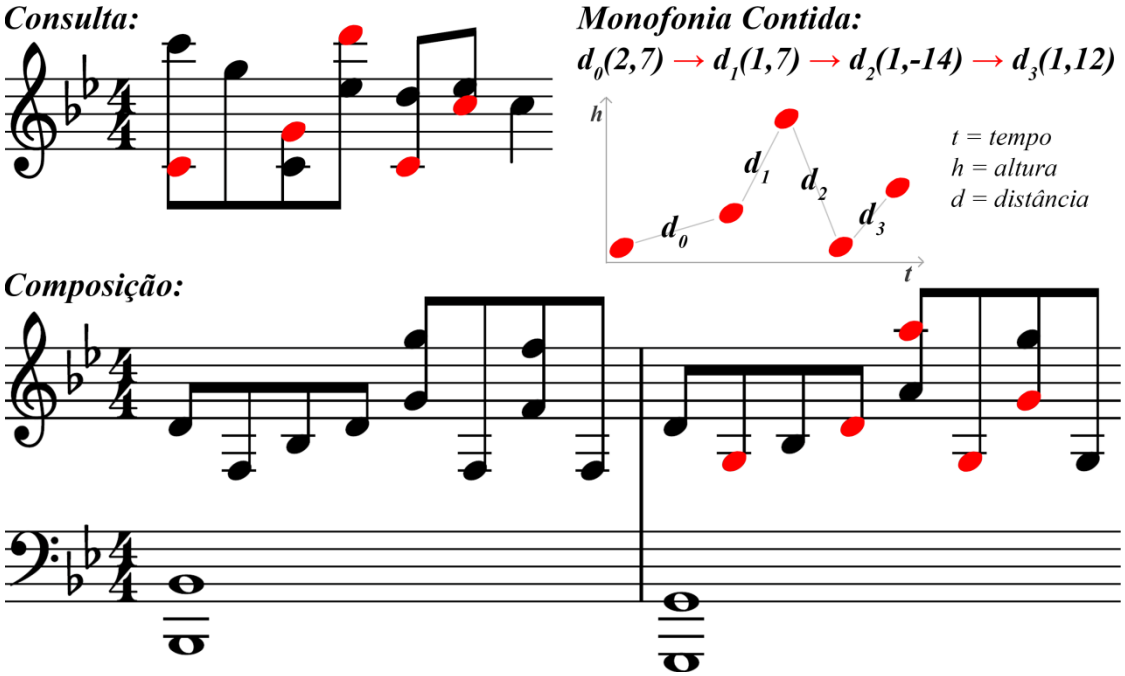


Figura 4.1: Consulta e composição com mesma monofonia contida.

Na Figura 4.1, tanto na consulta quanto na composição, estão destacadas (em vermelho) as notas que dão origem às distâncias que formam a monofonia contida em comum. Na consulta, a monofonia contida encontra-se entre a sequência de notas *C4*, *G4*, *D6*, *C4* e *C6*, enquanto na composição entre a sequência de notas *G3*, *D4*, *A5*, *G3* e *G4*. A primeira distância (d_1) da monofonia contida ocorre entre as notas *C4* e *G4* na consulta e entre *G3* e *D4* na composição, possuindo *2* como diferença de tempo e *7* como diferença de altura. A segunda distância entre notas (d_2) ocorre entre *G4* e *D6* na consulta e entre *D4* e *A5* na composição, com *1* para diferença de tempo e *7* para diferença de altura. A distância seguinte (d_3), entre as notas *D6* e *C4* da consulta e entre *A5* e *G3* da composição, contém *1* como diferença de tempo e *-14* para diferença de altura. A última distância da sequência (d_4) envolve as notas *C4* e *C6* da consulta e as notas *G3* e *G4* da composição, com diferença de tempo igual a *1* e diferença de altura igual a *12* (uma oitava acima). Para este exemplo, a composição será recuperada pelo método CMC, devido possuir uma monofonia contida em comum com a consulta.

4.2 Definições Preliminares

As definições apresentadas nesta seção formalizam os elementos básicos utilizados pelo método proposto.

Este trabalho utiliza um domínio de tempo *T* (equação 4.1), iniciando no valor *0*, referente ao eixo horizontal de uma melodia, a fim de posicionar cronologicamente suas notas. É importante ressaltar que este domínio *T* não tem relação com os conceitos de duração ou andamento, não devendo existir tempos sem notas (pausas).

$$T = \{x : 0 \leq x\} \qquad (4.1)$$

Já o eixo vertical está associado a um domínio de alturas *H* (equação 4.2), que utiliza valores do padrão MIDI, variando de *0* até *127*. A escolha do domínio de notas MIDI simplifica a representação e facilita a obtenção de informações de tom, oitava e distância entre notas.

$$H = \{x : 0 \leq x \leq 127\} \qquad (4.2)$$

Então, cada nota *v* (equação 4.3) de uma melodia corresponde a um ponto no plano formado pelos domínios tempo *T* e altura *H*. O tempo *0* não deve conter notas reais de uma melodia, devendo este tempo ser reservado para uma nota auxiliar do método, denominada nota falsa. A altura desta nota falsa sempre deve ser *0*.

$$v = (t, h) : (t, h) \in \{(0, 0)\} \cup \big((T - \{0\}) \times H\big) \qquad (4.3)$$

Uma melodia (equação 4.4) consiste de um conjunto de notas *v*, incluindo uma única nota falsa *(0,0)*.

$$V = \{v_0, v_1, \cdots, v_n\} : \exists! x \in V\big(x = (0, 0)\big) \qquad (4.4)$$

Uma função *D* (equação 4.5) foi definida para obtenção de uma distância entre duas notas v_a e v_b, gerando uma tupla formada pela diferença de tempo e pela diferença de altura das notas. Caso v_a seja uma nota falsa, a diferença de tempo será igual a *0*. A diferença de altura das notas é calculada em um intervalo de *r* valores, possibilitando não só calcular a diferença exata de alturas, quando *r* é igual a |*H*|, mas também a diferença desconsiderando oitavas, quando *r* é igual a *12*.

$$D(v_a, v_b, r) = \begin{cases} \big(0, \Pi_h(v_b \bmod r - v_a \bmod r)\big) & : v_a = (0,0) \\ \big(\Pi_t(v_b - v_a), \Pi_h(v_b \bmod r - v_a \bmod r)\big) & : v_a \neq (0,0) \end{cases} \quad (4.5)$$

A função *D* permite gerar uma distância entre notas, definida como *d* (equação 4.6), porém uma distância entre notas não necessariamente precisa ser gerada por esta função. Mesmo no caso de ser gerada pela função *D*, uma distância entre notas *d* é independente das notas que originaram seus valores.

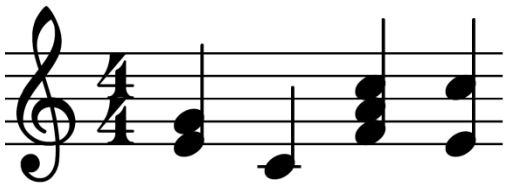
$$d = (\delta_t, \delta_h) \quad (4.6)$$

Devido uma monofonia (equação 4.7) poder ser transposta em qualquer tom, este trabalho a representa através de uma sequência progressiva de distâncias entre notas *d*. Somente a primeira distância entre notas (d_0) pode possuir diferença de tempo igual a *0*, que ocorre quando tal distância envolve uma nota falsa.

$$w = (d_0, d_1, \cdots, d_n) : \neg\exists x \in \{d_1, \cdots, d_n\}\big(\Pi_{\delta_t}(x) = 0\big) \quad (4.7)$$

Por utilizar distâncias entre notas *d*, uma monofonia *w* também é independente de notas. Porém, quando uma monofonia contém uma distância entre notas envolvendo a nota falsa, é possível descobrir as alturas de todas as notas que a originaram. A Figura 4.2 mostra uma melodia *V* juntamente com duas de suas monofonias contidas: w_0, sem envolver a nota falsa, e w_1 envolvendo a nota falsa.

Melodia:



$V = \{v_0(0,0),\ v_1(1,64),\ v_2(1,67),$
$v_3(2,60),\ v_4(3,65),\ v_5(3,69),$
$v_6(3,72),\ v_7(4,64),\ v_8(4,72)\}$

Monofonias Contidas (exemplos):

$w_0 = (d_1(1,-4),\ d_2(1,5))$

$w_1 = (d_0(0,64),\ d_1(1,-4),\ d_2(1,5))$

Notas da Melodia no plano T × H:

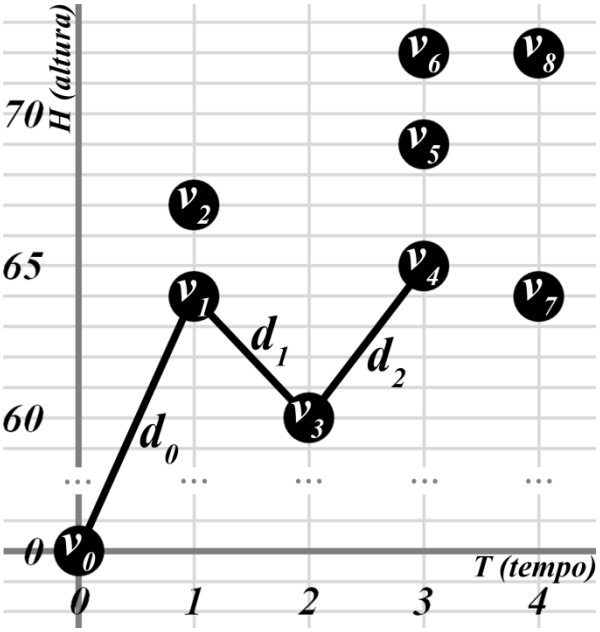


Figura 4.2: Melodia e monofonias contidas.

A monofonia contida w_0 da Figura 4.2 mostra que a diferença de altura das duas primeiras notas envolvidas é *-4* e entre a segunda e a terceira é *5*, porém não é possível saber a altura exata dessas notas. Já a monofonia w_1, que contém uma distância envolvendo a nota falsa, identificada através da diferença de tempo igual a *0*, permite saber que a altura exata da primeira nota é *64*, devido a altura da nota falsa sempre ser *0*, e que as alturas das notas seguintes envolvidas são *60* e *65*.

4.3 Descrição do Método

Esta seção descreve detalhes sobre o método Correspondência de Monofonia Contida (CMC), proposto por este trabalho.

O objetivo do método é recuperar composições, previamente armazenadas, que contenham linhas melódicas monofônicas similares às contidas em uma consulta.

O método é composto por uma série de processos, divididos em dois fluxos:

- Fluxo de Armazenamento de Composições – tem como objetivo representar e armazenar composições como Autômatos Finitos Não Determinísticos com Movimentos Vazios (AFNε), utilizados pelo Fluxo de Busca de Composições na verificação de similaridade com consultas;
- Fluxo de Busca de Composições – seu objetivo é gerar um conjunto de monofonias contidas em uma consulta, representada como um Grafo Acíclico Dirigido (GAD), e utilizá-las na verificação de similaridade junto às composições armazenadas, na forma de AFNs, pelo Fluxo de Armazenamento de Composições.

A Figura 4.3 apresenta os processos que compõem cada um dos dois fluxos do método proposto.

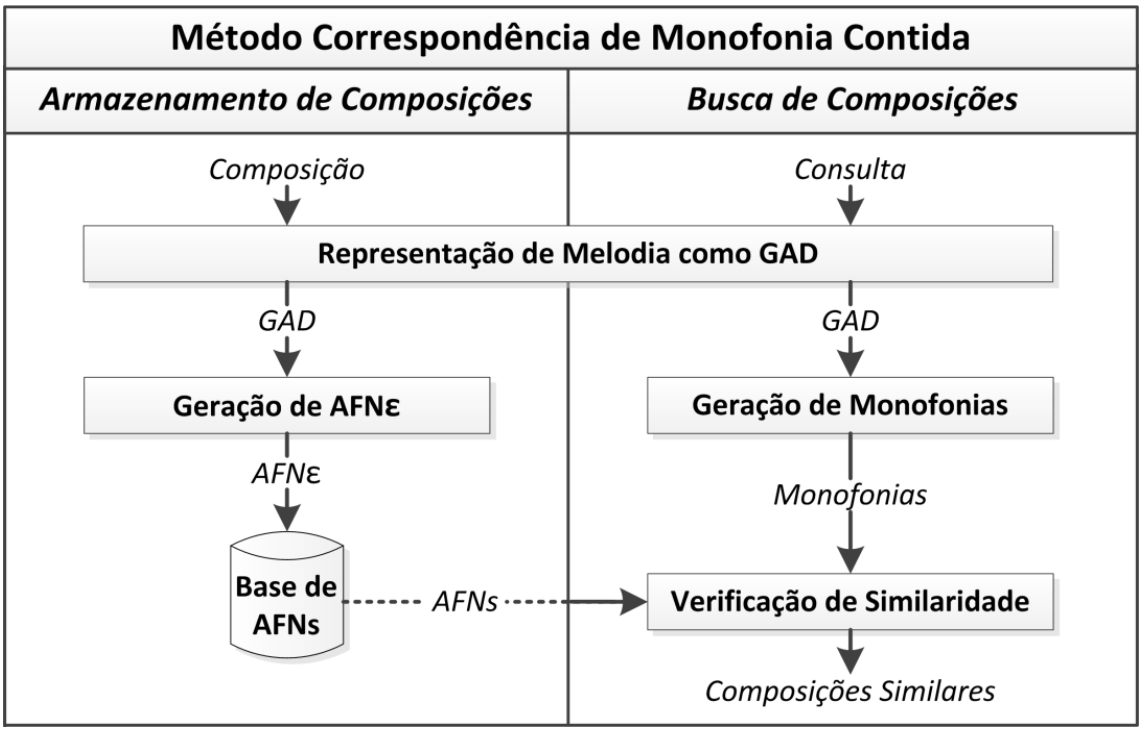


Figura 4.3: Fluxos do Método Correspondência de Monofonia Contida.

O fluxo de Armazenamento de Composições, mostrado na Figura 4.3, inicia pelo processo Representação de Melodia como GAD, que tem como entrada uma composição e como saída sua representação como um GAD. Em seguida, o processo Geração de AFNε utiliza o GAD recebido como entrada para gerar um AFNε, e o armazena em uma Base de AFNs. O fluxo de Busca de Composições também inicia pelo processo de Representação de Melodia como GAD, porém tendo como entrada uma consulta. O processo Geração de Monofonias utiliza o GAD recebido como entrada para gerar um conjunto de monofonias. O último processo, Verificação de Similaridade,

consiste em validar cada uma das monofonias geradas pelo processo de Geração de Monofonias, com cada um dos AFNs armazenados na Base de AFNs. Quando um AFNε, associado a uma composição, aceitar uma monofonia, associada a uma consulta, considera-se que a composição é similar à consulta. Um conjunto de composições similares é retornado pelo processo de Verificação de Similaridade.

As subseções seguintes detalham cada um dos processos definidos para o método Correspondência de Monofonia Contida.

4.3.1 Representação de Melodia como GAD

O processo Representação de Melodia como GAD tem como objetivo representar uma melodia *V* como um Grafo Acíclico Dirigido (GAD), de modo que seja possível estabelecer relações entre suas notas. Este processo é utilizado pelos dois fluxos de execução do método, Armazenamento de Composições e Busca de Composições.

A utilização da representação de melodias como GADs se justifica pela simplicidade de sua estrutura de grafo, facilitando a formalização do método, e por suas características, acíclico e direcionado, que permitem estabelecer a ordem progressiva de elementos, como ocorre em uma melodia.

Um GAD consiste de uma tupla *G* (equação 4.8), formada por um conjunto de nodos *V* e um conjunto de arestas dirigidas *E*. Uma aresta consiste de uma tupla, formada por um nodo de origem e um nodo de destino. Para este trabalho, o conjunto de nodos *V* de um GAD *G* refere-se a uma melodia *V*.

$$G = (V, E) \qquad (4.8)$$

Uma aresta refere-se a uma relação *e* (equação 4.9) entre duas notas *v*, sendo v_a a origem e v_b o destino desta relação. As relações entre notas *e* permitem que distâncias entre notas *d* sejam calculadas pela função *D*.

$$e = (v_a, v_b) \qquad (4.9)$$

O conjunto de arestas *E* (equação 4.10), de um GAD *G*, deve conter todas as possíveis relações *e* entre notas diferentes presentes em *V*, onde a distância calculada por *D* esteja entre *0* e um limite *k*. É importante ressaltar que o conjunto *E* possui relações da nota falsa com cada uma das outras notas da melodia, devido a diferença de tempo gerada por *D* ser igual a *0* quando envolve a nota falsa. O conjunto *E* de relações serve como base para a obtenção de monofonias contidas em uma melodia *V*.

$$\begin{aligned} E = \Big\{ & e \in V \times V : \Pi_{v_a}(e) \neq \Pi_{v_b}(e) \wedge \\ & \Pi_{\delta_t}\Big(D\big(\Pi_{v_a}(e), \Pi_{v_b}(e), |H|\big)\Big) \in [0, k] \Big\} \end{aligned} \qquad (4.10)$$

O limite *k* (equação 4.11) é utilizado para representar a diferença de tempo máxima entre duas notas reais em uma relação *e*, devendo ser maior ou igual a *1* e menor do que o maior tempo entre as notas de *V*. O limite *k* não se aplica à relações envolvendo a nota falsa. Este limite *k* é importante para que algumas monofonias, consideradas irrelevantes para uma avaliação de similaridade melódica, contendo diferenças de tempo muito grandes, não sejam identificadas em uma melodia.

$$k \in \left[\max\left(\Pi_t(V)\right)\right] \qquad (4.11)$$

A fim de obter a quantidade de relações *e*, entre um tempo t_m e um tempo t_n, para determinado *k*, foi definida a função C_E (equação 4.12). A função basicamente consiste em multiplicar a quantidade de notas de cada tempo, entre t_m+1 e t_n, pelo somatório da quantidade de notas que estejam entre o tempo *k* anterior e o tempo anterior, além de incluir na contagem relações da nota falsa com cada outra nota presente em *V*.

$$C_E(t_m, t_n) = \sum_{i \in [t_m+1, t_n]} \left|V \cap (i \times H)\right| \sum_{j \in \left(\{0\} \cup [i-k, i-1]\right) \cap \left[t_m, t_n-1\right]} \left|V \cap (j \times H)\right| \qquad (4.12)$$

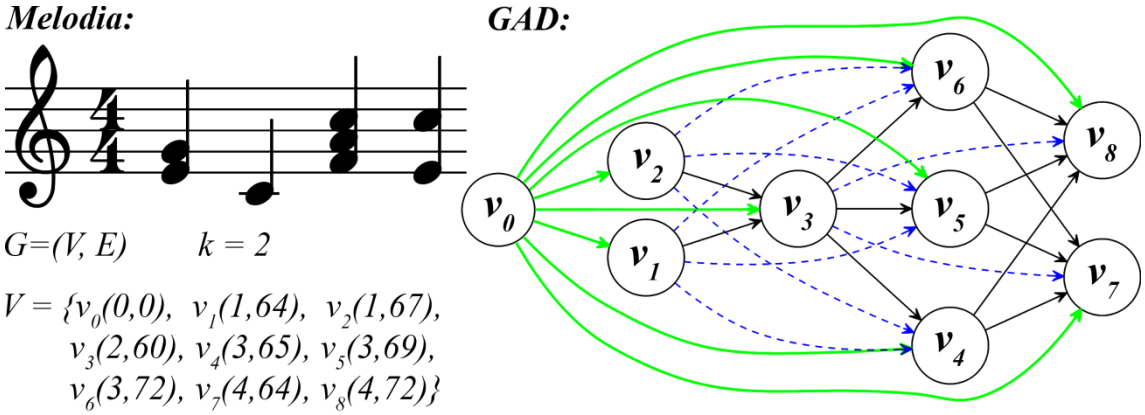
As relações *e*, para um conjunto *E* de um GAD *G*, são obtidas através da função *GeraRelacoes*, mostrada no Algoritmo 4.1.

Algoritmo 4.1 *GeraRelacoes*

```
1 : function GeraRelacoes(V, k)
2 :     var E
3 :     foreach v_a ∈ V do
4 :         foreach σ_(t>Π_t(v_a) ∧ (Π_t(v_a)=0 ∨ t≤Π_t(v_a)+k))(V) as v_b do
5 :             E ← E ∪ {(v_a, v_b)}
6 :         next
7 :     next
8 :     return E
9 : end function
```

A função *GeraRelacoes*, do Algoritmo 4.1, recebe como parâmetro uma melodia *V* e um valor para o limite *k* (linha 1). Uma variável representando um conjunto de arestas *E* é declarada logo no início da função (linha 2). Para cada nota em *V*, representada como v_a (linha 3), são encontradas todas outras notas, representadas como v_b, com tempo maior e, no caso do tempo de v_a não ser igual a *0* (nota falsa), somente àquelas notas que também tenham tempo menor ou igual ao tempo de v_a somado a *k* (linha 4). Então, a cada nota v_b encontrada para uma nota v_a, é adicionada uma relação *e* ao conjunto *E*, tendo como origem v_a e destino v_b (linha 5). Ao final da execução, a variável *E* é retornada (linha 8) contendo todas as possíveis relações entre notas de *V*, para determinado valor de *k*, incluindo relações da nota falsa com cada uma das outras notas.

A Figura 4.4 mostra uma melodia e seu respectivo GAD *G* contendo todas as relações para *k* igual *2*.



$G=(V, E) \qquad k = 2$

$V = \{v_0(0,0),\ v_1(1,64),\ v_2(1,67),$
$v_3(2,60),\ v_4(3,65),\ v_5(3,69),$
$v_6(3,72),\ v_7(4,64),\ v_8(4,72)\}$

$C_e(0,4) = 2.(1) + 1.(1 + 2) + 3.(1 + 2 + 1) + 2.(1 + 1 + 3) = 27$ *arestas*

$E = \{e_0(v_0,v_1), e_1(v_0,v_2), e_2(v_0,v_3), e_3(v_0,v_4), e_4(v_0,v_5), e_5(v_0,v_6), e_6(v_0,v_7), e_7(v_0,v_8), e_8(v_1,v_3),$
$e_9(v_1,v_4), e_{10}(v_1,v_5), e_{11}(v_1,v_6), e_{12}(v_2,v_3), e_{13}(v_2,v_4), e_{14}(v_2,v_5), e_{15}(v_2,v_6), e_{16}(v_3,v_4), e_{17}(v_3,v_5),$
$e_{18}(v_3,v_6), e_{19}(v_3,v_7), e_{20}(v_3,v_8), e_{21}(v_4,v_7), e_{22}(v_4,v_8), e_{23}(v_5,v_7), e_{24}(v_5,v_8), e_{25}(v_6,v_7), e_{26}(v_6,v_8)\}$

Figura 4.4: Melodia representada como GAD para limite k igual a 2.

No GAD da Figura 4.4, as arestas verdes grossas indicam relações envolvendo a nota falsa, que independem do limite *k*, já as pretas sólidas indicam relações com diferença de tempo igual a *1* e as azuis indicam relações com diferença de tempo igual a *2*. A Figura 4.4 também mostra o cálculo da função C_e, para obtenção da quantidade de arestas para *k* igual a *2*, resultando no valor 27.

4.3.2 Geração de AFNε

O objetivo do processo de Geração de AFNε é utilizar um GAD, representando uma composição, como base para geração de um Autômato Finito Não Determinístico com Movimentos Vazios (AFNε). O AFNε gerado é armazenado em uma Base de AFNs, sendo utilizado para validar monofonias *w*, a fim de determinar se tal monofonia encontra-se contida na composição.

Um AFNε consiste de uma quíntupla *M* (equação 4.13), formada por um alfabeto Σ, um conjunto finito de estados *q*, definido como *Q*, uma função de transição Δ, um estado inicial q_0, e um conjunto de estados finais *F*, subconjunto de *Q*.

$$M = (\Sigma, Q, \Delta, q_0, F) \qquad (4.13)$$

O alfabeto de entrada Σ (equação 4.14) de um AFNε consiste de um conjunto de distâncias entre notas *d*. O alfabeto Σ deve conter duas distâncias entre notas *d* para cada relação *e* do conjunto *E* do GAD *G*, uma dentro do intervalo |*H*| e outra em *12* (uma oitava), ambas geradas pela função *D*. Distâncias entre notas dentro do intervalo de uma oitava permitem que uma monofonia contida seja identificada em uma composição baseando-se somente na distância entre tons.

$$\begin{aligned}\Sigma = &\left\{D\left(\Pi_{v_a}(e), \Pi_{v_b}(e), |H|\right) : e \in \Pi_E(G)\right\} \\ &\cup \left\{D\left(\Pi_{v_a}(e), \Pi_{v_b}(e), 12\right) : e \in \Pi_E(G)\right\}\end{aligned} \qquad (4.14)$$

O conjunto *V* do GAD *G* é utilizado para o conjunto finito de estados possíveis *Q* (equação 4.15). Logo, um estado *q* possui a mesma estrutura que uma nota *v*, formada por um tempo *t* e uma altura *h*.

$$Q = \Pi_V(G) \qquad (4.15)$$

O estado inicial q_0 (equação 4.16) do AFNε sempre deve ser a nota falsa.

$$q_0 = (0, 0) \qquad (4.16)$$

Para formar o conjunto de estados finais F (equação 4.17), são utilizados todos os estados de Q, exceto q_0, para que seja possível encerrar a validação de uma monofonia em qualquer ponto do AFNε. Esta característica, de todos os estados associados a uma nota real serem finais, é importante, pois uma consulta, de onde são geradas monofonias, geralmente será menor que uma composição.

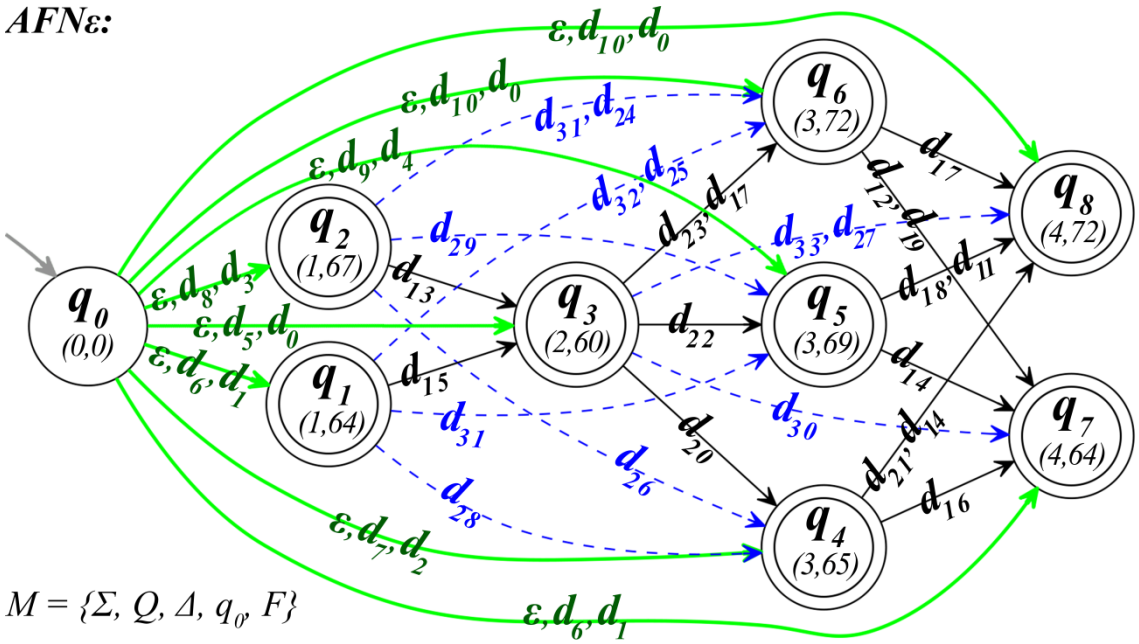
$$F = Q - \{q_0\} \qquad (4.17)$$

A função de transição Δ (equação 4.18) determina um conjunto de possíveis próximos estados para cada combinação entre estados q e elementos de Σ acrescido da palavra vazia ε. Na prática, as combinações são formadas por notas v da composição, representadas como estados q, e diferenças entre notas d, elementos de Σ, juntamente com a palavra vazia ε. Quando a combinação for formada pelo estado q_0 e pela palavra vazia ε, o conjunto de possíveis próximos estados será igual ao conjunto de estados Q, que inclui o próprio estado inicial q_0. Caso a combinação seja formada pelo estado q_0 e por uma distância entre notas d com diferença de tempo δ_t igual a 0, o conjunto de possíveis próximos estados será formado por todos os estados com altura h igual a diferença de altura δ_h de d. Já se a combinação for entre um estado q, diferente de q_0, e uma distância d, com diferença de tempo δ_t maior que 0, o conjunto de próximos estados será formado por um único estado com valores iguais ao resultado da soma entre as tuplas q e d.

$$\Delta(q, d) = \begin{cases} Q & : q = q_0 \wedge d = \varepsilon \\ \sigma_{\left(h=\Pi_{\delta_h}(d)\right)}(Q) & : q = q_0 \wedge \Pi_{\delta_t}(d) = 0 \\ \{q + d\} & : q \neq 0 \wedge \Pi_{\delta_t}(d) > 0 \end{cases} \qquad (4.18)$$

A Figura 4.5 mostra o AFNε gerado a partir do GAD da Figura 4.4.

A Figura 4.5 mostra uma representação gráfica do AFNε muito semelhante ao GAD da Figura 4.4, basicamente representando notas v como estados q, e relações e como transições definidas por Δ. Assim como no GAD, os padrões e cores das setas do AFNε servem simplesmente para facilitar a visualização das diferenças de tempo entre notas. O alfabeto Σ possui 34 distâncias entre notas d, parte calculada em um intervalo de $|H|$ (128) valores, parte calculada no intervalo 12 (uma oitava). Devido uma distância entre notas d ser independente de notas e o alfabeto Σ ser um conjunto, não distinguindo elementos com valores iguais, algumas distâncias entre notas d aparecem mais de uma vez na representação gráfica do AFNε, como d_{10}, que aparece entre os estados q_0 e q_6 e também entre q_0 e q_8.



$M = \{\Sigma, Q, \Delta, q_0, F\}$

$\Sigma=\{d_0(0,0), d_1(0,4), d_2(0,5), d_3(0,7), d_4(0,9), d_5(0,60), d_6(0,64), d_7(0,65), d_8(0,67), d_9(0,69), d_{10}(0,72), d_{11}(1,-9), d_{12}(1,-8), d_{13}(1,-7), d_{14}(1,-5), d_{15}(1,-4), d_{16}(1,-1), d_{17}(1,0), d_{18}(1,3), d_{19}(1,4), d_{20}(1,5), d_{21}(1,7), d_{22}(1,9), d_{23}(1,12), d_{24}(2,-7), d_{25}(2,-4), d_{26}(2,-2), d_{27}(2,0), d_{28}(2,1), d_{29}(2,2), d_{30}(2,4), d_{31}(2,5), d_{32}(2,8), d_{33}(2,12)\}$

$Q=\{q_0(0,0), q_1(1,64), q_2(1,67), q_3(2,60), q_4(3,65), q_5(3,69), q_6(3,72), q_7(4,64), q_8(4,72)\}$

$\Delta=\{(q_0,\varepsilon)\rightarrow\{q_1,q_2,q_3,q_4,q_5,q_6,q_7,q_8\}, (q_0,d_0)\rightarrow\{q_3,q_6,q_8\}, (q_0,d_1)\rightarrow\{q_1,q_7\}, (q_0,d_2)\rightarrow q_4, (q_0,d_3)\rightarrow q_2, (q_0,d_4)\rightarrow q_5, (q_0,d_5)\rightarrow q_3, (q_0,d_6)\rightarrow\{q_1,q_7\}, (q_0,d_7)\rightarrow q_4, (q_0,d_8)\rightarrow q_2, (q_0,d_9)\rightarrow q_5, (q_0,d_{10})\rightarrow\{q_6,q_8\}, (q_1,d_{15})\rightarrow q_3, (q_1,d_{25})\rightarrow q_6, (q_1,d_{28})\rightarrow q_4, (q_1,d_{31})\rightarrow q_5, (q_1,d_{32})\rightarrow q_6, (q_2,d_{13})\rightarrow q_3, (q_2,d_{24})\rightarrow q_6, (q_2,d_{26})\rightarrow q_4, (q_2,d_{29})\rightarrow q_5, (q_2,d_{31})\rightarrow q_6, (q_3,d_{17})\rightarrow q_6, (q_3,d_{20})\rightarrow q_4, (q_3,d_{22})\rightarrow q_5, (q_3,d_{23})\rightarrow q_6, (q_3,d_{27})\rightarrow q_8, (q_3,d_{30})\rightarrow q_7, (q_3,d_{33})\rightarrow q_8, (q_4,d_{14})\rightarrow q_8, (q_4,d_{16})\rightarrow q_7, (q_4,d_{21})\rightarrow q_8, (q_5,d_{11})\rightarrow q_8, (q_5,d_{14})\rightarrow q_7, (q_5,d_{18})\rightarrow q_8, (q_6,d_{12})\rightarrow q_7, (q_6,d_{17})\rightarrow q_8, (q_6,d_{20})\rightarrow q_7\}$

$q_0=(0,0)$

$F = \{q_1(1,64), q_2(1,67), q_3(2,60), q_4(3,65), q_5(3,69), q_6(3,72), q_7(4,64), q_8(4,72)\}$

Figura 4.5: AFNε gerado a partir de um GAD.

Assim como no GAD, o AFNε, mostrado na Figura 4.5, possui transições da nota falsa, representada pelo estado q_0, com cada uma das outras notas. O não determinismo do AFNε está somente relacionado às transições envolvendo a nota falsa e a palavra vazia ε, pois uma distância entre notas *d* sempre determinará um único novo estado *q*. Transições envolvendo a nota falsa (q_0) e a palavra vazia ε permitem que a aceitação de distâncias entre notas *d* de uma monofonia inicie em qualquer tempo da composição, representada pelo AFNε.

4.3.3 Geração de Monofonias

O processo de Geração de Monofonias tem como objetivo gerar um conjunto contendo todas as monofonias existentes em um GAD representando uma consulta. Um

conjunto de monofonias (equação 4.19) corresponde a um grupo de monofonias independentes, ou seja, monofonias que podem ou não estar contidas em uma melodia.

$$W = \{w_0, w_1, \cdots, w_n\} \qquad (4.19)$$

Duas funções foram definidas para o cálculo da quantidade de monofonias contidas em um GAD: C_w, que calcula a quantidade de monofonias contidas que iniciam em um tempo t_m e terminam em um tempo t_n; e a função C_W, que utiliza resultados de C_w para obter a quantidade total de monofonias contidas entre os tempos t_m e t_n.

Os resultados das funções C_w e C_W referem-se às quantidades de monofonias contidas com diferenças de altura em um único intervalo de valores, normalmente com $|H|$ valores, para diferenças de altura exatas, ou *12* valores, para diferenças de altura desconsiderando oitavas.

A função recursiva C_w (equação 4.20) realiza o somatório de produtos entre a quantidade de notas do tempo t_n e a quantidade de monofonias contidas, dada recursivamente por C_w, iniciando em t_m e terminando do tempo k anterior até o tempo anterior. Quando t_m for igual a *0* (nota falsa) ou a diferença entre t_n e t_m for menor ou igual a k, o resultado do produto entre a quantidade de notas do tempo t_n pela quantidade de notas do tempo t_m é somado ao resultado do somatório.

$$C_w(t_m, t_n) = \sum_{i \in [t_n - k, t_n - 1] \cap [t_m + 1, t_n - 1]} \left|V \cap (t_n \times H)\right| \cdot C_w(t_m, i) + [t_m = 0 \vee t_n - t_m \leq k] \cdot \left|V \cap (t_n \times H)\right| \cdot \left|V \cap (t_m \times H)\right| \qquad (4.20)$$

Já a função C_W (equação 4.21) soma os resultados da função recursiva C_w para cada intervalo possível entre os tempos t_m e t_n, obtendo a quantidade total de monofonias contidas neste intervalo de tempo. É importante ressaltar que a função C_W considera em sua contagem eventuais monofonias repetidas.

$$C_W(t_m, t_n) = \sum_{i \in [t_m, t_n - 1]} \sum_{j \in [i+1, t_n]} C_w(i, j) \qquad (4.21)$$

A Figura 4.6 demonstra os cálculos das funções C_W e C_w para a obtenção da quantidade de monofonias contidas no GAD da Figura 4.4.

Estão definidas duas funções para a obtenção de todas as monofonias contidas em um GAD: a função *MonofoniasNota*, mostrada no Algoritmo 4.2, que obtém todas as monofonias contidas que partam de uma nota *v* específica, ou seja, monofonias contidas que iniciem por distâncias entre notas *d* calculadas a partir de relações *e* com uma nota de origem v_a específica; e a função *GeraMonofonias*, mostrada no Algoritmo 4.3, que obtém todas as monofonias contidas em um GAD, sem quaisquer restrições. Ambas as funções obtém monofonias contidas compostas por distâncias entre notas *d*, geradas por *D*, com diferenças de altura δ_h em um intervalo de r valores. Quando o objetivo for recuperar composições para uma consulta, onde exista correspondência de monofonias contidas com diferenças de altura exatas, utiliza-se $|H|$ para valor de r. Para ignorar oitavas deve-se utilizar *12* como valor de r. Um conjunto de monofonias W pode conter monofonias contidas de ambos intervalos $|H|$ e *12*.

Algoritmo 4.2 *MonofoniasNota*

```
1 : function MonofoniasNota(E, v, ref w, r)
2 :     var W
3 :     foreach e ∈ σ_(v_a=v)(E) do
4 :         w ← (w, D(Π_{v_a}(e), Π_{v_b}(e), r))
5 :         W ← W ∪ {w} ∪ MonofoniasNota(E, Π_{v_b}(e), ref w, r)
6 :         w ← Π_{d_0..d_{n-1}}(w)
7 :     next
8 :     return W
9 : end function
```

A função *MonofoniasNota*, do Algoritmo 4.2, recebe como parâmetro (linha 1) o conjunto de relações *E* de um GAD, uma nota *v* que determinará de onde as monofonias contidas obtidas devem partir, uma monofonia *w* inicialmente vazia que será utilizada na recursão, e a quantidade *r* para o intervalo de diferenças de altura. Uma variável *W* é declarada para acumular as monofonias contidas obtidas (linha 2). Para cada relação *e* em *E* onde a nota de origem v_a for igual à nota *v* de parâmetro (linha 3) são realizadas as seguintes operações: uma distância entre notas, em um intervalo de altura de *r* valores, é gerada por *D* para a relação *e* corrente, e então adicionada ao final da monofonia *w* (linha 4); a variável *W* acumula a monofonia contida *w* juntamente com as monofonias contidas obtidas recursivamente por *MonofoniasNota*, tendo como parâmetros o conjunto de arestas *E*, a nota v_b de destino da relação *e*, a monofonia contida *w* por referência, e o valor *r* (linha 5); a última distância entre notas é removida de *w* (linha 6). Nota-se que *w* tem o comportamento de uma pilha no algoritmo, empilhando distâncias entre notas através da recursão (linha 4) e desempilhando ao final de cada iteração de cada execução (linha 6). Ao final da execução, a variável *W* é retornada com todas as monofonias contidas no GAD que partam da nota *v* passada por parâmetro (linha 8).

Algoritmo 4.3 *GeraMonofonias*

```
1 : function GeraMonofonias(G, r)
2 :     var W, w
3 :     foreach v ∈ Π_V(G) do
4 :         W ← W ∪ MonofoniasNota(Π_E(G), v, ref w, r)
5 :     next
6 :     return W
7 : end function
```

A função *GeraMonofonias* do Algoritmo 4.3 recebe por parâmetro um GAD *G* e um valor *r* para o cálculo das diferenças de altura (linha 1). Duas variáveis são declaradas no algoritmo (linha 2), sendo *W* para acumular as monofonias contidas obtidas e *w* para ser passada por referência nas chamadas da função *MonofoniasNota*. Para cada nota *v* do conjunto de notas *V* do GAD *G* (linha 3) é feita uma chamada à função *MonofoniasNota* (linha 4), mostrada no Algoritmo 4.2, acumulando as

monofonias contidas em *W*. Ao final da execução, a variável *W* é retornada com todas as monofonias contidas no GAD G (linha 6).

A Figura 4.7 mostra o cálculo das funções C_W e C_w, juntamente com todas as monofonias contidas no GAD da Figura 4.4 para *r* igual a |*H*|.

$$C_W(0,4)=(C_w(0,1)+C_w(0,2)+C_w(0,3)+C_w(0,4)) + (C_w(1,2)+C_w(1,3)+C_w(1,4)) + (C_w(2,3)+C_w(2,4)) + (C_w(3,4))$$
$$=(2+3+18+44) + (2+12+28) + (3+8)+(6)=126 \text{ monofonias contidas}$$

$C_w(0,1)=(0) + 2.1 = 2$
$C_w(0,2)=(1.C_w(0,1)) + 1.1 = 3$
$C_w(0,3)=(3.C_w(0,1)+3.C_w(0,2)) + 3.1 = 18$
$C_w(0,4)=(2.C_w(0,2)+2.C_w(0,3)) + 2.1 = 44$
$C_w(1,2)=(0) + 1.2 = 2$
$C_w(1,3)=(3.C_w(1,2)) + 3.2 = 12$
$C_w(1,4)=(2.C_w(1,2)+2.C_w(1,3)) + 0 = 28$
$C_w(2,3)=(0) + 3.1 = 3$
$C_w(2,4)=(2.C_w(2,3)) + 2.1 = 8$
$C_w(3,4)=(0) + 2.3 = 6$

$W=\{w_0((0,64)),w_1((0,64),(1,-4)),w_2((0,64),(1,-4),(1,5)),w_3((0,64),(1,-4),(1,5),(1,-1)),w_4((0,64),(1,-4),(1,5),(1,7)),$
$w_5((0,64),(1,-4),(1,9)),w_6((0,64),(1,-4),(1,9),(1,-5)),w_7((0,64),(1,-4),(1,9),(1,3)),w_8((0,64),(1,-4),(1,12)),$
$w_9((0,64),(1,-4),(1,12),(1,-8)),w_{10}((0,64),(1,-4),(1,12),(1,0)),w_{11}((0,64),(1,-4),(2,4)),w_{12}((0,64),(1,-4),(2,12)),$
$w_{13}((0,64),(2,1)),w_{14}((0,64),(2,1),(1,-1)),w_{15}((0,64),(2,1),(1,7)),w_{16}((0,64),(2,5)),w_{17}((0,64),(2,5),(1,-5)),$
$w_{18}((0,64),(2,5),(1,3)),w_{19}((0,64),(2,8)),w_{20}((0,64),(2,8),(1,-8)),w_{21}((0,64),(2,8),(1,0)),w_{22}((0,67)),w_{23}((0,67),(1,-7)),$
$w_{24}((0,67),(1,-7),(1,5)),w_{25}((0,67),(1,-7),(1,5),(1,-1)),w_{26}((0,67),(1,-7),(1,5),(1,7)),w_{27}((0,67),(1,-7),(1,9)),$
$w_{28}((0,67),(1,-7),(1,9),(1,-5)),w_{29}((0,67),(1,-7),(1,9),(1,3)),w_{30}((0,67),(1,-7),(1,12)),w_{31}((0,67),(1,-7),(1,12),(1,-8)),$
$w_{32}((0,67),(1,-7),(1,12),(1,0)),w_{33}((0,67),(1,-7),(2,4)),w_{34}((0,67),(1,-7),(2,12)),w_{35}((0,67),(2,-2)),$
$w_{36}((0,67),(2,-2),(1,-1)),w_{37}((0,67),(2,-2),(1,7)),w_{38}((0,67),(2,2)),w_{39}((0,67),(2,2),(1,-5)),w_{40}((0,67),(2,2),(1,3)),$
$w_{41}((0,67),(2,5)),w_{42}((0,67),(2,5),(1,-8)),w_{43}((0,67),(2,5),(1,0)),w_{44}((0,60)),w_{45}((0,60),(1,5)),w_{46}((0,60),(1,5),(1,-1)),$
$w_{47}((0,60),(1,5),(1,7)),w_{48}((0,60),(1,9)),w_{49}((0,60),(1,9),(1,-5)),w_{50}((0,60),(1,9),(1,3)),w_{51}((0,60),(1,12)),$
$w_{52}((0,60),(1,12),(1,-8)),w_{53}((0,60),(1,12),(1,0)),w_{54}((0,60),(2,4)),w_{55}((0,60),(2,12)),w_{56}((0,65)),w_{57}((0,65),(1,-1)),$
$w_{58}((0,65),(1,7)),w_{59}((0,69)),w_{60}((0,69),(1,-5)),w_{61}((0,69),(1,3)),w_{62}((0,72)),w_{63}((0,72),(1,-8)),w_{64}((0,72),(1,0)),$
$w_{65}((0,64)),w_{66}((0,72)),w_{67}((1,-4)),w_{68}((1,-4),(1,5)),w_{69}((1,-4),(1,5),(1,-1)),w_{70}((1,-4),(1,5),(1,7)),w_{71}((1,-4),(1,9)),$
$w_{72}((1,-4),(1,9),(1,-5)),w_{73}((1,-4),(1,9),(1,3)),w_{74}((1,-4),(1,12)),w_{75}((1,-4),(1,12),(1,-8)),w_{76}((1,-4),(1,12),(1,0)),$
$w_{77}((1,-4),(2,4)),w_{78}((1,-4),(2,12)),w_{79}((2,1)),w_{80}((2,1),(1,-1)),w_{81}((2,1),(1,7)),w_{82}((2,5)),w_{83}((2,5),(1,-5)),$
$w_{84}((2,5),(1,3)),w_{85}((2,8)),w_{86}((2,8),(1,-8)),w_{87}((2,8),(1,0)),w_{88}((1,-7)),w_{89}((1,-7),(1,5)),w_{90}((1,-7),(1,5),(1,-1)),$
$w_{91}((1,-7),(1,5),(1,7)),w_{92}((1,-7),(1,9)),w_{93}((1,-7),(1,9),(1,-5)),w_{94}((1,-7),(1,9),(1,3)),w_{95}((1,-7),(1,12)),$
$w_{96}((1,-7),(1,12),(1,-8)),w_{97}((1,-7),(1,12),(1,0)),w_{98}((1,-7),(2,4)),w_{99}((1,-7),(2,12)),w_{100}((2,-2)),w_{101}((2,-2),(1,-1)),$
$w_{102}((2,-2),(1,7)),w_{103}((2,2)),w_{104}((2,2),(1,-5)),w_{105}((2,2),(1,3)),w_{106}((2,5)),w_{107}((2,5),(1,-8)),w_{108}((2,5),(1,0)),$
$w_{109}((1,5)),w_{110}((1,5),(1,-1)),w_{111}((1,5),(1,7)),w_{112}((1,9)),w_{113}((1,9),(1,-5)),w_{114}((1,9),(1,3)),w_{115}((1,12)),$
$w_{116}((1,12),(1,-8)),w_{117}((1,12),(1,0)),w_{118}((2,4)),w_{119}((2,12)),w_{120}((1,-1)),w_{121}((1,7)),w_{122}((1,-5)),w_{123}((1,3)),$
$w_{124}((1,-8)),w_{125}((1,0))\}$

Figura 4.7: Cálculo da quantidade e monofonias contidas em um GAD.

O cálculo de C_W, mostrado na Figura 4.7, resulta na quantidade de 126 monofonias contidas no GAD da Figura 4.4. Todas estas 126 monofonias contidas também são mostradas no conjunto *W*.

4.3.3.1 Descarte de Monofonias

Algumas das monofonias obtidas de um GAD, representando uma consulta, podem não ser relevantes para uma determinada busca de composições. O objetivo do descarte de monofonias é eliminar, de um conjunto *W*, àquelas monofonias que não estejam de acordo com determinados critérios de relevância. Estão previstos para o método três critérios de relevância de monofonias:

1. Percentual Mínimo de Notas – são consideradas relevantes àquelas monofonias que envolverem um percentual mínimo de notas reais da consulta, em relação à maior quantidade possível de notas envolvidas para uma monofonia contida em tal consulta. A maior quantidade possível de notas envolvidas para uma monofonia contida é igual ao maior tempo da consulta (t_n);

2. Ausência de Saltos Consecutivos – uma monofonia contida é considerada relevante se não existirem saltos consecutivos, ou seja, se não existirem distâncias entre notas consecutivas com diferenças de tempo maiores que *1*. Somente consultas com tempo final maior que *4* contém monofonias com saltos consecutivos;
3. Alturas Exatas – são consideradas relevantes monofonias que possibilitem a recuperação de composições com alturas de notas exatas em relação à consulta. Somente monofonias que iniciem com uma distância entre notas envolvendo a nota falsa atendem a este critério.

Os dois primeiros critérios, Percentual Mínimo de Notas e Ausência de Saltos Consecutivos, visam auxiliar no descarte de monofonias que, mesmo contidas, estejam descaracterizadas melodicamente em relação à melodia da consulta.

O Algoritmo 4.4 mostra a função *DescarteMinimoNotas*, que elimina monofonias de acordo com o critério Percentual Mínimo de Notas.

Algoritmo 4.4 *DescarteMinimoNotas*

```
1 :  function DescarteMinimoNotas(W, p, t_n)
2 :      var W_aux, quant
3 :      foreach w ∈ W do
4 :          quant ← |σ_(Π_δt(d)≠0)(w)| + 1
5 :          if quant/t_n ≥ p then
6 :              W_aux ← W_aux ∪ {w}
7 :          end if
8 :      next
9 :      return W_aux
10 : end function
```

A função *DescarteMinimoNotas*, do Algoritmo 4.4, recebe como parâmetro um conjunto de monofonias W, um valor p para percentual mínimo de notas, e o maior tempo da consulta em t_n (linha 1). São declaradas as variáveis W_{aux}, para receber as monofonias de W que atendem ao critério de Percentual Mínimo de Notas, e *quant*, utilizada para armazenar a quantidade de notas envolvidas em cada monofonia presente em W (linha 2). Para cada monofonia w em W (linha 3), a variável *quant* recebe a quantidade de distâncias entre notas d com diferença de tempo δ_t diferente de *0* somado a *1* (linha 4), e, se o resultado da divisão entre *quant* e t_n for maior ou igual ao percentual p (linha 5), a variável W_{aux} acumula a monofonia w corrente (linha 6). Ao final da execução, o conjunto W_{aux}, contendo somente as monofonias que atendem ao critério de Percentual Mínimo de Notas, é retornado (linha 9).

Para o critério Ausência de Saltos Consecutivos, foi definida a função *DescarteSaltos*, mostrada no Algoritmo 4.5.

Algoritmo 4.5 *DescarteSaltos*

```
1 :  function DescarteSaltos(W)
2 :      var W_aux, saltos
3 :      foreach w ∈ σ_(|w|>1)(W) do
4 :          saltos ← false
5 :          for i ← 0 to |w| − 2 step 1 do
6 :              if Π_δt(Π_di(w)) > 1 and Π_δt(Π_di+1(w)) > 1 then
7 :                  saltos ← true
8 :              end if
9 :          next
10 :         if saltos = false then
11 :             W_aux ← W_aux ∪ {w}
12 :         end if
13 :     next
14 :     return W_aux
15 : end function
```

O Algoritmo 4.5 mostra a função *DescarteSaltos*, que recebe um único parâmetro, um conjunto de monofonias *W* (linha 1). Estão declaradas as variáveis W_{aux}, para receber as monofonias que atendem ao critério de Ausência de Saltos Consecutivos, e *saltos*, utilizada para armazenar o valor lógico se uma monofonia contém saltos consecutivos (linha 2). Para cada monofonia *w* com mais de uma distância entre notas (linha 3) são realizadas as seguintes operações: atribuição do valor lógico falso à variável *saltos* (linha 4); são percorridas todas as distâncias entre notas da monofonia *w* corrente (linha 5), verificando se existem diferenças de tempo δ_t consecutivas com valores maiores que *1* (linha 6), e, caso existam, *saltos* recebe o valor lógico verdadeiro (linha 7); em seguida, é verificado se *saltos* possui o valor falso, ou seja, se não contém saltos consecutivos (linha 10), e, neste caso, W_{aux} acumula a monofonia *w* corrente (linha 11). No final, W_{aux} é retornado contendo todas as monofonias que atendam ao critério de Ausência de Saltos Consecutivos (linha 14).

O terceiro e último critério, denominado Alturas Exatas, possui uma execução simples através da função *DescarteAlturas*, mostrada no Algoritmo 4.6.

Algoritmo 4.6 *DescarteAlturas*

```
1 :  function DescarteAlturas(W)
2 :      return σ_(Π_δt(Π_d0(w))=0)(W)
3 :  end function
```

A função *DescarteAlturas*, do Algoritmo 4.6, recebe como parâmetro um conjunto de monofonias *W* (linha 1) e retorna um conjunto contendo somente àquelas monofonias *w* que envolvam a nota falsa (linha 2).

A Figura 4.8 mostra resultados da execução sequencial das funções *DescarteMinimoNotas* e *DescarteAlturas*, para *W* da Figura 4.7.

W* ← *DescarteMinimoNotas(W, 0.75, 4)

80 Retornadas: *W={w2((0,64),(1,-4),(1,5)),w3((0,64),(1,-4),(1,5),(1,-1)),w4((0,64),(1,-4),(1,5),(1,7)), w5((0,64),(1,-4),(1,9)),w6((0,64),(1,-4),(1,9),(1,-5)),w7((0,64),(1,-4),(1,9),(1,3)),w8((0,64),(1,-4),(1,12)), w9((0,64),(1,-4),(1,12),(1,-8)),w10((0,64),(1,-4),(1,12),(1,0)),w11((0,64),(1,-4),(2,4)),w12((0,64),(1,-4),(2,12)), w14((0,64),(2,1),(1,-1)),w15((0,64),(2,1),(1,7)), w17((0,64),(2,5),(1,-5)),w18((0,64),(2,5),(1,3)), w20((0,64),(2,8),(1,-8)), w21((0,64),(2,8),(1,0)), w24((0,67),(1,-7),(1,5)),w25((0,67),(1,-7),(1,5),(1,-1)),w26((0,67),(1,-7),(1,5),(1,7)), w27((0,67),(1,-7),(1,9)),w28((0,67),(1,-7),(1,9),(1,-5)),w29((0,67),(1,-7),(1,9),(1,3)),w30((0,67),(1,-7),(1,12)), w31((0,67),(1,-7),(1,12),(1,-8)),w32((0,67),(1,-7),(1,12),(1,0)),w33((0,67),(1,-7),(2,4)),w34((0,67),(1,-7),(2,12)), w36((0,67),(2,-2),(1,-1)),w37((0,67),(2,-2),(1,7)), w39((0,67),(2,2),(1,-5)),w40((0,67),(2,2),(1,3)),w42((0,67),(2,5),(1,-8)), w43((0,67),(2,5),(1,0)), w46((0,60),(1,5),(1,-1)),w47((0,60),(1,5),(1,7)), w49((0,60),(1,9),(1,-5)),w50((0,60),(1,9),(1,3)), w52((0,60),(1,12),(1,-8)),w53((0,60),(1,12),(1,0)),w68((1,-4),(1,5)), w69((1,-4),(1,5),(1,-1)),w70((1,-4),(1,5),(1,7)), w71((1,-4),(1,9)),w72((1,-4),(1,9),(1,-5)),w73((1,-4),(1,9),(1,3)),w74((1,-4),(1,12)),w75((1,-4),(1,12),(1,-8)), w76((1,-4),(1,12),(1,0)),w77((1,-4),(2,4)),w78((1,-4),(2,12)), w80((2,1),(1,-1)),w81((2,1),(1,7)), w83((2,5),(1,-5)), w84((2,5),(1,3)),w86((2,8),(1,-8)),w87((2,8),(1,0)), w89((1,-7),(1,5)),w90((1,-7),(1,5),(1,-1)),w91((1,-7),(1,5),(1,7)), w92((1,-7),(1,9)),w93((1,-7),(1,9),(1,-5)),w94((1,-7),(1,9),(1,3)),w95((1,-7),(1,12)),w96((1,-7),(1,12),(1,-8)), w97((1,-7),(1,12),(1,0)),w98((1,-7),(2,4)),w99((1,-7),(2,12)), w101((2,-2),(1,-1)),w102((2,-2),(1,7)), w104((2,2),(1,-5)), w105((2,2),(1,3)),w107((2,5),(1,-8)),w108((2,5),(1,0)),w110((1,5),(1,-1)),w111((1,5),(1,7)), w113((1,9),(1,-5)), w114((1,9),(1,3)),w116((1,12),(1,-8)),w117((1,12),(1,0))}*

46 Descartadas: *{w0((0,64)), w1((0,64),(1,-4)), w13((0,64),(2,1)), w16((0,64),(2,5)), w19((0,64),(2,8)), w22((0,67)), w23((0,67),(1,-7)),w35((0,67),(2,-2)), w38((0,67),(2,2)), w41((0,67),(2,5)), w44((0,60)),w45((0,60),(1,5)), w48((0,60),(1,9)), w51((0,60),(1,12)), w54((0,60),(2,4)),w55((0,60),(2,12)),w56((0,65)),w57((0,65),(1,-1)),w58((0,65),(1,7)),w59((0,69)), w60((0,69),(1,-5)),w61((0,69),(1,3)),w62((0,72)),w63((0,72),(1,-8)),w64((0,72),(1,0)),w65((0,64)),w66((0,72)),w67((1,-4)), w79((2,1)),w82((2,5)),w85((2,8)), w88((1,-7)), w100((2,-2)), w103((2,2)), w106((2,5)), w109((1,5)), w112((1,9)), w115((1,12)), w118((2,4)), w119((2,12)),w120((1,-1)),w121((1,7)),w122((1,-5)),w123((1,3)),w124((1,-8)),w125((1,0))}*

W* ← *DescarteAlturas(W)

40 Retornadas: *W={w2((0,64),(1,-4),(1,5)),w3((0,64),(1,-4),(1,5),(1,-1)),w4((0,64),(1,-4),(1,5),(1,7)), w5((0,64),(1,-4),(1,9)),w6((0,64),(1,-4),(1,9),(1,-5)),w7((0,64),(1,-4),(1,9),(1,3)),w8((0,64),(1,-4),(1,12)), w9((0,64),(1,-4),(1,12),(1,-8)),w10((0,64),(1,-4),(1,12),(1,0)),w11((0,64),(1,-4),(2,4)),w12((0,64),(1,-4),(2,12)), w14((0,64),(2,1),(1,-1)),w15((0,64),(2,1),(1,7)),w17((0,64),(2,5),(1,-5)),w18((0,64),(2,5),(1,3)), w20((0,64),(2,8),(1,-8)), w21((0,64),(2,8),(1,0)), w24((0,67),(1,-7),(1,5)),w25((0,67),(1,-7),(1,5),(1,-1)),w26((0,67),(1,-7),(1,5),(1,7)), w27((0,67),(1,-7),(1,9)),w28((0,67),(1,-7),(1,9),(1,-5)),w29((0,67),(1,-7),(1,9),(1,3)),w30((0,67),(1,-7),(1,12)), w31((0,67),(1,-7),(1,12),(1,-8)),w32((0,67),(1,-7),(1,12),(1,0)),w33((0,67),(1,-7),(2,4)),w34((0,67),(1,-7),(2,12)), w36((0,67),(2,-2),(1,-1)),w37((0,67),(2,-2),(1,7)), w39((0,67),(2,2),(1,-5)),w40((0,67),(2,2),(1,3)),w42((0,67),(2,5),(1,-8)), w43((0,67),(2,5),(1,0)), w46((0,60),(1,5),(1,-1)),w47((0,60),(1,5),(1,7)), w49((0,60),(1,9),(1,-5)),w50((0,60),(1,9),(1,3)), w52((0,60),(1,12),(1,-8)),w53((0,60),(1,12),(1,0))}*

40 Descartadas: *{w68((1,-4),(1,5)),w69((1,-4),(1,5),(1,-1)),w70((1,-4),(1,5),(1,7)),w71((1,-4),(1,9)), w72((1,-4),(1,9),(1,-5)),w73((1,-4),(1,9),(1,3)),w74((1,-4),(1,12)),w75((1,-4),(1,12),(1,-8)),w76((1,-4),(1,12),(1,0)), w77((1,-4),(2,4)),w78((1,-4),(2,12)),w80((2,1),(1,-1)),w81((2,1),(1,7)), w83((2,5),(1,-5)),w84((2,5),(1,3)), w86((2,8),(1,-8)), w87((2,8),(1,0)), w89((1,-7),(1,5)),w90((1,-7),(1,5),(1,-1)),w91((1,-7),(1,5),(1,7)),w92((1,-7),(1,9)),w93((1,-7),(1,9),(1,-5)), w94((1,-7),(1,9),(1,3)),w95((1,-7),(1,12)),w96((1,-7),(1,12),(1,-8)),w97((1,-7),(1,12),(1,0)),w98((1,-7),(2,4)), w99((1,-7),(2,12)),w101((2,-2),(1,-1)),w102((2,-2),(1,7)), w104((2,2),(1,-5)),w105((2,2),(1,3)), w107((2,5),(1,-8)), w108((2,5),(1,0)),w110((1,5),(1,-1)),w111((1,5),(1,7)),w113((1,9),(1,-5)),w114((1,9),(1,3)),w116((1,12),(1,-8)), w117((1,12),(1,0))}*

Figura 4.8: Exemplo para as funções DescarteMinimoNotas e DescarteAlturas.

Na Figura 4.8, a função *DescarteMinimoNotas* recebe como parâmetro o conjunto de monofonias *W* da Figura 4.7, *0.75* para *percentual* mínimo de notas, e *4* para *total* de tempos da consulta, retornando 80 monofonias que atendem ao critério de Percentual Mínimo de Notas, tendo descartado as outras 46. O conjunto de monofonias *W*, resultante de *DescarteMinimoNotas*, é passado por parâmetro para a função *DescarteAlturas*, que acaba retornando 40 monofonias que atendem ao critério de Alturas Exatas, descartando as outras 40. Não foi apresentado exemplo para a função *DescarteSaltos* devido a consulta de exemplo não possuir tempo final maior que *4*, ou seja, ao executar a função *DescarteSaltos* para este exemplo nenhuma monofonia seria descartada.

4.3.4 Verificação de Similaridade

A Verificação de Similaridade é o último processo do fluxo de Busca de Composições, sendo responsável pela recuperação de composições similares a uma consulta. O processo efetua a verificação de similaridade através da validação de monofonias contidas na consulta junto a AFNs representando composições.

A validação de uma monofonia *w* por um AFNε ocorre através da sucessiva aplicação da função de transição *Δ*, definida como uma função de transição estendida Δ^* (equação 22). A função Δ^* recebe um conjunto de estados em *P* e uma monofonia *w*. No início da validação, quando conjunto *P* possui somente o estado inicial q_0, a função Δ^* retorna o resultado de uma chamada recursiva, tendo como parâmetro *P* o resultado da função de transição *Δ* (equação 4.18) para o estado q_0 e palavra vazia *ε*, e mesmo parâmetro *w*. Após o início da validação, quando *P* não contém somente q_0, a função Δ^* retorna o resultado de uma chamada recursiva, tendo como parâmetro *P* a união dos resultados da função de transição *Δ* para cada estado *q* em *P* e distância entre notas *d* mais à esquerda da monofonia *w*, e como parâmetro *w* todas as distâncias entre notas *d* da monofonia *w* corrente exceto a mais à esquerda. A monofonia w assume o valor de palavra vazia *ε* quando todas as distâncias entre notas *d* são computadas. A função Δ^* irá retornar o próprio conjunto passado no parâmetro *P* quando *w* for igual à palavra vazia *ε*, encerrando o processo de validação.

$$\Delta^*(P,w) = \begin{cases} \Delta^*\Big(\Delta\big(q_0,\varepsilon\big),w\Big) & : P = \{q_0\} \\ \Delta^*\Big(\bigcup\limits_{q\in P}\Delta\Big(q,\Pi_{d_0}(w)\Big),\Pi_{d_1..d_n}(w)\Big) & : P \neq \{q_0\} \\ P & : w = \varepsilon \end{cases} \quad (4.22)$$

Considera-se que a monofonia *w* é aceita pelo AFNε quando algum dos estados do conjunto resultante de Δ^* seja final, ou seja, pertença ao conjunto de estados finais *F*. Na prática, devido *F* ser formado por todos os estados de *Q* menos q_0, a monofonia *w* é aceita pelo AFNε se o conjunto resultante de Δ^* não for vazio. O conjunto de todas as monofonias *w* aceitas por um AFNε representando uma composição é expressa como *L(M)* (equação 4.23).

$$L(M) = \{w \in \Sigma^* : \Delta^*(\{q_0\},w) \cap F \neq \varnothing\} \quad (4.23)$$

No método, o grau de similaridade de uma monofonia *w* aceita por um AFNε *M* representando uma composição é dado pela função *S* (equação 4.24). A função de similaridade *S* consiste na divisão entre a quantidade de distâncias entre notas *d* com diferença de tempo δ_t diferente de *0* somado a *1*, e o maior tempo *t* dentre os estados de *Q* do AFNε.

$$S(w,M) = \frac{\big|\sigma_{(\Pi_{\delta t}(d)\neq 0)}(w)\big|+1}{\max\big(\Pi_t(\Pi_Q(M))\big)} : w \in L(M) \quad (4.24)$$

A Figura 4.9 apresenta um exemplo de validação da monofonia w_{12}, mostrada nas figuras 4.7 e 4.8, pela função estendida Δ^* do AFNε da Figura 4.5.

Validação de $w_{12}((0,64),(1,-4),(2,12))$:

$$
\begin{aligned}
&\Delta^*\Big(\{q_0\},\big((0,64),(1,-4),(2,12)\big)\Big) && =\Delta^*\Big(\{q_1,q_7\},\big((1,-4),(2,12)\big)\Big)\\
=&\Delta^*\Big(\Delta(q_0,\varepsilon),\big((0,64),(1,-4),(2,12)\big)\Big) && =\Delta^*\Big(\Delta\big(q_1,(1,-4)\big)\cup\Delta\big(q_7,(1,-4)\big),\big((2,12)\big)\Big)\\
=&\Delta^*\Big(Q,\big((0,64),(1,-4),(2,12)\big)\Big) && =\Delta^*\Big(\{q_3\},\big((2,12)\big)\Big)\\
=&\Delta^*\Big(\Delta\big(q_0,(0,64)\big)\cup\Delta\big(q_1,(0,64)\big)\cup && =\Delta^*\Big(\Delta\big(q_3,(2,12)\big),\varepsilon\Big)\\
&\quad\Delta\big(q_2,(0,64)\big)\cup\Delta\big(q_3,(0,64)\big)\cup && =\Delta^*\Big(\{q_8\},\varepsilon\Big)\\
&\quad\Delta\big(q_4,(0,64)\big)\cup\Delta\big(q_5,(0,64)\big)\cup && =\{q_8\}\\
&\quad\Delta\big(q_6,(0,64)\big)\cup\Delta\big(q_7,(0,64)\big),\\
&\quad\big((1,-4),(2,12)\big)\Big) && \{q_8\}\cap F\neq\varnothing\therefore w_{12}\in L(M)
\end{aligned}
$$

Similaridade de $w_{12}\big((0,64),(1,-4),(2,12)\big)$ com M :

$$
S\Big(\big((0,64),(1,-4),(2,12)\big),M\Big)
$$

$$
=\frac{\left|\sigma_{(\Pi_{\delta t}(d)\neq 0)}\big(\big((0,64),(1,-4),(2,12)\big)\big)\right|+1}{\max\big(\Pi_t(\Pi_Q(M))\big)}=\frac{\left|\big((1,-4),(2,12)\big)\right|+1}{4}=\frac{3}{4}=0.75
$$

Figura 4.9: Exemplo de validação de monofonia por AFNε e cálculo de similaridade.

No exemplo mostrado na Figura 4.9, a validação da monofonia w_{12} ocorre através de sucessivas execuções da função de transição estendida Δ^* do AFNε da Figura 4.5, sendo considerada como aceita, devido o conjunto resultante $\{q_8\}$ possuir um estado final de F. O do cálculo da função S da Figura 4.9 mostra que o método considera um grau de similaridade de 75% para monofonia w_{12} e o AFNε.

A função *VerificaSimilaridade* (Algoritmo 4.7) foi definida para recuperar composições, de uma base de composições B, similares à monofonias w contidas em uma consulta.

Algoritmo 4.7 *VerificaSimilaridade*

```
1 :  function VerificaSimilaridade(B, W)
2 :      var R
3 :      foreach M ∈ B do
4 :          foreach w ∈ τ_(|σ_(Π_δt(d)≠0)(w)| desc)(W) do
5 :              if w ∈ L(M) then
6 :                  R ← R ∪ {(M, S(w, M))}
7 :                  break
8 :              end if
9 :          next
10 :     next
11 :     return R
12 : end function
```

A função *VerificaSimilaridade*, mostrada no Algoritmo 4.7, recebe como parâmetro uma base B de AFNs M e um conjunto W de monofonias oriundas de uma

consulta (linha 1). Uma variável *R* é declarada para acumular os resultados, formados pelas composições similares e grau de similaridade (linha 2). Para cada AFNs *M* em *B* (linha 3), são realizadas verificações com cada monofonia *w* no conjunto *W* ordenado, de maneira decrescente, pela quantidade de distâncias entre notas *d* com diferença de tempo δ_t diferente de *0* (linha 4). Quando for identificada a primeira monofonia *w* aceita pelo AFNε *M* corrente (linha 5), o conjunto *R* acumulará uma tupla de resultado, formada por *M* e o grau de similaridade entre *w* e *M* dado por *S* (linha 6), e serão encerradas as verificações para o AFNε *M* corrente (linha 7). Ao final da execução, a função retorna o conjunto *R* (linha 11), contendo todas as composições similares (aceitas), representadas como AFNs, juntamente com o maior grau de similaridade dado por S para as monofonias de *W*.

4.4 Considerações Finais do Capítulo

O método Correspondência de Monofonia Contida possibilita buscar composições polifônicas a partir de uma consulta, também polifônica, utilizando uma abordagem de similaridade melódica baseada na correspondência de linhas melódicas monofônicas contidas nas melodias.

Após a definição do método, é interessante visualizar seu posicionamento em relação aos outros métodos estudados nos trabalhos relacionados (seção 3.1), a fim de estabelecer vantagens e desvantagens mais evidentes. A Tabela 4.1 apresenta as características do método proposto (linha 15), junto aos outros métodos.

Tabela 4.1: Correspondência de Monofonia Contida e outros métodos estudados.

#	Método	Abordagem	Consulta	Correspondência								
				Tipo	Padrão	Restr. Padrões	Tempo	Duração	Tom e Oitava	Intervalo e Oitava	Somente Tom	Somente Intervalo
1	*ShiftOrAnd*	*strings*	mono	exata	mono		ordem			X	X	
2	*DirectCheck*	*strings*	mono	exata	mono		ordem					X
3	*IntervalMatch*	*strings*	mono	exata	mono		ordem					X
4	*MonoPoly*	*strings*	mono	exata	mono		ordem					X
5	*Splitting*	*strings*	mono	exata	mono	X	ordem			X		X
6	LCTS	*strings*	mono	parcial	mono	X	ordem			X		X
7	PROMS	conjuntos	poli	parcial	poli	X	ataque		X	X	X	X
8	*P1*	conjuntos	poli	exata	poli		ataque	X		X		X
9	*P2*	conjuntos	poli	parcial	poli		ataque	X		X		X
10	*P3*	conjuntos	poli	parcial	poli		ataque	X		X		X
11	SIAMESE	conjuntos	poli	parcial	poli		ataque			X		X
12	MSM	conjuntos	poli	parcial	poli		ataque			X		X
13	TD-EMD	conjuntos	poli	parcial	poli		ataque	X		X		X
14	TD-PTD	conjuntos	poli	exata	poli		ataque	X		X		X
15	**CMC**	**conjuntos**	**poli**	**parcial**	**mono**	**X**	**ordem**		**X**	**X**	**X**	**X**

Como pode ser visto na Tabela 4.1, o método CMC se encaixa na abordagem de conjuntos, devido representar melodias como conjuntos de notas. Outra característica importante refere-se à possibilidade da utilização de consultas polifônicas. Considera-se que o método utiliza correspondência parcial, pois pode recuperar composições a partir de um subconjunto de notas de uma consulta, mais especificamente uma monofonia

contida. Porém, é importante salientar que, do ponto de vista de uma monofonia contida em uma consulta, a correspondência é exata. Logo, mesmo a consulta sendo polifônica, o método possui a desvantagem de trabalhar com correspondência de padrões monofônicos. Uma das principais características do método CMC consiste na possibilidade de restrição de padrões a serem correspondidos em uma busca, através da eliminação de monofonias contidas na consulta, sendo que somente outros três métodos (*Splitting*, LCTS e PROMS) possuem característica semelhante, por definição de limites de lacunas nas correspondências. O método CMC também possui a desvantagem de considerar somente a ordenação das notas, não considerando informações musicais temporais mais precisas para uma busca melódica, como o tempo de ataque das notas (início de emissão) e suas respectivas durações. Por fim, observando a Tabela 4.1, nota-se que, além do método CMC, somente o PROMS permite o uso dos quatro tipos de correspondência nas buscas: tom e oitava (altura exata); somente tom (escala cromática); distância e oitava (diferença de altura); e somente distância (diferença de altura dentro de uma oitava). Apesar de muitos trabalhos considerarem suficiente a correspondência de distâncias (intervalos) para a busca de composições melodicamente similares, considera-se esta característica uma vantagem, pois proporciona mais opções ao utilizador do método.

É importante ressaltar que a definição do método não leva em consideração aspectos relacionados à sua implementação, como performance na execução. Porém, utilizando as equações descritas neste capítulo, fica evidente que a quantidade de monofonias contidas em uma consulta aumenta significativamente a cada nota adicionada e com o aumento do valor do limite k. Logo, o descarte de monofonias mostra-se muito importante para a redução do esforço computacional do processo de Verificação de Similaridade, eliminando monofonias não consideradas relevantes.

5 BUSCADOR DE TABLATURAS TEXTUAIS DA WEB

Este capítulo apresenta detalhes do protótipo, nomeado como TabSim, desenvolvido para busca de tablaturas textuais da *web*, utilizando o método Correspondência de Monofonia Contida, proposto por este trabalho. O desenvolvimento deste protótipo tem como objetivo demonstrar a viabilidade de implementação do método proposto e servir como ferramenta base para os experimentos.

Inicialmente uma visão geral do Buscador de Tablaturas Textuais da *Web* é apresentada, seguido de uma seção das duas ações básicas previstas para o protótipo: coleta de documentos da *web* que contenham tablaturas textuais, e realização de buscas de composições similares a partir de uma consulta. Por fim, estão apresentadas as considerações finais sobre o protótipo, posicionando-o em relação às ferramentas apresentadas nos trabalhos relacionados (seção 3.2).

5.1 Visão Geral

O objetivo do Buscador de Tablatura Textuais da *Web* é localizar documentos da *web* que contenham tablaturas em formato textual, consideradas similares a uma consulta através do método Correspondência de Monofonia Contida. Este protótipo foi implementado considerando somente consultas monofônicas. Antes que as buscas sejam realizadas, dados musicais de documentos da *web* devem ser obtidos através de um *crawler*, mais especificamente de tablaturas textuais de violão e guitarra. A Figura 5.1 apresenta a arquitetura do buscador.

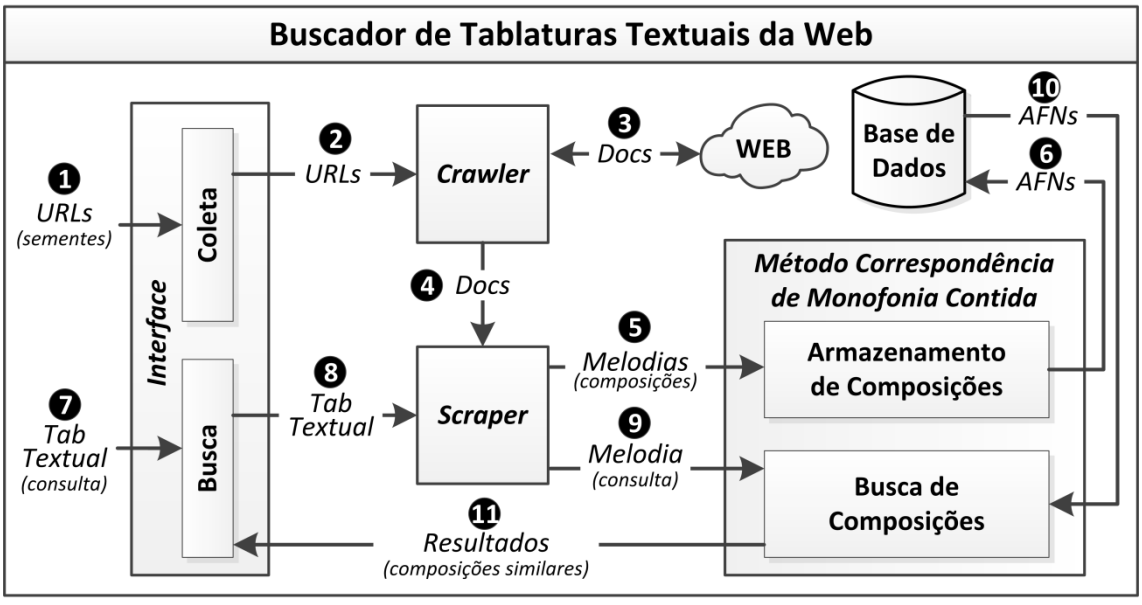


Figura 5.1: Visão geral da arquitetura do Buscador de Tablaturas Textuais da Web.

O Buscador de Tablaturas Textuais da *Web*, mostrado na Figura 5.1, tem como ações básicas a coleta de documentos da *web* contendo tablaturas textuais e a busca de composições (tablaturas textuais) similares a partir de uma consulta. Cada uma destas ações, coleta e busca, possui uma interface específica. A ação de coleta inicia pela entrada de sementes (1), na forma de URLs (*Uniform Resource Locator*), na interface de coleta. Em seguida, as URLs são repassadas ao *Crawler* (2), que explora todos os documentos do mesmo domínio de cada URL, recuperando aqueles documentos que contenham tablaturas textuais (3). Os documentos com tablaturas textuais são repassados ao *Scraper* (4) para que seus dados sejam extraídos. As melodias das tablaturas textuais (composições) são repassadas ao fluxo de Armazenamento de Composições do método Correspondência de Caminho Contido (5) para que sejam armazenadas em uma Base de Dados (6), na forma de Autômatos Finitos Não Determinísticos (AFN). Já a ação de busca inicia pela entrada de uma consulta (7), na forma de uma tablatura textual, na interface de busca. A tablatura textual (consulta) é repassada ao *Scraper* (8) para que seus dados sejam extraídos e repassados ao fluxo de Busca de Composições do método Correspondência de Caminho Contido (9). O fluxo de Busca de Composições recupera as composições similares da Base de Dados (10) e retorna os resultados à interface de busca (11).

O protótipo foi implementado na plataforma ASP.NET, utilizando .NET Framework 4.5 e sistema gerenciador de banco de dados MySQL 5.6. O equipamento utilizado foi um Notebook com Processador Core2 Duo 2.26 GHz com 4GB de memória RAM.

As subseções seguintes descrevem em detalhes a interação dos componentes do Buscador de Tablaturas Textuais da *Web* nas ações de coleta e busca.

5.2 Coleta

Esta seção tem como objetivo descrever cada um dos componentes envolvidos na ação de coleta de tablaturas textuais (composições) da *web*.

5.2.1 Interface

A interface de coleta permite que URLs sejam inseridas na ferramenta, a fim de recuperar e armazenar dados de documentos que possuem tablaturas textuais. A Figura 5.2 mostra a interface de coleta implementada.

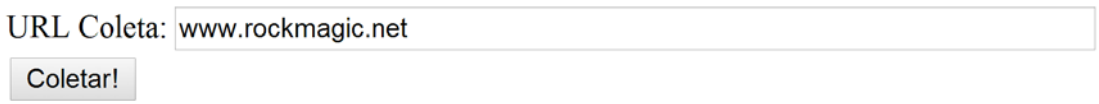


Figura 5.2: Interface de coleta do Buscador de Tablaturas Textuais da Web.

A interface de coleta, mostrada na Figura 5.2, possui somente uma caixa de texto para entrada da URL e um botão de submissão. A implementação desta interface pode ser simples devido a ação de coleta de documento ser essencialmente uma função administrativa.

5.2.2 Crawler

O *crawler* tem como objetivo explorar domínios das URLs (sementes) repassadas pela interface de coleta, recuperando documentos que contenham tablaturas textuais. Logo, dois processos compõem o *crawler*:

- Identificação de Tablatura Textual – processo que identifica se um documento contém ou não uma tablatura textual;
- Navegação – processo responsável por explorar documentos a partir de uma URL, através da navegação entre *links*.

As subseções seguintes descrevem detalhes dos processos de Identificação de Tablatura Textual e Navegação do *crawler*.

5.2.2.1 Identificação de Tablatura Textual

O processo de Identificação de Tablatura Textual recebe um documento baseado em texto e verifica se este documento possui uma tablatura textual.

Inicialmente, é realizada a padronização dos caracteres de hífen, utilizados na representação das cordas do instrumento musical, e quebra de linha, importante para identificação de blocos de tablatura textual. As principais representações para o caractere hífen são `U+2010`, `U+2011`, `U+2012`, `U+2013`, `U+2014`, `U+2015`, `U+002D`, `U+05BE`, `U+207B`, `U+208B`, `U+2212`, `U+FE58`, `U+FE63` e `U+FF0D`. Já as quebras de linha podem aparecer como combinações de caracteres *Carriage Return* (`\r`) e *Line Feed* (`\n`), além da *tag* `<br/>` em documentos HTML. É importante ressaltar que o conteúdo compreendido entre *tags* de texto pré-formatado (`<pre></pre>`) em um documento HTML utiliza caracteres `\r` e `\n` para representar quebras de linha, e não a *tag* `<br/>`. A implementação da padronização dos caracteres é feita através da captura de todas as diferentes formas de representação de hífen e quebra de linha por expressões regulares, e substituição dessas capturas por uma única forma de representação.

Finalizada a padronização dos caracteres de hífen e quebra de linha, são realizadas verificações, a cada linha do documento, a fim de identificar representações de corda de uma tablatura textual. Uma linha é identificada como uma corda quando possuir padrões de caracteres utilizados na representação de cordas de instrumento em tablaturas textuais. Os padrões procurados na identificação de linhas como cordas são formados por caracteres estruturais de tablaturas textuais, como a barra vertical (`|`), utilizada para dividir blocos de tablatura semelhantes a compassos, o dois pontos (`:`), utilizado após a definição de afinação de uma corda, e, principalmente, o hífen (`-`), utilizado na representação de cordas do instrumento. Os padrões de corda implementados são `--`, `:-`, `|-` e `-|`. Na implementação, uma linha é identificada como uma corda quando existir ao menos uma captura de padrão de corda por uma expressão regular.

Quando forem identificadas seis linhas consecutivas como cordas, seguida de uma linha não identificada como corda, considera-se que o documento possui uma tablatura textual, pois possui ao menos um bloco de tablatura. Tablaturas textuais geralmente são divididas em blocos para se adequar a orientação vertical de documentos. A implementação do Buscador de Tablaturas Textuais da *Web* foi direcionada à busca de tablaturas de violões e guitarras, por este motivo um bloco de tablatura textual consiste de uma sequência de 6 linhas consecutivas identificadas como cordas, justamente a quantidade de cordas dos instrumentos.

O Algoritmo 5.1 mostra a função *IdentificaTabTextual*, utilizada para a Identificação de Tablaturas Textuais em documentos.

Algoritmo 5.1 *IdentificaTabTextual*

```
1 :  function IdentificaTabTextual(doc)
2 :      var bloco ← 0
3 :      doc ← replaceER(doc, "[\u2010-\u2015\u002D\u05BE
             \u207B\u208B\u2212\uFE58\uFE63\uFF0D]", "-")
4 :      doc ← replaceER(doc, "<br */?>|(?<!  \r)\n\r(?!\n)|
             \r(?!\n)|(?<!\r)\n", "\r\n")
5 :      foreach linha ∈ getLines(doc) do
6 :          if matchER(linha, ":-|--|[|]-|-[|]") = true then
7 :              bloco ← bloco + 1
8 :          else
9 :              if bloco = 6 then
10 :                 return true
11 :             else
12 :                 bloco ← 0
13 :             end if
14 :         end if
15 :     next
16 :     return false
17 : end function
```

A função *IdentificaTabTextual*, mostrada no Algoritmo 5.1, recebe como parâmetro um documento baseado em texto (linha 1) e inicia pela declaração da variável *bloco* (linha 2), utilizada na contagem de linhas consecutivas identificadas como cordas de tablatura textual. Em seguida, é realizada a padronização dos caracteres de hífen através de substituições de capturas de uma expressão regular por um caractere hífen padrão (linha 3). Também são padronizadas as quebras de linha através da substituição de capturas de uma expressão regular por `\r\n` (linha 4). Para cada linha do documento (linha 5), se existir ao menos uma correspondência de padrão de linha (linha 6), dada por uma expressão regular, a variável *bloco* é incrementada (linha 7), caso contrário (linha 8), é verificado se a variável *bloco* possui o valor *6* (linha 9). Se a variável *bloco* possuir o valor *6* a função retorna o valor verdadeiro (*true*) para a identificação de tablatura textual no documento (linha 10), senão (linha 11) a variável *bloco* tem seu valor zerado (linha 12) e a execução do algoritmo prossegue. Ao final, não tendo sido identificado nenhum bloco de tablatura textual, a função retorna o valor falso (*false*) para a identificação de tablatura textual no documento (linha 16).

A Figura 5.3 apresenta um exemplo de execução do processo Identificação de Tablatura Textual, contendo resultados parciais após a padronização dos caracteres de hífen e quebra de linha, identificação de linhas como cordas e identificação de bloco de tablatura.

Documento original:

```
C C R - - - - - \r \n \r 4 \ 4   N . C \n e | - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - | - - - - - - - - - \n \r B
| — I — — — — — 8 ~   — — — — — — — 6 ~ ~ — — — — — — — — 5 ~ — — — — — — — 3
~ ~ — | — — — — — — — — — \r G | : N - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - | - - - - - - - : \r \n D | : T — / 9
~ ~ — — — — — / / 7 v — — — — — 3 / 5 ~ — — — — — — — / 3 ~ — — — — —   | —
— — — : | \n A | -   R - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - — — — — — 1 b — — — — — — — — | | \r \n E | - O — — — — — - - -
- - - - - - - - - - — - - - — — — — — — — — — — — — | - 3 ~ \r \n
```

Padronização de hífens (−) e quebras de linha (`\r\n`) (*substituições em destaque*):

ER hífens: `[\u2010-\u2015\u05BE\u207B\u208B\u2212\uFE58\uFE63\uFF0D]`

ER quebras de linha: `<br */?>|(?<!\r)\n\r(?!\n)|\r(?!\n)|(?<!\r)\n`

```
C C R - - - - - \r \n \r \n 4 \ 4   N . C \r \n e | - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | - - - - - - - - \r
\n B | - I - - - - - 8 ~   - - - - - - - 6 ~ ~ - - - - - - - - 5 ~ - - - - - - -
- 3 ~ ~ - | - - - - - - - - - \r \n G | : N - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - | - - - - - - - : \r \n D | : T
- / 9 ~ ~ - - - - - - / / 7 v - - - - - 3 / 5 ~ - - - - - - - / 3 ~ - - - - - -
   | - - - - : | \r \n A | -   R - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - 1 b - - - - - - - - | | \r \n E | - O - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | - 3 ~ \r \n
```

Identificação de padrões de corda nas linhas (*em destaque*)**:**

ER corda: `-{2,}|:-{1,}|[|]-{1,}|-{1,}[|]`

```
C C R - - - - - \r \n \r \n 4 \ 4   N . C \r \n e | - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | - - - - - - - - \r
\n B | - I - - - - - 8 ~   - - - - - - - 6 ~ ~ - - - - - - - - 5 ~ - - - - - - -
- 3 ~ ~ - | - - - - - - - - - \r \n G | : N - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - | - - - - - - - : \r \n D | : T
- / 9 ~ ~ - - - - - - / / 7 v - - - - - 3 / 5 ~ - - - - - - - / 3 ~ - - - - - -
   | - - - - : | \r \n A | -   R - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - 1 b - - - - - - - - | | \r \n E | - O - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | - 3 ~ \r \n
```

Identificação de bloco de tablatura textual com 6 cordas *(em destaque)*:

```
C C R - - - - - \r \n \r \n 4 \ 4   N . C \r \n e | - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | - - - - - - - - \r
\n B | - I - - - - - 8 ~   - - - - - - - 6 ~ ~ - - - - - - - - 5 ~ - - - - - - -
- 3 ~ ~ - | - - - - - - - - - \r \n G | : N - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - | - - - - - - - : \r \n D | : T
- / 9 ~ ~ - - - - - - / / 7 v - - - - - 3 / 5 ~ - - - - - - - / 3 ~ - - - - - -
   | - - - - : | \r \n A | -   R - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - 1 b - - - - - - - - | | \r \n E | - O - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | - 3 ~ \r \n
```

Figura 5.3: Exemplo de execução do processo de Identificação de Tablatura Textual.

A Figura 5.3 apresenta um exemplo de execução do processo de Identificação de Tablatura Textual para um documento. Após a padronização dos caracteres hífen e quebra de linha, são identificados os padrões de corda e, em seguida, é identificado um bloco de tablatura textual com 6 cordas. Logo, o documento original do exemplo possui uma tablatura textual.

5.2.2.2 Navegação

A implementação do processo de Navegação do *crawler* do Buscador de Tablaturas Textuais da *Web* utiliza o algoritmo de busca em profundidade (*Depth First Search* ou DFS). A navegação ocorre através de *links* contidos nos documentos. Somente são explorados *links* com nível de diretório maior ou igual ao documento que os contém. Todos os documentos baseados em texto são explorados pelo processo, porém, normalmente, consistem de documentos HTML. A implementação do algoritmo DFS inicia a partir da URL recebida pela interface de coleta, explorando ao máximo os *links* de cada documento (profundidade) antes de retroceder ao documento apontador (*backtraking*). O Algoritmo 5.2 mostra a implementação do algoritmo DFS para o processo de Navegação do *crawler*.

Algoritmo 5.2 *NavegaCrawler*

```
1 :  function NavegaCrawler(URL)
2 :      var D, doc ← getDocument(URL)
3 :      if IdentificaTabTextual(doc) = true then
4 :          D ← {doc}
5 :      end if
6 :      foreach link ∈ getLinks(doc) do
7 :          D ← D ∪ NavegaCrawler(link)
8 :      next
9 :      return D
10 : end function
```

A função *NavegaCrawler*, mostrada no Algoritmo 5.2, recebe uma *URL* para o início da navegação do *crawler* (linha 1). A variável *D* é declarada para acumular os documentos que possuem tablaturas textuais, já a variável *doc* recebe o documento relativo à *URL* inicial (linha 2). Caso seja identificada uma tablatura textual no documento contido na variável *doc* (linha 3), através da função *IdentificaTabTextual* (Algoritmo 5.1), a variável *D* acumula o documento (linha 4). Em seguida, para cada *link* válido (linha 6), não navegado e com nível de diretório maior ou igual à URL do documento contido em *doc*, a variável *D* acumula o resultado da chamada recursiva à função *NavegaCrawler*, tendo como parâmetro o *link* corrente (linha 7). Ao final da execução (linha 9), a função *NavegaCrawler* retorna a variável *D* contendo os documentos que possuem tablaturas textuais.

5.2.3 Scraper

O *scraper* tem como objetivo extrair dados dos documentos coletados pelo *crawler* para posterior armazenamento. A implementação do *scraper* extrai os títulos dos documentos de *tags* `<title></title>`, para o caso de documentos HTML, e melodias de tablaturas textuais.

A extração da melodia de um documento inicia pela extração de blocos de tablatura. Uma tablatura textual normalmente encontra-se dividida em blocos, devido sua totalidade ultrapassar o limite horizontal de visualização de documentos. Estes blocos, compostos por 6 linhas identificadas como cordas, consistem de trechos de

tablatura textual. Na implementação do *scraper*, números (trastes) contidos em todos os blocos de tablatura textual são estruturados em uma única matriz representando a tablatura textual contínua. Uma melodia *V* (equação 4.4 da seção 4.2) é gerada através da leitura da matriz representando a tablatura textual contínua. Cada número contido na matriz dá origem a uma nota *v* (equação 4.3 da seção 4.2). O tempo *t* de cada nota é definido pela ordem cronológica horizontal de aparição na matriz. Números de mesma coluna na matriz possuem tempos *t* iguais. A definição da altura *h* de cada nota é dada pela soma do número lido, indicando o traste no instrumento, com o valor MIDI correspondente à afinação da corda (linha da matriz) onde o número de lido ocorre. O *scraper* utiliza a afinação padrão de violões e guitarras, sendo, da primeira linha até a última, *E3*, *A3*, *D4*, *G4*, *B4* e *E5* (em MIDI, respectivamente, *40*, *45*, *50*, *55*, *59* e *64*).

A função *ExtraiMelodia*, mostrada no Algoritmo 5.3, extrai uma melodia *V* de uma matriz representando uma tablatura textual contínua extraída de um documento.

Algoritmo 5.3 *ExtraiMelodia*

```
1 :  function ExtraiMelodia(tab)
2 :      const afinacao ← (40, 45, 50, 55, 59, 64)
3 :      var V ← {(0, 0)}
4 :      for coluna ← 0 to coluna < |tab[0]| step 1 do
5 :          for linha ← 0 to linha < 6 step 1 do
6 :              if tab[linha, coluna] ∈ {0, ··· , 24} then
7 :                  V ← V ∪ {(coluna + 1,
                         tab[linha, coluna] + afinacao[coluna])}
8 :              end if
9 :          next
10 :     next
11 :     return V
12 : end function
```

No Algoritmo 5.3, a função *ExtraiMelodia* recebe por parâmetro uma matriz denominada *tab* (linha 1), representando uma tablatura textual contínua extraída de um documento. A constante *afinacao* é declarada com os valores MIDI referentes à afinação padrão das cordas da tablatura (linha 2). Uma variável *V* é declarada para acumular as notas extraídas da matriz *tab* (linha 3), já tendo como valor inicial a nota falsa *(0,0)*. Para cada coluna (linha 4) e linha (linha 5) da matriz, é verificado se o valor corrente está entre *0* e *24* (linha 6), quantidade padrão de trastes em violões e guitarras. Caso o valor esteja entre *0* e *24*, a variável *V* acumula uma nota *v* (linha 7) com tempo *t* igual ao valor da coluna corrente mais *1*, e altura *h* igual a soma entre o valor corrente de *tab* e o valor MIDI correspondente à afinação da linha (corda) corrente. Ao final da execução (linha 11), a variável *V* é retornada contendo a melodia extraída da tablatura textual.

5.2.4 Fluxo de Armazenamento de Composições do Método

A implementação do Fluxo de Armazenamento de Composições do método Correspondência de Monofonia Contida (seção 4.3) recebe melodias, extraídas de tablaturas textuais pelo *scraper*, as representa como Autômatos Finitos Não Determinísticos com Movimentos Vazios (AFNε), e armazena dados destes AFNs em uma Base de Dados relacional.

Para representar melodias como AFNε, foram implementados algoritmos para os processos de Representação de Melodia como GAD (seção 4.3.1) e Geração de AFNε (seção 4.3.2). Já no esquema de Banco de Dados relacional implementado, mostrado na Figura 5.4, optou-se por uma modelagem enxuta, armazenando somente dados das transições dos AFNε.

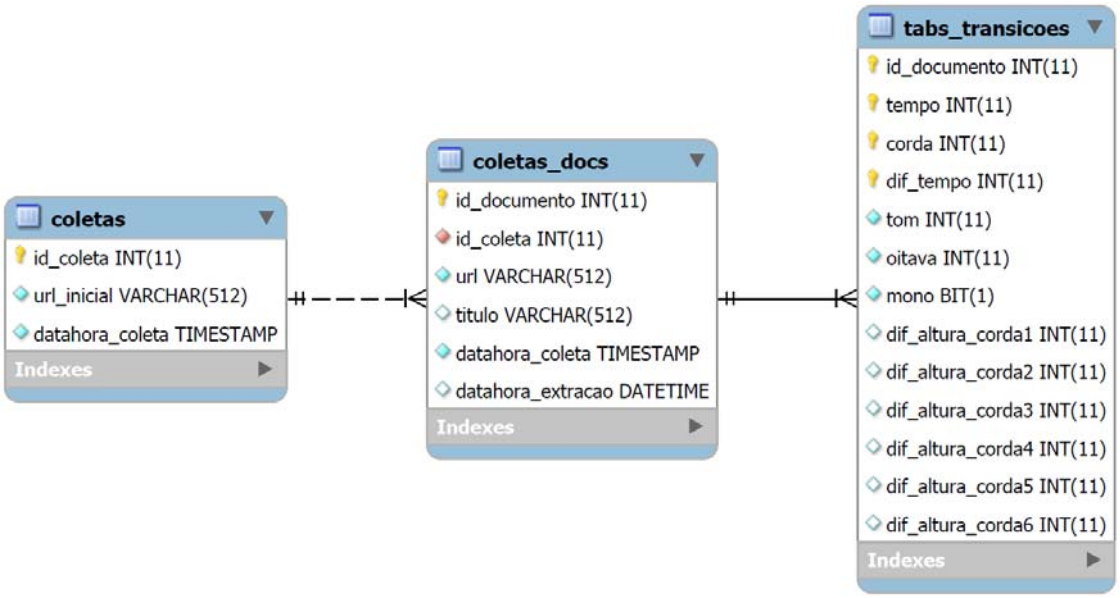


Figura 5.4: Esquema do Banco de Dados implementado no protótipo.

A tabela `coletas`, mostrada na Figura 5.4, é utilizada para armazenar dados iniciais de cada ação de coleta do protótipo, tendo uma chave primária sequencial (`id_coleta`), uma coluna obrigatória para guardar a URL inicial (`url_inicial`), e uma coluna obrigatória para guardar a data e a hora de realização da coleta (`datahora_coleta`).

Já a tabela `coletas_docs`, da Figura 5.4, armazena dados de navegação de cada documento visitado durante uma ação de coleta, tendo uma chave primária sequencial (`id_documento`), uma chave estrangeira obrigatória referenciando a tabela de `coleta` (`id_coleta`), uma coluna obrigatória para guardar a URL do documento visitado (`url`), um título opcional do documento (`titulo`), a data e hora obrigatória de coleta do documento (`datahora_coleta`), e uma coluna opcional para guardar data e hora do término de extração de dados, realizada pelo *scraper*, no caso do documento conter uma tablatura textual (`datahora_extracao`).

Na tabela `tabs_transicoes` da Figura 5.4, optou-se por armazenar transições de cada nota extraída para as notas a uma determinada diferença de tempo. A chave primária da tabela `tabs_transicoes` é composta por uma coluna chave estrangeira referenciando a tabela `coleta_docs` (`id_documento`), uma contendo o

tempo da nota de origem da transição (`tempo`), outra contendo a corda da tablatura onde a nota de origem ocorre (`corda`), e uma coluna indicando a diferença de tempo da transição (`dif_tempo`). A tabela `tabs_transicoes` também contém três colunas obrigatórias para guardar o tom (`tom`) e a oitava (`oitava`) da nota de origem da transição, e se a transição é monofônica (`mono`), ou seja, se a nota de origem e a nota de destino da transição são únicas em seus respectivos tempos. Devido o protótipo trabalhar com tablaturas textuais de violão e guitarra, instrumentos que contêm 6 cordas, optou-se por uma modelagem desnormalizada da tabela `tabs_transicoes` para guardar as diferenças de altura, em um intervalo de 128 valores (MIDI), entre a nota de origem da transição e cada uma das notas de destino que ocorram em cada corda a uma determinada diferença de tempo (colunas opcionais `dif_altura_corda1`, `dif_altura_corda2`, `dif_altura_corda3`, `dif_altura_corda4`, `dif_altura_corda5` e `dif_altura_corda6`). No protótipo, para cada nota em um tempo *t* da melodia, extraída da tablatura textual, foram inseridos três registros na tabela `tabs_transicoes`, correspondendo a transições para notas dos tempos *t+1*, *t+2* e *t+3*, devido o valor de *k* escolhido ser igual a *3* (seção 4.3.1).

5.3 Busca

Esta seção tem como objetivo descrever os componentes envolvidos na ação de busca de tablaturas textuais.

5.3.1 Interface

A interface de busca permite que um trecho de tablatura textual monofônica seja inserido na ferramenta, a fim de recuperar tablaturas textuais similares. A Figura 5.5 mostra a interface de coleta implementada.

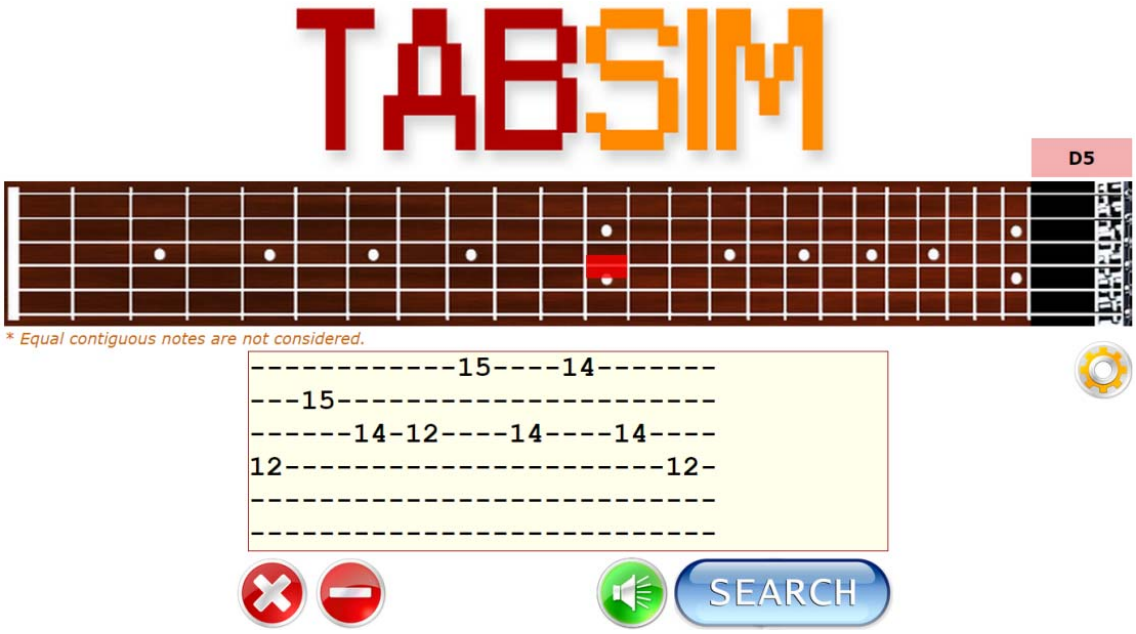


Figura 5.5: Interface de busca do Buscador de Tablaturas Textuais da Web.

A entrada de notas da tablatura textual de consulta, através da interface implementada, mostrada na Figura 5.5, ocorre com o uso do mouse em uma representação de um braço de guitarra. Na parte superior direita da interface, é exibida a nota correspondente à posição do ponteiro do mouse na representação do braço da

guitarra. Um clique na representação do braço da guitarra insere uma nota (ataque em uma determinada corda e determinado traste) na caixa de texto centralizada na interface. Por questões de performance no equipamento utilizado para implementação do protótipo, optou-se por limitar a tablatura textual de consulta a conter no máximo 12 notas e não considerar notas iguais consecutivas na consulta. Abaixo da caixa de texto, onde a tablatura textual é montada, está posicionado, da esquerda para a direita, um botão para limpar tablatura textual (botão "X"), um para apagar uma nota (botão "-"), outro para reproduzir o som da tablatura textual da caixa de texto (botão alto-falante), e o botão de busca (botão "*SEARCH*"). Na parte inferior direita da representação do braço da guitarra, encontra-se um botão para abrir o painel de configurações da busca, mostrado na Figura 5.6.

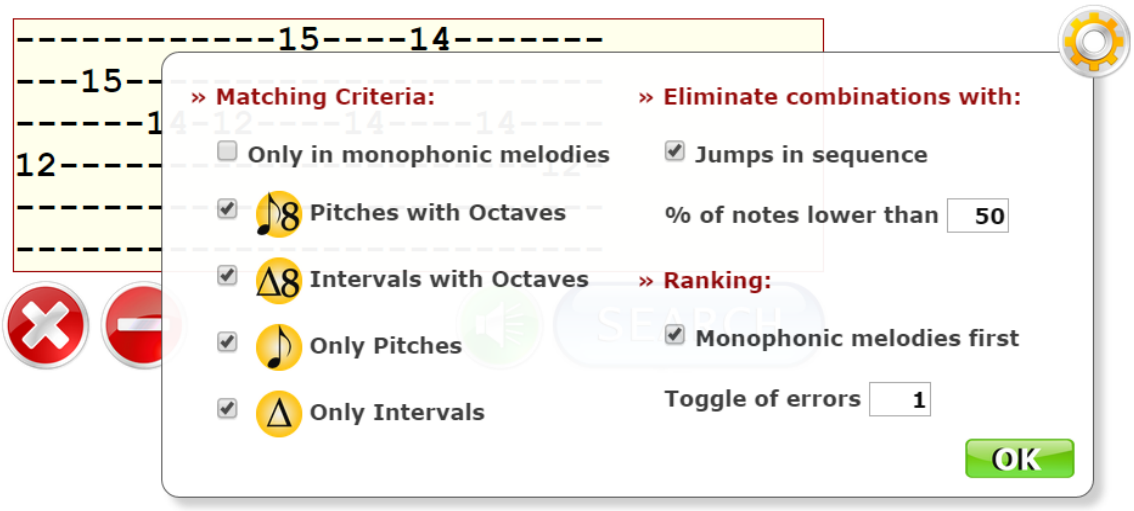


Figura 5.6: Painel de configurações do Buscador de Tablaturas Textuais da Web.

O painel de configurações, mostrado na Figura 5.6, apresenta opções relacionadas à correspondência de notas e intervalos (*Matching Criteria*) utilizados na busca. A opção "*Only in monophonic melodies*" (Somente em melodias monofônicas) permite determinar que as monofonias identificadas na consulta devam corresponder somente em trechos monofônicos da composição, não havendo mais de uma nota por tempo nas correspondências. Também estão disponíveis quatro opções para as correspondências de notas e intervalos:

1. *Pitches with Octaves* (Tons com oitavas) – determina que sejam utilizadas monofonias geradas com diferenças de altura entre notas em um intervalo *r* de 128 valores (notas MIDI) e que iniciem com uma distância entre notas envolvendo a nota falsa;
2. *Intervals with Octaves* (Intervalos com oitavas) – determina que sejam utilizadas monofonias geradas com diferenças de altura entre notas em um intervalo *r* de 128 valores (notas MIDI) e que não iniciem com uma distância entre notas envolvendo a nota falsa;
3. *Only Pitches* (Somente tons) – determina que sejam utilizadas monofonias geradas com diferenças de altura entre notas em um intervalo *r* de 12 valores (escala cromática) e que iniciem com uma distância entre notas envolvendo a nota falsa;
4. *Only Intervals* (Somente intervalos) – determina que sejam utilizadas monofonias geradas com diferenças de altura entre notas em um

intervalo *r* de 12 valores (escala cromática) e que não iniciem com uma distância entre notas envolvendo a nota falsa.

Na prática, os critérios de correspondência definem o intervalo de valores *r* para diferenças de alturas na geração de monofonias (seção 4.3.3) e a eliminação de monofonias pelo critério de Alturas Exatas (seção 4.3.3.1). Ainda, o painel de configurações apresenta opções para descarte de monofonias (seção 4.3.3.1) identificadas na melodia da tablatura textual de consulta. A opção "*Jumps in sequence*" permite eliminar monofonias que não estejam de acordo com o critério Ausência de Saltos Consecutivos, do método Correspondência de Caminho Contido. Também é possível estipular um percentual mínimo de notas em "*% of notes lower than*" (arredondamento utilizado), a fim de eliminar monofonias que não estejam de acordo com o critério Percentual Mínimo de Notas. Nas opções de ranqueamento (*Ranking*) do painel de configurações, pode-se determinar que correspondências de monofonias da consulta, com trechos monofônicos da composição, devam aparecer mais no topo (*Monophonic melodies first*), e também determinar de quantos e quantos erros (*Toggle of errors*) os critérios de correspondência selecionados devem ser alternados. Considere-se um erro a ausência do envolvimento de uma nota em uma monofonia. Quando *Toggle of errors* tiver valor *2*, por exemplo, a ferramenta irá retornar todos os resultados recuperados com monofonias contendo 0 e 1 erros, para cada critério selecionado, para somente depois retornar os resultados recuperados por monofonias com 2 e 3 erros, e assim sucessivamente, alternando de 2 em 2 erros.

5.3.2 Scraper

O *scraper* é utilizado na ação de busca para extrair a melodia da tablatura textual de consulta inserida na interface.

Diferente da extração na ação de coleta (seção 5.2.3), não é necessário iniciar pela extração blocos de tablatura, pois a tablatura textual de consulta inserida na interface de busca possui um único bloco. Na implementação do *scraper*, números (trastes) contidos na tablatura textual de consulta são estruturados em uma matriz. Uma melodia V (equação 4.4 da seção 4.2) é gerada através da leitura da matriz representando a tablatura textual de consulta. Cada número contido na matriz dá origem a uma nota v (equação 4.3 da seção 4.2). O tempo t de cada nota é definido pela ordem cronológica horizontal de aparição na matriz. Números de mesma coluna na matriz possuem tempos t iguais. A definição da altura h de cada nota é dada pela soma do número lido, indicando o traste no instrumento, com o valor MIDI correspondente à afinação da corda (linha da matriz) onde o número de lido ocorre. O *scraper* utiliza a afinação padrão de violões e guitarras, sendo, da primeira linha até a última, *E3*, *A3*, *D4*, *G4*, *B4* e *E5* (em MIDI, respectivamente, *40*, *45*, *50*, *55*, *59* e *64*).

A função *ExtraiMelodia*, mostrada no Algoritmo 5.3, utilizada na ação de coleta, também pode ser utilizada na ação de busca para extrair uma melodia V da matriz gerada para a tablatura textual de consulta.

5.3.3 Fluxo de Busca de Composições do Método

A implementação do Fluxo de Busca de Composições do método Correspondência de Monofonia Contida (seção 4.3) realiza buscas na Base de Dados, previamente populada pela ação de coleta (seção 5.2), a fim de obter composições similares a uma melodia extraída pelo *scraper* de uma tablatura textual de consulta.

A fim de se obter um conjunto de monofonias contidas na melodia da consulta, foram implementados algoritmos para os processos de Representação de Melodia como GAD (seção 4.3.1) e Geração de Monofonias (seção 4.3.3).

A implementação da verificação de similaridade (seção 4.3.4) foi adaptada para que a validação das monofonias contidas, feitas pelos AFNε armazenados na Base de Dados, ocorra em paralelo, agilizando a obtenção de resultados. Esta validação de monofonias em paralelo ocorre através de consultas SQL (*Structured Query Language*) na tabela `tabs_transicoes`, mostrada na Figura 5.4, que armazena as transições dos AFNε representando composições. Para cada monofonia é montada e executada uma consulta SQL.

A consulta SQL é montada dinamicamente de acordo com uma monofonia e seu respectivo critério de correspondência, conforme o diagrama mostrado na Figura 5.7. Tal consulta SQL permite recuperar dados de documentos, contendo tablaturas textuais consideradas similares a tablatura textual de consulta, através do método Correspondência de Monofonia Contida.

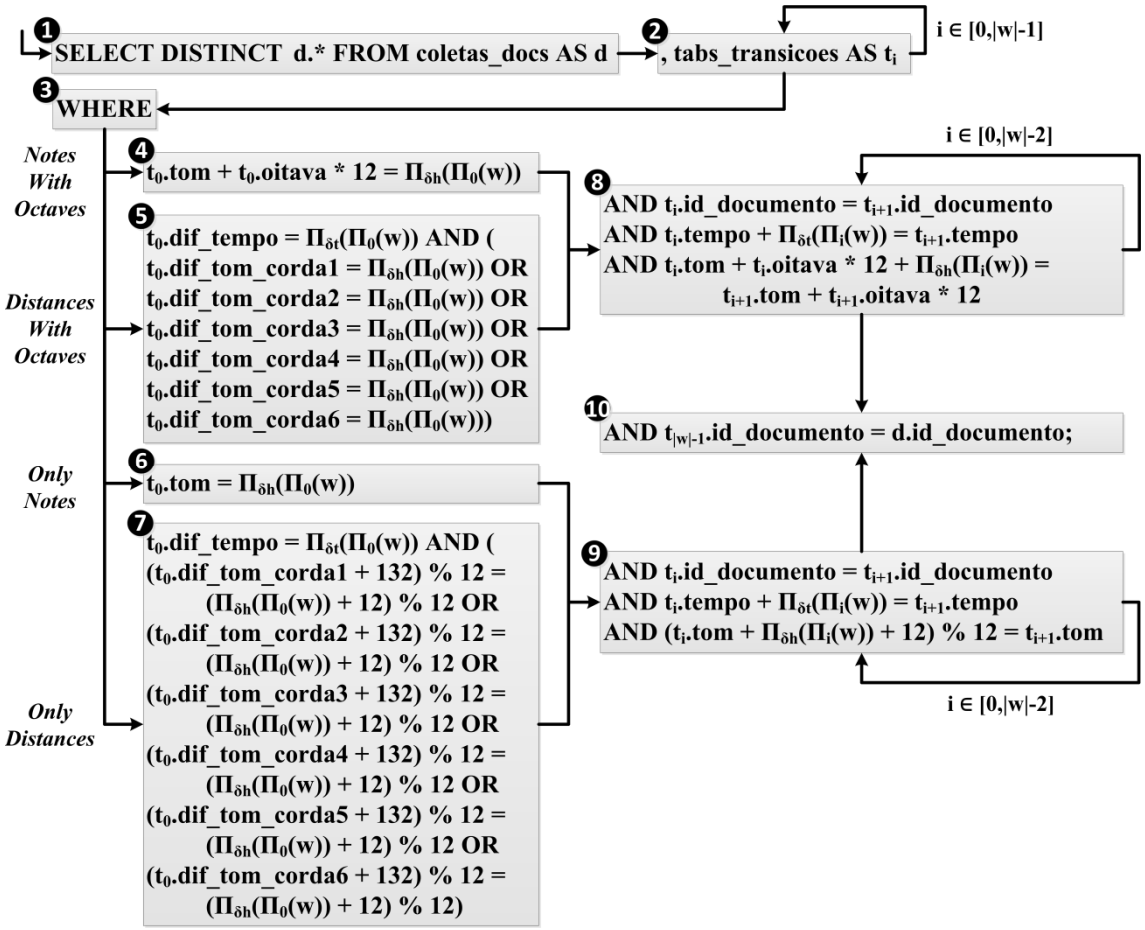


Figura 5.7: Fluxo de montagem dinâmica da consulta SQL para uma monofonia.

O diagrama da Figura 5.7 apresenta o fluxo de montagem de consultas SQL, de acordo com um critério de correspondência. Independente do critério de correspondência escolhido na interface, a consulta iniciará pelo bloco 1, que indica selecionar todos os dados de todos os registros distintos da tabela `coletas_docs`. O bloco 2 adiciona uma junção com a tabela `tabs_transicoes` para cada distância entre notas da monofonia *w* utilizada para montar a consulta. A partir do bloco 3 têm-se

as condições para seleção de registros. De acordo com o critério de correspondência, somente um entre os blocos 4, 5, 6 e 7 é adicionado na consulta. O bloco 4, utilizado para o critério de correspondência *Notes with Octaves*, faz com que sejam selecionados registros da tabela t_0 (`tabs_transicoes`) onde o valor de `tom` somado ao valor de `oitava` multiplicado por `12` seja igual a diferença de altura δ_h da primeira distância entre notas *d* de *w*, que consistirá de uma distância entre notas envolvendo a nota falsa. O bloco 5, do critério *Distances with Octaves*, faz com que sejam selecionados registros de t_0 onde o valor de `dif_tempo` seja igual a diferença de tempo δ_t e algum dos valores de `dif_tom_corda` seja igual a diferença de altura δ_h da primeira distância entre notas *d* de *w*. O bloco 6, do critério *Only Notes*, faz com que sejam selecionados de t_0 registros onde o valor de `tom` seja igual a diferença de altura δ_h da primeira distância entre notas *d* de *w*, que irá envolver a nota falsa. Já o bloco 7, utilizado para o critério de correspondência *Only Distances*, faz com que sejam selecionados registros de t_0 onde o valor de `dif_tempo` seja igual a diferença de tempo δ_t, e o resto da divisão por *12* da soma entre algum dos valores de `dif_tom_corda` com o valor *132* seja igual ao resto da divisão por *12* da soma da diferença de altura δ_h da primeira distância entre notas *d* de *w* com o valor *12*. As somas com os valores *132* e *12* que ocorrem no bloco 7 visam evitar que o resultado dos cálculos resultem em um valor negativo, para o caso de valores negativos de alguma coluna `dif_tom_corda` ou diferença de altura δ_h. Na sequência dos blocos 4 e 5, o bloco 8 estabelece condições para as autojunções da tabela `tabs_transicoes`, onde os valores de `id_documento` devem ser iguais; o `tempo` de t_i somado a diferença de tempo δ_t da *i*-ésima distância entre notas *d* de *w* deve ser igual ao valor de `tempo` de t_{i+1}; e a soma entre o valor de `tom` de t_i, o valor de `oitava` de t_i multiplicado por *12* e a diferença de altura δ_h da *i*-ésima distância entre notas *d* de *w*, deve ser igual a soma entre o valor de `tom` de t_{i+1} com o valor de `oitava` de t_{i+1} multiplicado por *12*. Assim como o bloco 8, o bloco 9, na sequência dos blocos 6 e 7, estabelece condições para as autojunções da tabela `tabs_transicoes`, onde os valores de `id_documento` devem ser iguais; o `tempo` de t_i somado a diferença de tempo δ_t da *i*-ésima distância entre notas *d* de *w* deve ser igual ao valor de `tempo` de t_{i+1}; e o resto da divisão por *12* da soma entre o valor de `tom` de t_i, a diferença de altura δ_h da *i*-ésima distância entre notas *d* de *w*, e o valor *12*, deve ser igual ao valor de `tom` de t_{i+1}. Ao final da montagem da consulta, o bloco 10 condiciona que a junção da última tabela `tabs_transicoes` deva ter o valor de `id_documento` igual ao valor de `id_documento` da tabela `coletas_docs`. Para buscar composições utilizando correspondências exclusivamente em trechos monofônicos, é adicionada uma condição em cada transição, determinando que a coluna `mono` seja verdadeira (igual a *1*).

5.3.4 Apresentação de Resultados

Após o Fluxo de Busca de Composições recuperar documentos contendo tablaturas textuais consideradas similares a tablatura textual de consulta, tais resultados são exibidos em uma listagem.

Mesmo a tablatura textual de consulta sendo monofônica, uma grande quantidade de monofonias é gerada, fazendo com que uma grande quantidade de consultas SQL tenham que ser montadas e executadas na Base de Dados, a fim de obter-se os resultados. O tempo total decorrido para a conclusão de todo este processo, parte

de uma única busca, mostra-se demasiadamente longo, na ordem de minutos, mesmo para um protótipo.

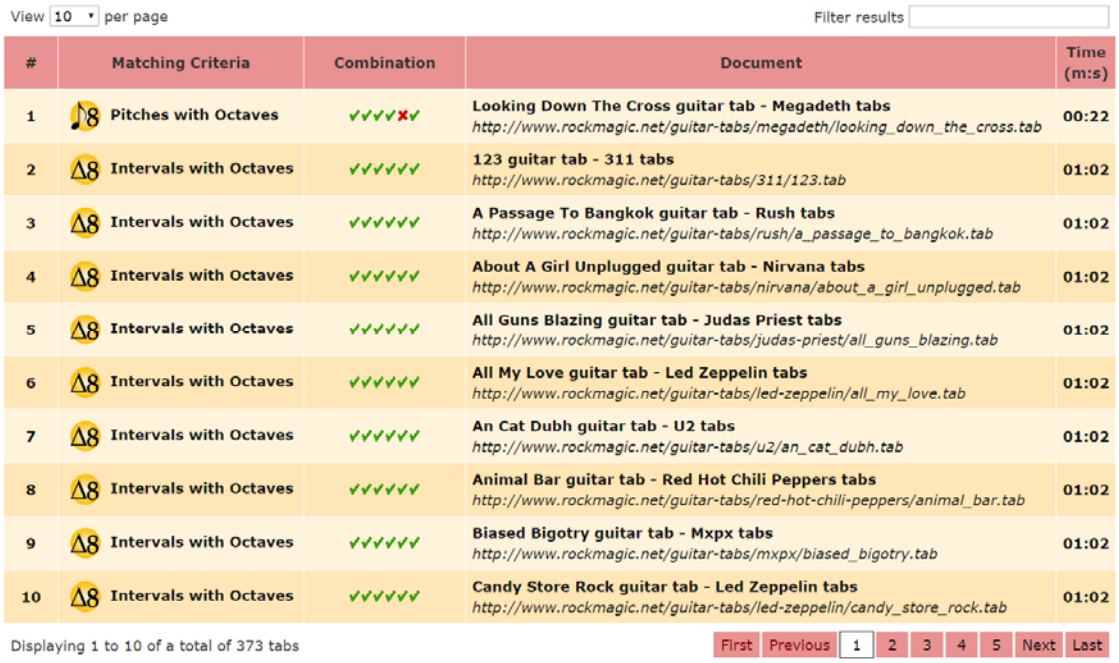
Para atenuar o problema da demora na apresentação de resultados das buscas, o protótipo apresenta tais resultados de maneira parcelada, retornando ao usuário os resultados de cada consulta SQL à medida que ela é executada. Na implementação deste retorno em parcelas, foi utilizada a biblioteca ASP.NET SignalR[9], que permite realizar atualizações assíncronas do lado cliente em uma aplicação *web* dinâmica, ou seja, permite enviar dados do servidor para o cliente mesmo sem o cliente realizar requisições de maneira explícita, abstraindo uma conexão persistente em ambiente *web*.

Os seguintes fatores influenciam no ranqueamento dos resultados da ferramenta TabSim:

- Quantidade de distâncias entre notas das monofonias – as monofonias são ordenadas de maneira decrescente pela sua quantidade de distâncias entre notas. Monofonias com mesma quantidade de distâncias entre notas são ordenadas de modo que monofonias com distâncias entre notas mais à esquerda, em relação à consulta, fiquem no início;
- Critérios de correspondência (*Matching Criteria*) – para cada monofonia é montada e executada uma consulta SQL para cada um dos critérios de correspondência selecionados na ferramenta, iniciando pelo critério *Pitches with Octaves* (Tons com oitavas), seguido de *Intervals with Octaves* (Intervalos com oitavas), *Only Pitches* (Somente tons) e *Only Intervals* (Somente intervalos);
- Alternância de erros (*Toggle of erros*) – determina de quantos e quantos erros os critérios de correspondência selecionados devem ser alternados, sendo um erro a ausência do envolvimento de uma nota em uma monofonia;
- Melodias monofônicas primeiro (*Monophonic melodies first*) – se utilizada esta opção na ferramenta TabSim, trechos monofônicos correspondidos por uma monofonia irão aparecer mais no topo dos resultados;
- Título da composição – composições retornadas por uma mesma monofonia são ordenadas alfabeticamente de acordo com o título da composição.

A Figura 5.8 mostra a interface de apresentação de resultados do Buscador de Tablaturas Textuais da *Web*, na forma de uma listagem paginada.

[9] Informações em: http://www.asp.net/signalr


View 10 per page

Filter results

#	Matching Criteria	Combination	Document	Time (m:s)
1	Pitches with Octaves	✓✓✓✓✗✓	**Looking Down The Cross guitar tab - Megadeth tabs** *http://www.rockmagic.net/guitar-tabs/megadeth/looking_down_the_cross.tab*	00:22
2	Intervals with Octaves	✓✓✓✓✓✓	**123 guitar tab - 311 tabs** *http://www.rockmagic.net/guitar-tabs/311/123.tab*	01:02
3	Intervals with Octaves	✓✓✓✓✓✓	**A Passage To Bangkok guitar tab - Rush tabs** *http://www.rockmagic.net/guitar-tabs/rush/a_passage_to_bangkok.tab*	01:02
4	Intervals with Octaves	✓✓✓✓✓✓	**About A Girl Unplugged guitar tab - Nirvana tabs** *http://www.rockmagic.net/guitar-tabs/nirvana/about_a_girl_unplugged.tab*	01:02
5	Intervals with Octaves	✓✓✓✓✓✓	**All Guns Blazing guitar tab - Judas Priest tabs** *http://www.rockmagic.net/guitar-tabs/judas-priest/all_guns_blazing.tab*	01:02
6	Intervals with Octaves	✓✓✓✓✓✓	**All My Love guitar tab - Led Zeppelin tabs** *http://www.rockmagic.net/guitar-tabs/led-zeppelin/all_my_love.tab*	01:02
7	Intervals with Octaves	✓✓✓✓✓✓	**An Cat Dubh guitar tab - U2 tabs** *http://www.rockmagic.net/guitar-tabs/u2/an_cat_dubh.tab*	01:02
8	Intervals with Octaves	✓✓✓✓✓✓	**Animal Bar guitar tab - Red Hot Chili Peppers tabs** *http://www.rockmagic.net/guitar-tabs/red-hot-chili-peppers/animal_bar.tab*	01:02
9	Intervals with Octaves	✓✓✓✓✓✓	**Biased Bigotry guitar tab - Mxpx tabs** *http://www.rockmagic.net/guitar-tabs/mxpx/biased_bigotry.tab*	01:02
10	Intervals with Octaves	✓✓✓✓✓✓	**Candy Store Rock guitar tab - Led Zeppelin tabs** *http://www.rockmagic.net/guitar-tabs/led-zeppelin/candy_store_rock.tab*	01:02

Figura 5.8: Interface dos resultados do Buscador de Tablatura Textuais da Web.

A interface de apresentação de resultados, mostrada na Figura 5.8, possui uma listagem paginada, além de controles para determinar a quantidade de resultado por página (canto superior esquerdo), filtro de resultados da listagem (canto superior direito), e navegação entre as páginas (canto inferior direito). A interface também permite ordenar a listagem de acordo com os valores de uma determinada coluna ao clicar em seu cabeçalho. A listagem possui informações de ordenação do resultado (coluna #); critério de correspondência utilizado na obtenção do resultado (coluna *Matching Criteria*); representação da monofonia utilizada, tendo um ícone associado a cada nota tablatura textual monofônica de consulta (coluna *Combination*); título e URL do documento considerado similar (coluna *Document*); e tempo de busca (coluna *Time*).

5.4 Considerações Finais do Capítulo

O buscador TabSim permite coletar documentos contendo tablaturas textuais da *web*, posteriormente possibilitando a realização de buscas de tais documentos através do método Correspondência de Monofonia Contida. O protótipo descrito neste capítulo tem como principal objetivo demonstrar a viabilidade de implementação do método CMC e servir como ferramenta base para experimentos.

Mesmo a ferramenta sendo um protótipo voltado à demonstração do método CMC, é importante salientar que *websites* que disponibilizam tablaturas textuais não possuem mecanismos para busca baseada no conteúdo musical dos documentos, normalmente possuindo somente buscas por palavras-chave, o que não demonstra-se interessante quando não se conhece informações como o compositor, letra ou título da composição.

Também é interessante entender o posicionamento da ferramenta TabSim em relação às ferramentas estudadas nos trabalhos relacionados (seção 3.2). A Tabela 5.1 apresenta características da ferramenta TabSim (linha 5), juntamente com as ferramentas C-Brahms, Kooplet, Meldex e Musipedia.

Tabela 5.1: TabSim e outras ferramentas estudadas.

#	Ferramenta	Interfaces	Base Particular	Busca na *Web*	Consulta Polifônica	Ajuste Params Corresp.	Analise Tablaturas
1	C-Brahms	1	X		X	X	
2	Kooplet	1		X			X
3	Meldex	2	X				
4	Musipedia	5	X	X	X		
5	**TabSim**	**1**		**X**		**X**	**X**

Como mostra a Tabela 5.1, assim como as ferramentas C-Brahms e Kooplet, a TabSim possui somente uma interface, com a diferença básica de utilizar a representação do braço de uma guitarra no lugar de um teclado virtual, justamente devido seu foco em tablaturas textuais. Outra semelhança da TabSim com a Kooplet refere-se às buscas serem realizadas somente na *web*, ambas não possuindo base particular. A desvantagem mais significativa da ferramenta TabSim em relação às outras, mais especificamente a C-Brahms e Musipedia, refere-se a não possibilidade do uso de consultas polifônicas na realização de buscas, mesmo o método CMC cobrindo esta possibilidade. Pode-se considerar como uma vantagem da TabSim, juntamente com a C-Brahms, dispor ao usuário controles para ajuste de parâmetros relacionados aos padrões de correspondência do método de busca utilizado. A última característica apresentada na Tabela 5.1, das ferramentas analisar tablaturas, é muito específica para ser considerada uma real vantagem, mas não deixa de ser um diferencial da TabSim, juntamente com a Kooplet, em relação às outras três. Mesmo a ferramenta Kooplet também analisar tablaturas em suas buscas, é interessante destacar que ela não analisa o conteúdo de documentos contendo tablaturas textuais, trabalhando somente com tablaturas em formatos específicos, como XML (*Extensible Markup Language*) e PDF (*Portable Document Format*).

É evidente que existe espaço para melhorias na implementação da ferramenta TabSim, descrita neste capítulo. Na ação de coleta pode-se implementar algoritmos mais avançados no *crawler* para uma navegação mais ágil. Já a ação de busca requer soluções mais avançadas, principalmente para melhora da performance, para que seja possível utilizar consultas polifônicas e aumento do valor do limite *k*, atualmente 3.

6 EXPERIMENTOS

Este capítulo descreve os experimentos realizados na ferramenta TabSim a fim de avaliar o método Correspondência de Monofonia Contida (CMC), proposto por este trabalho. Inicialmente, foram coletados documentos da *web* contendo tablaturas textuais, populando a base de dados de composições, utilizada na realização dos experimentos. Dois grupos de experimentos foram realizados, sendo um preliminar, responsável por determinar as configurações da ferramenta TabSim e intervalo de quantidade de notas para as consultas do segundo grupo, realizado com usuários.

6.1 Base de Dados para Experimentos

Para popular a base de dados de composições, inicialmente, realizou-se uma pesquisa por *websites* que contivessem um conjunto razoável de documentos contendo tablaturas textuais. *Websites* contendo poucos documentos com tablaturas textuais foram descartados, pois restringiriam muito as opções do usuário nas buscas e reduziriam as chances de existirem composições com trechos de melodias em comum. Já *websites* contendo muitos documentos com tablaturas textuais também foram descartados, pois seria despendido muito tempo na coleta desnecessariamente, visto que o objetivo da população da base de dados foi obter uma quantidade de composições considerada suficiente para a execução dos experimentos. Então, dentre os diversos *websites* que disponibilizam documentos contendo tablaturas textuais, optou-se pelo RockMagic[10], um *website* focado em tablaturas musicais do gênero *rock*. A Figura 6.1 mostra a página inicial do *website* RockMagic.

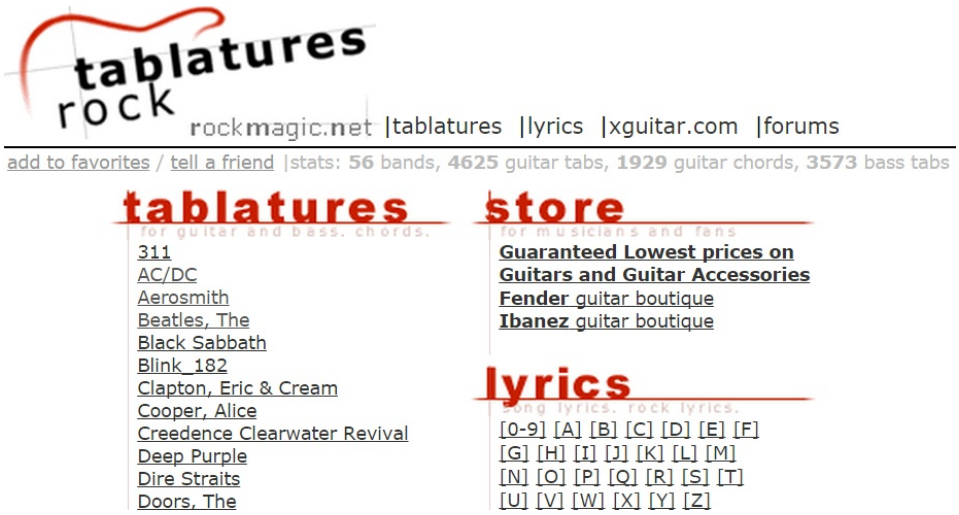


Figura 6.1: Página inicial do website de tablaturas textuais RockMagic.

[10] Disponível em http://www.rockmagic.net

Determinado o *website* de onde seriam coletadas as informações para popular a base de dados, foi executada a ação de coleta (seção 5.2) do Buscador de Tablaturas Textuais da *Web* em um Notebook com Processador Core2 Duo 2.26 GHz com 4GB de memória RAM conectado em um *link* de Internet com banda de 10 Mb/s (megabits por segundo). Após aproximadamente 42 horas de execução, analisando 10495 documentos, a ação de coleta recuperou 4828 contendo tablaturas textuais de guitarra, extraindo seus dados para a base de dados de composições. A quantidade de documentos recuperados contendo tablaturas textuais de guitarra (4828) é superior à quantidade de tablaturas informada pelo *website* (4625), devido alguns documentos de tablaturas para baixo também conter pequenos trechos de tablatura de guitarra. A Tabela 6.1 apresenta mais alguns números da ação de coleta realizada para popular a base de dados de composições.

Tabela 6.1: Números da ação de coleta para população da base de dados.

Informação	Valor
Documentos analisados	10.495
Documentos contendo tablaturas textuais de guitarra	4.828
Tempo total da ação de coleta (em horas)	41,59
Tempo médio de extração de tablatura de documento (em segundos)	24,62
Quantidade total de notas extraídas	1.135.686
Quantidade total de registros de transições ($k = 3$)	3.407.058
Média de notas por tablatura	235,57
Média de tamanho de tablatura (maior tempo com notas)	167,04
Média de notas por tempo	1,41
Tons mais frequentes em notas (em ordem)	E, A, D, G, B, C, F#, F, C#, A#, G#, D#
Oitavas mais frequentes em notas (em ordem)	4ª, 5ª, 3ª, 6ª, 7ª
Cordas do instrumento usadas mais frequentemente (em ordem)	G (3ª), D (4ª), A (5ª), B (2ª), e (1ª), E (6ª)
Tamanho da base de dados populada (em *Megabytes*)	179,34

Realizado o processo de coleta de tablaturas textuais e população da base de dados de composição, iniciou-se a execução dos experimentos preliminares.

6.2 Experimentos Preliminares

Os primeiros experimentos realizados na ferramenta TabSim tiveram como objetivo determinar quais valores para o painel de configurações da ferramenta TabSim e intervalo de quantidade de notas produzem os melhores resultados. Tais configurações e intervalo de quantidade de notas, resultantes destes experimentos preliminares, foram utilizados posteriormente nos experimentos com usuários.

6.2.1 Metodologia

Para determinar quais configurações e quais intervalos de quantidade de notas produzem os melhores resultados, foram realizados diversos experimentos, utilizando diferentes consultas, com diferentes combinações de configurações.

Para que os resultados dos experimentos pudessem ser comparados, foram definidas diversas consultas baseadas em uma única tablatura textual, determinada previamente através de um levantamento na base de dados.

Cada uma das consultas definidas para os experimentos foi submetida à ferramenta TabSim, utilizando diferentes configurações, coletando-se a posição da tablatura textual esperada nos resultados. Não seria viável realizar experimentos com todas as combinações de configurações disponíveis na ferramenta, então foram definidos valores fixos para algumas opções:

- *Only in monophonic melodies* – opção não utilizada, possibilitando a correspondência da consulta em trechos polifônicos;
- *Matching Criteria* – optou-se por utilizar somente os critérios de correspondência *Pitches with Octaves* e *Intervals with Octaves*, devido serem mais precisos;
- *Jumps in sequence* – opção utilizada para eliminar monofonias com trechos de não correspondências consecutivos (lacunas consecutivas);
- *% of notes lower than* – para que o percentual mínimo de notas não influenciasse nos experimentos preliminares, seu valor foi definido como 0%;
- *Monophonic melodies first* – opção utilizada para que resultados de correspondências em trechos monofônicos para uma mesma monofonia sejam colocados mais no topo.

A única opção variável durante os experimentos foi a alternância de critérios de correspondência de acordo com a quantidade de erros nas monofonias (*toggle of errors*). Foram realizados experimentos com valores de 1 até 4 para *toggle of errors*. Valores de *toggle of errors* alteram significativamente a posição dos resultados no ranqueamento final.

Ao final da execução destes experimentos preliminares, os dados foram analisados, utilizando a medida de avaliação de ranqueamento *Mean Reciprocal Rank* (MRR). O MRR é uma métrica que favorece ranqueamentos cujo primeiro resultado correto ocorra perto do topo (YATES, 2011). Para calcular o MRR, foi calculado inicialmente o *Reciprocal Rank* (RR) para cada experimento de cada configuração. A equação 6.1 mostra a função RR, onde $rank_q$ representa a posição do resultado esperado no ranqueamento de resultados e L um limiar arbitrário.

$$RR(q) = \begin{cases} \frac{1}{rank_q} & : rank_q \leq L \\ 0 & : rank_q > L \end{cases} \qquad (6.1)$$

A medida MRR, mostrada na equação 6.2, consiste na média aritmética dos RR de um grupo de experimentos Q.

$$MRR(Q) = \frac{1}{|Q|} \sum_{i \in [|Q|]} RR(q_i) \qquad (6.2)$$

Optou-se por pelo MRR, devido medidas comumente utilizadas para avaliar resultados de experimentos de recuperação de informação, como precisão e revocação, não poderem ser utilizadas perfeitamente no contexto da busca de composições musicais devido, geralmente, o usuário estar à procura de uma composição específica.

6.2.2 Definição de Consultas

Inicialmente, foi necessário determinar qual das tablaturas textuais armazenadas na base de dados serviria como base para a definição de consultas e realização dos experimentos preliminares.

Devido a ferramenta TabSim ordenar alfabeticamente pelo título resultados obtidos por uma mesma monofonia, foi realizado um levantamento na base de dados de modo a encontrar a tablatura textual que minimizasse a influência deste fator de ordenação nos resultados dos experimentos preliminares. Tal levantamento consistiu em determinar, sucessivamente, a cada posição de caractere nos títulos, o caractere mediano, sempre filtrando pelos caracteres medianos previamente encontrados. Após nove medianas, como mostra a Tabela 6.2, restou somente uma tablatura textual, a composição "*Little Dreamer*", da banda de rock Van Halen, sendo esta a selecionada para realização dos experimentos preliminares.

Tabela 6.2: Levantamento para encontrar a tablatura central da base de dados.

Char	Tabs	Distribuição	Mediana
1º	4828	18 '*1*', 15 '*2*', 3 '*3*', 4 '*4*', 4 '*5*', 1 '*7*', 4 '*8*', 1 '*9*', 247 '*A*', 328 '*B*', 256 '*C*', 295 '*D*', 110 '*E*', 182 '*F*', 166 '*G*', 221 '*H*', 264 '*I*', 63 '*J*', 60 '*K*', 236 '*L*', 252 '*M*', 138 '*N*', 126 '*O*', 198 '*P*', 5 '*Q*', 198 '*R*', 515 '*S*', 531 '*T*', 45 '*U*', 41 '*V*', 224 '*W*', 3 '*X*', 67 '*Y*', 7 '*Z*'	2414ª tab '*L*'
2º	236	1 ' ', 35 '*A*', 53 '*E*', 49 '*I*', 84 '*O*', 14 '*U*'	118ª tab '*I*'
3º	49	3 '*A*', 2 '*C*', 6 '*F*', 7 '*G*', 3 '*K*', 2 '*M*', 10 '*T*', 16 '*V*'	24ª tab '*T*'
4º	10	10 '*T*'	5ª tab '*T*'
5º	10	10 '*L*'	5ª tab '*L*'
6º	10	10 '*E*'	5ª tab '*E*'
7º	10	9 ' ', 1 '*W*'	5ª tab ' '
8º	9	2 '*B*', 2 '*D*', 1 '*G*', 1 '*L*', 1 '*S*', 1 '*T*', 1 '*W*'	4ª tab '*D*'
9º	2	1 '*R*', 1 '*U*'	1ª tab '*R*'

Na Tabela 6.2, a coluna *Char* indica a posição de caractere analisada no levantamento, *Tabs* a quantidade de tablaturas analisadas, *Distribuição* a quantidade encontrada de cada caractere para a posição indicada por *Char*, e *Mediana* o caractere mediano da posição *Char*.

Determinada a tablatura textual, foram criadas duas consultas monofônicas de 12 notas a partir de um trecho polifônico presente nesta tablatura: uma idêntica ao trecho (consulta Tom Original) e outra transposta em outro tom (consulta Transposta), ambas excluindo notas sucessivas repetidas. A Figura 6.2 apresenta o trecho escolhido da tablatura "*Little Dreamer*" e as consultas Tom Original e Transposta.

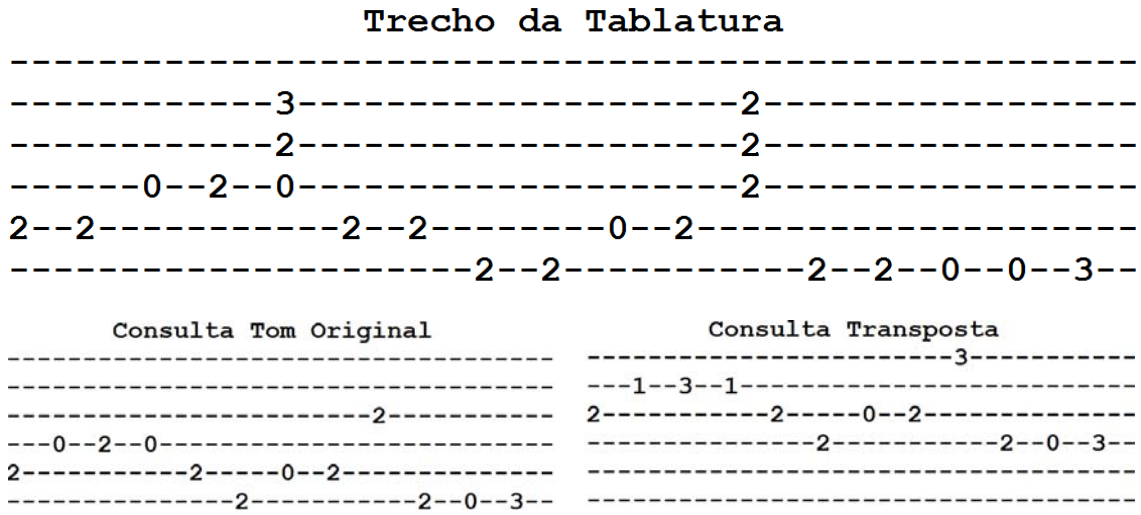


Figura 6.2: Consultas escolhidas para testes para definição de formato de consulta.

As consultas Tom Original e Transposta, mostradas na Figura 6.2, serviram como base para a criação de outras 62 consultas derivadas (31 de cada), variando a quantidade de notas, de 5 até 12, e quantidade de erros, de 0 até 3. Todas as variações das consultas Tom Original e Transposta foram realizadas no final, ou seja, mais a direita das consultas.

Para determinar quais tons poderiam ser utilizados como erros para a tablatura "*Little Dreamer*", foi realizado um levantamento de tons presentes na composição. A Tabela 6.3 mostra os resultados deste levantamento.

Tabela 6.3: Levantamento de tons presentes na tablatura textual "Little Dreamer".

Tom	C (0)	C# (1)	D (2)	D# (3)	E (4)	F (5)	F# (6)	G (7)	G# (8)	A (9)	A# (10)	B (11)
Quantidade	0	25	42	0	54	0	42	12	0	48	14	55

Nota-se que, no levantamento mostrado na Tabela 6.3, a tablatura textual "*Little Dreamer*" não possui notas com os tons C (0), D# (3), F (5) e G# (8), caracterizando uma composição em escala menor de B, com variações na escala menor harmônica, também de B. Estes tons, não presentes na composição, foram utilizados como erros nas consultas baseadas na consulta Tom Original. Para as consultas baseadas na consulta Transposta os tons dos erros também foram transpostos (10 semitons acima).

6.2.3 Execução e Resultados

O primeiro grupo dos experimentos preliminares utilizou todas as 32 combinações de consultas derivadas da consulta Tom Original (Figura 6.2), totalizando 128 experimentos, sendo 32 para cada valor diferente de alternância (*toggle*).

A Figura 6.3 mostra as posições da tablatura textual esperada "*Little Dreamer*" no *ranking* de resultados, para os respectivos experimentos com consultas derivadas da consulta Tom Original. Analisando os gráficos da Figura 6.3, pode-se facilmente inferir que a posição no *ranking* é inversamente proporcional à quantidade de notas corretas da consulta (quantidade de notas menos a quantidade de erros) para um mesmo valor de *toggle*, fazendo com que os melhores resultados tendam ao canto superior esquerdo da tabela de dados. Também é possível concluir que, no caso da consulta estar no tom

original, a posição dos resultados, para consultas com quantidade de erros menor do que o valor de alternância (*toggle*), é inversamente proporcional a este valor de alternância.

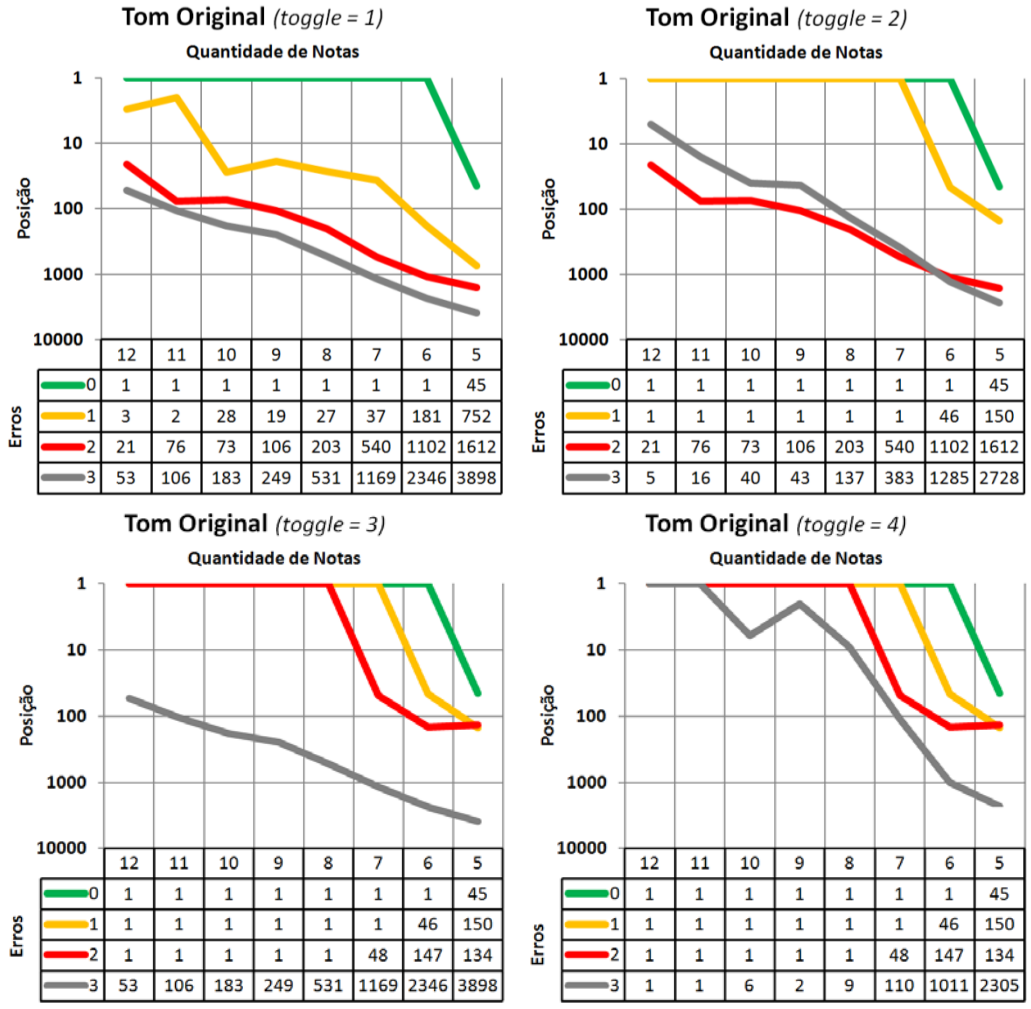


Figura 6.3: Resultados dos experimentos preliminares com consultas no Tom Original.

O segundo grupo dos experimentos preliminares também utilizou 32 consultas, porém, derivadas da consulta Transposta (Figura 6.2), totalizando 128 experimentos, sendo 32 para cada valor diferente de alternância (*toggle*). A Figura 6.4 mostra as posições da tablatura textual esperada "*Little Dreamer*" no *ranking* de resultados, para os respectivos experimentos com consultas derivadas da consulta Transposta. Assim como nos experimentos com a consulta Tom Original, os gráficos da Figura 6.4 mostram que a posição no *ranking* é inversamente proporcional à quantidade de notas corretas da consulta (quantidade de notas menos a quantidade de erros) para um mesmo valor de *toggle*. Diferentemente dos experimentos com consultas no tom original, nota-se uma semelhança nos resultados dos diferentes valores de *toggle* dos experimentos com consultas transpostas. Esta semelhança deve-se ao fato de que consultas transpostas irão corresponder somente quando utilizado o critério de correspondência *Intervals with Octaves*, enquanto consultas no tom original podem corresponder com os dois critérios utilizados nestes experimentos, *Pitches with Octaves* e *Intervals with Octaves*.

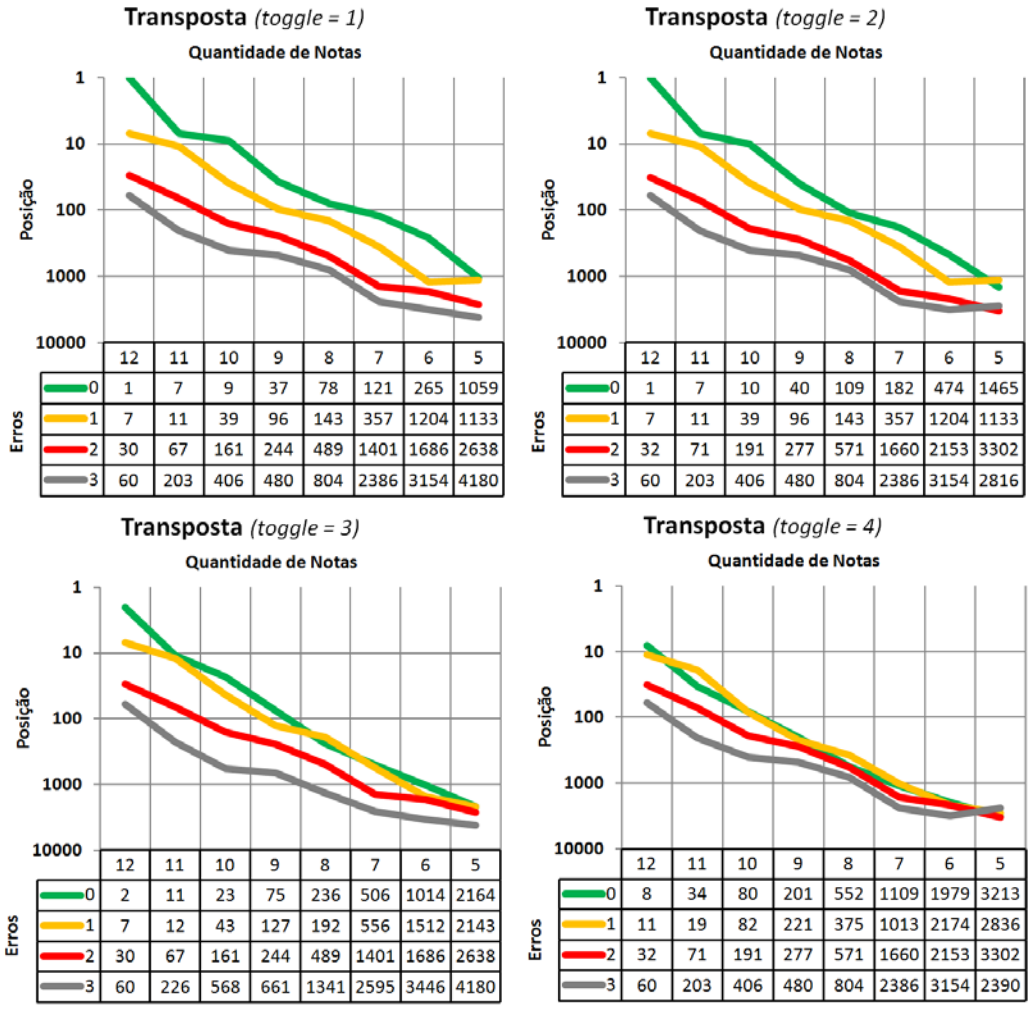


Figura 6.4: Resultados dos experimentos preliminares com consultas Transpostas.

Para determinar quais seriam os intervalos de quantidade de notas e quantidade de erros permitidos nos experimentos com usuários, foram calculados valores de MRR para três limites *L*, determinando os intervalos de ranqueamento utilizados: MRR das primeiras 10 posições (MRR@10), determinando um intervalo de ranqueamento geralmente utilizado em avaliações de recuperação de informação (RI); MRR das primeiras 25 posições (MRR@25), sendo um intervalo de ranqueamento intermediário; e MRR das primeiras 50 posições (MRR@50), escolhido com base no valor relativo a 1% da quantidade de composições da base de dados de experimentos (arredondado na dezena).

Primeiramente, foram calculados os valores de MRR, nos três limites de *L*, dos resultados de consultas com mesma quantidade de notas e mesma quantidade de erros. Também, para cada intervalo de ranqueamento, foram calculadas as médias dos valores de MRR por quantidade de notas e a média geral dos valores de MRR.

A Figura 6.5 mostra os valores de MRR referentes ao primeiro intervalo de ranqueamento (MRR@10), tendo as seguintes médias de valores de MRR, de 12 a 5 notas respectivamente: 0.42, 0.34, 0.29, 0.30, 0.28, 0.22, 0.13, e 0.00. Já a média geral dos valores de MRR ficou em 0.25 para este primeiro intervalo de ranqueamento. É possível observar que, em MRR@10, somente valores de MRR para 0, 1 e 2 erros

ficaram superiores ou iguais a média geral, e que somente as médias de valores de MRR para 12, 11, 10, 9 e 8 quantidade de notas ficaram acima da média geral.

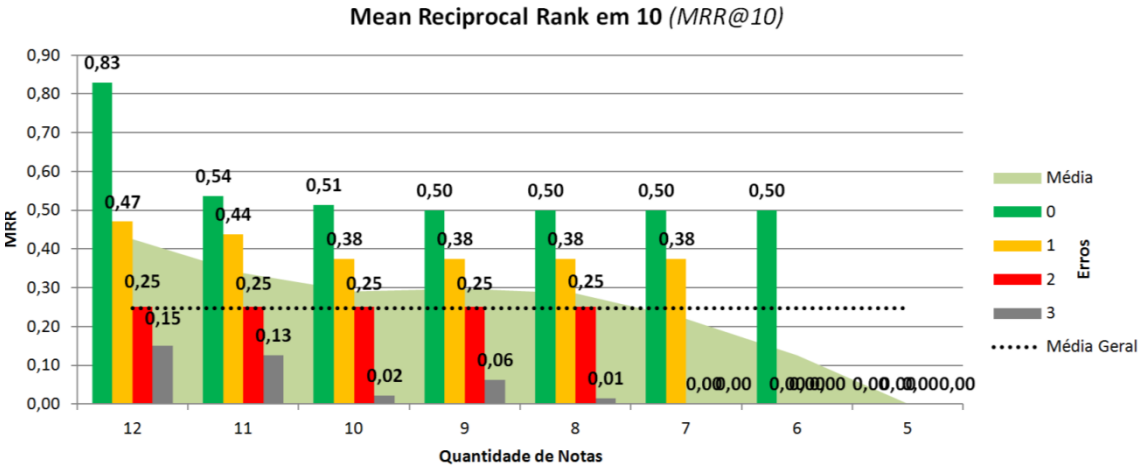


Figura 6.5: Valores de MRR para limite 10 (MRR@10).

Na Figura 6.6 pode-se visualizar os valores de MRR para o segundo intervalo de ranqueamento (MRR@25), tendo as seguintes médias de valores de MRR, de 12 a 5 notas respectivamente: 0.43, 0.35, 0.29, 0.30, 0.28, 0.22, 0.13, e 0.00. Assim como em MRR@10, a média geral dos valores de MRR@25 ficou em 0.25, apenas os valores de MRR para 0, 1 e 2 erros ficaram superiores ou iguais a média geral, e somente as médias de valores de MRR para 12, 11, 10, 9 e 8 quantidade de notas ficaram acima da média geral.

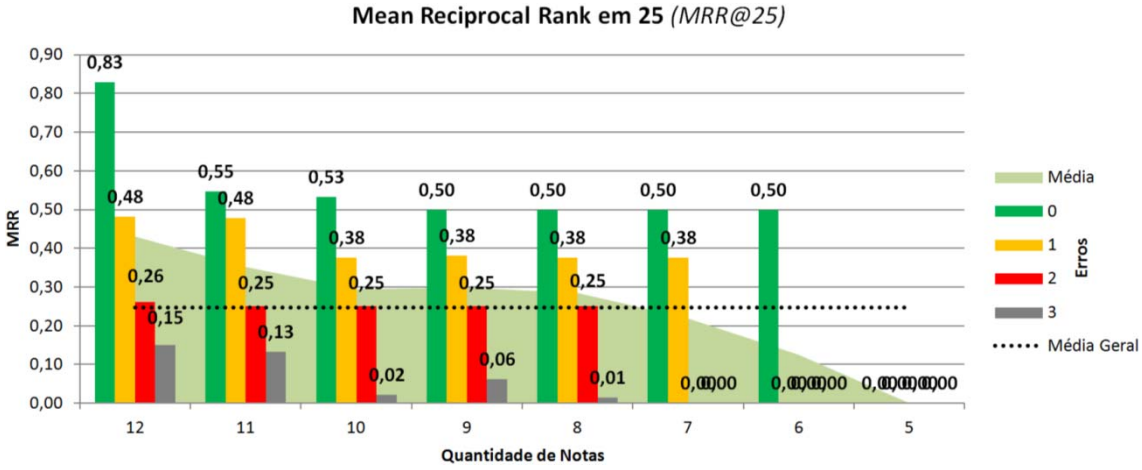


Figura 6.6: Valores de MRR para limite 25 (MRR@25).

Os valores do último intervalo de ranqueamento (MRR@50) podem ser visualizados na Figura 6.7. As médias de valores de MRR, para consultas de 12 a 5 notas foram respectivamente: 0.43, 0.35, 0.30, 0.30, 0.29, 0.22, 0.13, e 0.00. Assim como os intervalos de ranqueamento anteriores, a média geral dos valores de MRR@50 ficou em 0.25. Também, apenas os valores de MRR para 0, 1 e 2 erros ficaram superiores ou iguais a média geral, e somente as médias de valores de MRR para 12, 11, 10, 9 e 8 quantidade de notas ficaram acima da média geral.

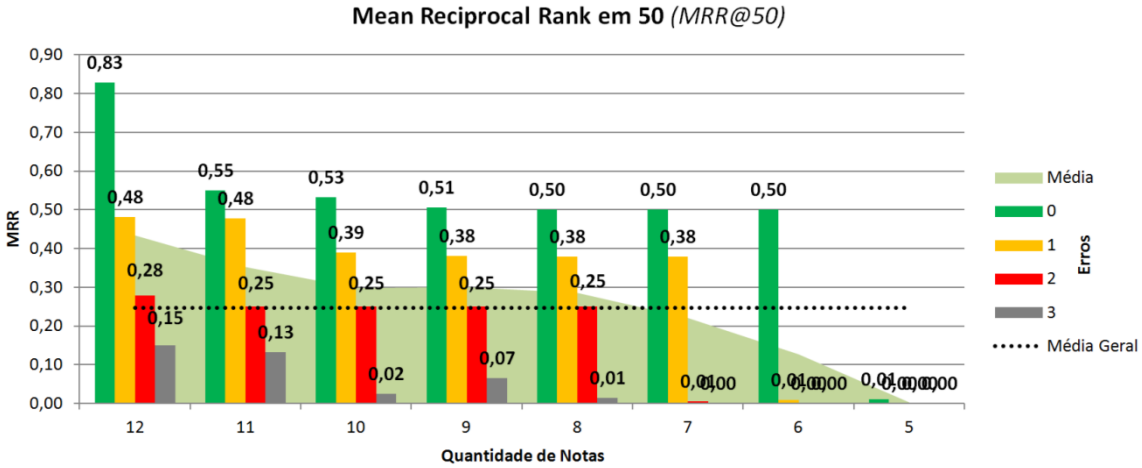


Figura 6.7: Valores de MRR para limite 50 (MRR@50).

Após a análise individual de valores MRR nos intervalos de ranqueamento 10, 25 e 50, ficou clara a quebra na qualidade do ranqueamento dos resultados na passagem da quantidade de notas 8 para a quantidade 7, e na passagem de 2 para 3 erros. A baixa qualidade no ranqueamento do resultado esperado nos resultados para consultas com quantidade menor de notas pode ser explicada pelo fato de que, quanto menor a consulta, menos restritiva ela é, sendo maior a probabilidade de que ela corresponda em uma composição, fazendo o resultado esperado ficar disperso em meio a outras composições recuperadas.

Estipulou-se então, para os experimentos com usuários, um intervalo de quantidade de notas permitido em consultas variando de 8 até 12 notas, e quantidade de erros variando de 0 até 2.

Determinado o intervalo de quantidade de notas e intervalo de erros permitidos nas consultas com usuários, buscou-se escolher o melhor valor de alternância de critérios de correspondência (*toggle*). Para possibilitar uma comparação entre os resultados obtidos com diferentes valores de *toggle*, foram calculados os MRR, nos três limites de *L*, dos resultados de consultas de mesmo *toggle*, incluindo consultas com tom original e consultas transpostas. É importante ressaltar que somente foram utilizados os resultados de consultas com quantidade de notas de 8 até 12 e quantidade de erros de 0 até 2, valores estes estipulados pelos primeiros experimentos preliminares descritos nesta seção. Também, para cada intervalo de ranqueamento, foram calculadas as médias dos valores de MRR por valor de *toggle* e a média geral dos valores de MRR.

A Figura 6.8 mostra os valores de MRR para cada valor de *toggle* nos intervalos de ranqueamento 10, 25 e 50. As médias de valores de MRR por intervalo de ranqueamento MRR se mantiveram praticamente constantes, sendo respectivamente, para MRR@10, MRR@25 e MRR@50: 0.41, 0.42, e 0.42. A média geral dos valores de MRR ficou em 0.42. É possível observar que, em todos os intervalos de ranqueamento, somente valores de MRR para *toggle* 3 e 4 ficaram superiores à média geral. Mesmo o MRR do *toggle* 4 tendo obtido bons resultados nos experimentos, é evidente a superioridade dos resultados com *toggle* 3, tendo sido este o valor escolhido para a realização dos experimentos com os usuários.

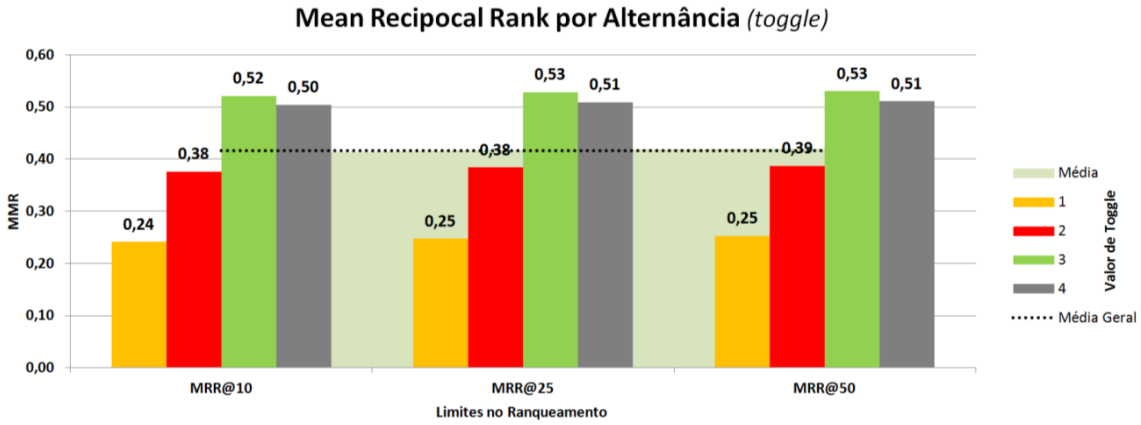


Figura 6.8: Valores de MRR por Alternância (toggle).

Finalizados os experimentos preliminares, ficaram definidas as configurações e intervalo de quantidade de notas permitido nos experimentos com usuários:

- Intervalo de quantidade de notas – mínimo de 8 e máximo de 12 notas para uma consulta;
- Intervalo de erros – mínimo de 0 e máximo de 2 erros em uma busca;
- *Matching Criteria* – utilizado somente os critérios de correspondência *Pitches with Octaves* e *Intervals with Octaves*, devido serem mais precisos;
- *Toggle of errors* – definido o valor de 3 para a alternância dos critérios de correspondência;
- *% of notes lower than* – utilizando os intervalos de quantidade de notas e quantidade de erros definidos por estes experimentos preliminares, procurou-se determinar um percentual mínimo de notas para monofonias que se ajustasse a todos os casos, sendo definido o valor de 80%;
- *Only in monophonic melodies* – opção não utilizada, possibilitando a correspondência da consulta em trechos polifônicos;
- *Jumps in sequence* – opção utilizada para eliminar monofonias com trechos de não correspondências consecutivos (lacunas consecutivas);
- *Monophonic melodies first* – opção utilizada para que resultados de correspondências em trechos monofônicos para uma mesma monofonia sejam colocados mais no topo.

Os experimentos preliminares também foram importantes para identificar pequenos problemas de implementação do protótipo, permitindo que fossem feitas correções antes da execução dos experimentos com usuários.

6.3 Experimentos com Usuários

A experimentação com usuários é importante para coletar informações provenientes de comportamentos particulares às pessoas, principalmente quando envolve fatores subjetivos como os implícitos a percepção de similaridade melódica musical. A busca de composições não se encaixa perfeitamente em avaliações de recuperação de informação, não sendo possível o apontamento de diversos documentos relevantes em resultados, devido, geralmente, estar-se à procura de uma única resposta. Mesmo assim, a execução de experimentos com usuários demonstra-se importante, para

que seja possível avaliar resultados obtidos com o método CMC e o protótipo TabSim para diferentes perfis de pessoas.

6.3.1 Metodologia

Os experimentos com usuários se deram através da submissão de consultas com quantidade de notas entre 8 e 12, utilizando as configurações definidas nos experimentos preliminares. O objetivo destes experimentos foi levantar dados sobre buscas na ferramenta TabSim, de modo entender e avaliar resultados produzidos com o método CMC. Não fizeram parte destes experimentos com usuários aspectos relacionados à interface e usabilidade da ferramenta TabSim.

Este conjunto de experimentos contou com a participação de seis usuários, sendo todos eles músicos amadores, possuindo conhecimento prévio sobre tablaturas textuais, suficiente para o uso da ferramenta.

Para que a quantidade de buscas sem sucesso não fosse exageradamente alta, influenciando na avaliação do método e ferramenta, optou-se por informar aos usuários quais bandas/artistas possuíam tablaturas textuais na base de dados, não sendo disponibilizada nenhuma informação sobre as composições.

Antes da realização dos experimentos, os usuários foram alertados sobre alguns aspectos relacionados às consultas:

1. Somente seriam permitidas consultas contendo entre 8 e 12 notas, excluindo-se desta contagem notas consecutivas repetidas;
2. A posição da tablatura textual esperada nos resultados tende a ser maior para consultas menores;
3. Em casos onde uma melodia se repete na composição, é interessante também repetir o trecho na tablatura textual de consulta, de modo a determinar uma consulta maior;
4. Mesmo a consulta sendo monofônica, seriam buscadas correspondências em trechos polifônicos nas composições;
5. Consultas com mais de 2 erros não iriam retornar o resultado esperado.

Cada usuário realizou seis buscas, sempre indicando previamente qual tablatura textual esperada nos resultados. Para cada busca, foram coletadas todas as informações da página de resultados TabSim até a posição 50, juntamente com o trecho de tablatura de consulta. Devido à ferramenta demorar um tempo significativo para conclusão do processo de busca, determinou-se que a tablatura textual procurada deveria aparecer dentre os 50 primeiros resultados, caso contrário, seria considerada como tablatura não encontrada, mesmo com o processo de busca ainda incompleto.

A análise dos experimentos com usuários foi feita com base em valores absolutos e frequências de informações coletadas dos resultados.

6.3.2 Execução e Resultados

Ao finalizar a execução dos 36 experimentos com os usuários (seis com cada), realizou-se a tabulação de dados importantes para a análise, dados estes provenientes das 50 primeiras posições dos resultados de cada busca, sendo eles:

- Posição – posição do resultado esperado dentre as 50 posições de ranqueamento;

- Topo da mesma monofonia – posição do primeiro resultado recuperado pela mesma monofonia que o resultado esperado;
- Base da mesma monofonia – posição do último resultado recuperado pela mesma monofonia que o resultado esperado;
- Topo da mesma quantidade de erros – posição do primeiro resultado recuperado dentre as monofonias com mesma quantidade de erros da monofonia que recuperou o resultado esperado;
- Base da mesma quantidade de erros – posição do último resultado recuperado dentre as monofonias com mesma quantidade de erros da monofonia que recuperou o resultado esperado.

Dados de posições de topo e base da mesma monofonia que recuperou o resultado esperado permitem determinar o intervalo de posições influenciado pela ordenação alfabética pelo título, estabelecendo uma espécie de margem de variabilidade para o resultado esperado. Já os dados de posições de topo e base de monofonias com mesma quantidade de erros da monofonia que recuperou o resultado esperado permite determinar o intervalo de posições influenciado pela ordenação de monofonias (usado erros mais no final primeiro), estabelecendo uma margem de variabilidade mais externa para o resultado esperado do que a margem de variabilidade da mesma monofonia. A Figura 6.9 apresenta uma representação das margens de variabilidade de mesma monofonia (b) e variabilidade de mesma quantidade de erros (c) nos primeiros 50 resultados de uma busca (d), juntamente com a posição do resultado esperado (a).

Figura 6.9: Exemplo de margens de variabilidade para os resultados e uma busca.

A Figura 6.10 apresenta um gráfico contendo os dados de posição, margem de variabilidade de mesma monofonia, margem de variabilidade de mesma quantidade de erros. O gráfico também apresenta as seguintes médias: média de posição considerando todas as consultas submetidas (22.97); média de posição considerando somente as consultas submetidas que encontraram o resultado esperado (10.64); média do topo da margem de variabilidade de mesma monofonia considerando todas as consultas submetidas (21.42); média do topo da margem de variabilidade de mesma monofonia considerando somente as consultas submetidas que encontraram o resultado esperado (8.40); média do topo da margem de variabilidade de mesma quantidade de erros considerando todas as consultas submetidas (19.28); e média do topo da margem de variabilidade de mesma quantidade de erros considerando somente as consultas submetidas que encontraram o resultado esperado (5.32). Para que as médias considerando todas as consultas submetidas não fossem beneficiadas ao considerar a

posição do resultado esperado como zero, para o caso do resultado não estar dentre as 50 primeiras posições, utilizou-se o valor 51.

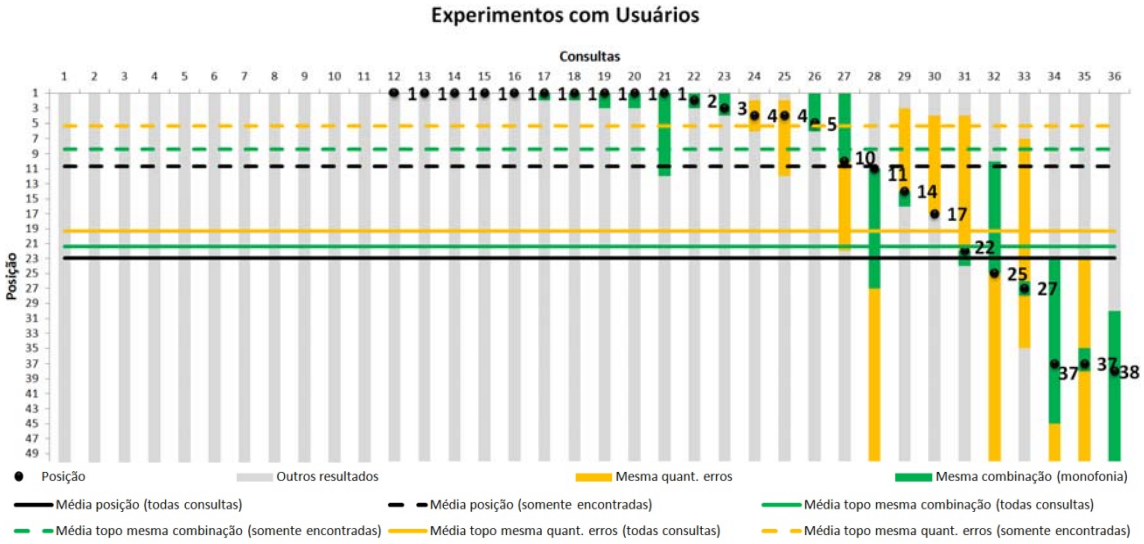


Figura 6.10: Resultados dos experimentos com usuários.

Analisando as médias de posição do resultado esperado, mostradas na Figura 6.10, é possível concluir que, mesmo considerando os resultados de consultas que não encontraram a tablatura textual procurada, o método CMC produz resultados razoáveis, posicionando o resultado esperado no início da terceira página de resultados, considerando páginas com 10 resultados cada. Considerando somente aquelas consultas onde o resultado esperado ficou entre os 50 primeiros, pode-se concluir que o método CMC produz resultados bons, posicionando o resultado esperado no início da segunda página de resultados, considerando páginas com 10 resultados cada. Ao analisar as médias das margens de variabilidade pode-se concluir que, o método CMC possui espaço para melhorias do posicionamento do resultado esperado no ranqueamento geral, podendo produzir ótimos resultados se o resultado esperado for colocado mais ao topo da margem de variabilidade de quantidade de erros. Também é interessante analisar a distribuição dos resultados esperados pelas posições de ranqueamento, como mostra a Figura 6.11.

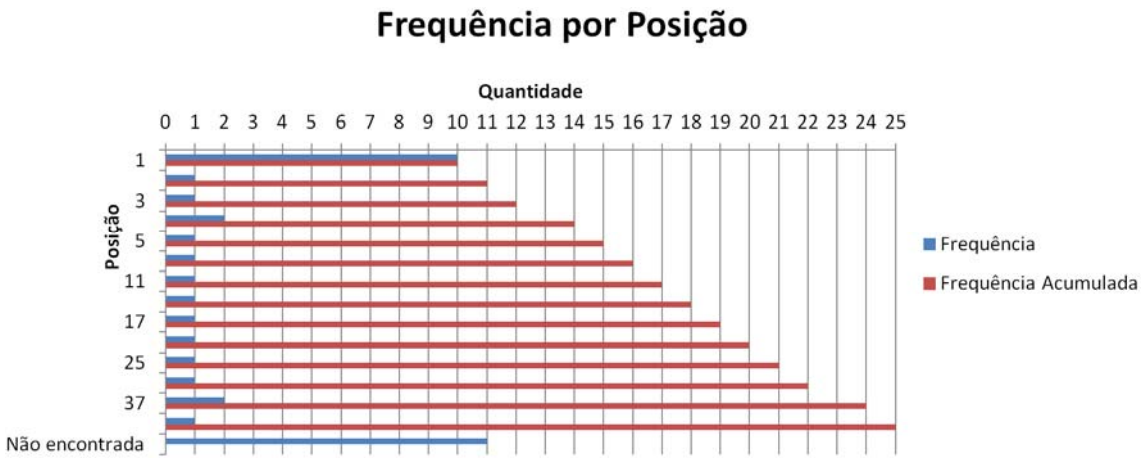


Figura 6.11: Frequências do resultado esperado por posição de ranqueamento.

Percebe-se, na Figura 6.11, que o método CMC, juntamente com as opções implementadas na ferramenta TabSim, produziu uma frequência de posicionamento do

resultado esperado na primeira colocação muito semelhante à frequência de consultas que não encontraram o resultado esperado. Também percebe-se que as outras posições tiveram frequências do resultado esperado muito semelhantes. Analisando as frequências mostradas na Figura 6.11, fica clara a influência das margens de variabilidade, distribuindo os resultados esperados obtidos por consultas não idênticas pelas primeiras posições de ranqueamento. Além disso, pode-se considerar boa a taxa de localização do resultado esperado dentro dos primeiros 50 resultados, sendo de 69,44%, contra 30,56% de não localização.

7 CONCLUSÕES

Este trabalho apresentou o método Correspondência de Monofonia Contida (CMC), que permite recuperar composições polifônicas a partir de uma consulta, também polifônica, utilizando similaridade melódica. O método CMC identifica todas as monofonias contidas na consulta, permitindo que sejam eliminadas monofonias não consideradas melodicamente relevantes. O conjunto restante de monofonias, identificadas na consulta, é utilizado para recuperação de composições presentes em uma base de composições representadas como Autômatos Finitos Não Determinísticos com Movimentos Vazios (AFNε).

A principal contribuição deste trabalho é a definição formal de um método para busca de composições polifônicas, através de similaridade melódica, que possibilite a eliminação de combinações melódicas não consideradas relevantes nas consultas. Não pode-se também deixar de destacar a descrição do buscador de tablaturas textuais da *web* TabSim, principalmente devido a carência de soluções similares.

Através da realização de experimentos foi possível concluir que o método CMC produz, em geral, bons resultados, localizando composições na grande maioria das buscas e posicionando em média o resultado esperado dentre as primeiras 11 posições de ranqueamento, ao considerar somente os casos em que o resultado esperado foi encontrado. Também, ficou claro que o método possui grande espaço para melhorias, podendo produzir ótimos resultados, posicionando em média o resultado esperado dentre as 5 primeiras posições de ranqueamento. Mesmo após a realização dos experimentos, acredita-se que o protótipo TabSim consegue demonstrar apenas parte do potencial do método CMC.

Durante o período de realização deste trabalho, foi escrito um artigo contendo uma proposta inicial, demonstrando a arquitetura geral da solução para busca de tablaturas textuais da *web*. Este artigo foi submetido em maio de 2013 à 14ª edição da *International Society for Music Information Retrieval* (ISMIR) *Conference*, realizada entre os dias 4 e 8 de novembro em Curitiba. O congresso anual ISMIR é o principal evento internacional da área de computação musical, possuindo nível *Qualis* A1 pela CAPES (Coordenação de Aperfeiçoamento de Pessoal de Ensino Superior).

Dentre as possibilidades de trabalhos futuros, para avaliação, continuidade ou aperfeiçoamento do método CMC, estão:

- Melhorar a determinação do grau de similaridade e processo de Verificação de Similaridade (seção 4.3.4). Uma opção seria, em correspondências de notas, levar em consideração a quantidade de notas por tempo na composição, e não somente a quantidade de tempos,

fazendo o grau de similaridade reduzir nos casos de correspondência de monofonias em meio a acordes;

- Definir mais opções para eliminação de monofonias;
- Tratamento dos casos onde existam notas extras erradas em meio a uma consulta correta, e dos casos onde estejam faltando notas em meio às notas da consulta;
- Definir formas de colocar o resultado esperado mais ao topo das margens de variabilidade;
- Definir uma função de ranqueamento que melhorasse o posicionamento do resultado esperado de acordo com a quantidade de monofonias correspondidas em seu conteúdo;
- Adaptação para verificar uma polifonia contida, através de correspondências de grupos de monofonias contidas;
- Adaptação do método para considerar durações musicais na avaliação de similaridade;
- A definição do método CMC, apresentada neste trabalho, não leva em consideração aspectos relacionados ao seu desempenho, porém, sua estrutura, baseada em conjuntos, e seu comportamento, de segmentar a consulta em consultas menores, demonstram-se favoráveis para a realização de experimentos e melhorias relacionadas a este aspecto (desempenho).

REFERÊNCIAS

ANGELUS, J. **Musipedia**: rhythm, parsons code, musical instrument digital interface, GNU lilypond. [S.l.]: International Book Marketing Service, 2012.

CLAUSEN, M. et al. PROMS: a web-based tool for searching in polyphonic music. In: INTERNATIONAL CONFERENCE ON MUSIC INFORMATION RETRIEVAL, 2000… **Proceedings** [S.l.:s.n.], 2000. 10 p.

CLIFFORD, R. et al. A fast randomized maximum subset matching algorithm for document-level music retrieval. In: INTERNATIONAL CONFERENCE ON MUSIC INFORMATION RETRIEVAL, 2006… **Proceedings** [S.l.:s.n.], 2006. 6 p.

DEMOPOULOS, R; KATCHABAW, M. J. Music information retrieval: a survey of issues and approaches. **Technical Report** n.677. Ontario: University of Western Ontario, 2007.

LEMSTRÖM, K.; MEREDITH, D.; WIGGINS, G. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. **Journal of New Music Research**, v. 31, 2002. p.321-345.

LEMSTRÖM, K.; TARHIO, J. Transposition invariant pattern matching for multi-track strings. **Nordic Journal of Computing**, v.10, 2003. p. 185-205.

LEMSTRÖM, K.; MAKINEN, V. On minimizing pattern splitting in multi-track string matching. In: **Lecture Notes in Computer Science**, v. 2676. [S.l.]: Springer, 2003. p. 237-253.

LEMSTRÖM, K. et al. **The C-BRAHMS Project**. Helsinki: University of Helsink, 2003.

LEMSTRÖM, K.; NAVARRO, G.; PINZON, Y. Practical algorithms for transposition-invariant string-matching. **Journal of Discrete Algorithms**, v. 3, 2005. p. 267-292.

MCNAB, R. J. et al. The New Zealand Digital Library Melody index. **D-Lib Magazine**, v. 3, n. 5, 1997.

MED, B. **Teoria da Música**. Brasília: Musimed, 1996. 4ed.

MIDI Manufactures Association (MMA). **MIDI 1.0 detailed specification document**. Version 4.2. Los Angeles: MMA, 1995.

MULLER, M. **Information retrieval for music and motion**. Berlin Heidelberg: Springer, 2007.

MYRIAD. **Kooplet**: help and advanced options. Acesso em: 13 dezembro de 2012. Disponível em: http://www.kooplet.com/en/kooplet.html.

SELFRIDGE-FIELD, E. Describing musical information. In: **Beyound MIDI**: the handbook of musical codes. Cambridge: MIT Press, 1997. p. 3-38.

SELFRIDGE-FIELD, E. Conceptual and representational issues in melodic comparison. In: **Melodic similarity**: concepts, procedures, and applications. Cambridge: MIT Press, 1998. p. 3-64. Computing in Musicology n. 11.

TYPKE, R.; VELTKAMP, R. C.; WIERING, F. Searching notated polyphonic music using transportation distances. In: INTERNATIONAL CONFERENCE ON MULTIMEDIA 12th, 2004… **Proceedings** New York: ACM, 2004. p. 128-135.

TYPKE, R. **Music retrieval based on melodic similarity**. Waiblingen: Universiteit Utrecht, 2007.

UKKONEN, E.; LEMSTRÖM, K.; MAKINEN, K. **Sweepline the Music!** Helsink: University of Helsink, 2003.

YATES, R.; NETO, B. **Recuperação de informação**: conceitos e tecnologia das máquinas de busca. Porto Alegre: Pearson, 2011. 2ed.