Universidade Federal de Minas Gerais
Centro de Pesquisa e Desenvolvimento em Engenharia Elétrica
Programa de Pós-Graduação em Engenharia Elétrica

Implementação Computacional de uma Gramática Gerativa para Música Tonal

Antônio Gilberto Machado de Carvalho
Orientador: Homero Nogueira Guimarães

Tese submetida à banca examinadora designada pelo Colegiado
do Programa de Pós-Graduação em Engenharia Elétrica da
Universidade Federal de Minas Gerais como parte dos requisitos necessários à
obtenção do grau de Doutor em Engenharia Elétrica.

09 de maio de 2008

Resumo

O presente trabalho apresenta a implementação computacional de uma gramática gerativa que visa a análise automática de trechos musicais. É baseado na Teoria Gerativa de Música Tonal (Lerdahl, F. & Jackendoff, R., 1996, *A Generative Theory of Tonal Music*, second ed., MIT Press, Cambridge, Massachusets), daqui por diante chamada de TGMT, a qual tem por objetivo a análise de peças de música sob o ponto de vista da percepção de um ouvinte experiente. Seguindo a proposta original da TGMT, a implementação do sistema tem por pilares quatro programas, um para cada componente da teoria, que rodam em paralelo e são dependentes, através de um sistema de comunicação empregando arquivos, dos resultados dos demais para a obtenção dos seus próprios. Naturalmente, existe um estado inicial a partir do qual todos os processos são inicializados. Para o componente *Estrutura de Agrupamento* foi empregado um sistema nebuloso e uma ferramenta de reconhecimento de padrões (visando o reconhecimento de padrões melódicos) baseada em uma matriz correlativa. No componente *Estrutura Métrica* foi utilizado um algoritmo genético multi-objetivo cuja tarefa é minimizar o conflito entre as regras preferenciais do componente. O componente *Redução Temporal* procura encontrar o melhor casamento entre os resultados dos dois componente anteriores visando a construção de uma árvore de redução temporal. Este componente, ainda que autônomo, trabalha também como parte de uma das regras da Estrutura Métrica, isto através de um segmento de memória compartilhada. Finalmente, o componente *Redução Prolongacional*, através da consulta a todos os resultados dos componentes anteriores, visa a construção de uma árvore prolongacional para uma música sendo analisada, ou seja, uma árvore que represente, em diversos níveis, os estados de tensão/relaxamento existentes na estrutura tonal da amostra. Com o objetivo de gerar resultados para o sistema sendo implementado foram empregados testes de validação, constituídos de fragmentos de obras, e trechos musicais completos.

Abstract

This work deals with the computational implementation of a generative grammar aiming at the automatic analysis of musical samples. It is based on *A Generative Theory of Tonal Music* (Lerdahl, F. & Jackendoff, R., 1996, second ed., MIT Press, Cambridge, Massachusets), from now on named GTTM, which has as its main objective the analysis of musical samples from the point of view of an experienced listener perception. Following the original proposal of GTTM, the system implementation is built on four programs, one for each theory component, that run concurrently and are dependent, by means of a pooling system, of the other components results aiming the processing of their own. Obviously, there is a initial state in which from there on all the components are initialized. For the component *Grouping Structure* was employed a fuzzy system and tool for melodic pattern recognition based on a correlative matrix. In the *Metrical Structure* a multi-objective genetic algorithm was used whose task is to minimize the conflict between the component preferential rules. The *Time-Span Reduction* tries to find the best union between the results of the two former components aiming to build a time-span reduction tree. This component, besides its own autonomy, works also as a part of one of the Metrical Structure preferential rules, by means of a shared memory segment. Finally, the component *Prolongational Reduction*, scanning all the former results, aims at building a prolongational tree, that is to say, a tree that represents, at several levels, the states of tension/relaxation that exists in the tonal structure of the processed sample. As sample tests for the system *Validation Tests* were employed, for fragments of works, and *Analysis*, for complete pieces.

Nomenclatura

Siglas

TGMT	*Teoria Gerativa de Música Tonal.*
MIDI	*Musical Instrument Digital Interface.*
CSP	*Constraint Satisfaction Problem.*
RBFA	*Regras de Boa–Formatividade de Agrupamento.*
RPA	*Regras Preferenciais de Agrupamento.*
RBFM	*Regras de Boa–Formatividade de Métrica.*
RPM	*Regras Preferenciais de Métrica.*
RBFRT	*Regras de Boa–Formatividade de Redução Temporal.*
RPRT	*Regras Preferenciais de Redução Temporal.*
RBFRP	*Regras de Boa–Formatividade de Redução Prolongacional.*
RPRP	*Regras Preferenciais de Redução Prolongacional.*
NSGA	*Nondominated Sorting Genetic Algorithm.*

Variáveis

$\mathbf{x}$	Vetor de entrada.
	Vetor de variáveis de decisão.
$\mathbf{E}$	Função de erro a ser minimizada.
	Vetor de erros.
p	Número de padrões de treinamento.
y	Saída de um sistema nebuloso.
N	Coeficiente determinante do número de regiões nebulosas.
	Coeficiente determinante de um limiar de vizinhança.
	Número de eventos de uma amostra.
	Número de níveis de uma estrutura métrica.
n	Número de elementos off-line em uma matriz esparsa.
	Expoente para o cálculo do limar de vizinhança.
Θ	Limar de vizinhança.
δ	Quantum de duração.
	Distância entre dois acordes pertencentes a regiões diferentes.
m	Grau dos componentes de uma regra num sistema nebuloso.
	Grau de controle de saída num sistema nebuloso.
D	Grau de uma regra num sistema nebuloso.
	Dimensão de um vetor de fronteiras de grupos.
I	Região de entrada de um sistema nebuloso.
	Instante de tempo sobre o qual se aplica um limiar de vizinhança.
O	Região de saída de um sistema nebuloso.
$\overline{y}$	Centro de uma região nebulosa.

K	Número de regras numa base de um sistema nebuloso.
d	Dimensão do vetor de fronteiras de grupos.
L_v	Limiar de vizinhança.
$\mathbf{G}$	Vetor de fronteiras de grupos.
n	Dimensão de um vetor de alturas (motivo).
NB_c	Conjunto dos índices vizinhos ao nodo vencedor c.
$\mathbf{w}$	Vetor de pesos.
X	Padrão repetitivo.
	Espaço de decisão.
$\mathbf{S}$	Seqüência de notas.
	Espaço necessário para codificação de matriz esparsa.
	Conjunto de pontos de um problema multi-objetivo.
$\mathbf{T}$	Matriz correlativa.
	Tamanho de uma matriz.
	Duração de uma amostra musical.
$\mathbf{CS}$	Conjunto de padrões candidatos.
f_p	Freqüência de um padrão repetitivo não-trivial.
$\mathbf{G}$	Vetor de fronteiras de grupos.
$\mathbf{u}$	Vetor de variáveis de decisão.
$\mathbf{v}$	Vetor de variáveis de decisão.
$\mathbf{D}$	Matriz de dominância.
	Matriz com somatório de funções.
	Conjunto de domínios.
$F(x)$	Somatório de funções em problema multi-objetivo.
f_i	i-ésimo elemento do front f.
r	Rank de um front.
M	Número de indivíduos do front corrente.
m_i	i-ésimo elemento do front corrente.
$Sh()$	Função de compartilhamento.
σ_{share}	Distância máxima em um nicho.
$d(i,j)$	Distância Euclidiana Normalizada.
m_{dg}	Medida de diversidade genética.
NG_i	Número de grupos do i-ésimo nível.
NL	Número de níveis de uma estrutura métrica.
NA	Número de eventos-altura.
$\mathbf{dur}$	Vetor de durações.
C	Conflito entre as regras preferenciais da Redução Temporal.
$\mathbf{V}$	Conjunto de variáveia.
$\mathbf{C}$	Conjunto de restrições.
$G(V,C)$	Grafo de restrições.
L	Número de níveis da Redução Temporal.
$\mathbf{I}$	Matriz com classes de altura e durações.

Agradecimentos

Um trabalho como o que está sendo apresentado, envolvendo mais de uma área de conhecimento e chegando mesmo, em alguns momentos, a fazer uso de várias, não é uma tarefa individual, no sentido correto do termo, mas antes uma tarefa coletiva corporificada, aí sim, por apenas um indivíduo. Sendo assim, nesta seção do trabalho, não faz-se apenas necessário, mas também de premente justiça, que sejam formalmente citados aqueles que, de uma forma ou de outra, contribuíram para a realização desta tese.

Inicialmente, gostaria de citar o Professor Homero Nogueira Guimarães, orientador e primeiro a sugerir e vislumbrar a possibilidade deste trabalho, posteriormente incentivando-o e propondo questões de crucial importância para sua viabilidade de realização.

Em seguida, devem ser lembrados os professores e os colegas do CPDEE com os quais tive oportunidade conviver, os primeiros contribuindo com seus respectivos conhecimentos e técnicas e os outros através de conversas nas quais pontos de dúvida eram substituídos por curvas de certeza. Assim, desta forma, ambos os grupos também tiveram sua participação na produção deste trabalho.

Recordo ainda a compreensão de minha família no que tange às longas horas de presença ausente, simultaneamente estando sem estar, em épocas e momentos nos quais talvez outros em iguais circunstâncias não pudessem ser presenteados com as horas de trabalho necessárias.

Finalmente, lembro aqueles a quem é dedicado este trabalho, meus pais, os quais, da maneira a mais indireta, porém, ao mesmo tempo, a mais veemente, tornaram possível, com usa compreensão e apoio, a realização deste trabalho. Desta forma, apesar de não estarem mais entre nós, seguramente estão vivos em cada linha escrita, de texto ou de código e em cada resultado alcançado, seja musical ou computacional.

Aos meus pais, Antônio e Ivone, in memoriam

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Considerações Gerais

Neste trabalho são apresentados os métodos e os resultados da implementação computacional de uma gramática gerativa para música tonal, sendo que, para esta, foi tomada como base a teoria elaborada por Lerdahl & Jackendoff (1996). Trata-se de uma continuação do trabalho já iniciado por Carvalho (2001), no qual foi implementado o primeiro componente da teoria, a Estrutura de Agrupamento, tarefa na qual foram empregadas diversas ferramentas de inteligência computacional. Sobre o emprego desta disciplina em aplicações musicais, ver os trabalhos de Meehan (1979), Goldman et al. (1999) e Holland(1989, 2000), o primeiro versando sobre teoria musical, o segundo sobre representação de conhecimento musical e os dois restantes sobre aplicações em educação musical.

Ainda que existam inúmeros enfoques computacionais para a análise musical, indo desde compilações de métodos, como o trabalho de Gerhard (2002), passando por estudos completos envolvendo modelos cognitivos (Lartillot, 2002) ou processamento de sinais (Monti, 2001; Franz, 1998), até implementações completas (Maidin, 1995), a abordagem computacional do trabalho de Lerdahl & Jackendoff (1996) mostra-se bastante promissora em seus resultados, já que aceita diversos tipos de enfoque na implementação de suas regras. Além disto, tendo em vista que os componentes operam de uma forma paralela, uns dependendo do resultado dos outros, sua implementação apresenta o desafio de alcançar-se uma representação eficiente para este paralelismo, o que é discutido no capítulo 3, página 9.

A TGMT[1] é dividida em quatro componentes (hierárquicos, segundo seus autores[2]) principais (Lerdahl & Jackendoff, 1996, pag. 8):

1. A *Estrutura de Agrupamento*, "que expressa uma segmentação hierárquica da peça

[1]Apesar de ser principalmente direcionada para a música dita tonal, esta teoria pode, ainda que com restrições, ser também empregada em outros tipos de música, como pode ser comprovado por Lerdahl (1989) e também por uma análise de uma peça de Stravinsky apresentada na seção 4.7.2, página 41 deste trabalho.

[2]Mas que, na realidade, operam paralelamente, pois cada componente possui regras que, para seu correto funcionamento, dependem dos resultados dos outros componentes.

em motivos, frases e seções";

2. A *Estrutura Métrica*, "que expressa a intuição de que os eventos de uma peça estão relacionados a uma alternância regular de tempos fortes e fracos em vários níveis hierárquicos";

3. A *Redução Temporal*, "que atribui às alturas de uma peça uma hierarquia de *importância estrutural* no que diz respeito à sua posição nas estruturas de agrupamento e métrica";

4. A *Redução Prolongacional*, "que atribui às alturas de uma peça uma hierarquia que expressa padrões harmônicos e melódicos de tensão/relaxamento e continuidade/progressão".

No que diz respeito ao conjunto de regras da teoria, ele é dividido em dois grupos:

Regras de Boa–Formatividade, que especificam as possibilidades de descrição estrutural da peça;

Regras Preferenciais, que selecionam do conjunto das possibilidades de descrição estrutural aquelas que correspondem ao modo no qual um ouvinte experimentado ouve uma determinada peça.

A utilização das Regras Preferenciais, as quais realizam o maior trabalho analítico dentro da teoria, é a maior diferença entre a presente teoria e a Gramática Gerativa, sua contrapartida lingüística. Enquanto que, nesta última, a principal questão é saber se uma cadeia de palavras é ou não uma sentença, no caso musical existe uma tal ambigüidade nas possibilidades de estruturação que se faz necessário levar em conta modos coerentes de audição de uma determinada peça para, a partir daí, analisar-se sua estrutura.

Além dos dois tipos de regras mencionados, existe um terceiro, a saber, as *Regras Transformacionais*, as quais têm por objetivo flexibilizar a aplicação das Regras de Boa–Formatividade. Mais uma vez de modo diverso da Lingüística, na qual possuem papel relevante ou mesmo central, as Regras Transformacionais desempenham um papel periférico na presente teoria.

A interação dos componentes da teoria gera um sistema cuja topologia pode ser vista na figura 1.1, retirada da obra original. Nela podem ser vistos os três conjuntos de regras (de Boa–Formatividade, Preferenciais e Transformacionais) e sua interação visando a análise de uma peça musical. No dizer dos autores (Lerdahl & Jackendoff, 1996, pag. 11):

> *"De um modo geral, o sistema pode ser pensado como tendo uma superfície musical como entrada e produzindo como saída a estrutura que o ouvinte percebe".*

1.2 Estado da Arte

Dentre os trabalhos precursores ao que agora está sendo apresentado destacam-se aqueles de Temperley (2001), de Cambouropoulos (1998) e de Hamanaka et al. (2007). Destes,

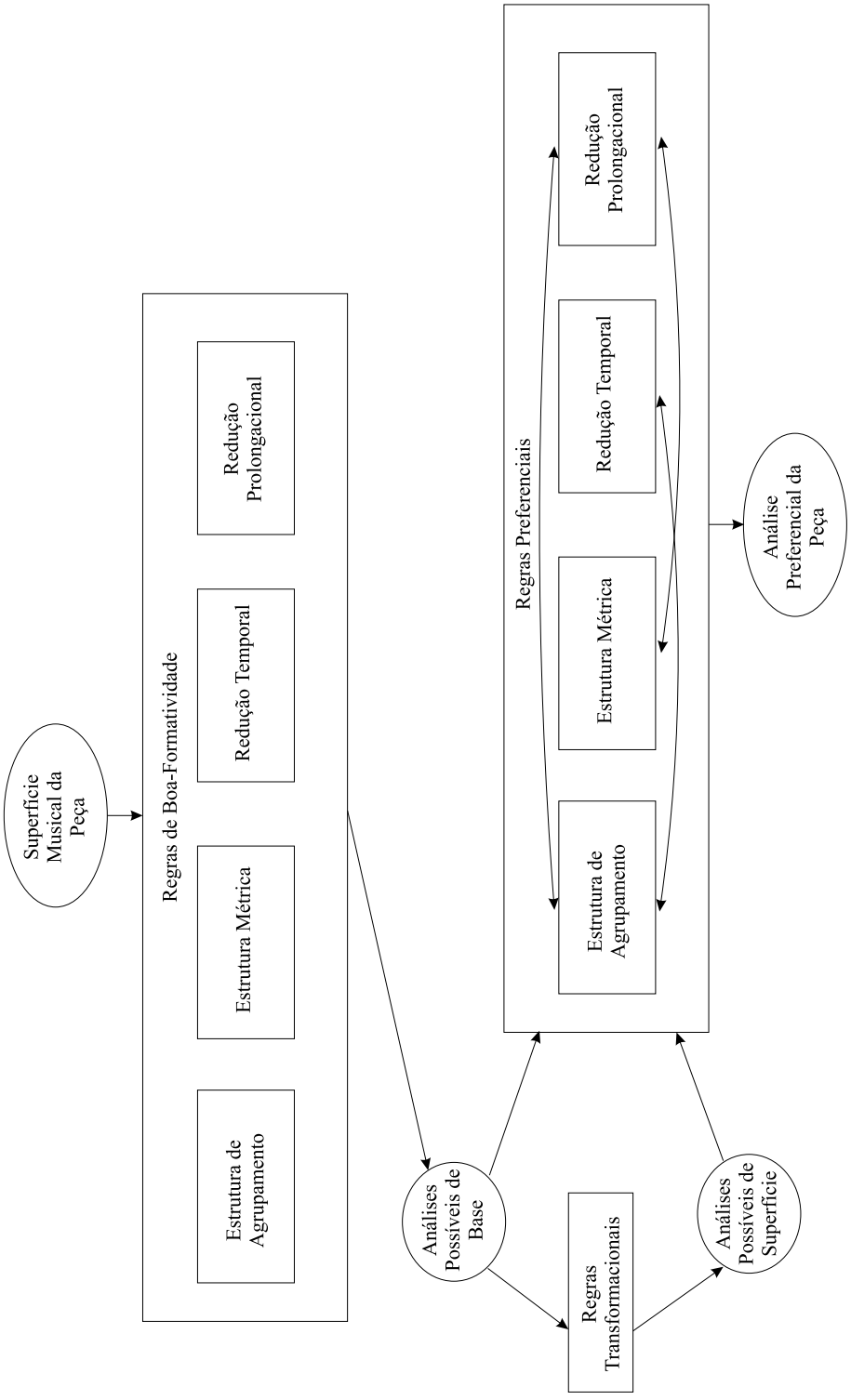

Figura 1.1: Forma global da teoria.

o último trabalho é aquele que apresenta maior proximidade com este ora apresentado. Entretanto, apesar disto, nesta seção é comentado integralmente o conjunto dos três com o objetivo de melhor situar este trabalho no contexto da disciplina de análise musical automática.

A proximidade da abordagem de Temperley (2001) com a deste trabalho consta, principalmente, na utilização de regras igualmente divididas entre de Boa–Formatividade e Preferenciais. Entretanto, ainda que fortemente influenciado pela TGMT, Temperley trabalha com seu próprio conjunto de regras, ao invés de implementar aquelas presentes na TGMT. Outra característica de seu trabalho é o uso de programação dinâmica (Lew & Mauch, 2007) visando a otimização dos processos. Tal emprego, visando a integração de métrica e harmonia pode ser visto em Temperley & Sleator (1999). Ainda de modo diverso ao empregado neste trabalho, no sistema implementado por Temperley o usuário deve seguir uma ordem na execução dos programas, o que significa que eles não operam em paralelo. Além dos trabalhos já citados, abordou também com mais especificidade o tema da *métrica*, indo desde questões de *hipermétrica*[3] (Temperley, 1996) até uma discussão acerca da avaliação de métodos visando o tratamento (Temperley, 2004).

Em sua tese de doutorado, Cambouropoulos (1998) descreve uma teoria, ao mesmo tempo com caráter geral e computacional, voltada à análise de uma superfície musical[4]. Com este fim e a partir de uma série de considerações tanto musicais quanto lógicas e cognitivas, o autor criou um conjunto de heurísticas para o tratamento de importantes categorias presentes em análise musical, tais como segmentação, paralelismo e métrica. Uma restrição na teoria de Cambouropoulos e que a diferencia do presente trabalho e também daquele de Temperley, é sua aplicabilidade unicamente a amostras melódicas, ou seja, não é possível a realização de análises de peças nas quais seja considerada sua harmonia. Por outro lado, uma grande vantagem que a teoria apresenta é sua possibilidade de emprego em quaisquer tipos de amostras, independentemente de estilos ou épocas históricas.

Os trabalhos de Hamanaka e seu grupo (2007, 2002, 2004, 2005b, 2005a), no sentido de uma implementação da TGMT visam, além de possibilitar a análise automática de amostras musicais, também seu emprego como ferramenta de recuperação de informação em bases de dados musicais. Três elementos diferenciais podem ser apontados entre o trabalho de Hamanaka e o que ora está sendo apresentado. Inicialmente, no primeiro trabalho existem um grande número de parâmetros (dezessete) que podem ser ajustados visando a obtenção de uma melhor qualidade analítica. Já no presente trabalho o número de parâmetros é muito menor (sete) e estão pensados como elementos que devem possuir uma certa persistência, isto é, somente devem ser ajustados em caso de obtenção de resultados considerados falhos ou errôneos. Em segundo lugar, no trabalho de Hamanaka todo o processamento é realizado através de heurísticas de natureza determinística, enquanto no presente trabalho existe um misto de tratamento determinístico e

[3]Denomina-se de hipermetro aquela região da grade métrica na qual ocorrem pulsos de duração igual ou maior à do compasso do trecho sendo analisado (ver capítulo 5).

[4]Em análise musical, chama-se superfície aos dados brutos que chegam aos ouvidos do analista. Isto é, trata-se da matéria musical antes de sofrer qualquer análise ou processamento.

de ferramentas de inteligência computacional (de natureza estocástica). Finalmente, no trabalho de Hamanaka já existe a leitura automática dos dados de saída para uma interface gráfica em XML, possibilidade ainda ausente no trabalho que está sendo apresentado. Duas principais restrições podem ser indicadas no trabalho de Hamanaka. Em primeiro lugar, o grande número de parâmetros das regras, apesar de permitir maior flexibilidade, devem ser ajustados manualmente, o que pode fazer com que o usuário perca rapidamente o controle sobre o sistema. Uma ferramenta para ajuste automático dos parâmetros seria de grande valia para solucionar esta situação. Em seguida, o trabalho de Hamanaka, assim como o de Cambouropoulos, somente trata de amostras melódicas, o que pode não ser uma grande limitação quanto se trata de recuperação de informação, mas certamente o é no caso da análise musical.

1.3 Estrutura do Trabalho

Este trabalho está dividido em 8 capítulos. Em primeiro lugar, nesta Introdução, mostra-se uma visão da matéria, discute-se sobre o *estado-da-arte* e indica-se a estrutura formal do texto. Em seguida, são explicitados os Objetivos Gerais do projeto, aos quais segue um capítulo sobre a Arquitetura do Sistema implementado. Nos próximos quatro capítulos, um para cada componente da teoria, é dada uma Introdução contendo um *Sistema Analítico* compreendendo conceitos e notações relativos ao componente e uma *Gramática Formal*, composta do seu sistema de regras. A isto segue-se os Objetivos Específicos de cada componente, uma Metodologia e os Resultados, divididos em Testes de Validação e Análises completas. No capítulo seguinte, é apresentada uma Discussão dos Resultados alcançados até o momento. Finalmente, no último capítulo, discorre-se acerca das Possibilidades Futuras do presente trabalho.

Capítulo 2

Objetivos Gerais

Na medida em que no estudo anterior que deu origem a este projeto (Carvalho, 2001) foi trabalhado unicamente o primeiro componente da teoria, o mais importante objetivo do presente trabalho é uma implementação que integre os quatro componentes da teoria elaborada por Lerdahl & Jackendoff (1996) num único sistema analítico musical. Espera-se, portanto, ao final do trabalho, a integração de cada componente da teoria formando um sistema analítico único que, a partir da entrada de um trecho musical codificado no protoloco MIDI devolva como saída um conjunto de dados correspondentes à análise sintática do trecho em questão, tal como aparece na figura 1.1, da Introdução. Isto envolve a criação e desenvolvimento de estruturas de dados e representações musicais que possam ser manipuladas individualmente por cada componente, possibilitando, assim, a troca de informações. Entretanto, antes de passar para o segundo objetivo geral deste trabalho, e na medida em que a análise musical é, por sua vez, o objeto do sistema sendo desenvolvido, tornam-se relevantes algumas palavras acerca do tema.

A disciplina da Análise Musical possui um amplo espectro no que tange à sua real necessidade e aos seus diversos tipos e aplicações. Além disto, existem também razões e motivações que levam a, ou pelo menos justificam, a automação do processo analítico.

Em primeiro lugar, pode-se pensar na necessidade da análise musical como sendo de três ordens, a saber, aquela referente à interpretação musical, em seguida aquela referente às demandas que formam o arcabouço teórico da formação de um compositor e, finalmente, o enfoque analítico que considera a obra musical como um objeto produto de um conhecimento, isto é, um enfoque que considera a construção ou composição musical sob a ótica da epistemologia. Esta última abordagem da análise musical é típica da musicologia em suas diversas subdivisões internas.

Considerando-se a prática musical de um instrumentista, a análise musical surge como fundamento teórico, e muitas vezes estilístico, para seu estudo e interpretação de uma obra musical, fundamento sem o qual sua interpretação não fará jus ao nome, na medida em que será apenas uma cópia daquela elaborada por outro instrumentista, algo, aliás, bastante comum no meio musical.

Já no que tange ao compositor, a análise musical adquire sua importância ao proporcionar a possibilidade da compreensão e aquisição de técnicas de construção de diversas

épocas e complexidades. Além disto, ao empregar diversas metodologias aplicadas a diferentes épocas e estilos, a análise musical faz surgir na mente do compositor, através da engenharia reversa, hábitos e costumes típicos do *métier* de compositor e ligados principalmente a questões de construção, forma e elaboração de material.

Finalmente, a análise musical, agora em sentido amplo, pode ser considerada como a principal base para a Musicologia, na medida em que nesta, sendo a Música uma expressão do conhecimento humano, sua formalização através de metodologias estritas torna-se um procedimento fundamental caso se queira colocar esta disciplina em pé de igualdade com as outras ciências. Diferentes abordagens musicológicas exigirão, naturalmente, diferentes objetos de estudo e, conseqüentemente, diferentes metodologias. Porém, todas estas abordagens devem possuir a característica comum da rigorosa metodologia de análise, não importando o objeto a ser analisado, sob pena de sacrificarem o próprio objetivo da disciplina Musicologia, o qual é a consideração do objeto musical como uma forma de conhecimento, além de artístico, também epistemológico. Após esta exposição sobre a necessidade da Análise Musical, passamos a discutir seus tipos e aplicações.

Para cada uma das abordagens oriundas das práticas musicais principais (composição, interpretação e estudo musicológico), existem aplicações típicas que podem exemplificar seu emprego. Uma utilização da análise musical que pode ser considerada como elemento comum entre todas aquelas práticas mencionadas é o seu emprego na educação musical. Ainda que variando seus enfoques e prioridades, o uso da análise na educação musical pode ser considerado como fundamental desde os primeiros anos de estudo até os graus mais avançados da pesquisa musical. Inicialmente, no caso da composição, a análise musical, através de diferentes metodologias, mostra-se a base para o estudo da forma e sua plena compreensão. Em segundo lugar, tratando-se da interpretação, a análise pode voltar-se para a compreensão de determinadas sintaxes e de suas aplicações num texto musical, viabilizando, assim, uma mais forte apreensão do texto e, conseqüentemente, uma melhor interpretação do mesmo. Em último lugar, no universo da musicologia, seria limitador pretender enumerar os tipos de análise musical que podem constar e serem úteis ao seu escopo, já que o próprio objeto musical, assumindo incontáveis formas, exige tratamentos em número tão diversos quanto estas. Resta agora um tópico que liga a análise musical ao trabalho ora sendo desenvolvido: trata-se da automação do processo analítico.

Considerando-se os tipos e aplicações da análise musical citadas acima, automatizar seu processo pode trazer um certo número de vantagens. Do ponto de vista educacional, por exemplo, poderá facilitar a construção de testes nos quais respostas realizadas por analistas humanos seriam comparadas àquelas produzidas automaticamente. Sob o enfoque da composição, análises exaustivas de obras complexas poderiam ser realizadas, proporcionando uma melhor e mais vasta compreensão destas obras. Pensando na interpretação musical, detalhes cuja natureza complexa os deixassem ocultos poderiam ser revelados, facilitando, assim, a construção de uma interpretação mais robusta e teoricamente sólida. Finalmente, sob a ótica da musicologia, a automação do processo analítico poderia, além de trazer novas visões e questionamentos sobre análises já realizadas por analistas humanos, também trazer a enorme contribuição de permitir a análise de uma grande quantidade de amostras (obras musicais) sem que houvesse a interferência humana,

análise cujos resultados, posteriormente, poderiam ter incontáveis aplicações.

Além do objetivo primordial deste trabalho, o qual é, como já dito antes, a criação de um sistema automático de análise musical baseado na TGMT, existe também o desejo do emprego dos algoritmos e ferramentas trabalhadas num contexto complementar, ou seja, a intenção de utilizá-los como meios auxiliares à composição musical[1].

A utilização de procedimentos automatizados na composição musical não é uma novidade na música clássica ocidental. O que diferencia os métodos das épocas mais antigas daquelas mais recentes é somente o grau de automação. Ao construir uma fuga, o compositor do século XVIII sabia que, via de regra, nos momentos de aparição do sujeito nas diferentes tonalidades, estava sempre a acompanhar-lhe o contrassujeito. Como este já estava composto desde o começo de elaboração da fuga, sua aplicação ao sujeito tornava-se quase automática, bastando algumas alterações devidas às mudanças de modo. Continuando com o exemplo, poder-se-ia dizer que, nos dias de hoje, o almejado é a própria construção do contrassujeito (dado o sujeito) e não apenas seu sincronismo com o elemento principal. Trata-se, portanto, de um grau de automação muito superior àquele já existente na elaboração da clássica forma contrapontística.

A partir deste exemplo simples é possível compreender o quanto a automação de métodos analíticos poderá contribuir nos diversos aspectos da elaboração de uma composição musical, delimitando o segundo objetivo geral do presente trabalho.

[1] À qual poderia chamar-se, sem dificuldade, *síntese musical*, já que se trata, realmente, do contexto complementar daquilo a que se chama de análise. Todavia, o termo *composição* é de uso tão corrente nas várias vertentes ocidentais da prática musical que não seria aconselhável deixar de dar continuidade a seu emprego.

Capítulo 3

Arquitetura do Sistema

Uma das maiores dificuldades na implementação computacional da TGMT é o interrelacionamento entre seus quatro módulos. Em outras palavras, longe de possuir a linearidade e hierarquização proposta por seus autores, tal teoria opera com seus quatro componentes numa arquitetura em paralelo, uns dependendo dos resultados obtidos pelos outros. Assim, tal implementação implica em problemas de três ordens, os quais são independência no processamento dos componentes, sincronismo e comunicação entre os processos e escolha do ponto inicial. Destes, o último é o mais complexo e sutil, pois, tratando-se a teoria, na realidade, de uma malha sem começo nem fim[1], é necessário para sua utilização prática a escolha de um ponto onde deverá se iniciar o processamento.

3.1 Independência no Processamento

Dentre as possibilidades de processamento multi-tarefa (fork, threads e multi-processadores) optou-se por realizar o processamento paralelo em *background* dos quatro componentes. Através de um *script* estes recebem o mesmo nome de um arquivo de entrada e passam a rodar independentes uns dos outros, porém trocando entre si os resultados individualmente obtidos. Tal arquitetura pode ser vista na figura 3.1. Nela podem ser vistos os quatro componentes da TGMT e suas rotas de comunicação. Todas estas são realizadas através de arquivos, com exceção da ligação entre a Redução Temporal e a Estrutura Métrica, de caráter bi-direcional, e que é realizada através de um segmento de memória compartilhada (ver a seção 3.2.2). Pode-se ver ainda que todos os componentes recebem o mesmo arquivo de entrada, o qual é um arquivo MIDI e que é transformado, em cada um dos componentes, numa lista de eventos que são, então, processados.

3.2 Sincronismo e Comunicação

Na medida em que os processos rodam em paralelo e, além disso, necessitam cambiar informações e dados, é necessário que sejam empregados alguns mecanismos que favoreçam

[1]Sendo um problema semelhante ao conhecido "Quem nasceu antes, o ovo ou a galinha?".

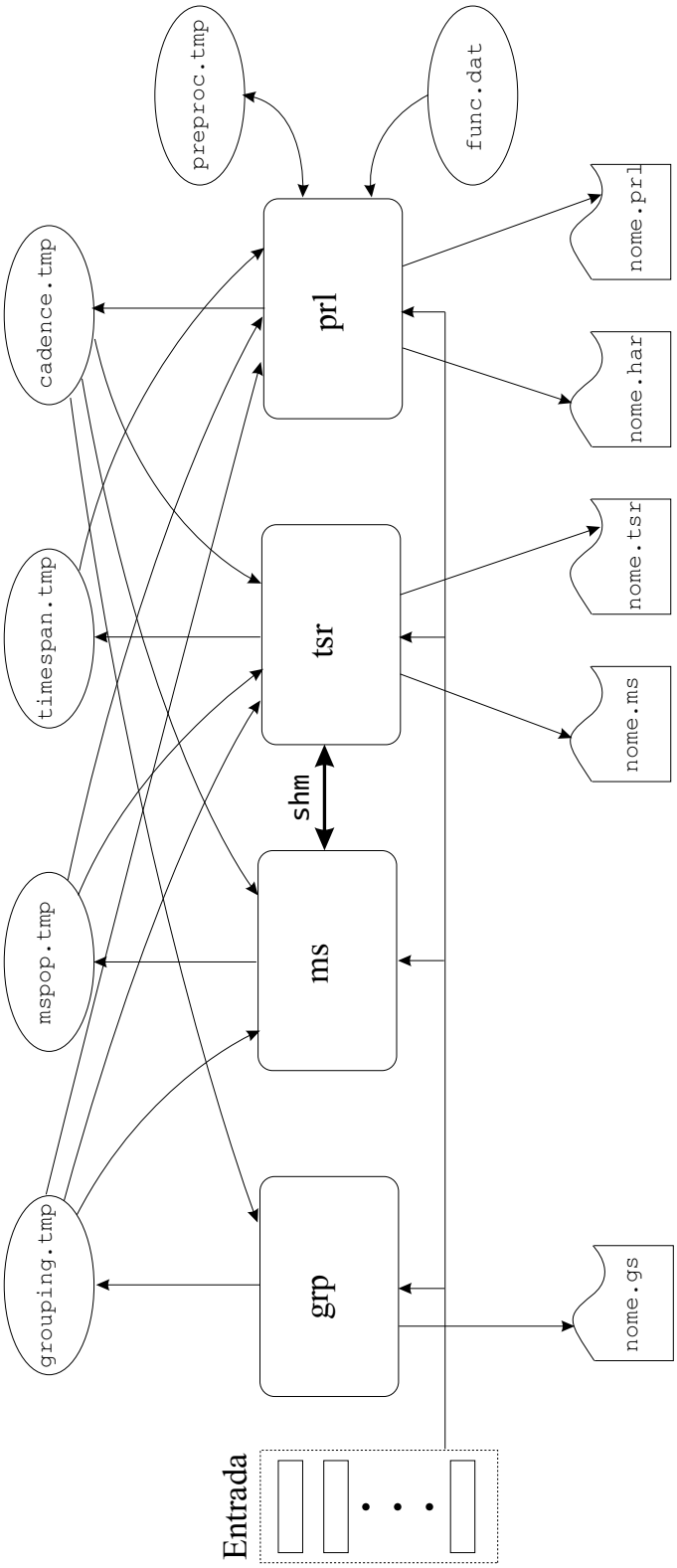

Figura 3.1: Arquitetura da implementação do sistema.

esta troca e possibilitem o sincronismo necessário. Neste trabalho, dois métodos de comunicação foram empregados, a saber, o travamento de arquivos (*file locking*) e memória compartilhada (em conjunto com semáforos). Dos dois o primeiro foi o quase exclusivamente empregado, restando o segundo para um caso particular o qual será exposto a seguir.

3.2.1 Travamento de Arquivos

Considerando que a comunicação entre os componentes é realizada através de arquivos, existe o problema de um componente desejar ler um arquivo antes que o mesmo tenha sido inteiramente escrito. Isto pode ser evitado empregando-se o recurso conhecido como *travamento de arquivos* (*file locking*). Neste recurso, um processo qualquer somente tem acesso a um determinado arquivo quando este não estiver sendo utilizado (lido ou escrito) por outro processo. Desta forma, assegura-se a inexistência de conflitos de informação.

3.2.2 Memória Compartilhada

O recurso de memória compartilhada foi usado na comunicação entre o processo encarregado do cálculo da Estrutura Métrica, mais especificamente a nona Regra Preferencial de Métrica (RPM 9), e aquele a cargo da Redução Temporal. A necessidade de emprego deste recurso surge do fato dele ser de acesso aleatório e de grande velocidade, o que viabiliza e justifica seu uso no presente caso.

Na figura 3.2 pode ser visto o esquema de funcionamento do recurso de memória compartilhada. Inicialmente, um processo reserva um segmento para compartilhamento de memória especificando seu tamanho e alocando uma *chave*[2], a qual é um valor numérico, que permitirá que outros processos tenham acesso ao segmento. Após a criação do segmento, qualquer processo que tenha acesso a sua chave poderá ter acesso ao mesmo anexando-o ao seu próprio espaço de endereços. Isto é feito através de um apontador que poderá ser empregado de maneira semelhante aos apontadores empregados para acessar endereços de memória do *heap*.

Todavia, na medida em que qualquer número de processos que possuam a chave podem acessar o segmento de memória compartilhada, existe a possibilidade bastante real de um processo alterar parcialmente os dados antes que outro tenha acesso aos mesmo em sua integridade. Assim, em outras palavras, é necessária a criação de seções críticas de código controladas por semáforos e que evitem o problema citado acima.

[2]No caso exemplificado na figura 3.2 a chave é 1111, porém, naturalmente, poderia ser qualquer outro valor numérico que não interferisse com outras chaves já presentes no sistema onde os processos estiverem rodando.

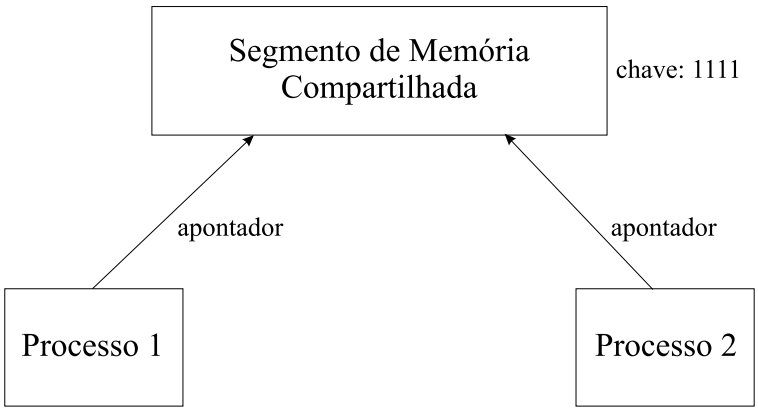

Figura 3.2: Operação da memória compartilhada.

3.3 Leitor de arquivos MIDI

3.3.1 Implementação

Na medida em que todos os quatro componentes recebem como entrada o mesmo arquivo MIDI, nesta seção são descritas suas estruturas de dados e algumas características de sua implementação.

O leitor de arquivos MIDI possui como entrada um arquivo MIDI e retorna uma lista de estruturas contendo os dados musicais contidos no mesmo arquivo. Na versão atual, o leitor aceita como entrada arquivos MIDI nos dois formatos mais utilizados[3], o que facilita seu emprego com arquivos MIDI provenientes de uma maior variedade de fontes. As estruturas de dados que constituem a saída do leitor (lista de estruturas) são apresentadas a seguir.

A estrutura inicial leva os dados de definição do próprio arquivo:

```
struct tagFileData {
   float TotalDuration; /* Duração total do arquivo    */
   float Tempo;         /* Tempo metronômico           */
   float qd;            /* Quantum de duração          */
   float InitialAttack; /* Instante de ataque inicial  */
   int Tonality;        /* Tonalidade                  */
   int Mode;            /* Modalidade                  */
   TIMESIG ts;          /* Metro (fórmula de compasso) */
}FILEDATA;
```

Os dois primeiro membros são a duração total do arquivo (em segundos) e seu tempo metronômico (em unidade metronômica). O membro seguinte, denominado *quantum de duração* (ou *unidade de análise de duração*), armazena o menor valor temporal presente no trecho musical sendo analisado. A variável `InitialAttack` guarda o valor do ataque

[3]Os quais são os formatos 0 e 1. O formato 2 é empregado correntemente para programas de percussão.

inicial (em segundos). Os membros `Tonality` e `Mode` correspondem, respectivamente, à tonalidade e ao modo do trecho. A variável `Tonality` pode assumir quinze valores inteiros no intervalo $[-7, 7]$, sendo os valores negativos as tonalidades com bemóis e os positivos as tonalidades com sustenidos. A variável `Mode` pode assumir os valores zero (modo maior) ou um (modo menor).

A estrutura do tipo `TIMESIG` armazena os dados do compasso (metro) do trecho sendo analisado:

```
typedef struct tagTimeSig {
   unsigned char numerator;     /* Numerador                          */
   unsigned char denominator;   /* Denominador                        */
   unsigned char nummidiclocks; /* MIDI clocks por pulso metronômico */
   unsigned char noted32;       /* Fusas por semínima MIDI            */
}TIMESIG;
```

Uma outra estrutura de dados importante para o sistema é aquela que determina os parâmetros de cada evento:

```
struct tagCelula {
  BYTE channel;       /* Canal do evento                      */
  BYTE pitch;         /* Altura do evento                     */
  BYTE velocity;      /* Intensidade do evento                */
  float rel_attack;   /* Ataque relativo ao evento precedente */
  float abs_attack;   /* Ataque absoluto do evento            */
  float duration;     /* Duração do evento                    */
}CELULA;
```

A variável `channel` pode assumir valores inteiros no intervalo $[0, 15]$ (correspondendo aos dezesseis canais MIDI); a variável `pitch` pode assumir valores inteiros no intervalo $[0, 127]$ representando as alturas de uma escala musical tendo o dó central como valor igual a 60; a variável `velocity` também pode assumir valores inteiros no intervalo $[0, 127]$ e representa a intensidade de um evento. Neste trabalho foi usada a referência abaixo:

Intensidade	*pppp*	*ppp*	*pp*	*p*	*mp*	*mf*	*f*	*ff*	*fff*	*ffff*
Velocity	10	23	36	49	62	75	88	101	114	127

Tabela 3.1: Relação entre intensidade e velocity.

As três variáveis seguintes são exclusivas do sistema e não pertencem ao protocolo MIDI. A variável `rel_attack` armazena o ataque relativo entre o evento precedente e o evento atual; a variável `abs_attack` armazena o ataque absoluto do evento considerando como zero o início do arquivo; finalmente, a variável `duration` armazena a duração do evento. Todos os três valores são dados em segundos.

Cada um dos eventos de cada uma das vozes do trecho sendo analisado é armazenado na memória do computador como uma célula do tipo descrito, formando o trecho completo uma lista encadeada de eventos.

3.3.2 Preparação do Arquivo – Edição

Na medida em que o sistema propõe-se a simular o comportamento de um ouvinte experimentado de música tonal, faz-se necessário que a informação codificada que a ele chega deva possuir as características musicais daquela sintaxe. Em outras palavras, a afinação, o ritmo, as características melódicas e harmônicas e as realizações de fraseado deverão todos estar consistentes com o estilo em questão.

Como este trabalho faz uso exclusivo de arquivos MIDI como entrada, o único ponto problemático dentre os apresentados corresponde à realização do fraseado, o qual inexiste num arquivo proveniente de um programa de notação musical por computador. Sendo assim, foi necessário realizar uma "interpretação" de cada um dos trechos sendo analisado com o objetivo de incluir dados que ajudassem na simulação do processo perceptivo de um ouvinte humano.

O tipo de edição realizado limitou-se a trabalhar com os dados MIDI que tivessem um efeito direto na percepção do fraseado, a saber, as intensidades (*crescendi*, *decrescendi* e mudanças de intensidade ao mudar de frase) e as durações (respirações ao final de frases). Com estes recursos simples de edição foi possível o envio de dados muito mais coerentes para o sistema.

Um outro tópico delicado com relação à edição dos dados refere-se à presença da informação correta da modalidade do trecho como um parâmetro do arquivo MIDI, já que se trata de uma informação importante no sentido de facilitar a confrontação com os dados obtidos pelo sistema. Assim, caso se queira experimentar com outros trechos além dos que foram disponibilizados com este trabalho, deve-se ter o cuidado de averiguar se o sequenciador ou o editor de partitura que se está utilizando permite a manipulação e a inclusão de ambos os parâmetros, tonalidade e modalidade, num arquivo MIDI. Esta observação faz-se necessária porque, enquanto indiscriminadamente permitem o arquivamento da tonalidade de um trecho musical, nem todos os programas que trabalham com MIDI manipulam corretamente o parâmetro correspondente à modalidade.

3.4 Estruturas de Dados Básicas

Nesta seção são introduzidas as estruturas de dados básicas empregadas em cada um dos quatro componentes da TGMT. Ainda que haja alguma redundância no que diz respeito às possíveis repetições, tal introdução faz-se necessária com o objetivo de tornar o mais claro possível a operação do sistema como um todo.

3.4.1 Estrutura de Agrupamento

Para a armazenagem das fronteiras de grupos e de seus respectivos valores (necessários para o processamento da RPA 4 – Intensificação), foi empregada a estrutura de dados:

```
typedef struct tagGroup {
  float position;  /* Fronteira de grupo */
```

```
  float value;     /* Valor da fronteira */
}GROUP;
```

onde a variável `position` é uma fronteira de grupo e `value` o seu valor. Como a *Estrutura de Agrupamento*, tal como proposta pela teoria, apresenta um caráter hierárquico e multi-escala, faz-se necessária uma estrutura de dados mais abrangente do que a anterior e através da qual estas características tornem-se evidentes:

```
typedef struct tagGroupingStructure {
  int level;      /* Nível hierárquico do agrupamento */
  int tam;        /* Número de fronteiras de grupos   */
  GROUP *g;       /* Vetor de fronteiras de grupos    */
}GROUPINGSTRUCTURE;
```

Nesta estrutura de dados, a variável `level` é o nível hierárquico correspondente, `tam` é o número de grupos no presente nível e `g` é um vetor de `tam` estruturas de fronteiras de grupos. Assim, através de uma lista de variáveis do tipo `GROUPINGSTRUCTURE`, é possível representar fielmente uma estrutura de agrupamento tal como descrita na TGMT.

Para o processamento da RPA 6, a qual trata dos paralelismos motívicos, é necessária ainda uma outra estrutura de dados que melhor se adapte às suas necessidades:

```
typedef struct tagMotive {
  Celula *motive;  /* Vetor dos componentes de um motivo */
  int length;      /* Número de componentes de um motivo */
  float begin;     /* Instante de ataque de um motivo    */
  float duration;  /* Duração de um motivo               */
}MOTIVE;
```

Os `length` componentes do motivo são armazenados no vetor `motive` de estruturas do tipo `CELULA` (iguais àquelas empregadas no leitor MIDI). Este mesmo motivo inicia-se no instante `begin` e a soma da duração de todos os seus componentes é igual a `duration`.

Para o processamento da RPA 7, a qual trata de questões harmônicas foi empregado o analisador harmônico descrito no capitulo 7. Através do arquivo temporário `cadences.tmp` o analisador harmônico envia dados para o componente Estrutura de Agrupamento, com os quais processa a RPA 7.

3.4.2 Estrutura Métrica

Além das estruturas de dados já mostradas e descritas anteriormente, o componente encarregado da Estrutura Métrica possui ainda duas outras de natureza específica.

Inicialmente, uma estrutura que permite a codificação dos dados necessários à definição de uma grade métrica, codificação esta necessária levando-se em conta que será utilizada uma população de estruturas métricas no algoritmo genético multi-objetivo.

Em segundo lugar, um conjunto de estruturas que permita a comunicação entre o processo responsável pela Estrutura Métrica e aquele outro responsável pela Redução

Temporal, comunicação esta estabelecida através de um segmento de memória compartilhada, tal como citado anteriormente.

A primeira destas estruturas diz respeito à codificação de um nível da grade métrica:

```
typedef struct tagMetricalLevel{
  int level_number;
  int initial_position;
  int spacing;
}MetricalLevel;
```

onde `level_number` é o índice no nível na grade métrica, `initial_position` é seu *offset* em relação ao nível anterior e `spacing` é o espaçamento entre os pulsos do nível, o qual pode assumir os valores típicos da Música Clássica Ocidental, que são 2 (metro binário) ou 3 (metro ternário).

A estrutura seguinte configura já uma grade métrica completa, pois possui a informação do número de níveis da mesma e a codificação de cada nível:

```
typedef struct tagMetricalGrid {
  int number_of_levels;
  MetricalLevel *level;
}MetricalGrid;
```

onde `number_of_levels` é o número de níveis constituintes da grade métrica e `level` é um vetor de estruturas do tipo `MetricalLevel`, a qual, como visto acima, codifica um nível métrico.

Finalmente, é necessária a existência de uma estrutura que permita a comunicação entre a Estrutura Métrica e a Redução Temporal (RPM 9). Esta, além das estruturas já vistas para a codificação da grade métrica, também possui a informação acerca da direção da comunicação e do erro encontrado na Redução Temporal para uma dada Estrutura Métrica. Assim:

```
typedef struct tagMsTsr {
  boolean direct;
  double error;
  MetricalGrid mg;
}MsTsr;
```

onde `direct` é uma variável booleana indicando a direção da comunicação entre os dois componentes, Estrutura Métrica e Redução Temporal, `error` é o erro devolvido pela Redução Temporal à Estrutura Métrica e `mg` é uma variável do tipo `MetricalGrid`, tal como foi definida acima.

3.4.3 Redução Temporal e Redução Prolongacional

Estes dois componentes não apresentam novas estruturas de dados em relação àquelas dos componentes já apresentados.

3.5 Implementação e Utilização do Sistema

O sistema descrito neste trabalho foi inteiramente implementado na linguagem C em um ambiente Linux Mandrake 9.0, sendo que a ele foi dado no nome de `Pierre` em homenagem ao grande compositor, pedagogo e regente francês Pierre Boulez. Para utilizá-lo na realização de uma análise basta digitar

```
pierre arquivo
```

onde `arquivo` é o nome de um arquivo MIDI dado sem extensão. Como resultado da análise `Pierre` dará quatro arquivos texto, cada um correspondendo a um componente da teoria de Lerdahl & Jackendoff (1996) e cujas estruturas podem ser vistas no apêndice C, página 198. Maiores detalhes acerca de como utilizar `Pierre` podem ser vistos no apêndice E, página 226.

Capítulo 4

Estrutura de Agrupamento

4.1 Introdução

Neste capítulo será apresentado o componente *Estrutura de Agrupamento* da Teoria Gerativa de Música Tonal, de Lerdahl & Jackendoff (1996).

Neste e nos capítulos subseqüentes relativos aos demais componentes da teoria serão expostos, inicialmente, o Sistema Analítico e a Gramática Formal do componente, seguindo a isto o seu Objetivo Específico. Em seguida, apresenta-se a Metodologia empregada e uma descrição das ferramentas utilizadas na Implementação. Finalmente, são mostrados os Resultados, divididos sob a forma de Testes de Validação e Análises de obras.

Para uma abordagem mais profunda deste componente, assim como dos demais, consultar o capítulo 2 de Carvalho (2001) ou, preferencialmente, o texto original de Lerdahl & Jackendoff (1996).

4.2 Sistema Analítico

Ao ouvir uma peça, um ouvinte organiza-a em unidades de informação cujo nome genérico é *grupo*. Estas unidades de informação (de diferentes dimensões temporais) são conhecidas na terminologia musical tradicional como motivos, temas, frases, períodos, grupo de temas, seções e a própria peça em sua integridade. Como será visto adiante nesta mesma subseção, as regras das estruturas de agrupamento organizam hierárquica e recursivamente estas unidades de informação. A noção de hierarquia merece alguns comentários adicionais.

Segundo a definição dos autores, o termo hierarquia diz respeito a *"uma organização composta de elementos discretos ou regiões relacionados de tal forma que um elemento ou região contém ou está contida em outros elementos ou regiões"* (pag. 13). Esta definição cria dentro da estrutura hierárquica dois tipos de elementos, a saber: os elementos *subordinados* e os elementos *supraordenados*, sendo os primeiros contidos pelos últimos. Ela também possibilita o surgimento de níveis hierárquicos que são função do tamanho de seus elementos ou regiões. Tais níveis dividem-se em *pequena-escala* e *larga-escala*. Devido ao

evidente caráter recursivo deste tipo de estruturação, tanto a segmentação da peça em elementos ou regiões quanto o surgimento de níveis hierárquicos podem atingir um estado de grande detalhamento.

Outro ponto importante a ser observado é que, em qualquer nível da estrutura de agrupamento, os grupos que a compõem devem ser adjacentes (podendo ou não compartilhar um evento, como foi dito). Isto significa que a audição de grupos, para um dado nível, não é *polifônica*, isto é, grupos são sempre ouvidos seqüencialmente.

Finalmente, completando este resumo sobre o sistema analítico da estrutura de agrupamento, pode-se dizer que:

1. A estrutura de agrupamento é hierárquica não admitindo a sobreposição de grupos;
2. A estrutura de agrupamento é recursiva;
3. Os grupos e cada um de seus componentes devem ser adjacentes.

4.3 Gramática Formal

Duas das características mais importantes de um agrupamento musical são sua semelhança com agrupamentos visuais e a sua independência com relação ao idioma nos quais os grupos que o constituem são elaborados.

A semelhança dos agrupamentos musicais com os agrupamentos visuais será a base para o subconjunto de regras gramaticais chamadas *regras preferenciais*, já que envolvem possibilidades diferentes do ponto de vista de percepção. Os dois princípios básicos envolvidos neste processo são o da *proximidade* e o da *similaridade*.

No caso da percepção visual, a proximidade ocorre quando objetos semelhantes distanciam-se desigualmente no espaço separando-se, a partir daí, em agrupamentos naturalmente formados. Já no caso musical, o distanciamento entre os objetos (eventos, neste caso) é, de fato, um intervalo temporal, os grupos formando-se através da proximidade entre os pontos de ataques dos eventos envolvidos.

O outro princípio básico, o da similaridade, ocorre quando, num conjunto de objetos igualmente espaçados, aqueles que são semelhantes tendem a agrupar-se. Também neste caso é possível observar que a invariância no espaço para um objeto visual é equivalente à invariância do tempo para um objeto acústico. Não obstante, é importante notar que a invariância entre os objetos pode ser gradual: à medida que ela varia os agrupamentos também variam e podem tornar-se ambíguos.

Após considerar os casos isolados dos dois princípios básicos envolvidos nos processos de agrupamento é necessário abordar sua percepção conjunta e conseqüentes ambigüidades. Quando percebidos em conjunto, os princípios da proximidade e da similaridade podem variar desde a mais completa contradição (significando, no que diz respeito aos agrupamentos, total ambigüidade por parte da percepção) até o máximo reforço (traduzido como agrupamentos inequívocos).

Partindo do que foi dito até agora, é possível intuir três propriedades importantes dos princípios de agrupamento:

1. Intuições sobre agrupamentos são graduais;

2. Os princípios básicos de proximidade e de similaridade podem reforçar-se ou estar em conflito;

3. Quando em conflito, um princípio pode sobrepor-se ao outro.

O paralelismo realizado acima entre as percepções visual e musical será a base para a formulação das *Regras Preferenciais de Agrupamento*. Entretanto, antes que estas sejam abordadas, é necessário que sejam enunciadas as chamadas *Regras de Boa–Formatividade*.

As *Regras de Boa–Formatividade de Agrupamento* (RBFA) formam um conjunto de condições às quais todo e qualquer agrupamento deve satisfazer. Como já foi dito anteriormente, estas condições estabelecem uma hierarquia estrita, recursiva e baseada na não-sobreposição de seus componentes. A partir de agora serão enunciadas cada uma das cinco RBFA.

RBFA 1 Qualquer seqüência contígua de eventos-altura[1], pulsações ou semelhantes constitui um grupo e somente seqüências contíguas podem constituir um grupo.

RBFA 2 Uma peça constitui um grupo.

RBFA 3 Um grupo pode conter grupos menores.

RBFA 4 Se um grupo G_1 contém parte de um grupo G_2, então deve conter integralmente G_2.

RBFA 5 Se um grupo G_1 contém um grupo menor G_2, então deve ser feita a partição exaustiva de G_1 em grupos menores.

Antes de prosseguir com os enunciados das *Regras Preferenciais de Agrupamento* (RPA), é importante que seja explicitada a razão da escolha da expressão *Regra Preferencial*. De modo diverso das RBFA as regras preferenciais não estabelecem decisões inflexíveis acerca da estrutura, mas, antes, preferências relativas dentre as numerosas análises logicamente possíveis de um ouvinte experimentado perceber uma peça. A análise resultante da combinação de todas as regras é chamada a "mais estável". Disto resulta que o formalismo das regras preferenciais, longe de ser um dispositivo arbitrário, é uma hipótese empírica sobre a natureza da percepção humana.

As regras preferenciais dividem-se em dois tipos de evidência determinantes dos agrupamentos percebidos pelo ouvinte:

Regras de Detalhes Locais, correspondentes aos padrões de ataque, articulação, dinâmica e tessitura que conduzem à percepção das fronteiras dos grupos;

[1]Um evento-altura é um complexo de alturas, de densidade 1 até n possuindo todos os componentes o mesmo instante de ataque e a mesma duração.

Organização Global de Agrupamento, tais como simetrias e paralelismos motívicos, temáticos, rítmicos e harmônicos.

Existem três RPA que caracterizam a percepção de evidências locais.

RPA 1. Evite enfaticamente grupos contendo um único evento.

As regras 2 e 3 tratam, respectivamente, da elaboração e aplicação dos princípios da proximidade e da similaridade.

RPA 2 (Proximidade). Considere uma seqüência de quatro notas $n_1n_2n_3n_4$. Tudo mais sendo igual, a transição n_2–n_3 pode ser ouvida como a fronteira de um grupo se

a. (Ligadura de expressão/Pausa) o intervalo de tempo do fim de n_2 ao começo de n_3 é maior do que aquele do fim de n_1 ao começo de n_2 e do que aquele do fim de n_3 ao começo de n_4, ou se

b. (Ponto de Ataque) o intervalo de tempo entre os pontos de ataque de n_2 e n_3 é maior do que aquele entre os pontos de ataque de n_1 e n_2 e do que aquele entre os pontos de ataque de n_3 e n_4.

RPA 3 (Similaridade). Considere uma seqüência de quatro notas $n_1n_2n_3n_4$. Tudo mais sendo igual, a transição n_2–n_3 pode ser ouvida como a fronteira de um grupo se

a. (Tessitura) a transição n_2–n_3 possui uma distância interválica maior do que n_1–n_2 e n_3–n_4, ou se

b. (Dinâmica) a transição n_2–n_3 possui uma mudança na dinâmica e n_1–n_2 e n_3–n_4 não possuem, ou se

c. (Articulação) a transição n_2–n_3 possui uma mudança de articulação e n_1–n_2 e n_3–n_4 não possuem, ou se

d. (Comprimento) n_2 e n_3 são de diferentes comprimento e os pares n_1,n_2 e n_3,n_4 não diferem em comprimento.

Não obstante as regras preferenciais enunciadas acima já serem suficientemente completas para realizarem agrupamentos que correspondem com precisão à audição que o ouvinte experimentado teria de tais grupos, elas tratam de agrupamentos de pequena extensão (4 notas), o que é limitador quando se quer tratar de entidades formais de maior dimensão. Para isto são necessárias regras preferenciais que atuem no segundo tipo de evidência perceptiva, ou seja, regras que visem uma *Organização Global de Agrupamento.*

RPA 4 (Intensificação). Onde os efeitos evidenciados (*picked out*) por RPA 2 e RPA 3 são relativamente mais pronunciados pode ser colocada a fronteira de um grupo de nível maior.

RPA 5 (Simetria). Devem ser preferidas análises de grupo que sejam as mais próximas possíveis da subdivisão ideal de um grupo em duas partes iguais.

RPA 6 (Paralelismo). Onde dois ou mais segmentos de música podem ser construídos como paralelos eles, preferivelmente, formam partes paralelas de grupos.

A última regra diz respeito à influência dos agrupamentos nas reduções temporais e prolongacionais (páginas 100 e 127, respectivamente).

RPA 7 (Estabilidade Temporal e Prolongacional). Prefira uma estrutura de agrupamento que resulte em reduções temporais e/ou prolongacionais mais estáveis.

4.4 Objetivo Específico

O objetivo específico desta parte do trabalho é a elaboração e implementação de algoritmos visando a segmentação de um trecho musical em agrupamentos de natureza hierárquica e recursiva. Ainda que existam várias abordagens para realizar este trabalho, como, além das já citadas anteriormente, (Aucouturier, 2001; Thom et al., 2002; Weyde, 2002), aqui seguir-se-á o caminho já iniciado em Carvalho (2001) aperfeiçoando-se algumas das ferramentas já anteriormente implementadas.

4.5 Metodologia

A metodologia empregada na implementação consistiu, inicialmente, em estudar e verificar de quais modos acontece a interação entre os dois conjuntos de regras e entre as regras entre si. Em seguida, estudou-se quais tipos de ferramentas seriam mais adequadas para cada regra (preferencial). A partir daí, partiu-se para a implementação propriamente dita. Dos métodos utilizados na implementação da Estrutura de Agrupamento destacam-se duas ferramentas, a saber, um Identificador de Padrões Não-Triviais (responsável pela RPA 6) e um Sistema Nebuloso (responsável pelas RPA 2 e RPA 3). Considerando que o primeiro método citado faz uso de matrizes altamente esparsas, foi implementado também um método utilizado para o tratamento destas, o qual pode ser visto no apêndice A, página 181.

As RPA 1, RPA 4 e RPA 5, assim como as Regras de Boa–Formatividade, utilizam heurísticas que têm como entrada os resultados das demais regras. A RPA 7 emprega um analisador harmônico descrito no capítulo 7, página 127.

Na medida em que cada um destes itens pode apresentar, individualmente, uma complexidade interna considerável, eles são expostos em seções diferentes visando maior clareza na demonstração de seus processos e estruturas.

4.5.1 Identificador de Padrões Não-Triviais

Histórico

Um dos principais problemas durante a implementação dos algoritmos utilizados para a detecção de padrões em Carvalho (2001) foi a tendência a sobrevalorizar pequenos intervalos durante a procura por padrões. Isto era causado pela própria metodologia empregada, a qual considerava como mais relevantes os padrões que estivessem mais próximos, em distância euclidiana, dos centros de *clusters* calculados por meio de um SOM (*Self-Organising Map*), tal como proposto por Kohonen et al. (1996). Na medida em que a detecção de padrões é um importante passo na implementação das regras de Lerdahl & Jackendoff (1996), este problema demanda por uma solução que leve a uma melhor precisão dos resultados.

Com este objetivo em mente, imaginou-se uma solução de caráter híbrido que pudesse caracterizar-se tanto pela precisão na descoberta de padrões quanto por uma flexibilidade que, através de generalização, permitisse a detecção de variações dos mesmos. Assim, mostrou-se como promissor um algoritmo que é constituído pelos passos seguintes:

1. Inicialmente, com a ajuda de uma heurística faz-se a varredura do trecho a ser analisado à procura de *padrões repetitivos não-triviais* (ver definições adiante).

2. A partir dos padrões encontrados cria-se um ou mais conjuntos de treinamento (dependendo da dimensão dos padrões) constituídos dos padrões originais acrescidos de variações dos mesmos obtidas através da introdução de ruído.

3. Com os conjuntos de treinamento obtidos no item anterior treina-se uma rede neural ou sistema nebuloso.

4. A partir dos resultados dos treinamentos varre-se novamente o trecho a ser analisado à procura de variantes dos padrões principais.

Neste trabalho é descrita primeira parte do algoritmo acima e é apresentada uma implementação da mesma. Uma descrição detalhada dos procedimentos pode ser encontrada em Hsu et al. (2001).

Padrões Repetitivos Não-triviais – Definições

Antes que seja descrita a heurística empregada faz-se necessária a exposição das definições de *padrões repetitivos* e da *não-trivialidade* dos mesmos (Hsu et al., 2001).

Definição 1

Para um subconjunto X de uma seqüência de notas $\mathbf{S}$, se X aparece mais do que uma vez em $\mathbf{S}$ nós denominamos X um *padrão repetitivo* em $\mathbf{S}$. A *freqüência de repetição* do padrão repetitivo X é o número de aparições de X em $\mathbf{S}$. O comprimento do padrão repetitivo X, representado como $\|X\|$ é o número de notas em X.

Definição 2

Um padrão repetitivo X é *não-trivial* se e somente se não existe outro padrão repetitivo Y tal que $freq(X) = freq(Y)$ e X é uma *substring* de Y.

Descrição da Heurística

Nesta subseção será descrita a seqüência de passos na qual opera o identificador de padrões não-triviais. A entrada do algoritmo é uma lista de notas (das quais, neste primeiro estágio, são desconsideradas as durações) e a saída é uma lista de duplas cujo primeiro elemento é o padrão e o segundo sua freqüência de repetição:

$$\{(p_1, f_1), (p_2, f_2), \ldots, (p_i, f_i)\}$$

O algoritmo em questão constitui-se de duas seções fundamentais, as quais são expostas a seguir:

1. Construção da Matriz Correlativa
   (a) Considere uma lista de notas **S**, de comprimento n e cuja i-ésima nota é S_i.
   (b) Considere uma matriz **T** (a matriz correlativa), de tamanho $n \times n$ e inicializada com zeros.
   (c) Para a primeira linha de **T**, considerando $2 \leq j \leq n$, se $S_1 = S_j$, então, $T_{1,j} = 1$.
   (d) Para o restante da parte triangular superior de **T**:

$$T_{i,j} = \begin{cases} \mathbf{T}_{i-1,j-1} + 1 & \text{se } S_i = S_j. \\ & \text{considerando } 2 \leq i \leq (n-1),\ 3 \leq j \leq n, \text{ e } i < j. \\ 0 & \text{em outro caso.} \end{cases}$$

2. Detecção dos Padrões a partir da Matriz
   (a) Considere $\mathbf{S}[a : b]$ um subconjunto de **S** indo da a-ésima até a b-ésima nota.
   (b) Considere um conjunto de padrões candidatos **CS** inicializado como vazio.
   (c) $\forall\ T_{i,j} > 0\ \exists\ \mathbf{P} = \mathbf{S}[j - T_{i,j} : j]$, sendo que **P** e todas as suas substrings são padrões repetitivos.
   (d) Na medida em que todas as substrings que não são sufixos[2] de **P** são calculados durante o processamento de outras células, somente é necessário o processamento do próprio **P** e de suas substrings sufixos. Sendo assim, para cada padrão *pat*, subconjunto sufixo de **P**, acumula-se o número de repetições (`rep_count`), verifica-se se é subconjunto de outro padrão (`sub_count`) e armazena-se o resultado no conjunto **CS**, o que pode ocorrer através de um dos quatro casos:

[2]São chamados aqui aqui de sufixos as substrings cujo último elemento é também o último elemento de **P**.

Caso 1
Se $T_{i+1,j+1} = 0 \ \wedge \ pat \notin \mathbf{CS}$, $insere(pat, 1, 0)$ em **CS**.

Caso 2
Se $T_{i+1,j+1} = 0 \ \wedge \ pat \in \mathbf{CS}$, atualiza $(pat, rep\_count, sub\_count)$ para $(pat, rep\_count + 1, sub\_count)$.

Caso 3
Se $T_{i+1,j+1} \neq 0 \ \wedge \ pat \notin \mathbf{CS}$, $insere(pat, 1, 1)$ em **CS**.

Caso 4
Se $T_{i+1,j+1} \neq 0 \ \wedge \ pat \in \mathbf{CS}$, atualiza $(pat, rep\_count, sub\_count)$ para $(pat, rep\_count + 1, sub\_count + 1)$.

(e) Após a inserção de todos os padrões repetitivos no conjunto candidato **CS** é necessário remover os padrões triviais. Isto é feito simplesmente eliminando os padrões que possuem os mesmos índices `rep_count` e `sub_count`.

(f) Finalmente, é necessário calcular a freqüência dos padrões restantes e não-triviais, através da equação (Hsu et al., 2001):

$$f_p = \frac{(1 + \sqrt{1 + 8 \times rep\_count_p})}{2}$$

Implementação

Apesar de sua eficiência, método apresentado originalmente possui uma limitação digna de nota. Esta provém do modo como música é normalmente estruturada. De acordo com a praxis tradicional, um determinado padrão pode (e normalmente o faz) aparecer, durante o transcurso de uma obra, com pequenas variações em sua forma original. Sendo assim, um sistema visando a detecção de padrões deve ser capaz de lidar com estas pequenas variações nas diversas aparições de um determinado padrão. Este não é o caso, entretanto, da presente heurística em sua forma original, a qual somente percebe repetições literais de um padrão. No sentido de superar este problema três melhorias foram implementadas.

O trabalho original, tratando exclusivamente da detecção de padrões de alturas, elimina a possibilidade do reconhecimento do padrão transposto (musicalmente falando) como sendo apenas uma variante do mesmo. Com o objetivo de incrementar a capacidade do algoritmo implementou-se, além da detecção de alturas somente, também a detecção de padrões intervalares, o que elimina a falha de reconhecimento em padrões transpostos[3].

Na mesma linha de pensamento, considerou-se interessante e enriquecedor a inclusão

[3]É importante ressaltar aqui que o termo *transposição*, em Música, significa o deslocamento de um dado conjunto de alturas a um dado intervalo. Assim, ao utilizar-se uma representação numérica para as alturas (em inteiros), *transpor* um trecho significa simplesmente realizar a soma algébrica de uma constante (representando o intervalo ao qual se deseja transpor o trecho) aos valores representantes das alturas do trecho. Por exemplo, se o padrão [72–67–78] for *transposto* ascendentemente de um intervalo de 3 (*terça-maior*), tornar-se-á [75–70–81], enquanto se for transposto descendentemente por um intervalo de -2 (*segunda-maior*), tornar-se-á [70–65–76].

das variações temáticas *retrogradação*, *inversão* e *retrogradação da inversão*[4], todas tanto do ponto de vista de alturas quanto do ponto de vista intervalar. Desta forma, o identificador torna-se mais adequado no tratamento analítico de obras polifônicas, nas quais são comuns tais tipos de variação e tratamento.

Finalmente, foi considerado durante o cálculo uma banda de passagem ao invés de uma única altura. Este método torna possível detectar pequenas variações intervalares no interior de padrões perceptivamente idênticos. Visando seguir a prática musical real, a largura de banda aplicada a uma altura é proporcional à dimensão do intervalo entre ela e a próxima altura.

4.5.2 Sistema Nebuloso

O objetivo do sistema nebuloso utilizado é a criação automática de uma base de regras nebulosas em ambientes para os quais não existe um modelo matemático ou nos quais este é altamente não-linear. Trata-se, portanto, de um sistema adequado para ser empregado em aplicações musicais.

No que diz respeito à sua implementação, esta pode ser dividida em cinco partes:

1. Divisão dos espaços de entrada e saída em regiões nebulosas.
2. Geração de regras nebulosas a partir de pares de dados entrada/saída.
3. Atribuição de graus às regras geradas.
4. Criação de uma base de regras.
5. Desnebulização.

Para a clarificar a descrição de cada passo suporemos um sistema com duas entradas e uma saída, constituído pelos pares de dados:

$$(x_1^{(1)}, x_2^{(1)}; y^{(1)}), (x_1^{(2)}, x_2^{(2)}; y^{(2)}), \ldots$$

onde x_i^k são as entradas e y^k é a saída.

A tarefa a ser realizada, portanto, é a geração de um conjunto de regras nebulosas a partir dos dados acima e utilizá-lo na determinação de um mapeamento do tipo:

$$f : (x_1, x_2) \rightarrow y$$

[4]As quatro formas usuais de um padrão melódico são sua forma original, sua retrogradação, sua inversão e a retrogradação da inversão. A retrogradação é a leitura do padrão do final ao começo, a inversão é a multiplicação de todos os intervalos ordenados (Straus, 1990) entre os pares de alturas seqüenciais do padrão por -1 e, finalmente, a retrogradação da inversão é a leitura do padrão invertido partindo do final e indo até o começo. Assim, considerando o padrão original [72–67–78], utilizado como exemplo em nota anterior, tem-se que sua retrogradação é [78–67–72], sua inversão (em torno do valor inicial) é [72–77–66] e sua retrogradação da inversão [66–77–72].

Passo 1 – Divisão dos espaços de entrada e saída em regiões nebulosas

O primeiro passo corresponde ao estabelecimento de intervalos de domínio para as variáveis de entrada e saída, onde intervalo de domínio de uma variável significa que existe uma alta probabilidade da variável assumir valores no intervalo em questão.

Para os pares de dados em questão, tem-se:

$$[x_1^-, x_1^+], [x_2^-, x_2^+] \text{ e } [y^-, y^+]$$

onde $[x_i^-, x_i^+]$ é o domínio da i-ésima entrada e $[y^-, y^+]$ é o domínio da saída.

Após a determinação dos intervalos de domínio, dividir cada um deles em $2N + 1$ regiões, sendo que N poderá ser diferente para cada variável e os comprimentos das regiões poderão ser iguais ou não. As diferentes regiões serão denominadas:

$$S_N(SmallN), ..., S_1(Small1), C_E(Center), B_1(Big1), ..., B_N(BigN).$$

Por exemplo, se $N = 2$ ter-se-á o conjunto que pode ser visto na figura 4.1.

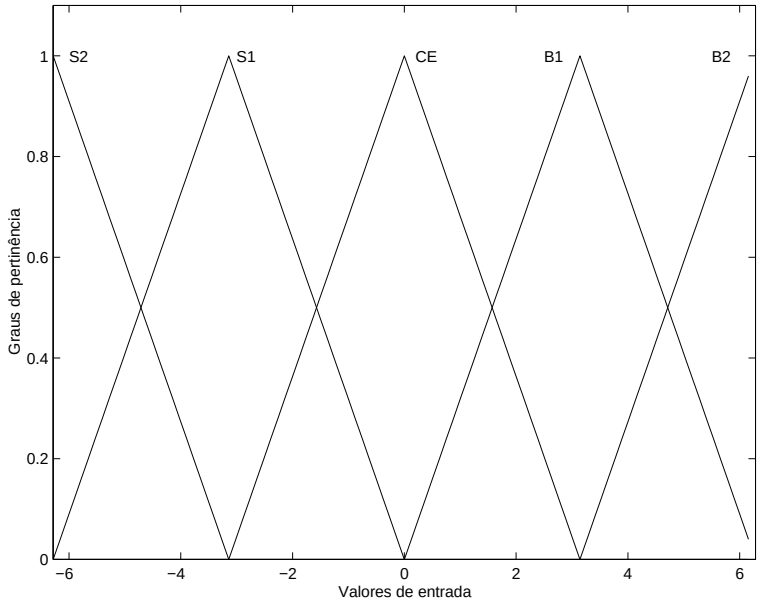

Figura 4.1: Regiões nebulosas para $N = 2$.

Passo 2 – Geração de regras nebulosas a partir de pares de dados

Inicialmente determina-se o grau de pertinência de cada variável em cada região, guardando-se o máximo dentre os encontrados e, em seguida, gera-se uma regra a partir deles. Por exemplo, se

$$(x_1^{(1)}, x_2^{(1)}) \Rightarrow [x_1^{(1)}(0,8/B1, max), x_2^{(1)}(0,7/S1, max); y^{(1)}(0,9/CE, max)]$$

isto implica na regra:

SE x_1 é B1 e x_2 é S1, ENTÃO y é CE.

que representa o modo como a saída está relacionada com as duas entradas.

Passo 3 – Atribuição de graus às regras geradas

Na medida em que, possivelmente, existirá uma grande quantidade de pares de dados e que cada um deles gerará uma regra, é altamente provável que ocorrerão regras conflitantes, ou seja, regras com a mesma premissa. Os autores do método propõem um modo de resolver este conflito que é baseado na atribuição de graus às regras encontradas, aceitando, dentro de um grupo de regras conflitantes, somente aquela que possuir o máximo grau. Além de resolver o problema do conflito entre regras, esta operação traz a vantagem de reduzir grandemente o seu número.

Para o cálculo do grau de uma regra a estratégia adotada é fazer o produto dos graus de seus componentes. Assim:

$$D(Regra) = m_A(x_1)m_B(x_2)m_C(y)$$

onde $m_A(x_1)$ é o grau de pertinência da primeira entrada, $m_B(x_2)$ é o grau de pertinência da segunda entrada e $m_C(y)$ o grau de pertinência da saída.

Considerando-se o exemplo anterior, ter-se-á:

$$D(Regra1) = m_{B1}(x_1)m_{S1}(x_2)m_{CE}(y) = 0,8 \times 0,7 \times 0,9 = 0,504$$

Passo 4 – Criação de uma base de regras

O próximo passo é a criação de uma base a partir das regras geradas nos dois passos anteriores. A forma da base de regras pode ser vista na tabela 4.1. Nesta está exemplificada a regra exposta anteriormente:

SE x_1 é B1 e x_2 é S1, ENTÃO y é CE.

B3					
B2					
B1					
X_2 CE					
S1				CE	
S2					
S3					
	S2	S1	CE X_1	B1	B2

Tabela 4.1: Base de Regras.

Passo 5 – Desnebulização

A estratégia de desnebulização empregada, por sua vez, pode ser dividida em dois passos principais. Em primeiro lugar, dadas duas entradas (x_1, x_2), combinar os antecedentes da i-ésima regra nebulosa, através de seu produto, para determinar o grau de controle de saída correspondente a (x_1, x_2), ou seja:

$$m^i_{O^i} = m^i_{I^i_1}(x_1) m^i_{I^i_2}(x_2)$$

onde O^i é a região de saída da i-ésima regra e I^i_j é a região de entrada da i-ésima regra para o j-ésimo componente. Considerando nosso exemplo anterior:

$$m^1_{CE} = m_{B1}(x_1) m_{S1}(x_2)$$

Em seguida, empregar a seguinte equação de desnebulização do tipo centróide:

$$y = \frac{\sum_{i=1}^{K} m^i_{O^i} \overline{y}^i}{\sum_{i=1}^{K} m^i_{O^i}}$$

onde $\overline{y}^i$ é o centro[5] da região O^i e K é o número de regras na base.

Treinamento e Utilização do Sistema

Com o objetivo de facilitar o entendimento do treinamento do sistema, iniciamos esta subseção com uma repetição da RPA 2 e da RPA 3.

RPA 2 (Proximidade) Considere uma seqüência de quatro notas $n_1 n_2 n_3 n_4$. Tudo mais sendo igual, a transição n_2–n_3 pode ser ouvida como a fronteira de um grupo se

a. (Ligadura de expressão/Pausa) o intervalo de tempo do fim de n_2 ao começo de n_3 é maior do que aquele do fim de n_1 ao começo de n_2 e do que aquele do fim de n_3 ao começo de n_4.

b. (Ponto de Ataque) o intervalo de tempo entre os pontos de ataque de n_2 e n_3 é maior do que aquele entre os pontos de ataque de n_1 e n_2 e do que aquele entre os pontos de ataque de n_3 e n_4.

RPA 3 (Similaridade) Considere uma seqüência de quatro notas $n_1 n_2 n_3 n_4$. Tudo mais sendo igual, a transição n_2–n_3 pode ser ouvida como a fronteira de um grupo se

[5]Compreende-se por centro de uma região nebulosa como o ponto que tem o menor valor absoluto entre todos os pontos para os quais a função de pertinência para esta região tem valor de pertinência igual a um. Como, no nosso caso, trataremos somente com funções de pertinência triangulares, existe somente um ponto para o qual uma função de pertinência possui valor igual a 1.

a. (Tessitura) a transição n_2–n_3 possui uma distância interválica maior do que n_1–n_2 e n_3–n_4, ou se

b. (Dinâmica) a transição n_2–n_3 possui uma mudança na dinâmica e n_1–n_2 e n_3–n_4 não possuem, ou se

c. (Articulação) a transição n_2–n_3 possui uma mudança de articulação e n_1–n_2 e n_3–n_4 não possuem, ou se

d. (Comprimento) n_2 e n_3 são de diferentes comprimento e os pares n_1,n_2 e n_3,n_4 não diferem em comprimento.

É possível notar, imediatamente, dois pontos importantes. Trabalha-se sempre com as diferenças entre certas características de quatro eventos (o que sugere um tratamento vetorial tridimensional) e existe, para cada uma das regras, subdivisão em sub-regras (duas para a RPA 2 e quatro para a RPA 3).

Considerando que os sistemas nebulosos são aproximadores universais de funções (Wang & Mendel, 1992; Jang et al., 1997; Wang, 1994), o melhor seria pensar em funções que representassem cada uma das sub-regras e, com a ajuda das mesmas, realizar o treinamento do sistema. Como todas as regras são baseadas na diferença entre características de eventos, foi empregada uma única função que pudesse ser uma representação de todas elas. Foi, então, definida a função abaixo,

$$f(d_1, d_2, d_3) = \begin{cases} 1 - ((d_1 + d_3)/2)/d_2 & \text{se } d_2 > d_1 \text{ e } d_2 > d_3 \\ 0 & \text{senão} \end{cases}$$

onde d_1, d_2 e d_3 são, respectivamente, as diferenças características dos pares de eventos n_1–n_2, n_2–n_3 e n_3–n_4. Do ponto de vista das regras, o significado da função é que, quanto maior for d_2 em relação a d_1 e d_3, tanto maior será probabilidade de existir uma fronteira de grupo entre os eventos n_2–n_3.

A geração dos pontos de treinamento foi, então, realizada através de um algoritmo cuja forma geral é a que segue:

Para cada ponto de treinamento:

1. Gerar três números aleatórios no intervalo $[0, 1]$ correspondentes às diferenças d_1, d_2 e d_3.
2. Utilizar os números encontrados no item anterior como entrada para a função mostrada acima e calcular a saída da mesma.

Alguma diferenciação, entretanto, fez-se necessária devido à diversidade dos dados empregados em cada uma das regras.

Para a RPA 2 e a RPA 3d a aplicação do algoritmo mostrado acima é imediata, já que estas trabalham com proporções entre durações, ou, em outras palavras, são regras cujas entradas são valores contínuos.

Tal não é o caso das três sub-regras iniciais da RPA 3, que têm como entrada, respectivamente, diferenças de registro e de intensidades (números inteiros no protocolo MIDI).

Para estes três casos particulares, optou-se por utilizar os valores de ocorrência possíveis para cada um deles e depois proceder a uma normalização pela norma unitária com o intuito de utilizar a função já mostrada acima.

Com os meios descritos acima, foram gerados seis conjuntos de 50.000 pontos de treinamento, um para cada uma das sub-regras. Em seguida, a partir desses conjuntos, foram calculadas seis bases de regras, mais uma vez, uma para cada uma das sub-regras. Estas bases de regras são utilizadas diretamente pelo programa principal responsável pela Estrutura de Agrupamento.

As características gerais das bases de regras podem ser vistas a seguir:

1. Dimensão da entrada igual a 3.
2. Dimensão da saída igual a 1.
3. Valores mínimo e máximo de cada dimensão da entrada iguais 0 e 1, respectivamente.
4. Valores mínimo e máximo da saída iguais 0 e 1, respectivamente.
5. Onze funções de pertinência triangulares tanto para a entrada quanto para a saída.

Finalmente, duas observações fazem-se ainda necessárias para a plena compreensão da implementação das RPA 2 e 3. A primeira delas diz respeito ao emprego, dentro da função principal de cada uma das regras, de uma constante nomeada *limiar de percepção*. Esta constante determina o valor a partir do qual uma certa localização dentro do discurso musical pode ser considerada uma fronteira de grupo. Em outras palavras, significa o limiar a partir do qual uma fronteira de um grupo será perceptível como tal. O valor do limiar de percepção fica situado no intervalo $[0, 1]$, sendo seu valor encontrado empiricamente. Tanto na RPA 2 quanto na RPA 3 o limiar de percepção foi fixado em $0,1$.

A segunda observação refere-se à importância relativa das RPA 2 e 3. Na página 47 do principal texto no qual se baseia este trabalho (Lerdahl & Jackendoff, 1996), é dito que *"de modo geral, todos os casos da RPA 3, com a possível exceção da regra de dinâmica, parecem possuir efeitos mais fracos do que a RPA 2"*. De fato, durante a elaboração da implementação das regras a afirmação acima foi confirmada. Sendo assim, optou-se por utilizar os resultados da RPA 3 multiplicados por um *fator de relevância* que evidenciasse a diferença entre as duas regras. Após experimentos, chegou-se, para o fator citado acima, a um valor de $1,5 \times 10^{-1}$.

4.6 Implementação da Estrutura de Agrupamento

4.6.1 Arquitetura do Componente

A proposta deste componente é receber como entrada uma seqüência de eventos musicais e calcular a respectiva Estrutura de Agrupamento em seus diversos níveis.

Com o objetivo de implementar os dois conjuntos de regras e sua interação é proposta uma arquitetura como a mostrada na figura 4.2, onde as RBFAs agem como *checkers* no

que diz respeito às RPAs e moldam as saídas das últimas visando os requerimentos da Boa–Formatividade.

Desta forma, o sistema recebe como entrada um arquivo MIDI e apresenta na saía uma matriz cujas colunas são as possíveis fronteiras entre os grupos e as linhas são os níveis de agrupamento (no que diz respeito à geração de níveis ver 4.6.3). Para cada nível suas fronteiras podem ser zero (inativas ou inexistente) ou ter um valor real $0 < b \leq 1$ (ativas), sendo sua importância diretamente proporcional a este valor.

Na medida em que as fronteiras são definidas a partir de diferenças entre dois pares de eventos, e também que entre cada par existe uma possível fronteira, caso tenha-se uma seqüência de x eventos existirão $x - 3$ fronteiras possíveis, tal como é mostrado na figura 4.3. Nesta é possível ver uma análise completa (ainda que hipotética) da estrutura de agrupamento de uma amostra com onze eventos e oito possíveis fronteiras.

4.6.2 Regras de Boa–Formatividade de Agrupamento

Como mostrado na figura 4.2, as Regras de Boa-Formatividade atuam como *checkers* do processamento realizado pelas Regras Preferenciais. Com exceção e RBFA 1, a qual recebe como entrada os dados do arquivo MIDI, todas as outras RBFAs recebem como entrada a saída de RBFA 5. Desde que as cinco RBFAs são, comparadas às RPAs, heurísticas simples, nesta subseção são mencionadas não mais do que as descrições de suas operações:

RBFA 1
: Verifica para todos os eventos se a soma de seus ataques mais suas durações é maior do que o ataque do próximo evento. Se o for, corrige o problema diminuindo a duração do evento atual e envia uma mensagem de alerta.

RBFA 2
: Verifica se o mais baixo nível de análise corresponde à totalidade da peça. Se não, envia uma mensagem de erro indicando sua localização.

RBFA 3
: Faz a contagem de quantos níveis de análise e de quantos grupos existem em cada nível. Se existe somente um nível ou grupo (exceto o último) envia uma mensagem de alerta.

RBFA 4
: Verifica se existe alguma fronteira que está localizada no interior de algum grupo em algum nível anterior. Caso exista, envia uma mensagem de erro indicando sua localização.

RBFA 5
: Verifica se a duração de um grupo é igual à soma das durações de todos os seus subgrupos. Caso não seja, envia uma mensagem de erro indicando sua localização.

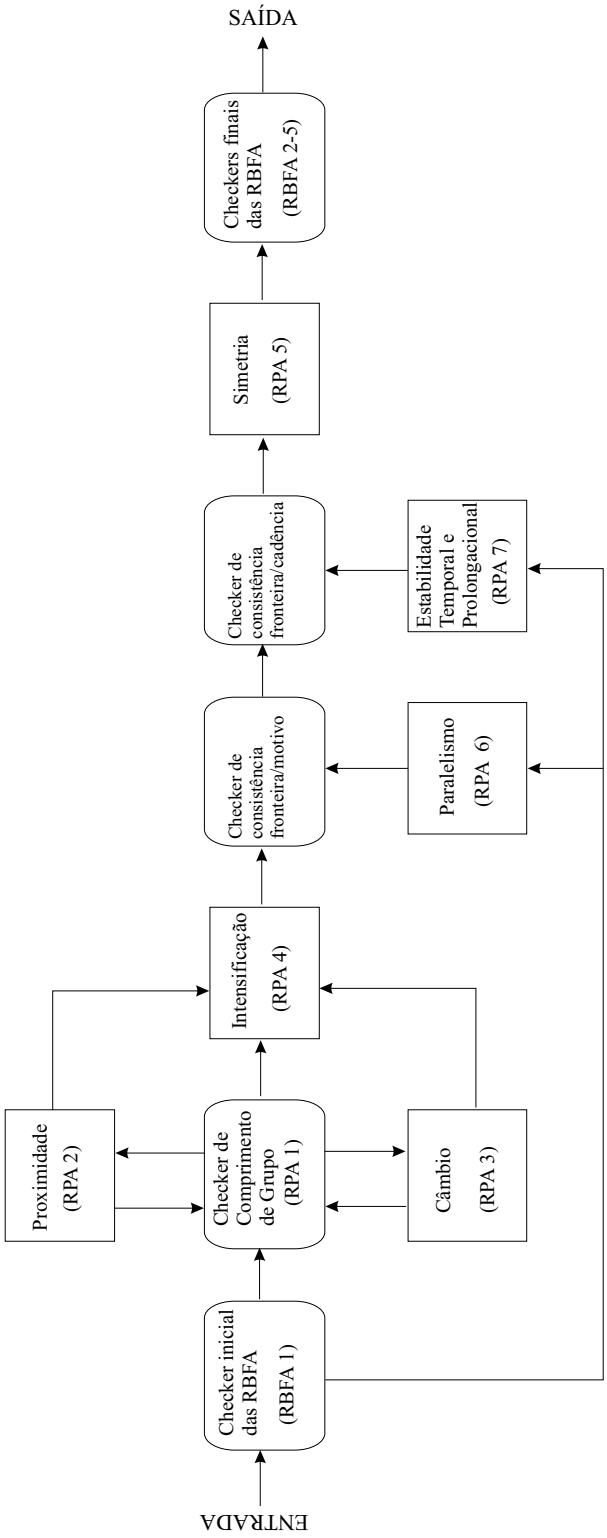

Figura 4.2: Arquitetura da Estrutura de Agrupamento.

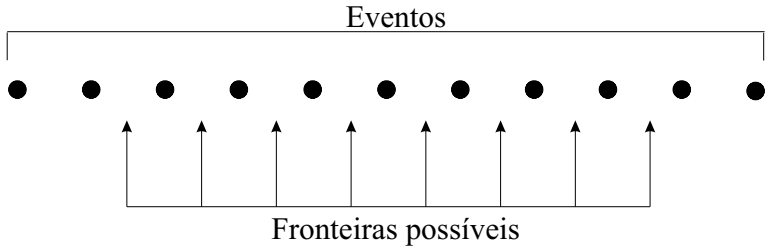

Figura 4.3: Fronteiras e eventos.

4.6.3 Regras Preferenciais de Agrupamento

RPA 1 – Comprimento de Grupo

Como dito antes e mostrado na figura 4.2, RPA 1 atua apenas como um *checker* em relação a RPA 2 e RPA 3 visando assegurar que nenhum grupo possua somente um elemento. A suposição básica para atingir este objetivo é considerar que, em uma lista de fronteiras, *duas fronteiras adjacentes com valores maiores do que zero caracteriza um grupo com apenas um único elemento.* O algoritmo empregado na implementação desta regra é o seguinte:

1. Atribua a um apontador a posição do segundo elemento da lista de fronteiras;
2. Se tanto o valor da posição atual quanto o da anterior são maiores do que zero, some os dois valores e coloque o resultado na fronteira que possui o maior valor;
3. Repita as operações anteriores trocando a posição anterior pela posterior à posição atual;
4. Se o apontador está na penúltima fronteira, então vá para o passo 6;
5. Senão, incremente o apontador e vá para o passo 2;
6. Normalize a lista de fronteiras pela norma unitária;

Regras 2 e 3 - Proximidade e Similaridade

Tanto a RPA 2 quanto RPA 3 utilizam, na sua implementação, o sistema nebuloso desenvolvido por Wang & Mendel (1992) e descrito na seção 4.5.2. Nesta pode ser visto como foi treinado e utilizado o sistema no processamento das duas regras.

RPA 4 - Intensificação

Após o processamento de RPA 2 and RPA 3 é necessário realizar sua *intensificação.* Este procedimento visa resolver os conflitos e ambigüidades que emergem devido ao uso concorrente das duas regras.

São três os fundamentos que tornam possível o algoritmo de intensificação, a saber, uma estrutura de dados que represente uma fronteira de grupo e seu valor característico, o conceito de *limiar de percepção* e o conceito de *limiar de vizinhança*. Na medida em que os dois primeiros já foram anteriormente abordados, resta aqui a necessidade de definir-se o último deles.

O limiar de vizinhança representa o "domínio" que uma fronteira de grupo exerce sobre uma outra sempre que ambas estão no interior de um mesmo intervalo temporal. Em outras palavras, dentro de um mesmo limiar de vizinhança, somente a fronteira com maior valor permanece ativa, sendo as demais desativadas (recebem valor igual a zero). O limiar de vizinhança é calculado assim:

$$\theta = 2^n \delta \tag{4.1}$$

onde θ é o limiar de vizinhança, $n \in \mathbb{N}$ e δ é o *quantum de duração*, o qual é a menor duração existente na amostra musical a ser analisada. Nos testes realizados neste trabalho n foi feito igual a 1, o que significa que o limiar de vizinhança θ ficou igual a dois quanta de duração.

Estando definidos os conceitos necessários para a implementação de RPA 4, o algoritmo completo pode ser agora apresentado:

1. Alocar um vetor $\mathbf{G}$ de fronteiras de grupo com dimensão:

$$d = \frac{T}{\delta} \tag{4.2}$$

   onde T é a duração total da amostra musical, e o inicializar com zeros.

2. Para cada uma das estruturas de agrupamento a serem intensificadas, varrê-las em quanta de duração e marcar em $\mathbf{G}$ onde existem fronteiras de grupos. Em caso de coincidências, os valores devem ser somados.

3. Normalizar $\mathbf{G}$ na norma unitária observando o *limiar de percepção* considerado.

4. Aplicar o algoritmo de *Eliminação de Vizinhanças*:

   (a) Fazer $I = \theta$, onde I é uma variável auxiliar.

   (b) Varrer, em quanta de duração, o intervalo $[I - \theta, I + \theta]$ armazenando a posição (fronteira de grupo) que tem o maior valor.

   (c) Varrer novamente o intervalo atribuindo zero a todas posições que não tem o valor máximo.

   (d) Fazer $I = I + \theta$.

   (e) Volte ao passo 4a até que $I + \theta > T$.

RPA 5 – Simetria

Inicialmente, para implementação da RPA 5 foi imaginado o método descrito no Apêndice B. Apesar da elegância e eficiência deste método, ele foi, entretanto, posteriormente substituído pelo algoritmo mostrado abaixo por este ser menos complexo e com custo computacional inverso.

A RPA 5 recebe da RPA 4 uma estrutura de agrupamento resultante da intensificação entre a RPA 2 e a RPA 3. Sua tarefa é calcular os próximos níveis de análise até que todas as fronteiras sejam iguais a zero, ou seja, o único grupo do último nível coincide com a peça em sua integridade.

Entretanto, é importante lembrar que antes que saída de RPA 4 possa chegar na entrada de RPA 5 é feita uma verificação se existem fronteiras que quebrem a continuidade de algum padrão (RPA 6) ou cadência (RPA 7). Se isto acontece, a fronteira é desativada (seu valor torna-se zero). Segue o algoritmo para a RPA 5:

1. Considerar a saída dos *checkers* (para a RPA 6 e a RPA 7) como o primeiro nível da análise (primeira linha da matriz níveis × fronteiras).
2. Ir para o próximo nível:
    a. Copiar o array de fronteiras do nível anterior para o nível atual.
    b. Encontrar o mínimo valor (maior do que zero) do nível atual.
    c. Multiplicar seu valor por 1.2 (o *limiar de fronteira*) e armazenar o resultado.
    d. Atribuir zero a todas as fronteiras cujos valores são iguais ou menores do que o limiar de fronteira.
3. Se todas as fronteiras são zero, ir para o passo 4; senão, ir para passo 2.

RPA 6 – Paralelismo

Na TGMT o conceito de "paralelismo" é vago e abrangente. Neste trabalho o termo paralelismo significa simplesmente a reiteração de motivos, ou, em outras palavras, somente as estruturas melódicas são consideradas. O método empregado para esta tarefa é a heurística já descrita em 4.5.1, a qual emprega uma matriz correlativa visando a detecção de estruturas motívicas.

RPA 7 – Estabilidade Temporal e Prolongacional

A função de RPA 7 é optar por fronteiras de grupo que minimizem conflitos com a estabilidade harmônica da amostra sendo analisada. Esta estabilidade harmônica reflete-se, principalmente, na manutenção da integridade nas cadências. Visando a realização deste trabalho tem-se como primeito passo a análise harmônica da amostra.

O presente trabalho emprega um analisador harmônico baseado numa técnica conhecida como *Constraint Satisfaction Problem* (CSP) e sugerida por Hoffman & Birmingham (2000) e que está descrito detalhadamente no capítulo 7, página 127.

4.7 Resultados

4.7.1 Testes de Validação

Sistema Nebuloso

Considerando que as RPA 2 e RPA 3 são processadas utilizando o sistema nebuloso descrito na seção 4.5.2 e interagem com as RPA 1, RPA 4 e RPA 5, na figura 4.4 é mostrado um exemplo completo da aplicação das cinco primeiras regras preferenciais. Nesta são mostrados os valores numéricos gerados por cada regra com o objetivo de proporcionar uma melhor compreensão dos seus processos individuais e de como elas funcionam. Onde os cruzamentos entre fronteiras e regras estão sem números significa que as fronteiras tem valor zero (desativada).

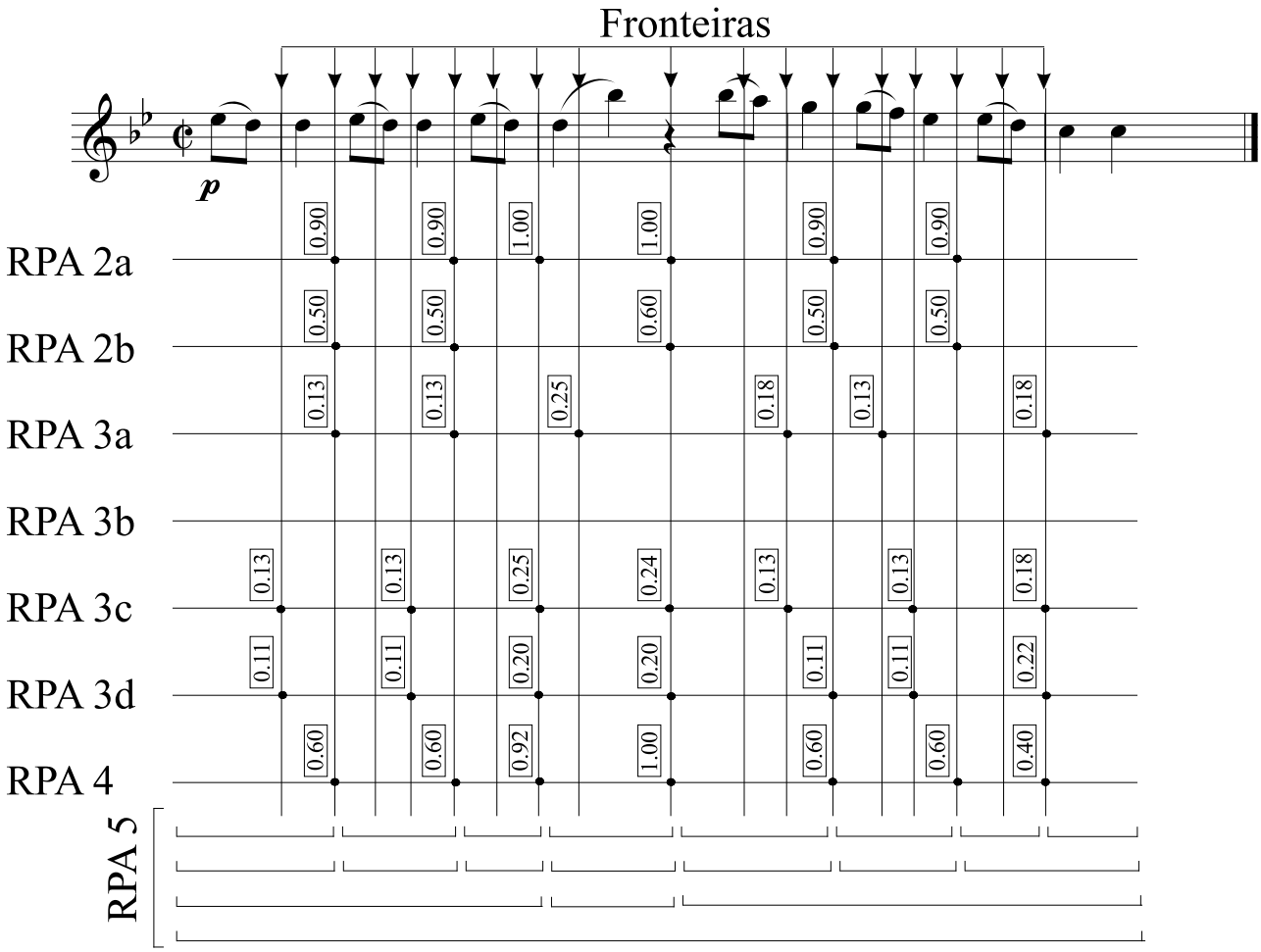

Figura 4.4: Exemplo da atuação conjunta das primeiras cinco RPAs.

Identificador de Padrões Não-Triviais

Para os testes com o identificador de padrões não-triviais foram elaborados seis exemplos melódicos com crescente nível de complexidade na identificação.

Para o primeiro exemplo, apresentado na figura 4.5 são mostrados a sua matriz correlativa, os resultados da busca por padrões de alturas e os resultados da busca por padrões intervalares. Tanto para a busca por padrões de alturas quanto para aquela por padrões intervalares são consideradas as formas original, retrógrado, inversão e retrógrado da inversão. Para os demais exemplos, apresentados nas figuras 4.6 até 4.10, somente é dada a representação musical, sendo os padrões indicados por letras gregas e/ou colchetes.

Os resultados, com suas respetivas representações musicais, estão a seguir.

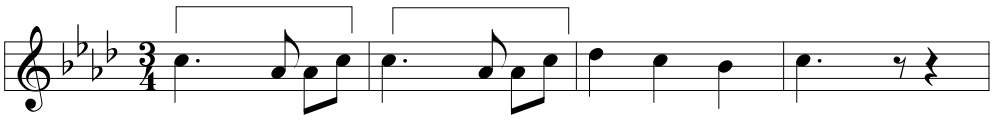

Figura 4.5: Identificador de Padrões Não-Triviais: representação musical do exemplo 1.

```
Analise dos padroes de alturas:

- - - 1 1 - - 1 - 1 - 1
- - 1 - - 2 1 - - - - -
- - - - - 1 3 - - - - -
- - - - 1 - - 4 - 1 - 1
- - - - - - - 1 - 1 - 1
- - - - - - 1 - - - - -
- - - - - - - - - - - -
- - - - - - - - - 1 - 1
- - - - - - - - - - - -
- - - - - - - - - - - 1
- - - - - - - - - - - -
- - - - - - - - - - - -
PADROES REPETITIVOS:
Padrao: {72}
Frequencia: 6
Posicoes = 0 3 4 7 9 11
Padrao: {68}
Frequencia: 4
Posicoes = 1 2 5 6
Padrao: {72, 68, 68, 72}
Frequencia: 2
Posicoes = 0 4

PADROES INVERTIDOS:
Padrao: {72}
Frequencia: 6
Posicoes = 0 3 4 7 9 11
```

```
Padrao: {68}
Frequencia: 4
Posicoes = 1 2 5 6

PADROES RETROGRADADOS:

PADROES RETROGRADADOS NA INVERSAO:


-----------------------------------------------


Analise dos padroes intervalares:

- - - - 1 - - - - - -
- - - 1 - 2 - - - - -
- - - - - - 3 - - - -
- - - - - 1 - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
- - - - - - - - - - -
PADROES REPETITIVOS:
Padrao: {0}
Frequencia: 3
Posicoes = 1 3 5
Padrao: {-4, 0, 4}
Frequencia: 2
Posicoes = 0 4

PADROES INVERTIDOS:
Padrao: {0}
Frequencia: 3
Posicoes = 1 3 5
Padrao: {-4, 0, 4}
Frequencia: 1
Posicoes = 2

PADROES RETROGRADADOS:
Padrao: {0}
Frequencia: 3
Posicoes = 1 3 5
Padrao: {-4, 0, 4}
Frequencia: 2
Posicoes = 0 4

PADROES RETROGRADADOS NA INVERSAO:
Padrao: {0}
Frequencia: 3
```

```
Posicoes = 1 3 5
Padrao: {-4, 0, 4}
Frequencia: 1
Posicoes = 2
```

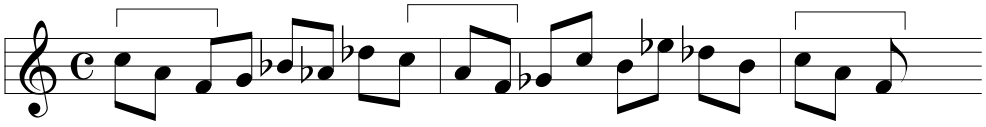

Figura 4.6: Identificador de Padrões Não-Triviais: representação musical do exemplo 2.

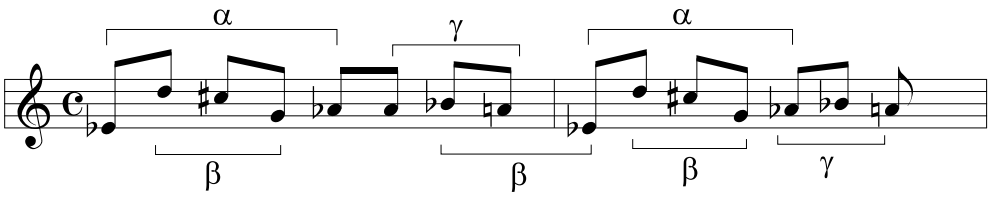

Figura 4.7: Identificador de Padrões Não-Triviais: representação musical do exemplo 3.

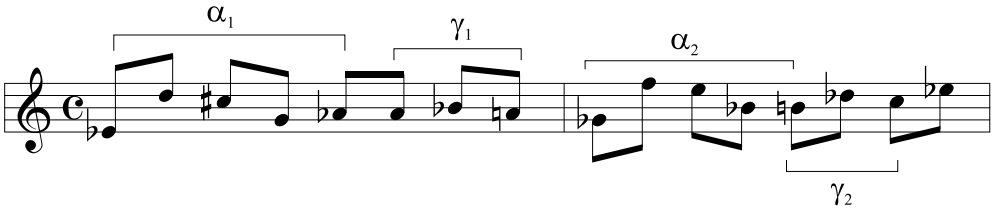

Figura 4.8: Identificador de Padrões Não-Triviais: representação musical do exemplo 4.

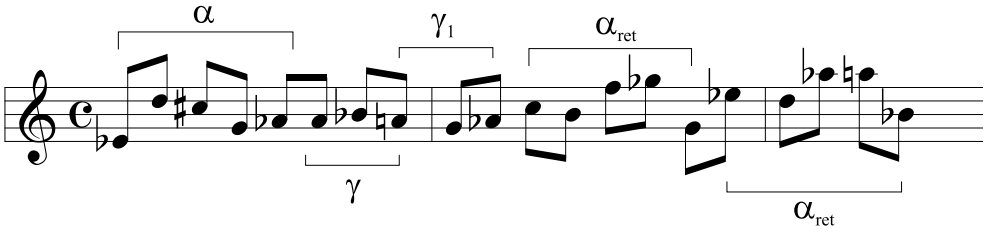

Figura 4.9: Identificador de Padrões Não-Triviais: representação musical do exemplo 5. O termo *ret* significa a retrogradação de um padrão.

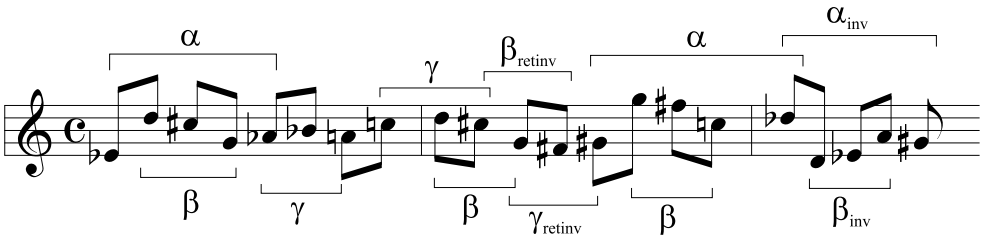

Figura 4.10: Identificador de Padrões Não-Triviais: representação musical do exemplo 6. Os termos *inv* e *retinv* significam, respectivamente, a inversão e a retrogradação da inversão de um padrão.

4.7.2 Análises

Com o intuito de testar o sistema proposto foram selecionadas três obras de diferentes períodos históricos. Elas são o coral *Christus, der ist mein Leben* (BWV 95), de Johann Sebastian Bach, o tema do 3º movimento do Quarteto de Cordas Op. 18 Nº5, de Ludwig van Beethoven e a primeira das *Três Peças (para clarinete solo)*, de Igor Stravinsky.

Na medida em que os resultados, tal como encontrados na saída do sistema, são seqüências de números, aqui eles são apresentados sob a forma de figuras em notação musical clássica, almejando, desta forma, obter uma melhor representação da análise. Além disto, com o objetivo de mostrar os resultados em uma forma mais apropriada de visualização, os primeiros dois trabalhos são apresentados inicialmente em seu texto original (figura 4.11 e figura 4.14), depois é mostrada sua análise harmônica (figura 4.12 e figura 4.15) e finalmente seus vários níveis de agrupamento mais sua análise motívica (figura 4.13 e figura 4.16), esta última tomando como referência somente as partes superiores de cada peça para simplificar a apresentação.

No que diz respeito ao trabalho de Stravinsky, considerando que se trata de uma obra para uma única linha melódica e sem uma estrutura harmônica explícita, o passo descrito acima torna-se desnecessário e os resultados de sua análise são apresentados diretamente no texto original (figura 4.17).

No que tange à análise harmônica, a saída do sistema é um conjunto de informações que inclui, para cada acorde, a tonalidade e o modo (maior ou menor), a estrutura interna (a classe do acorde), a função harmônica dentro da tonalidade, a função formal (se o acorde é ou não cadencial), a fundamental e o estado do acorde[6]. Nas presentes análises todas estas informações são traduzidas na figura através da simbologia clássica atualmente empregada na análise musical. Desta forma, nas figuras correspondentes às análises harmônicas estão indicadas todas as informações necessárias para o correto entendimento do texto musical.

Quanto à análise motívica, somente as partes superiores das duas obras polifônicas foram consideradas na análise e mostradas nas correspondentes figuras, tanto porque isto é suficiente para ilustrar o processo quanto para não sobrecarregar a figura. Além disto,

[6]Este tipo de saída pode ser examinado nos Teste de Validação da Redução Prolongacional, capítulo 7.

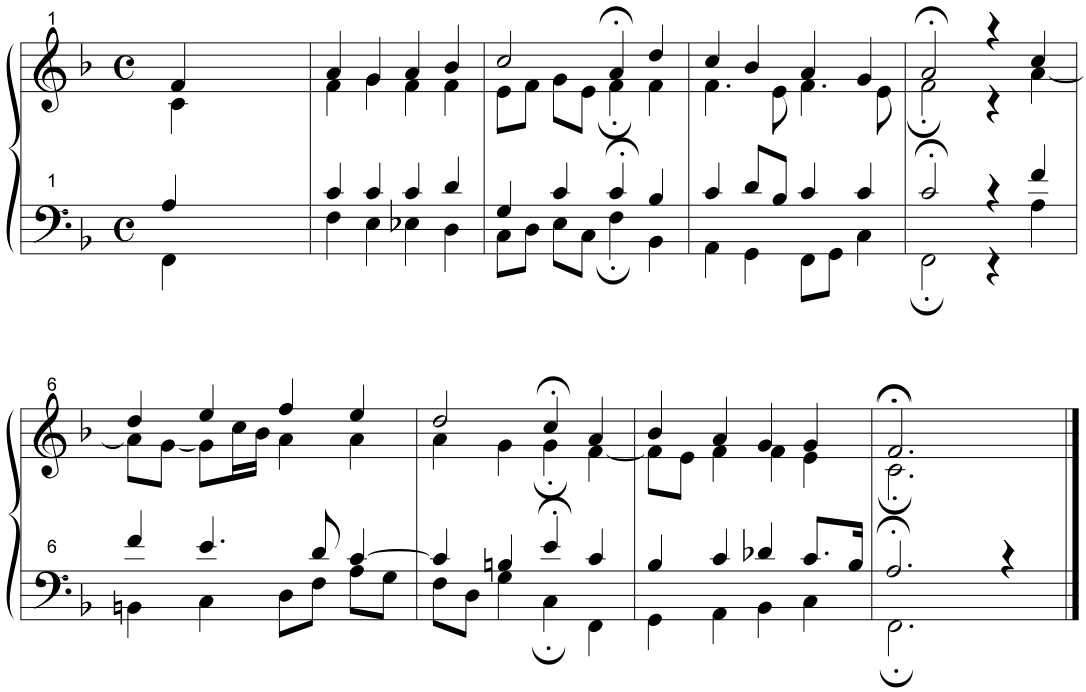

Figura 4.11: Texto original de *Christus, der ist mein Leben* (BWV 95) de J.S. Bach.

com o mesmo propósito, nem todas as relações motívicas encontradas foram representadas graficamente, já que o sistema indica, para cada amostra, um grande número de combinações de alturas. Considerando que para o coral de Bach foram encontradas vinte e sete combinações (para três e quatro elementos), para o quarteto de Beethoven uma centena de combinações (para três, quatro, cinco, seis e oito elementos) e para a peça de Stravinsky oitenta e três combinações (para três, quatro, cinco e seis elementos), vê-se que a representação em notação analítico-musical destes conjuntos seria impraticável. Desta forma, em cada caso, foi feita uma seleção aproveitando os resultados mais relevantes. Cada motivo é indicado com um colchete e uma letra grega (mais um algarismo, quando apropriado) acima da pauta.

A análise dos agrupamentos, com seus diversos níveis, é apresentada por meio de colchetes abaixo da pauta. À medida em que os níveis crescem existem cada vez menos grupos em seu interior, até que, no último nível, existe apenas um grupo, o qual representa a integralidade da peça sendo analisada.

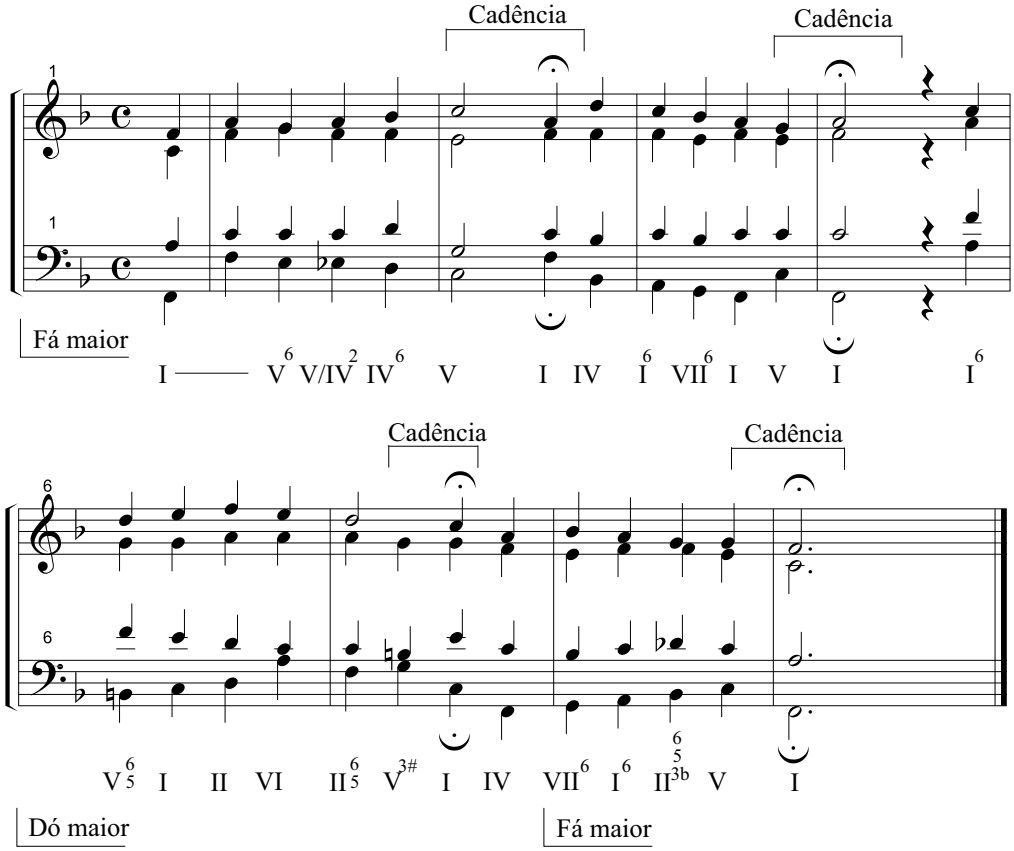

Figura 4.12: Estrutura harmônica de *Christus, der ist mein Leben* (BWV 95), de J.S. Bach.

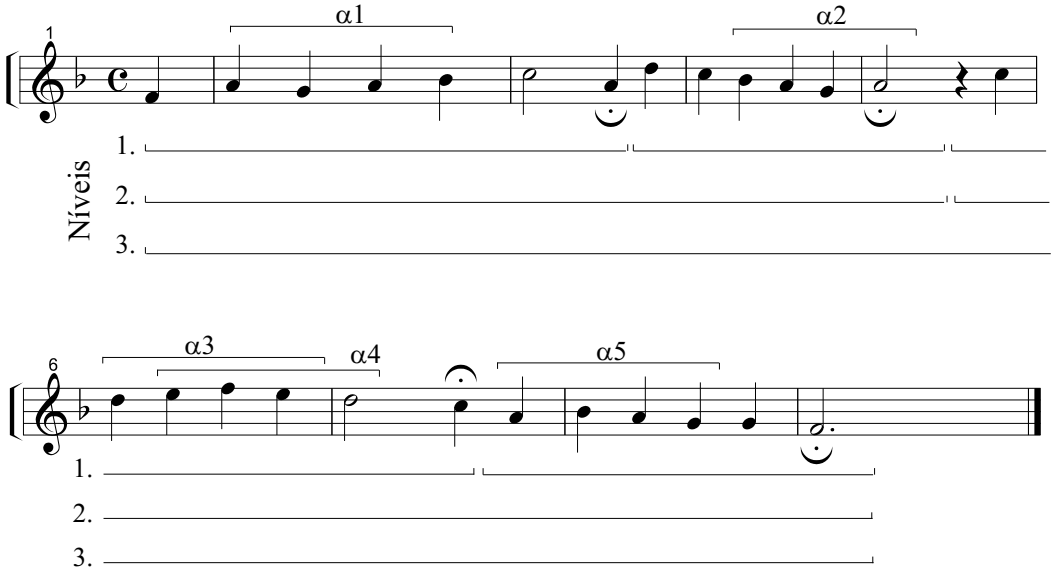

Figura 4.13: Estrutura de agrupamento de *Christus, der ist mein Leben* (BWV 95), de J.S. Bach.

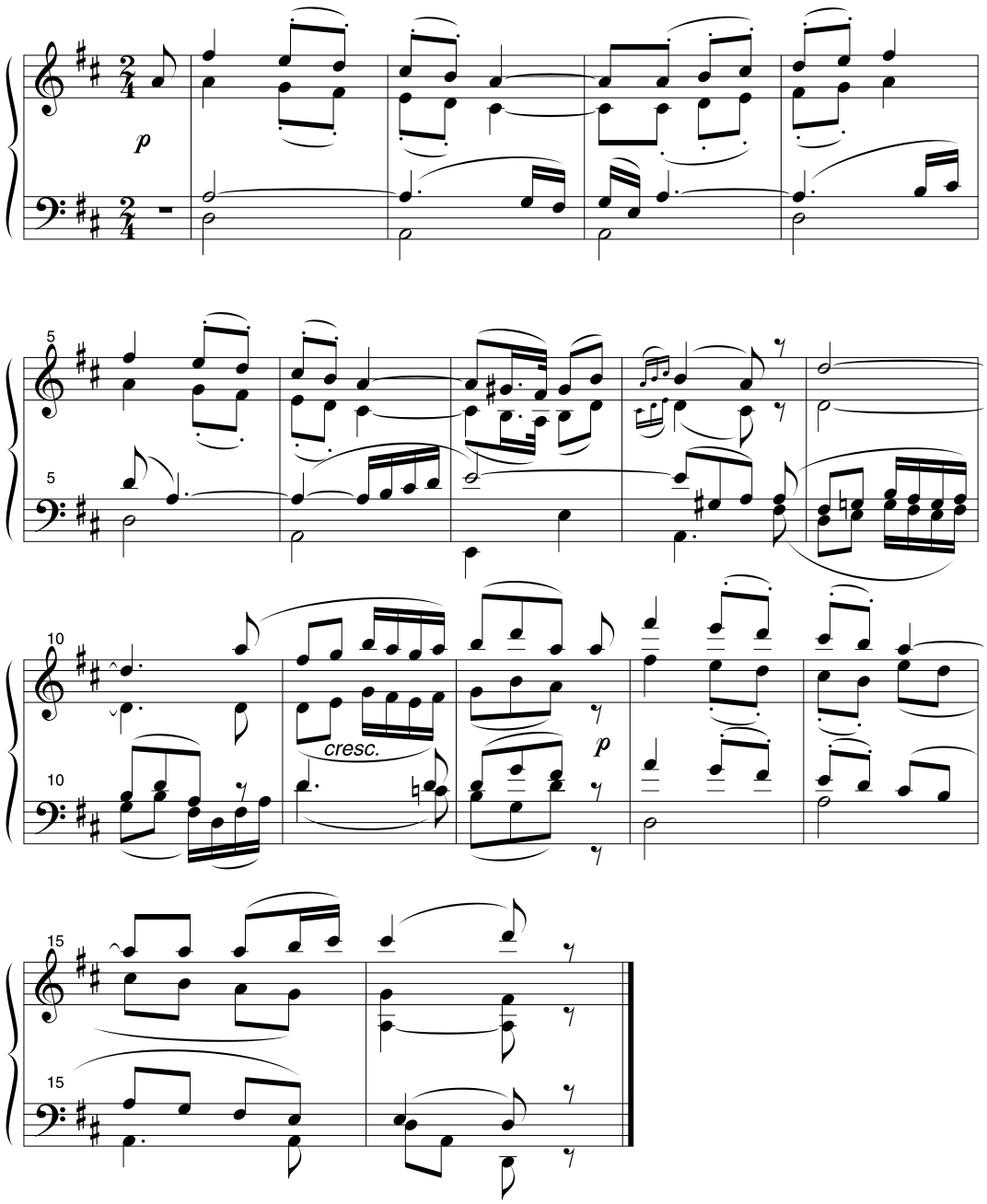

Figura 4.14: Texto original do tema 3º movimento do Quarteto de Cordas Op. 18 nº5, de L.van Beethoven.

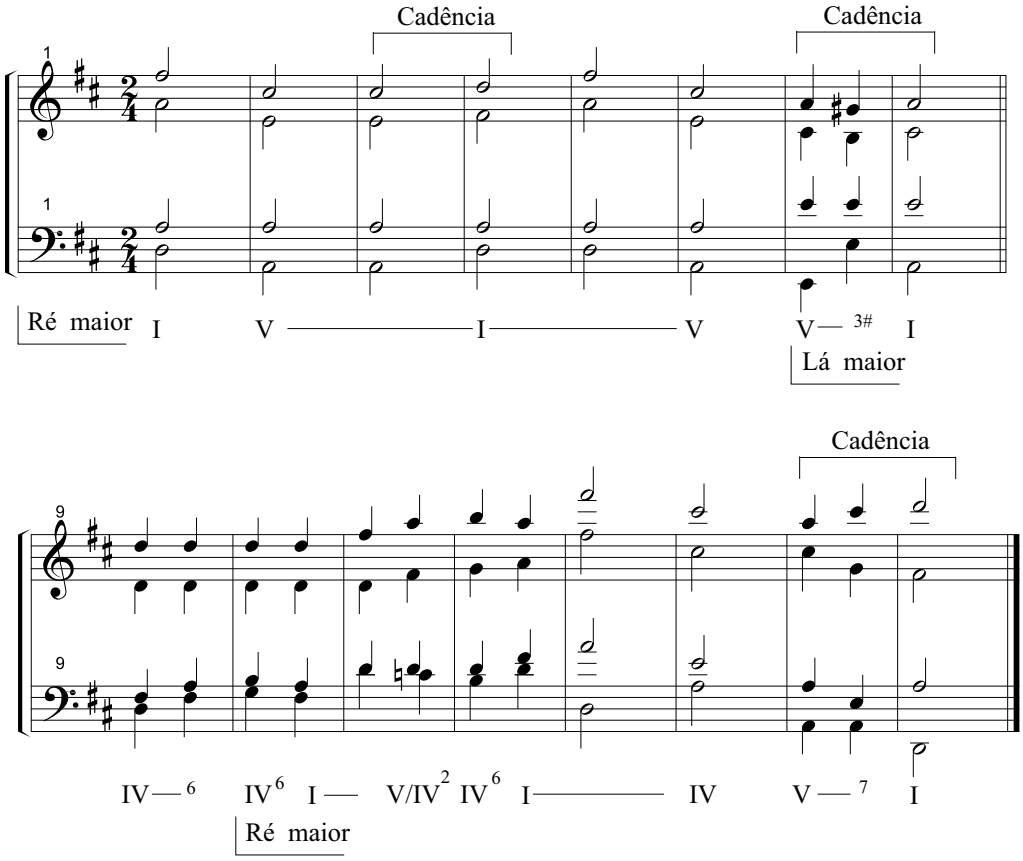

Figura 4.15: Estrutura harmônica do tema 3º movimento do Quarteto de Cordas Op. 18 nº5, de L.van Beethoven.

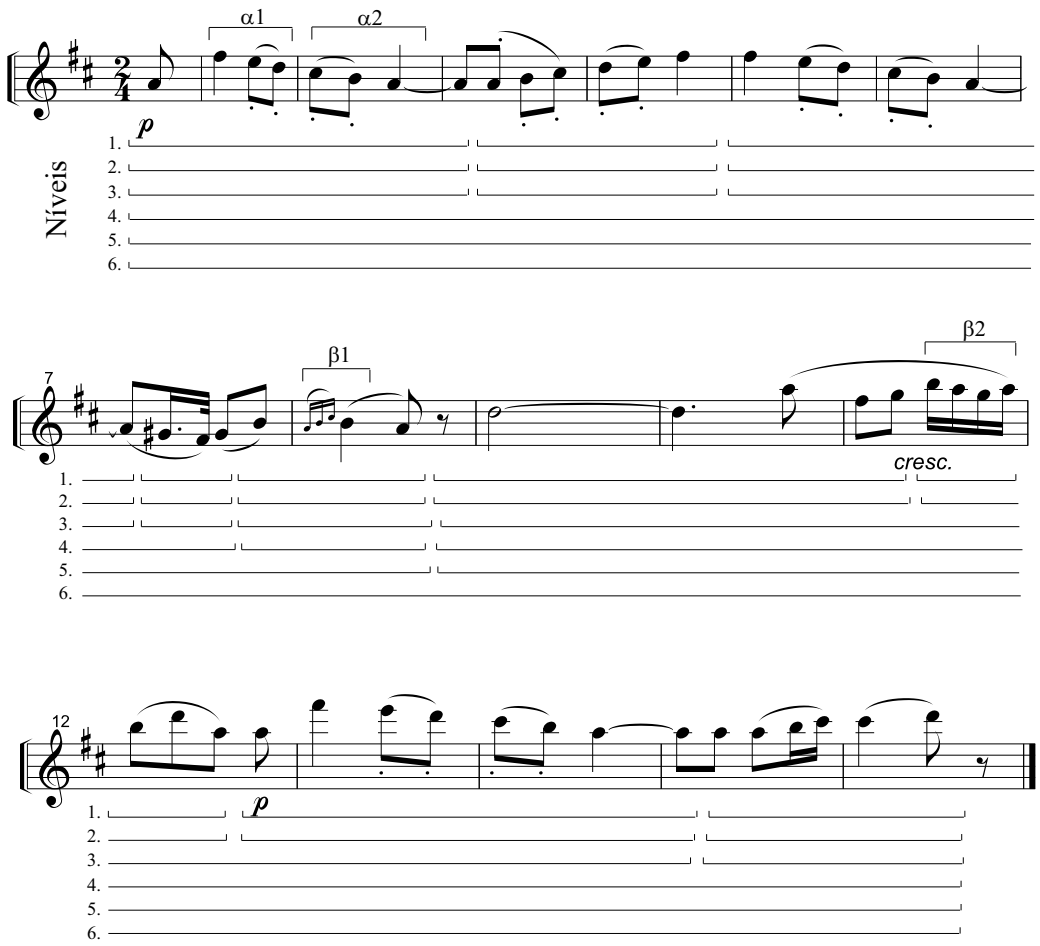

Figura 4.16: Estrutura de Agrupamento do tema 3º movimento do Quarteto de Cordas Op. 18 nº5, de L.van Beethoven.

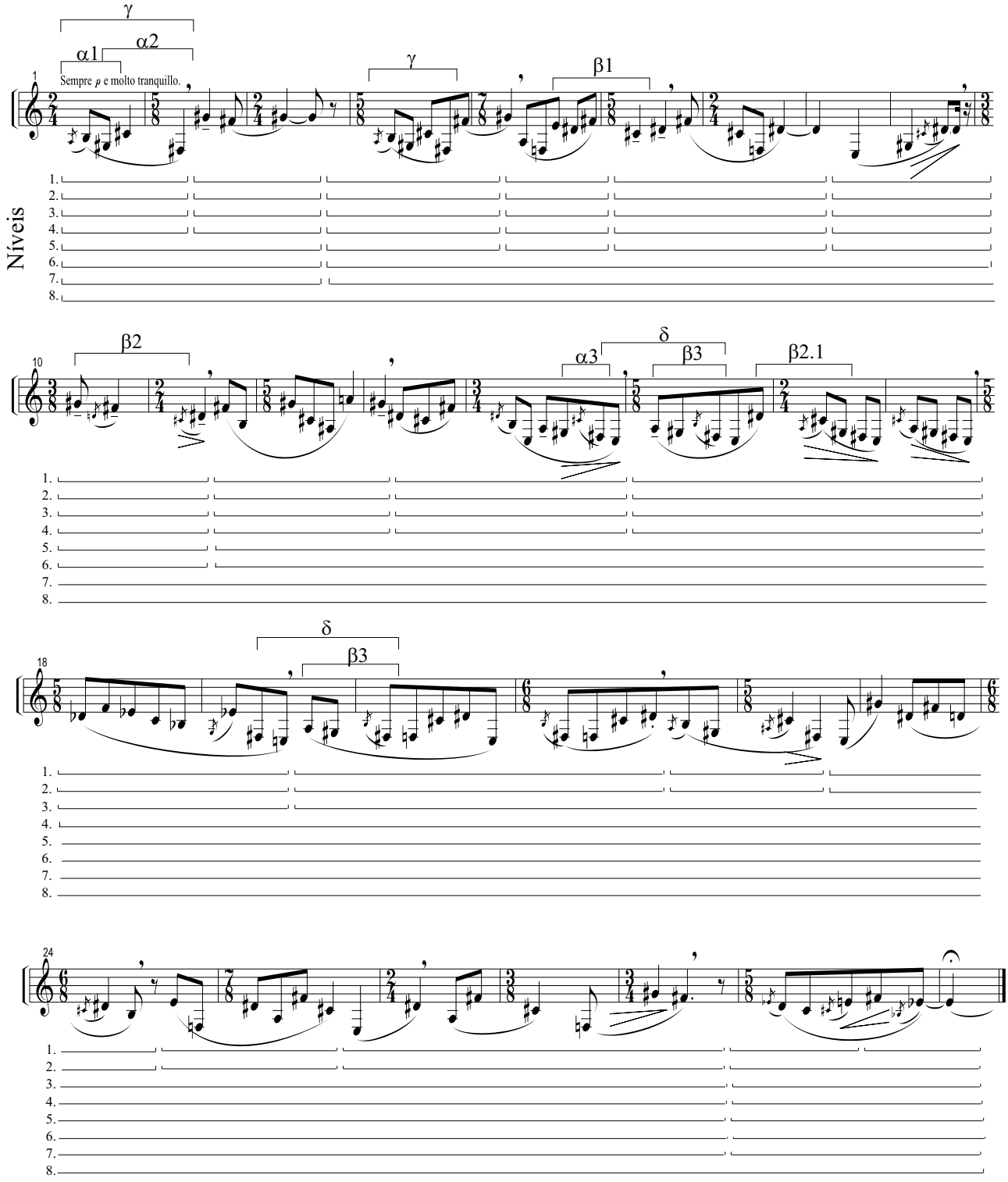

Figura 4.17: Análise motívica e estrutura de agrupamento de *Três Peças (para clarinete solo)* – I, de Igor de Stravinsky.

4.8 Discussão dos Resultados

4.8.1 Testes de Validação

Sistema Nebuloso

Com o objetivo de testar a funcionalidade do sistema nebuloso foi construído o trecho musical apresentado na figura 4.4 (página 37), contendo a segmentação de uma amostra musical e os respectivos valores das cinco primeiras Regras Preferenciais de Agrupamento. Quanto à estrutura de agrupamento resultante, o único aspecto questionável é, no primeiro nível, a divisão da primeira seção do tema apresentado em quatro partes ao invés de em três. Como pode ser visto na mesma figura, isto é causado pela forte influência da RPA 2a, a qual pontua fortemente a fronteira que separa o terceiro do quarto agrupamento. Na medida em que nem os sinais de expressão nem os elementos harmônicos são levados em conta, o resultado apresenta-se bastante razoável. Entretanto, caso se quisesse alterá-lo, o mais indicado seria atuar no parâmetro *limiar de vizinhança*, o qual controla o comportamento da RPA 4.

Resultados do Identificador de Padrões Não-Triviais

Na discussão sobre os resultados do identificador serão desprezados padrões menores do que aqueles formados por três alturas ou dois intervalos por julgar-se que não são suficientemente pertinentes para serem caracterizados como motivos temáticos. A presença, nos resultados, de padrões menores do que os indicados deve-se unicamente ao fato do programa ser um protótipo e ainda estar sob testes. Na medida em que foram testados seis exemplos melódicos diferentes, os comentários também serão realizados separadamente e na mesma ordem.

Exemplo 1 No exemplo 1, tirado de Hsu et al. (2001), pode-se verificar que, na parte da análise baseada em padrões de alturas, o algoritmo funcionou corretamente, encontrando o padrão principal. Na parte baseada em padrões intervalares, é interessante observar que o algoritmo encontrou uma inversão do padrão principal a partir do terceiro elemento, assim como os retrógrados de ambos, o original e a sua inversão. Mesmo neste exemplo simples é possível ver como a busca intervalar é superior e mais reveladora do que aquela simplesmente baseada em alturas.

Exemplo 2 O exemplo 2 ilusta um caso no qual existe total coincidência entre a busca por alturas e a busca intervalar. Em ambos os casos o único padrão não-trivial encontrado é aquele formado pelas alturas `<dó,lá,fá>`.

Exemplo 3 O exemplo 3 é um trecho baseado em três motivos melódicos, chamados aqui α, β e γ. Pode-se verificar que, na busca por padrões de alturas, somente são detectados α e γ, já que β é uma *substring* de α. Entretanto, na busca intervalar, β

aparece como tendo uma freqüência igual a 3, pois também aparece transposto a partir do sétimo elemento, não sendo neste caso uma *substring* de α.

Exemplo 4 De modo complementar ao exemplo anterior, o exemplo 4 mostra como a busca por alturas pode chegar a um resultado vazio enquanto a busca intervalar pode detectar padrões não-triviais. Isto ocorre porque a essa pode encontrar padrões através de transposições dos originais enquanto que aquela não o pode. Realmente, é possível se ver neste exemplo que ele é composto de dois padrões cuja segunda aparição é uma transposição (3 semitons acima) da primeira a aparição. Os padrões em questão são os padrões α e γ do exemplo 2.

Exemplo 5 O exemplo 5 é algo mais complexo. Como no caso anterior, a busca por alturas não revela nada da estrutura do trecho musical. A busca intervalar, ao contrário, mostra como existem dois retrógrados transpostos do padrão α[7]. Por outro lado, a aparição de γ e de de sua inversão γ_1 não são detectados. Isto se deve ao fato de nenhum dos dois aparecerem mais do que uma vez. Tal fato vem enfatizar a necessidade da identificador de padrões não-triviais, fazer parte de uma ferramenta que tenha uma maior capacidade de generalização (ver subseção 4.5.1, página 23), ao invés de ser empregado isoladamente.

Exemplo 6 O exemplo 6 é o mais complexo de todos. A busca por alturas revelou-se, neste caso, como totalmente incapaz (ainda que tenha detectado uma repetição de β) de decifrar a intrincada relação entre os padrões α, β e γ. Mais uma vez, a busca intervalar foi capaz de revelar integralmente a estrutura motívica do trecho detectar.

4.8.2 Análises

Visando simplificar esta discussão, será observada a mesma ordem da seção 4.7.2 no comentário dos três casos. Inicialmente, será abordada a análise harmônica da amostra (naturalmente, somente para os dois primeiros casos). Em seguida, será comentada a análise motívica. Finalmente, a análise da estrutura de agrupamento resultante, a qual é o objetivo último do componente.

Christus, der ist mein Leben (BWV 95), de J.S. Bach

O coral de Bach *Christus, der ist mein Leben* é a amostra mais simples do grupo de três, e foi incluído devido à sua clareza em estrutura, tanto do ponto de vista harmônico quanto melódico. A peça original pode ser vista na figura 4.11, página 42.

Em sua análise harmônica, mostrada na figura 4.12, página 43, pode-se observar, ao lado da classificação e análise dos acordes, a indicação das cadências, o que é decisivo na cooperação com as outras regras do componente na busca pelas corretas fronteiras entre

[7]Apesar de que α é que aparece como sendo uma retrogradação dos dois! Isto ocorre porque existe uma repetição dos retrógrados enquanto α aparece uma única vez.

os grupos (a análise harmônica é feita principalmente em função de RPA 7 em sua busca por cadências). Ao lado destas considerações é importante dizer que as modulações, uma dominante secundária (quarto acorde) e um acorde alterado (penúltimo acorde), foram corretamente detectados pelo sistema.

No que diz respeito à análise motívica, um importante conjunto de quatro alturas é relevante. Na figura 4.13 (página 43) ele é indicado como motivo α e apresenta-se em cinco formas diferentes. Considerando-se $\alpha 1$ como a forma original, $\alpha 2$ é o retrógrado, $\alpha 3$ é o retrógrado da inversão transposto, $\alpha 4$ é a inversão transposta e $\alpha 5$ é a inversão. Assim, cada frase do coral (ou seja, cada grupo do primeiro nível) tem, no mínimo, uma forma diferente do motivo α. Também é importante observar que o motivo original e seus variantes nunca quebram quaisquer fronteiras de grupos, apesar de eles não formarem "partes paralelas de agrupamentos" (ver RPA 6 na seção 4.3, página 22). Mesmo esta característica pode ser interessante, na medida em que ela pode denotar uma grande flexibilidade métrica devido ao motivo ter início em pontos diferentes do compasso.

Na análise de agrupamento existe somente três níveis de grupos (figura 4.13, página 43). É importante observar que todos os três níveis de agrupamento coincidem com os pontos cadenciais e que o segundo nível indica a segunda cadência como de mais importância do que a primeira e a terceira. O terceiro nível, como esperado, considera o coral inteiro como um grupo, ou seja, como uma única e grande cadência.

Tema do 3º Movimento do Quarteto de Cordas Op. 18 nº 5, de L.van Beethoven)

Na figura 4.14, página 44, é mostrado o tema do 3º movimento do Quarteto de Cordas Op. 18 nº5, de Beethoven. Esta é uma peça com uma estrutura harmônica muito mais simples do que a precedente, porém com um tratamento motívico e uma segmentação em grupos muito mais intrincados.

A análise harmônica divide a amostra em duas partes, sendo que a segunda, por sua vez, pode ser subdividida em duas. A primeira parte oscila entre as duas funções de tônica e dominante e termina com uma modulação à região da dominante. A segunda parte retorna à região principal por meio de um acorde de subdominante, atravessa a região da subdominante (por meio de uma dominante secundária, compasso 10, primeiro tempo) e tem uma cadência final na região principal. A forma em três partes é assim reforçada pela seqüência harmônica. Existe uma parte onde são expostos os acordes e regiões; em seguida, uma outra onde os elementos anteriores são elaborados e, finalmente, uma terceira parte que funciona como uma recapitulação. Para a estrutura de agrupamento, entretanto, o mais relevante é a localização das cadências.

Sob o ponto de vista do tratamento motívico, o tema em sua integridade é composto pelo único motivo α, mostrado da figura 4.16 (página 46). Também está indicada uma variante rítmica $\alpha 2$ do original $\alpha 1$. Mesmo onde parecem existir variantes independentes, tais como $\beta 1$ e $\beta 2$ nos compassos 8 e 11, respectivamente, é possível ver que, caso se remova os tons ornamentais, o motivo α surge novamente. Assim, no compasso 11, por exemplo, se forem removidos si e sol do segundo tempo obtém-se fá♯–sol–lá, ou seja, α.

A estrutura de agrupamento do tema apresenta seis diferentes níveis. Não obstante,

ao invés de apresentar separadamente cada nível, parece ser melhor, neste caso, optar por destacar certos grupos importantes e característicos e procurar compreender sua importância na estrutura. Nesta situação encontram-se quatro grupos. Os candidatos iniciais são os dois grupos cujo primeiro começa na metade do primeiro tempo do compasso 7 e o segundo na metade do segundo tempo do mesmo compasso. Estes dois grupos quebram a continuidade entre a primeira cadência e a próxima, na região da dominante. O primeiro dos dois persiste por três níveis e outro por quatro níveis. Parece que sua ocorrência na análise é motivada por uma tendência do sistema em enfatizar a modulação à dominante. Quaisquer que sejam os motivos, esta tendência é suficientemente poderosa para forçar estes agrupamentos até o antepenúltimo nível.

Os outros dois agrupamentos estão no segundo tempo do compasso 11 e no primeiro tempo do compasso 12. De modo semelhante aos dois primeiros, parece que tais agrupamentos foram selecionados pelo sistema em função do tratamento motívico. Entretanto, diversamente dos anteriores, eles persistem somente por dois níveis, o que indica que são hierarquicamente menos relevantes do que os outros dois.

Os demais agrupamentos da análise são muito mais triviais, indicando cadências e enfatizando o tratamento motívico e a estrutura sub-harmônica. Entretanto, é importante observar o nível cinco (penúltimo nível), onde a única fronteira localiza-se no fim da cadência à dominante, o que reforça a perspectiva de considerar esta modulação como o ponto estrutural central da peça.

Primeira das *Três Peças para Clarinete Solo*, de I. Stravinsky

A peça de Stravinsky é a única amostra das três que possui uma sintaxe não-tonal, tanto do ponto de vista harmônico quanto melódico. Além disto, a peça é monofônica, o que torna impossível a detecção de uma estrutura harmônica de suporte (pelo menos no sentido clássico da expressão). Essas assertivas possuem duas implicações ligadas entre si. A primeira diz que a aplicação de técnicas baseadas em harmonia tradicional é impossível (aplicação de RPA 7). Conseqüentemente, e esta é a segunda implicação, nenhuma cadência no discurso será detectada. Na medida em que cadências são necessárias para filtrar (por importância) motivos de alturas selecionados pela RPA 6, este último processo também é comprometido. Assim, em outras palavras, todo o trabalho deve ser feito pelas RBFAs e pelas RPA 1 a 5. Entretanto, a RPA 6 é empregada para detectar motivos relevantes, suas formas e variações. A peça e sua análise é mostrada na figura 4.17, página 47.

Da profusão de motivos e variantes encontrados pelo sistema, são ilustrados alguns com três (motivos α), quatro (motivos β), cinco (motivos γ) e seis (motivos δ) alturas. O real gerador deste conjunto é o motivo γ e, desta forma, todos os outros são a ele subsidiários. Em alguns casos existe uma repetição literal (como γ no compasso 1 e no compasso 4 e como $\alpha 1$ no compasso 1 e no compasso 21) ou uma transposição literal (como aquela entre $\beta 2$ no compasso 10 e $\beta 2.1$ nos compassos 15 e 16). Em outros casos existe alguma espécie de variação (como aquela entre $\alpha 2$ no compasso 1 e $\alpha 3$ no compasso 14). Em quaisquer dos casos é possível relacioná-los com o motivo γ. Igualmente digno de

nota é o motivo δ, porque, a despeito de ocorrer entre a fronteira entre dois agrupamentos, ele está localizado em pontos paralelos no interior dos agrupamentos cuja fronteira foi por ele quebrada.

A discussão acerca da análise da estrutura de agrupamento está dividida em duas partes. A primeira faz uma comparação entre os resultados encontrados e aqueles mostrados na seção 3.6 de Lerdahl & Jackendoff (1996). Isto é realizado para os nove compassos iniciais (dimensão da análise proporcionada por Lerdahl & Jackendoff). A segunda parte é composta de comentários sobre o restante da análise, mais um comentário geral sobre forma e agrupamento.

Ao comparar a análise proposta por Lerdahl & Jackendoff (1996) com aquela realizada pelo sistema proposto surgem três pontos de divergência.

O primeiro está na passagem entre compassos 4 e 5. Na primeira análise existe uma fronteira entre os dois fá♯. Para sustentar esta fronteira os autores sustentam argumentos baseados na forte transição (salto de oitava) e paralelismo motívico. Apesar disto, o ponto de articulação real é após o sol♯. Considerando como válida esta assertiva, o sistema proposto marcou corretamente a fronteira no compasso 5, ou seja, após o sol♯.

A segunda divergência acontece nos compassos 5–6. A primeira análise marca corretamente a fronteira do agrupamento no ponto de articulação no compasso 6. Entretanto, existe uma maior concentração de regras (Lerdahl & Jackendoff, 1996, figura 3.34) antes do dó♯ (três regras) do que antes do ré♯ (duas regras). Seguindo esta tendência, o sistema marcou uma fronteira de agrupamento após o fá♯ (última nota do compasso 5).

A última divergência diz respeito aos agrupamentos de níveis maiores (*Larger-Level Grouping*). Levando em conta que a RPA 5, ao lado das simetrias, foi implementada observando a importância de cada fronteira através de seu valor, os agrupamentos de níveis mais altos não coincidem completamente com aqueles da primeira análise (a qual emprega somente simetria em sua tarefa). Entretanto, mesmo nesta as ambigüidades são muitas e os autores propõem três possibilidades. Dentre elas, a terceira é a que se apresenta como mais próxima à análise do sistema proposto neste trabalho.

Ao lado das divergências mostradas antes, outras duas importantes diferenças entre a análise original e a realizada neste trabalho são o número de níveis de agrupamento (oito neste trabalho e quatro no outro) e o tipo de troca de nível do terceiro agrupamento em diante (repentino neste trabalho e gradual na outra análise).

Enquanto que o resultado distinto para os agrupamentos pode ser explicado pele emprego de uma abordagem diferente e conseqüente implementação da RPA 5, o maior número de níveis encontrado (o dobro) neste trabalho pode ser justificado através do tamanho da amostra considerada, a saber, a peça completa no caso deste trabalho e nove compassos na outra análise.

Finalmente, são necessárias algumas palavras acerca da análise de agrupamento proposta. Em primeiro lugar, é importante notar a forte persistência dos agrupamentos inicial (compassos 1 a 3) e final (compassos 29 e 30) no que diz respeito aos diferentes níveis e também observar que eles somente desaparecem no penúltimo nível. Isto pode sugerir a forma da peça, isto é, uma forma bipartida com uma *coda* (exposição, elaboração e coda). Em seguida, os agrupamentos das seções centrais apresentam-se como razoavelmente cor-

retos com a única exceção do agrupamento começando coma altura mi, no compasso 25 e terminando com fá♯ no compasso 28. Uma delimitação correta das fronteiras seria considerar tanto o mi quanto o ré♯ que lhe segue como pertencentes ao agrupamento anterior. Talvez o agrupamento correto tenha sido preterido porque, como no caso indicado antes, a forte densidade das regras na fronteira entre dó♯ e o mi no compasso 25 deve foi ainda maior do que aquela existente entre ré♯ e lá no compasso 26.

Capítulo 5

Estrutura Métrica

5.1 Introdução

Este capítulo examina o componente que trata das questões métricas tais como aparecem na TGMT (Lerdahl & Jackendoff, 1996). Com este objetivo, é apresentado um resumo de seu suporte teórico, dos quais constam o seu Sistema Analítico e sua Gramática Formal[1]. Em seguida, são expostos seu Objetivo Específico, a Metodologia empregada e sua Implementação. Finalmente, são apresentados os Resultados obtidos através de Testes de Validação e Análises de obras musicais reais.

5.2 Sistema Analítico

A Estrutura Métrica, dentro da TGMT, é definida como sendo um padrão hierárquico e regular de pulsos aos quais o ouvinte relaciona os eventos musicais. Para uma correta compreensão da Estrutura Métrica é necessário definir-se o conceito de acento.

Na TGMT existem três tipos de acentos:

1. O *acento fenomenal* é resultante de qualquer evento na superfície musical que enfatiza ou coloca em relevo um momento no fluxo musical. Exemplos de acentos fenomenais são instantes de ataques de alturas, ênfases locais (tais como *sforzandi*), mudanças bruscas de dinâmica ou de timbre, notas longas, grandes saltos, mudanças harmônicas e outros fenômenos de características similares;

2. O *acento estrutural* é resultante dos pontos de gravidade melódicos/harmônicos dentro de uma frase ou seção, especialmente a cadência;

3. O *acento métrico* é resultante de qualquer pulso que é relativamente forte em seu contexto métrico.

[1]Como antes, para uma abordagem mais profunda do assunto, consultar o capítulo 2 de Carvalho (2001) ou, preferencialmente, o texto original de Lerdahl & Jackendoff (1996).

Para a formação da Estrutura Métrica contribuem os três tipos de acento, mas não da mesma forma ou na mesma proporção. Das regularidades presentes no acento fenomenal, por exemplo, o ouvinte pode extrapolar os padrões regulares que formam o acento métrico. A partir do momento em que ouvinte reconhece tais padrões, qualquer mudança na regularidade dos acentos fenomenais produz ambigüidades e fenômenos de caráter contraditório, como a síncope. Caso tal conflito persista, é estabelecido, pela percepção do ouvinte, um novo acento métrico.

Não obstante, para que a Estrutura Métrica possa ser plenamente compreendida, é necessário ainda definir, além dos diferentes tipos de acento, os conceitos de *pulso* e de *hierarquia métrica*.

O pulso é o elemento básico dos padrões métricos. Um pulso não tem duração e é separado de outros pulsos por *intervalo-temporal*[2]. Utilizando novamente a analogia visual, os pulsos podem ser comparados aos pontos igualmente distanciados no espaço geométrico e os *intervalos-temporais* que os separam, às respectivas distâncias. Disto pode-se concluir que uma Estrutura Métrica é inerentemente periódica.

Uma representação gráfica adequada para uma seqüência de pulsos pode ser vista na figura 5.1.

Figura 5.1: Seqüências de pulsos.

É possível notar que, apesar de ambas as estruturas serem possíveis, somente a primeira (figura 5.1a) pode ser chamada de métrica, pois nela existe eqüidistância entre os pulsos, do ponto de vista espacial, e periodicidade, do ponto de vista temporal.

Isto pode ser melhor compreendido se visualizado na figura 5.2. Nesta estrutura de

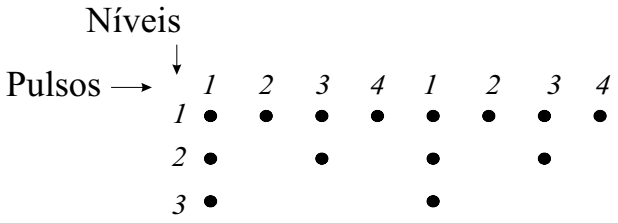

Figura 5.2: Relação entre pulsos.

três níveis, é possível observar como todos os pulsos fortes do nível 1 (considerando este

[2]Neste trabalho, *intervalo-temporal* foi empregado como tradução para expressão inglesa *time-span*.

como o de menor dimensão) são também pulsos no nível 2 e os pulsos fortes deste são também pulsos no nível 3.

Além disso, para que um padrão de pulsos seja considerado métrico é necessário que haja uma alternância periódica de pulsos fortes e pulsos fracos. Para que esta exista, por sua vez, é preciso que seja estabelecida uma *hierarquia métrica* composta de níveis diferenciados de pulsos. A relação entre os pulsos e os níveis da hierarquia pode ser estabelecida desta forma: um pulso é considerado "forte" num determinado nível se no nível de maior dimensão ele é um pulso. Estabelece-se, assim, uma "grade métrica" multidimensional na qual a periodicidade dos pulsos é reforçada de nível a nível. Um exemplo de uma grade métrica pode ser visto na figura 5.3.

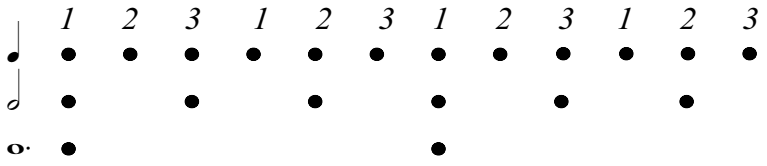

Figura 5.3: Grade métrica.

Dos cinco ou seis níveis métricos existentes numa peça, os níveis intermediários são os que, normalmente, mais facilmente percebemos. A fórmula de compasso da peça é um destes níveis intermediários, sendo eqüidistante tanto do menor quanto do maior nível. O nível intermediário mais facilmente perceptível leva o nome de *tactus*, termo de origem renascentista e adaptado à presente teoria.

É no que diz respeito à interação entre agrupamento e metro que os elementos básicos destas duas estruturas são diferentes: as estruturas de agrupamento consistem de *unidades* organizadas hierarquicamente enquanto que as estruturas métricas consistem de *pulsos* organizados hierarquicamente. Disto decorre que grupos não possuem acentos e pulsos não possuem um agrupamento inerente.

A interação entre a Estrutura de Agrupamento e Estrutura Métrica, é realizada através do grau de coincidência de inícios de grupos (em qualquer nível de agrupamento) com "pulsos fortes" (em qualquer nível da grade métrica). Quando a coincidência é total, diz-se que as Estruturas de Agrupamento e Métrica estão "em fase". Quando existe um determinado deslocamento entre ambas diz-se que estão "fora de fase" por um determinado número de intervalos-temporais de um determinado nível.

5.3 Gramática Formal

Da mesma forma que para a Estrutura de Agrupamento, é definida para a Estrutura Métrica uma gramática formal baseada em *Regras de Boa–Formatividade Métrica* (RBFM), as quais definem estruturas métricas possíveis, e *Regras Preferenciais de Métrica* (RPM),

as quais modelam os critérios do ouvinte experimentado com o objetivo de escolher a Estrutura Métrica mais estável para uma determinada superfície musical. Os dois conjuntos de regras são descritos abaixo.

RBFM 1 Todo ponto de ataque deve ser associado com um pulso no menor nível métrico presente naquele ponto da peça.

RBFM 2 Todo pulso em um dado nível deve também ser um pulso em todos os níveis menores presentes naquele ponto da peça.

RBFM 3 Em cada nível métrico, pulsos fortes ocorrem de dois em dois ou de três em três.

RBFM 4 O tactus e os níveis métricos imediatamente maiores do que ele devem consistir de pulsos igualmente espaçados durante toda a extensão da peça. Em níveis métricos abaixo do tactus os pulsos fracos devem ser igualmente espaçados entre os pulsos fortes que os circundam.

A primeira Regra Preferencial da Estrutura Métrica diz respeito ao paralelismo, o que traz uma inevitável conexão com a Estrutura de Agrupamento.

RPM 1 (Paralelismo) Onde dois ou mais grupos ou partes de grupos podem ser construídos em paralelo, eles preferivelmente recebem Estrutura Métrica paralela.

A segunda Regra Preferencial também estabelece uma conexão com a estrutura de agrupamento, só que desta vez indicando o local dentro de um grupo onde ocorre o pulso mais forte.

RPM 2 (Pulso Forte Antes) Prefira, não enfaticamente, uma Estrutura Métrica na qual o pulso mais forte de um grupo apareça relativamente cedo no grupo.

As RPM 3 e 4 dizem respeito, respectivamente, ao momento de surgimento dos eventos na superfície musical e à ênfase individual de cada um deles na mesma.

RPM 3 (Evento) Prefira uma Estrutura Métrica na qual pulsos que coincidam com o surgimento de eventos-altura sejam pulsos fortes.

RPM 4 (Ênfase) Prefira uma Estrutura Métrica na qual pulsos que são enfatizados sejam pulsos fortes.

A RPM 5 trata das durações de eventos na superfície musical.

RPM 5 (Comprimento) Prefira uma Estrutura Métrica na qual um pulso relativamente forte ocorra no surgimento de:

a. um evento-altura relativamente longo,

b. uma duração relativamente longa de uma dinâmica,

c. uma ligadura de expressão relativamente longa,

d. um padrão de articulação temporal relativamente longo,

e. uma duração relativamente longa de uma altura nos níveis relevantes da Redução Temporal, ou

f. uma duração relativamente longa de uma harmonia nos níveis relevantes da Redução Temporal (ritmo harmônico).

Além das Regras Preferenciais expostas acima, existem outras quatro que tratam de problemas específicos da música tonal. A primeira delas trata da importância do baixo para a percepção do metro.

RPM 6 (Baixo) Prefira um baixo metricamente estável.

A Regra Preferencial seguinte é baseada no fato de que *"cadências são um fator importante para fixação tanto do metro quanto da estrutura tonal"* (pag. 88).

RPM 7 (Cadências) Dê forte preferência a uma Estrutura Métrica na qual cadências são metricamente estáveis; isto é, evite fortemente, dentro de cadências, violações de regras de prefêrencias locais.

Um fator de grande importância para a escritura da música tonal, notadamente a música de caráter contrapontístico, é a utilização de suspensões (retardos). Como trata-se de um recurso cujo uso é extremamente normalizado estilisticamente, é necessário a formulação de uma Regra Preferencial para sua análise métrica.

RPM 8 (Suspensão) Dê forte preferência a uma Estrutura Métrica na qual uma suspensão está num pulso mais forte do que sua resolução.

Finalmente, para que as três regras anteriores sejam operacionais, torna-se necessária uma regra que permita a interação da Estrutura Métrica com o componente que viabiliza a análise dos eventos-altura em função do tempo, a saber, a Redução Temporal.

RPM 9 (Interação Temporal) Prefira uma análise métrica que minimize o conflito na Redução Temporal.

Finalmente, a regra dez tenta garantir a periodicidade do metro.

RPM 10 (Regularidade Binária) Prefira estruturas métricas nas quais exista uma alternância de pulsos fortes e fracos.

5.4 Objetivo Específico

O objetivo específico desta parte do trabalho é a criação e implementação de um método para detecção da Estrutura Métrica. Tal método, seguindo a orientação original da teoria sendo implementada, leva em conta os resultados obtidos por alguns dos outros componentes, alcançando, assim, uma integração dos diversos aspectos analíticos presentes.

5.5 Metodologia

Existem inúmeras abordagens referentes ao tratamento do metro e do ritmo partindo de um enfoque computacional. Tais abordagens partem de diferentes enfoques do fenômeno do ritmo e do metro afim de melhor sistematizá-lo. Tanto em Meredith (2004) quanto em Temperley (2004) é possível observar um conjunto das várias metodologias vigentes sobre o tratamento do metro musical. No que tange a este último trabalho, existe também a proposta de um método visando avaliar a eficiência de cada uma das várias metodologias expostas.

Assim, trabalhos como os de Large & Kolen (1999), Weyde (2001) e Snyder & Large (2002) tratam diretamente da questão da percepção do ritmo e do metro por um ser humano. Já o trabalho de Meudic (2001) almeja a modelagem de estruturas rítmicas visando sua utilização em análise musical.

Um importante trabalho presente neste grupo é aquele de Temperley & Sleator (1999), o qual emprega um conjunto de regras de boa–formatividade e preferenciais fortemente influenciado pela TGMT. Entretanto, duas diferenças relevantes ocorrem entre a sua abordagem e aquela do presente trabalho. Inicialmente, de modo diverso ao que acontece com este último, existe uma integração entre as regras de metro e de harmonia e não uma interação entre os dois conjuntos de regras (como acontece com a TGMT). Em segundo lugar, para a otimização dos conjunto de regras são empregadas soluções baseadas em programação dinâmica, soluções estas que tornar-se-iam impraticáveis caso se mantivesse os conjuntos de regras com as mesmas dimensões daquelas constantes da TGMT.

Enfoque diverso, voltado para uma abordagem típica de engenharia, existe nos trabalhos de Eck (2002), Port et al. (1998) e Orife (2001). Dois resumos de várias destas abordagens podem ser encontrados em Seppánnen (2001), o qual é uma coletânea de métodos de reconhecimento de metro, incluindo a detecção num arquivo MIDI ou diretamente num sinal de áudio, e em Meredith (2004), o qual é um tutorial em processos de formalização do metro musical.

No presente trabalho deverá ser seguido um método cujo objetivo será a otimização do conjunto de Regras Preferenciais da Estrutura Métrica. Como estas são em número de dez e apresentam um caráter muitas vezes conflitante, trata-se claramente de um problema carecendo de um método multi-objetivo para sua resolução. Visando este caminho, foi empregado neste componente da teoria um algoritmo genético multi-objetivo baseado, por sua vez, no conceito de *front* de Pareto, os quais serão explicados a seguir.

5.5.1 Problemas Multi-Objetivo

Problema Multi-Objetivo Geral

De acordo com Marco et al. (1999), um problema multi-objetivo consiste de um conjunto de funções objetivo que devem ser otimizadas (minimizadas ou maximizadas) simultaneamente e associadas com um conjunto de restrições representadas através de igualdades e desigualdades. Assim, um problema multi-objetivo pode ser formalizado como:

$$\text{Minimizar ou Maximizar } f_i(x) \quad i = 1, \ldots, N$$

$$\text{sujeito a : } \begin{cases} g_j(x) = 0 & j = 1, \ldots, M \\ h_k(x) \leq 0 & k = 1, \ldots, K \end{cases}$$

onde N é o número de funções objetivo f_i e x é um vetor de p elementos, os quais são as variáveis de decisão e, finalmente, g_j e h_k formam o conjunto de restrições.

O conjunto Pareto-ótimo

De modo diverso de um problema com apenas uma função objetivo, um problema multi-objetivo pode apresentar diversas soluções que são igualmente válidas. Este conjunto de soluções é chamado *conjunto de Pareto-ótimo*, ou *conjunto de soluções não-dominadas*, sendo que seu conceito foi formulado no século XIX por Vilfrido Pareto (Coello, 2000). O conceito de dominância de Pareto é ilustrado a seguir.

Inicialmente, cada parâmetro x de um problema multi-objetivo geral pode ser considerado como um vetor p-dimensional contendo p variáveis de decisão. Em um problema de minimização (o qual é o caso deste trabalho), um vetor $x^{(1)}$ é parcialmente menor do que um vetor $x^{(2)}$ ($x^{(1)} \succ x^{(2)}$), quando nenhum valor de $x^{(2)}$ é menor do que $x^{(1)}$ e, no mínimo, um valor de $x^{(2)}$ é estritamente maior do que $x^{(1)}$. Dito formalmente, considerando-se:

$$\mathbf{u} = \mathbf{f}(\mathbf{x_u}) = (u_1, \ldots, u_p)$$

e

$$\mathbf{v} = \mathbf{f}(\mathbf{x_v}) = (v_1, \ldots, v_p)$$

então $\mathbf{x_u} \succ \mathbf{x_v}$, ou seja, $\mathbf{x_u}$ *domina* $\mathbf{x_v}$, se e somente se

$$\forall i \in \{1, \ldots, p\}, v_i \geq u_i \text{ e } \exists i \in \{1, \ldots, p\}, v_i > u_i$$

Então, se $x^{(1)}$ parcialmente menor do que $x^{(2)}$, diz-se que $x^{(1)}$ *domina* $x^{(2)}$ ou $x^{(2)}$ é *inferior* a $x^{(1)}$. Considerando o espaço de decisão, quaisquer vetores que não são dominados por outros são denominados *não-dominados* ou *não-inferiores*.

Como exemplo, considere-se um problema multi-objetivo qualquer de minimização, com duas dimensões e cujo resultado obtido após sua resolução seja o conjunto de pontos:

$$\mathbf{S} = \begin{bmatrix} 1 & 0.5 & 2 & 3 & 1 & 2.5 \\ 4 & 3 & 2 & 1.5 & 1 & 0.5 \end{bmatrix}$$

no qual o número de linhas corresponde ao número de dimensões do problema e o número de colunas, por sua vez, corresponde ao número de soluções (pontos). Com o intuito de encontrar-se quais soluções fazem parte do conjunto de Pareto-ótimo pode ser definida uma *matriz de dominância* **D**, do tipo booleano, com dimensão $|S| \times |S|$ e preenchida de acordo com a equação abaixo:

$$D_{ij} = \begin{cases} 1 & \text{se } S_i \succ S_j \text{ e } i,j \in \{0,\ldots,|S|-1\} \\ 0 & \text{em caso contrário} \end{cases}$$

No caso deste exemplo, a matrix **D** adquire a forma:

$$\mathbf{D} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Nesta matriz, os índices dos pontos não dominados correspondem às colunas onde cada elemento é zero. Assim, neste exemplo, o conjunto de Pareto-ótimo é composto de três soluções, a saber, aquelas de números 1, 4 e 5 (considerando que a indexação dos pontos pertence ao intervalo $[0, |S| - 1]$). Graficamente, tanto o conjunto integral de soluções quanto o conjunto de Pareto-ótimo podem ser vistos na figura 5.4. Nela podem ser vistos os pontos que formam o conjunto total de soluções e aqueles que formam o conjunto de Pareto-ótimo (*front* de Pareto). Os pontos 0, 2 e 3 são dominados, respectivamente, pelos pontos 1, 4 e 5. O ponto 1, não é dominado pelos pontos 4 e 5 porque apresenta melhor solução do que ambos em f_1. O ponto 4 não é dominado pelo ponto 1 porque apresenta melhor solução do que este em em f_2 e nem pelo ponto 5 porque apresenta melhor solução do que este em f_1. O ponto 5 não é dominado pelos pontos 4 e 1 porque apresenta melhor solução do que ambos em f_2.

Pode ser notado também que tal tipo de matriz tende a possuir um alto grau de dispersão. Sendo assim, em sua implementação, faz-se necessário levar em conta o algoritmo para o tratamento de matrizes esparsas mostrado no apêndice A, página 181.

Métodos Clássicos de Resolução

A mais característica dificuldade na resolução de problemas multi-objetivo é aquela que diz respeito ao conflito entre as funções objetivo. Isto deve-se ao fato de cada uma delas, comumente, apresentar individualmente uma solução ótima diferente. O conjunto Pareto-ótimo, portanto, oferece uma espécie de conciliação ao proporcionar um conjunto de soluções que apresenta o menor conflito entre as funções objetivo. Os três métodos clássicos mais comuns para resolução de problemas multi-objetivo são o método de ponderação de funções objetivo, o método de distância funcional e o método min-max. Tais métodos apresentam como ponto comum a redução do conjunto das funções objetivo em uma única função. Em outras palavras, transformam um problema vetorial (vetor das

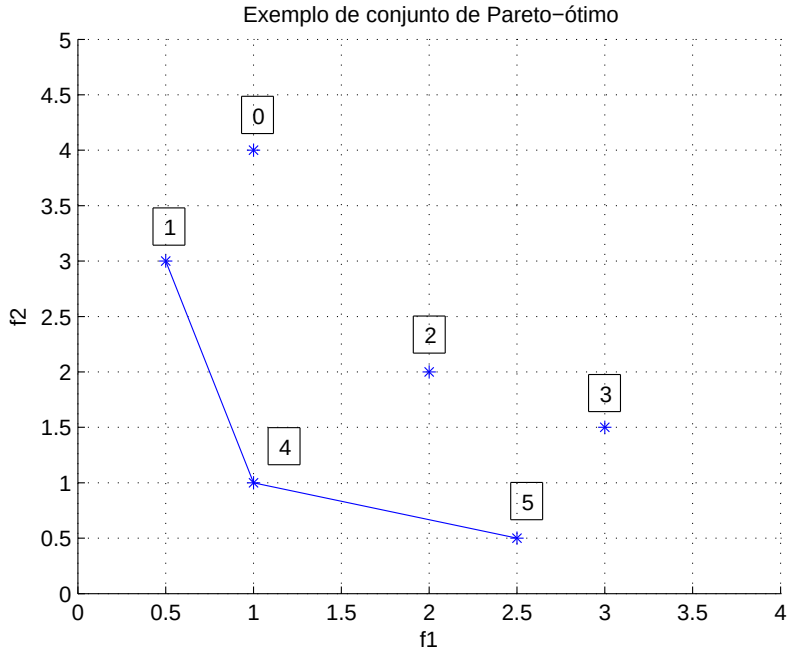

Figura 5.4: Conjunto de soluções e o *front* de Pareto.

funções objetivo) em escalar (com uma única função objetivo). Uma descrição detalhada de tais métodos pode ser encontrada em Srinivas & Deb (1994). A seguir, como ilustração, será descrito o método de ponderação de funções objetivo.

O método de ponderação de funções objetivo é um dos métodos mais comuns para a resolução de um problema multi-objetivo e opera atribuindo diferentes pesos a cada uma das funções objetivo com o intuito de poder-se manipular sua influência no resultado final. Trata-se de um método clássico, porque já em 1900 foi empregado por economistas ligados ao estudo da prosperidade social (Marco et al., 1999). No presente método, as funções objetivo são combinadas em uma única função $F(x)$, de modo que:

$$\begin{cases} F(x) = \sum_{i=1}^{N} \omega_i f_i(x) \\ \text{onde } X \text{ é o espaço de decisão e } x \in X \\ 0 \leq \omega_i \leq 1 \text{ e } \sum_{i=1}^{N} \omega_i = 1 \end{cases}$$

As principais vantagens deste método é que os pesos podem ser manipulados, permitindo, assim, uma comparação entre as funções objetivo e que, sendo a soma dos pesos igual a um, as soluções encontradas pertencerão ao conjunto de Pareto-ótimo. Isto pode ser facilmente verificado através do exemplo anterior. Fazendo os pesos de cada uma das duas funções objetivo variarem, respectivamente, nos intervalos $[1, 0]$ e $[0, 1]$ e considerando o índice do valor mínimo de $F(x)$, observa-se que este corresponde sempre a um ponto do *front* de Pareto. Isto pode ser verificado na lista abaixo, onde, em cada grupo de três

linhas, a primeira corresponde aos pesos de cada função objetivo (cuja soma é sempre um), a segunda aos valores de $F(x)$ e a terceira ao mínimo de $F(x)$ (com seu índice entre parêntesis).

```
w(0) = 1.0    w(1) = 0.0
F(x) = [1.0 0.5 2.0 3.0 1.0 2.5]
Valor mínimo de F(x) = 0.5 (Ponto 1)

w(0) = 0.9    w(1) = 0.1
F(x) = [1.3 0.8 2.0 2.9 1.0 2.3]
Valor mínimo de F(x) = 0.8 (Ponto 1)

w(0) = 0.8    w(1) = 0.2
F(x) = [1.6 1.0 2.0 2.7 1.0 2.1]
Valor mínimo de F(x) = 1.0 (Ponto 1)

w(0) = 0.7    w(1) = 0.3
F(x) = [1.9 1.2 2.0 2.5 1.0 1.9]
Valor mínimo de F(x) = 1.0 (Ponto 4)

w(0) = 0.6    w(1) = 0.4
F(x) = [2.2 1.5 2.0 2.4 1.0 1.7]
Valor mínimo de F(x) = 1.0 (Ponto 4)

w(0) = 0.5    w(1) = 0.5
F(x) = [2.5 1.8 2.0 2.2 1.0 1.5]
Valor mínimo de F(x) = 1.0 (Ponto 4)

w(0) = 0.4    w(1) = 0.6
F(x) = [2.8 2.0 2.0 2.1 1.0 1.3]
Valor mínimo de F(x) = 1.0 (Ponto 4)

w(0) = 0.3    w(1) = 0.7
F(x) = [3.1 2.2 2.0 1.9 1.0 1.1]
Valor mínimo de F(x) = 1.0 (Ponto 4)

w(0) = 0.2    w(1) = 0.8
F(x) = [3.4 2.5 2.0 1.8 1.0 0.9]
Valor mínimo de F(x) = 0.9 (Ponto 5)

w(0) = 0.1    w(1) = 0.9
F(x) = [3.7 2.8 2.0 1.6 1.0 0.7]
Valor mínimo de F(x) = 0.7 (Ponto 5)
```

```
w(0) = 0.0    w(1) = 1.0
F(x) = [4.0 3.0 2.0 1.5 1.0 0.5]
Valor mínimo de F(x) = 0.5 (Ponto 5)
```

Ainda com respeito ao método de ponderação de funções objetivo, suas principais desvantagens são, em primeiro lugar, que não se acham todas soluções e, além disso, caso se desconheça inteiramente o problema representado pelas funções objetivo, é possível dar-se valores totalmente ilógicos ao pesos de cada uma das mesmas.

Problema de otimização com duas funções objetivo

Como exemplo, é mostrado agora um problema de minimização com duas funções objetivo. Este problema foi originalmente apresentado por Schaffer (1984) em sua tese de doutorado. Trata-se da minimização da função f de uma variável e definida como:

$$f = (f_1(x), f_2(x))$$

sendo que $x \in [-6, 6]$ e $f_1(x) = x^2$ e $f_2(x) = (x-2)^2$.

Para que seja observado e compreendido o comportamento da função pode ser valiosa uma solução analítica. Esta pode ser realizada minimizando a função $f(x) = \lambda f_1(x) + (1-\lambda) f_2(x)$ e considerando $\lambda \in [0, 1]$ como um parâmetro. Desta forma, o *front* de Pareto (conjunto de Pareto-ótimo) pode ser representado parametricamente através da resolução deste problema (minimização da função $f(x)$):

$$\frac{df}{dx} = f'(x) = 2\lambda x + 2(1-\lambda)(x-2) = 2x - 4 + 4\lambda = 0$$

o que dá $x = 2 - 2\lambda$. Assim, $x \in [0, 2]$, $f_1(x) \in [0, 4]$ e $f_2(x) \in [0, 4]$, o que significa que o *front* de Pareto é constituído por todos os valores funcionais gerados por $x \in [0, 2]$. Fora deste intervalos os valores obtidos em ambas as funções não pertencem ao conjunto de Pareto-ótimo[3].

5.5.2 O Algoritmo Genético Multi-Objetivo

De modo diverso dos métodos clássicos citados anteriormente, os quais são capazes de gerar como solução um único ponto no interior do espaço de decisão, o emprego de algoritmos genéticos apresenta-se como uma alternativa importante na medida em que estes são capazes de trabalhar com uma população de vetores de variáveis de decisão, o que indica que podem entregar como resultado, após sua otimização (minimização ou maximização), um determinado número de soluções, todas elas pertencendo ao conjunto de Pareto-ótimo.

[3]Pode-se observar que este método de resolução equivale àquele de ponderação das funções objetivo, as quais são aqui as derivadas das funções originais.

Histórico

A primeira aparição na literatura da tentativa de resolução de um problema através de uma simulação genética foi por Rosenberg, em 1967 (Srinivas & Deb, 1994). Seu trabalho consiste na busca genética visando a simulação da genética e da química de uma população de organismos unicelulares possuindo múltiplas propriedades (funções objetivo). Entretanto, Rosenberg não chegou a implementar seu método.

Assim, o primeiro algoritmo idealizado e implementado surgiu com Schaffer (1984). Este algoritmo, intitulado *Vector Evaluated Genetic Algorithm* (VEGA), entretanto, apresenta como grave problema sua tendência a algumas soluções do conjunto Pareto-ótimo.

Com o objetivo de suplantar esta tendência de VEGA, foi sugerido por Goldberg (1989) um procedimento denominado *classificação por não-dominação* (*nondominated sorting*), o qual se baseia em dois pontos principais. Em primeiro lugar, o procedimento emprega um método de seleção por *ranking* que enfatiza bons pontos (soluções) e, além disso, usa um método de nicho com o objetivo de manter subpopulações estáveis de bons pontos. A partir da sugestão de Goldberg, dois importantes métodos foram idealizados e implementados, a saber, o *Multi Objective Genetic Algorithm* (MOGA) e o *Nondominated Sorting Genetic Algorithm* (NSGA).

Apresentado por Fonseca e Fleming em 1993 (Coello, 2000), MOGA possui uma implementação cujo procedimento de classificação é similar ao NSGA (apresentada detalhadamente mais adiante), porém com algumas diferenças que são importantes. A primeira delas é que a classificação opera-se em blocos (indivíduos com o mesmo *ranking*), o que, por ocasião da criação de uma nova geração (seleção), pode apresentar grande pressão de seleção o que, por sua vez, tem grande probabilidade de causar convergência prematura. Em seguida, ao contrário de, como NSGA, efetuar o compartilhamento nos valores dos parâmetros, MOGA o faz nos valores das funções objetivo, o que, ainda que mantendo a diversidade nos valores das funções, não necessariamente a mantém no conjunto dos parâmetros, o que pode ser importante para a escolha da solução definitiva. Finalmente, o algoritmo é insensível a problemas onde diferentes pontos do conjunto de Pareto-ótimo correspondem a valores idênticos de uma mesma função objetivo, o que indica que o algoritmo, nestes casos, pode ser incapaz de gerar múltiplas soluções.

Visando superar alguns dos problemas de MOGA citados acima, Srinivas & Deb (1994) desenvolveram NSGA. Este algoritmo, além das características já citadas acima, apresenta, também, a vantagem de evitar a convergência para um pequeno conjuntos de soluções (ou mesmo uma única solução), o que pode ser muito relevante quanto se trabalha com um número elevado de funções objetivo. Posteriormente, os mesmos autores (Deb et al., 2002) apresentaram o algoritmo NSGAII, o qual possui características de elitismo ausentes no primeiro.

Para um tutorial completo em algoritmos genéticos multi-objetivo os trabalhos de Zitzler et al. (2004), Sbalzarini et al. (2000) e Dias & Vasconcelos (2002) podem ser consultados. No que tange a elaboração de funções de teste existem os trabalhos de Lu & Yen (2003) e de Kalyanmoy (1998). Estudos de caso podem se encontrados em Zizler & Thiele (1999) e uma outra pespectiva sobre otimização multi-objetivo com algoritmos

genéticos em Zizler et al. (2002).

Na medida em que NSGA é o algoritmo no qual está baseada a implementação deste componente da teoria (o elitismo presente em NSGAII não justificaria em termos de resultados as novas dificuldades de implementação), ele será descrito detalhadamente na próxima seção.

Implementação do Algoritmo NSGA

Como foi dito antes, Srinivas & Deb (1994) idealizaram o NSGA visando a supressão de alguns dos problemas apresentados pelo MOGA. De modo diverso aos algoritmos genéticos mono-objetivo, nos quais o processo de seleção é efetuado através de um valor de aptidão (*fitness*) característico de cada possível solução, os algoritmos genéticos multi-objetivo empregam um processo no qual um conjunto de soluções recebem um mesmo valor, chamado *dummy fitness*. Este conjunto de soluções é o conjunto de Pareto-ótimo para uma dada população de pontos. Entretanto, na medida em que é necessário que toda a população receba um valor de aptidão, faz-se necessário um outro procedimento, denominado *ranking por fronts*. Neste procedimento, todos os pontos da população são previamente classificados por *fronts* sendo que a cada *front* corresponderá um valor de *dummy fitness*. Este pode ser descrito assim:

1. Considerar a população completa e achar os pontos não-dominados.
2. Atribuir a estes pontos um valor *dummy fitness*.
3. Retirar da população os pontos não-dominados encontrados anteriormente.
4. Se a população foi inteiramente classificada terminar.
5. Senão, voltar para o passo 1.

Nos casos de minimização os valores de *dummy fitness* são decrescentes e naqueles de maximização, crescentes. Uma possível forma para calcular estes valores pode ser:

$$f_i = \begin{cases} 1/r & \text{para minimização} \\ r & \text{para maximização} \end{cases}$$

onde i é o i-ésimo elemento do *front* f e r o *rank* deste último.

Além do *ranking* por *fronts*, o qual enfatiza os pontos ótimos, o NSGA também utiliza estratégias de *compartilhamento* ou *nichos*, as quais, por sua vez, estabilizam subpopulações de pontos "bons". Isto minimiza a possibilidade de convergência prematura e o surgimento de um "super-indivíduo", o que poderia gerar populações compostas unicamente de cópias de si mesmo. O algoritmo completo de NSGA está a seguir:

1. Considerando a integralidade da população de possíveis soluções, encontrar os pontos não-dominados.

2. Atribuir o mesmo valor de *dummy fitness* a todos os i indivíduos não-dominados do *front* 1 (em geral, este valor é igual a 1).

3. O próximo passo é o compartilhamento do valor de *dummy fitness* dos indivíduos, visando manter a diversidade. Isto é feito dividindo-o por uma quantidade proprorcional ao número de indivíduos em seu entorno, segundo um determinado raio, o que leva o nome de nicho e é quantificado como:

$$m_i = \sum_{j=0}^{M-1} Sh(d(i,j))$$

onde i e M são, respectivamente, o i-ésimo indivíduo e o número de indivíduos do *front* corrente e $Sh(d)$ é a *função de compartilhamento*. A função de compartilhamento $Sh(d)$, a qual é uma função linear decrescente e definida como:

$$Sh(d(i,j)) = \begin{cases} Sh(0) = 1 \\ \\ Sh(d(i,j)) = \begin{cases} 1 - \frac{d(i,j)}{\sigma_{share}} \text{ se } d(i,j) < \sigma_{share} \\ 0 \text{ se } d(i,j) \geq \sigma_{share} \end{cases} \end{cases}$$

onde:

- σ_{share} é a máxima distância fenotípica permitida entre dois indivíduos para que sejam pertencentes a um mesmo nicho.
- $d(i,j)$ é a *Distância Euclidiana Normalizada* e que é dada por:

$$d(i,j) = \sqrt{\sum_{k=0}^{np-1} (\frac{x_k^i - x_k^j}{max(k) - min(k)})^2}$$

onde np é o número de parâmetros (variáveis de decisão) definindo os cromossomos i e j; x_k^i e x_k^j são, respectivamente, os valores reais dos parâmetros k dos cromossomos i e j, e os valores máximo e mínimo do k-ésimo parâmetro do conjunto de indivíduos (população):

$$max(k) = \max_{i=0,\ldots,M-1} (x_k^i)$$

e

$$min(k) = \min_{i=0,\ldots,M-1} (x_k^i)$$

sendo M o número de indivíduos pertencendo ao *front* atual. O uso da distância normalizada é de interesse pois permite que $\sigma_{share} \in [0,1]$.

4. Na medida em que todo o processo é altamente dependente do valor de σ_{share}, é de interesse o emprego de um processo adaptativo em função do *front* atual. O método utilizado está a seguir:

(a) Considerando cada parâmetro de cada cromossomo, calcular seus valores máximo (max) e mínimo (min).

(b) Sabendo-se que n_{front} é o número de cromossomos do *front* atual, calcular:

$$\sigma_{share} = \frac{max - min}{n_{front} \times max}$$

O σ_{share} assim encontrado deve ser o utilizado no *front* atual.

5. Após o processamento do nicho para cada indivíduo do *front* atual é atribuído um novo valor de *dummy fitness*, a saber, f_i/m_i.

6. Neste estágio os pontos não-dominados que formam o *front* atual são temporariamente separados da população. Nesta, um novo *front* com soluções não-dominadas é calculado e o processo é repetido até que toda a população tenha sido classificada.

7. Após toda a população ter sido processada e a cada indivíduo ter sido atribuído um valor de *dummy fitness* os operadores de seleção, cruzamento e mutação são aplicados da maneira usual, como se tratasse de uma algoritmo genético mono-objetivo.

Um fluxograma do funcionamento do NSGA pode ser visto na figura 5.5.

Como já dito antes, excetuando-se o processo de classificação da população em *fronts*, o NSGA opera de modo semelhante ao algoritmo genético simples. Entretanto, com o objetivo de melhorar a diversidade genética e, conseqüentemente, incrementar o número de decisões pertencentes ao conjunto de Pareto-ótimo, foi implementada uma variação dinâmica das taxas dos operadores de cruzamento e de mutação (Vasconcelos, 2003). Além de manter a diversidade genética da população em um nível adequado, este procedimento também tem por objetivo evitar uma possível convergência prematura. Dentre as duas possibilidades existentes, que são a adaptação dinâmica por indivíduo e aquela baseada no comportamento médio da população, optou-se pela segunda, já que o contexto geral é aquele da otimização de um problema multi-objetivo.

O método baseia-se na chamada *medida de diversidade genética*, a partir da qual as probabilidades dos operadores de cruzamento e de mutação variam de acordo com os valores de desempenho médio e máximo encontrados numa população:

$$m_{dg} = \frac{(f_{max} - f_{min})}{f_{max}}$$

Quando $m_{dg} << 1$ tem-se pouca diversidade genética e quando $m_{dg} \cong 1$ tem-se, ao contrário, muita diversidade genética. Em outras palavras, isto significa que, ao longo do processo, quer-se variar as probabilidades dos operadores de cruzamento e mutação de tal forma que o valor médio de desempenho seja convergente ao valor máximo de desempenho (para cada geração).O procedimento completo é descrito no pseudo-código abaixo:

Se $m_{dg} \geq V_{sup}$

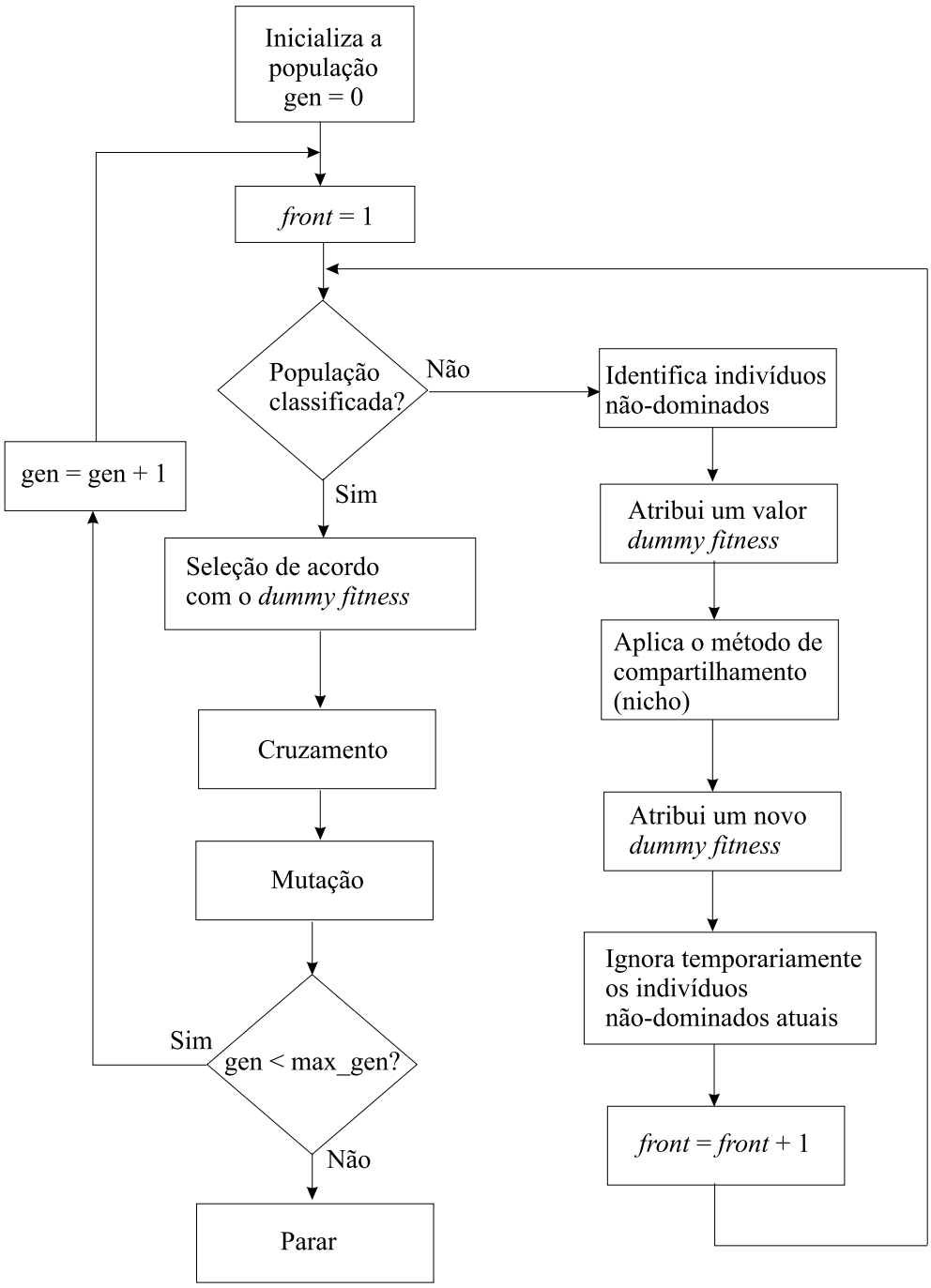

Figura 5.5: NSGA(*Nondominated Sorting Genetic Algorithm*).

$$pm = pm_{min}$$
$$pc = pc_{max}$$

Senão se $m_{dg} \leq V_{inf}$

$$pm = pm_{max}$$
$$pc = pc_{min}$$

Senão

$$pm = interpola(m_{dg}, V_{inf}, V_{sup}, pm_{min}, pm_{max})$$
$$pc = interpola(m_{dg}, V_{inf}, V_{sup}, pc_{min}, pc_{max})$$

onde pm é a probabilidade do operador de mutação e pm_{min} e pm_{max}, respectivamente, seus valores mínimo e máximo, pc é a probabilidade do operador de cruzamento e pc_{min} e pc_{max}, respectivamente, seus valores mínimo e máximo, m_{dg} é a medida de diversidade genética da faixa escolhida e $interpola()$ uma função de interpolação linear.

Na figura 5.6 pode ser observado o comportamento do processo descrito acima. Nesta

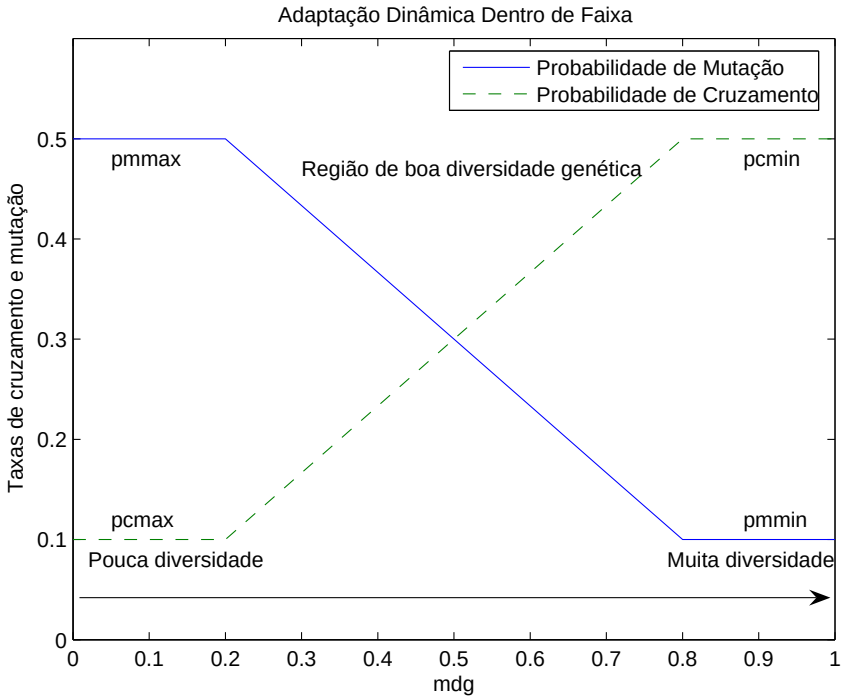

Figura 5.6: Adaptação Dinâmica Pela Média da População Dentro da Faixa $[V_{min}, V_{max}]$.

figura optou-se por valores meramente ilustrativos como delimitadores de faixa e das probabilidades de mutação e de cruzamento com o objetivo de facilitar a visualização do

método através da mesma. Na prática, principalmente no caso das probabilidades dos operadores, outros valores devem ser empregados. Antes de ilustrar o comportamento deste algoritmo através de um exemplo, resta ainda uma palavra acerca do método empregado para determinar a precisão de cálculo.

Considerando que cada elemento da população é uma cadeia de *bits* representando uma ou mais variáveis, a precisão de cálculo realizada com as variáveis representadas por esta cadeia é diretamente proporcional ao seu comprimento. Supondo $\mathbf{x}$ um vetor de variáveis de decisão e considerando l_i e $[x_i^{min}, x_i^{max}]$ como representando, respectivamente, o comprimento da cadeia de *bits* e a faixa de variação da i-ésima variável, a distância entre dois pontos da malha desta variável pode ser calculada como:

$$\Delta x_i = \frac{x_i^{max} - x_i^{min}}{2^{l_i} - 1}$$

Com base nesta equação é possível calcular-se, dada uma precisão, o comprimento de uma cadeia necessária para alcançar-se uma determinada precisão. Desta forma, tem-se:

$$l_i = ceil(log_2(\frac{x_i^{max} - x_i^{min}}{\Delta x_i} + 1))$$

onde $ceil()$ retorna o próximo inteiro maior ou igual ao seu argumento. Assim, caso seja desejada uma precisão igual a p casas decimais, basta fazer $\Delta x_i = 10^{-(p+1)}$ e calcular o comprimento da cadeia de *bits* necessário para alcançar-se tal precisão. Se, por exemplo, $p = 6$ e x_i varia no intervalo $[0, 1]$, será necessária um cadeia de 24 *bits* para alcançar-se a precisão desejada nesta variável.

A seguir, como exemplo de aplicação, é mostrado o emprego do NSGA na resolução do problema com duas funções objetivo mostrado na sub-seção 5.5.1, página 64. Para o processo de otimização o algoritmo foi inicializado com os parâmetros abaixo:

- Número de indivíduos (população): 500
- Faixa de variação da variável independente: $[-6, 6]$
- Tamanho do cromossomo: 22 (precisão de 10^{-5})
- Número de gerações: 10
- Adaptação dinâmica:
  - Probabilidade mínima do operador de mutação: 0.001
  - Probabilidade máxima do operador de mutação: 0.05
  - Probabilidade mínima do operador de cruzamento: 0.1
  - Probabilidade máxima do operador de cruzamento: 0.9
  - Valor inferior da faixa: 0.1
  - Valor superior da faixa: 0.9

Após dez gerações todos os indivíduos da população convergiram para o *front* de Pareto, resultado que pode ser visto na figuras 5.7 e 5.8. Na primeira pode ser visto a concordância deste resultado com a resolução analítica mostrada na sub-seção 5.5.1 e na qual se observa que o valor da variável independente x fica restrita ao intervalo $[0, 2]$. Na outra figura pode-se ver o *front* de Pareto e também a grande diversidade de resultados possíveis, o que mostra, por sua vez, a eficiência do método de nicho (compartilhamento) empregado.

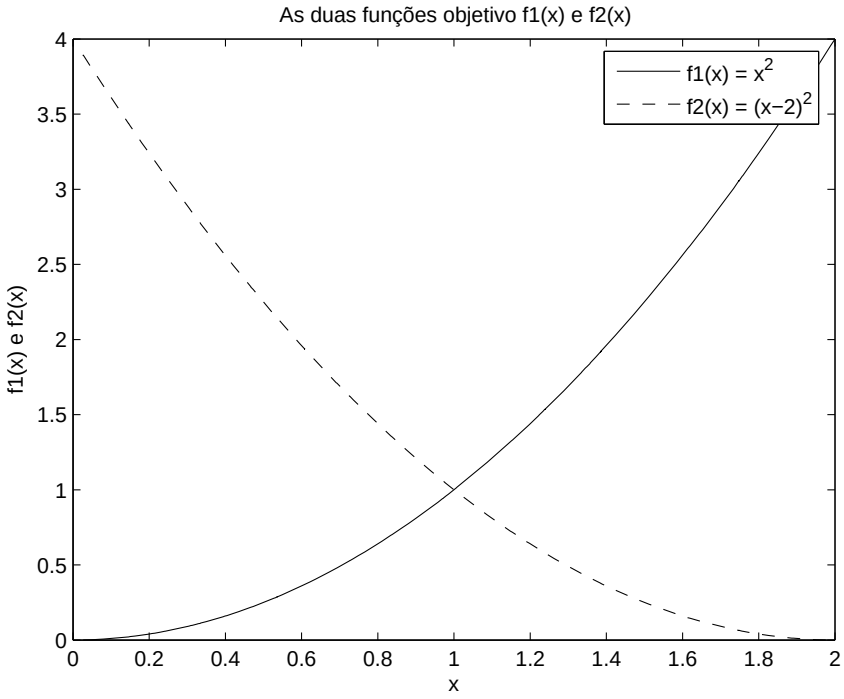

Figura 5.7: Universo de decisão para as funções f1 e f2.

5.6 Implementação da Estrutura Métrica

5.6.1 Arquitetura do Componente

Conforme foi visto anteriormente (figura 5.3), uma grade métrica é uma matriz na qual o número de linhas é o número de níveis métricos e o número de colunas é a interpolação (através de quanta de duração) do número de eventos. Implementar, porém, o algoritmo multi-objetivo através de uma população de grades métricas representadas por matrizes seria extremamente dispendioso do ponto de vista de memória. Para ter-se uma idéia das dimensões do problema, considere-se uma grade métrica (de resto, pequena) representando um trecho com 100 eventos e 5 níveis métricos. Sabendo que cada valor booleano ocupa 4 bytes, tem-se, portando, 2000 bytes como tamanho da grade. Considere-se, agora, uma grade desta dimensão fazendo parte de uma população em um algoritmo genético,

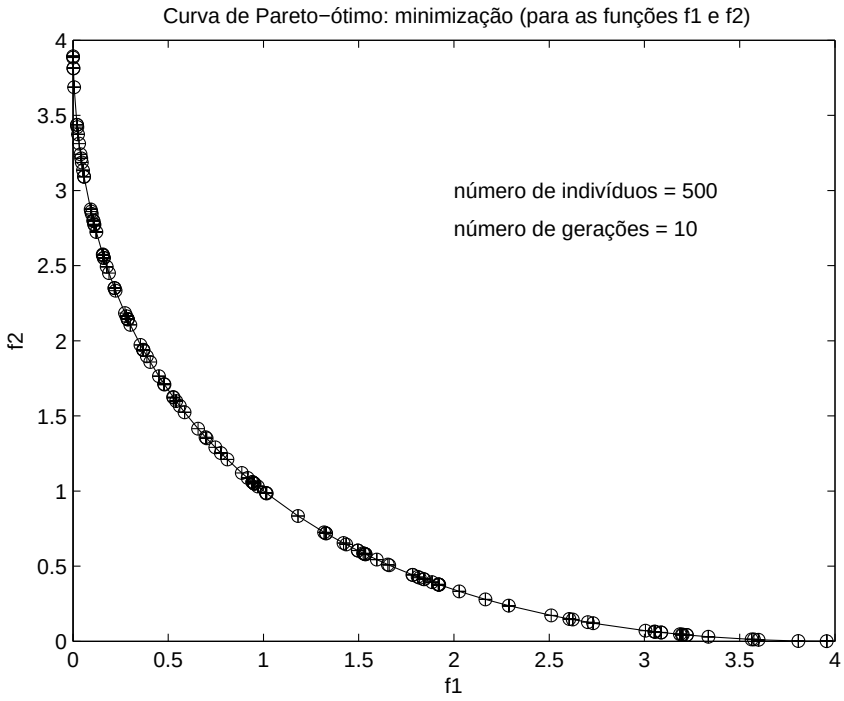

Figura 5.8: *Front* de Pareto (para as funções f1 e f2).

população esta com 500 indivíduos. Ter-se-á, então, um espaço de 1000000 bytes ocupado somente pela população de grades métricas. Dois caminhos surgem como possíveis soluções para este problema. A primeira via é o emprego de matrizes esparsas, tal como mostrado no apêndice A, página 181, o que contribuiria para uma considerável economia de memória. Esta solução, porém, apresenta o inconveniente de necessitar de um grande número de cálculos referentes às constantes conversões entre a forma original da matriz e sua forma compactada, o que faz-se necessário devido à forma de operação de ambos os conjuntos de regras (de *Boa–Formatividade* e *Preferenciais*) do componente. A segunda solução é alcançada através do uso de uma codificação da grade métrica, a qual é descrita a seguir.

Ao examinar-se a estrutura de uma grade métrica pode-se notar que, devido ao seu alto grau de periodicidade, ela pode ser codificada armazenando-se unicamente alguns de seus parâmetros, os quais são o número de elementos do menor nível, o número de níveis e, para cada nível, o offset em relação ao nível precedente e a separação entre os pulsos (2 ou 3). Estes parâmetros definindo uma Estrutura Métrica podem ser implementados assim[4]:

```
/*
 * Estruturas que codificam uma Estrutura Métrica (Grade Métrica).
 * Estas estruturas contém as informações para a geração de
```

[4]Já vistos no capítulo 3, página 15 e repetidos aqui.

```
 * uma grade métrica.
 */
typedef struct tagMetricalLevel {
  int level_number;     /* nivel da grade */
  int initial_position; /* posicao onde comeca em relacao ao nivel anterior */
  int spacing;          /* separacao entre os pulsos: 2 ou 3 */
} MetricalLevel;
typedef struct tagMetricalGrid {
  int number_of_levels;
  MetricalLevel level[MAX_METRICAL_LEVELS]; /* array com os niveis metricos */
} MetricalGrid;
```

Na medida em que o número de níveis não varia durante uma análise (são necessárias duas variáveis para definir a estrutura de cada nível) e além disto, o menor nível não possui separações entre os pulsos, tem-se como estrutura individual para compor a população do algoritmo genético uma entrada com $(l-1)*2$ variáveis, sendo l o número de níveis. Isto significa que, para uma análise de 6 níveis o algoritmo genético terá como entrada um conjunto de 10 variáveis.

Assim como no caso da Estrutura de Agrupamento, a implementação da Estrutura Métrica tem como característica sua divisão em dois momentos distintos, a saber, a implementação das Regras de Boa–Formatividade que, atuando como filtros, indicam se uma determinada estrutura pode ser considerada possível ou não, e a implementação das Regras Preferenciais, as quais, dentre todas as possíveis, escolhem a mais adequada.

Porém, de modo diverso à Estrutura de Agrupamento, a Estrutura Métrica tem suas Regras Preferenciais implementadas como um conjunto de funções de devem ser minimizadas. Na medida em que muitas vezes são conflitantes, configuram-se como um problema multi-objetivo de minimização.

O algoritmo NSGA foi escolhido porque apresenta uma característica importante que é, segundo seus autores (Srinivas & Deb, 1994), manter sua eficiência independente do número de funções objetivo. Como no caso deste trabalho existem 10 funções objetivo (dez Regras Preferenciais de Métrica), tal característica é muito relevante.

As principais diferenças entre a implementação do NSGA mostrado antes e aquele utilizado na implementação da Estrutura Métrica diz respeito ao número de funções objetivo e ao número de variáveis independentes. No primeiro tem-se duas funções objetivo e uma única variável independente e no segundo tem-se dez funções objetivo e um número de variáveis independentes que é função da quantidade de níveis da grade métrica.

Uma representação gráfica da arquitetura do componente Estrutura Métrica pode ser vista na figura 5.9.

5.6.2 Implementação do Algoritmo Genético

Além da implementação das possibilidades mostradas nas seções 5.5.2 (adaptação dinâmica de mutação, cruzamento e sigma dinâmico para melhorar a eficiência no nicho) e 5.6.1 (compactação da informação necessária à implementação da grade métrica), foi também

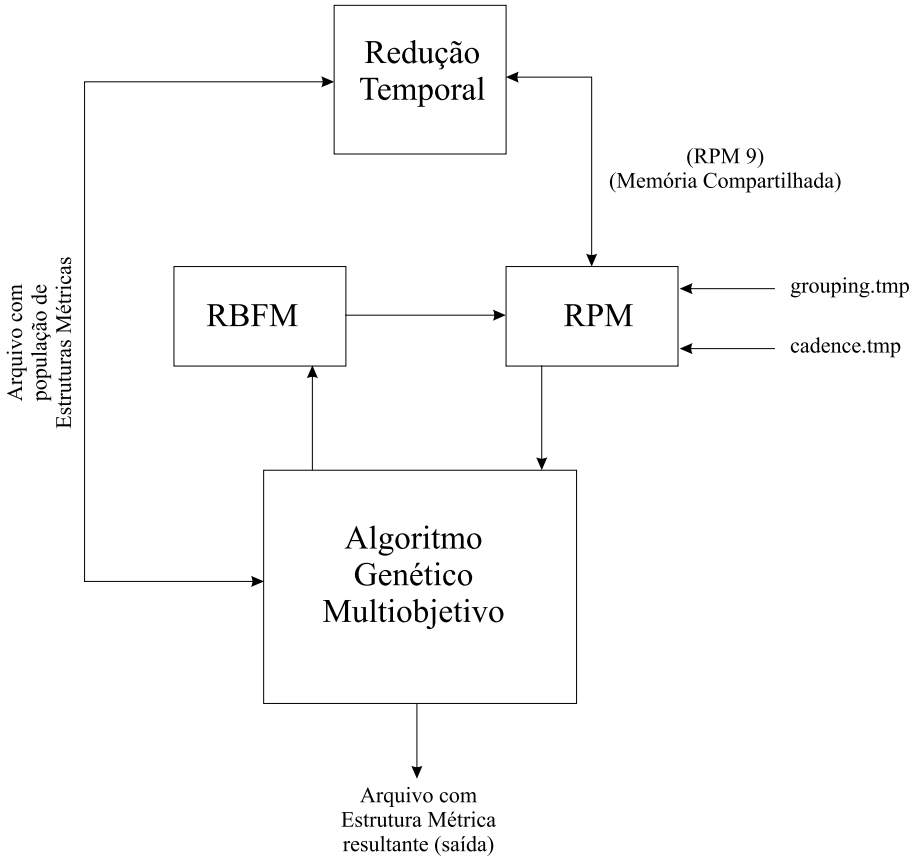

Figura 5.9: Arquitetura da Estrutura Métrica.

implementado a técnica de crossover por variável[5] (Vasconcelos, 2003), a qual mostrou-se necessária para garantir a diversidade genética pertinente à cada variável (cujo número, como foi visto antes, é função da quantidade de níveis métricos presentes na análise).

5.6.3 Regras de Boa–Formatividade

Assim como no caso do componente anterior, as regras de Boa–Formatividade agem como filtros avaliando se estruturas métricas são possíveis ou não. No caso específico deste componente, as regras de Boa–Formatividade aferem se as estruturas métricas criadas aleatoriamente como indivíduos da população de estruturas no algoritmo genético multiobjetivo são possíveis dentro das perspectivas da TGTM.

[5]A qual não foi citada na mesma ocasião dos dois outros casos devido ao fato do problema sendo mostrado como exemplo ter uma única variável independente.

RBFM 1

Esta regra diz que todo ponto de ataque do trecho sendo analisado deve coincidir com um pulso em seu menor nível métrico. A implementação desta regra é simples, pois apenas verifica se a primeira linha da grade métrica (representando a Estrutura Métrica) é composta somente de valores verdadeiros (pulsos igualmente espaçados por quanta de duração)[6]. Caso esta condição não seja satisfeita, a regra retorna falso, o que corresponde a um erro.

RBFM 2

Esta regra dita que todo pulso nos níveis acima do primeiro é também um pulso em todos os níveis menores do que ele. Para isto e em sua implementação, a regra verifica todos os níveis a partir do segundo e, quando encontra um pulso, verifica se, em todos os outros níveis anteriores (linhas), esta posição (coluna) é também um pulso. Não sendo satisfeita a condição, retorna falso.

RBFM 3

A RBFM 3 limita a periodicidade dos diversos níveis da Estrutura Métrica em períodos de 2 ou 3. Para isto, na sua implementação, percorre cada nível a partir do segundo conferindo a distância d (em quanta de duração) entre cada um de seus pulsos. Se, para cada nível e para cada par de pulsos no nível atual (d mod 2) e (d mod 3) forem diferentes de zero a regra retorna retorna falso.

RBFM 4

Esta regra de Boa–Formatividade, assim como em outros casos, é implementada sem levar em conta certas especificidades da teoria. Neste caso particular, trata-se de considerar, para efeito de análise, todos os níveis métricos (com exceção do primeiro, naturalmente) e não somente o nível do tactus e aqueles imediatamente maiores do que ele. Dito de outra forma, isto significa que *todos* os níveis métricos de uma peça devem ser constituídos de pulsos igualmente espaçados. Segue o resumo do algoritmo utilizado:

1. Para cada nível da grade métrica:
   (a) Acha a posição do primeiro pulso.
   (b) Calcula a distância d (em quanta de duração) até o próximo pulso.
   (c) Percorre o restante do nível atual e verifica se a distância d é mantida entre os demais pulsos.
   (d) Se a distância d entre os pulsos foi alterada, retornar falso.

[6]Neste ponto, lembrar que a grade métrica é do tipo booleano.

2. Se for o último nível da grade métrica, ir para o próximo passo; senão, ir para o passo 1.

3. Retornar verdadeiro.

5.6.4 Regras Preferenciais

As Regras Preferenciais são implementadas como funções que retornam um valor que representa o erro de sua adequação a uma determinada situação. Por exemplo, a RBFM 1 diz que grupos ou partes de grupos devem, preferencialmente, ocupar a mesma posição métrica (dentro de uma dada Estrutura Métrica). Conseqüentemente, se isto ocorre para uma dada estrutura de grupos, o erro retornado por esta regra será zero e crescerá proporcionalmente ao afastamento desta situação ideal durante a interação entre a Estrutura de Agrupamento e a Estrutura Métrica que está sendo avaliada. A seguir é mostrada a implementação de cada uma das Regras Preferenciais de Métrica.

RPM 1 – Paralelismo

A RPM 1 procura privilegiar a construção paralela, do ponto de vista métrico, de agrupamentos e motivos. Para isto, retorna um erro que significa o quanto uma dada Estrutura de Agrupamento se adapta a uma Estrutura Métrica sendo avaliada. Evidentemente, para que esta regra possa operar corretamente o primeiro componente da teoria, a *Estrutura de Agrupamento*, já deverá ter sido calculado.

Basicamente, o procedimento empregado na regra verifica quantos niveis métricos, de uma Estrutura Métrica sendo avaliada, estão presentes em cada par de grupos de cada nível da estrutura de agrupamento. Quanto maior a diferença entre os níveis métricos encontrados para cada par de grupos, maior o erro. Naturalmente, quando dois grupos de um mesmo nível ocuparem posição métrica equivalente o erro será igual a zero.

O algoritmo empregado é o seguinte:

1. Inicializar o contador de operações: $c = 0$.

2. Para cada nível da Estrutura de Agrupamento:

   (a) Para cada par de grupos do nível atual da Estrutura de Agrupamento:

      i. Calcular quantos níveis métricos existem em cada uma das posições iniciais de cada um dos grupos do par.

      ii. Verificar qual dos dois valores encontrados é o maior (max) e qual o menor (min).

      iii. Calcular e armazenar o erro para o par: $e_c = 1 - (min/max)$.

      iv. Incrementar o contador de operações: $c = c + 1$.

   (b) Se existem pares a serem processados no nível atual da estrutura de agrupamento, voltar ao item anterior.

(c) Senão, termina o processamento do nível atual.

3. Se existem níveis da Estrutura de Agrupamento a serem processados, voltar ao item 2.

4. Senão:

   (a) Somar todos os erros encontrados (todos os pares de todos os níveis):

$$E_s = \sum_{i=0}^{c} e_i$$

   (b) Calcular e retornar o erro final: $E_f = E_s/c$.

Como exemplo de aplicação deste algoritmo estão mostradas na figura 5.10 uma Estrutura de Agrupamento e uma estrutura métrica que deverá ser avaliada em relação à primeira. Antes, entretanto, de apresentar o exemplo completo com o cálculo dos erros, alguns comentários sobre estas estruturas são necessários.

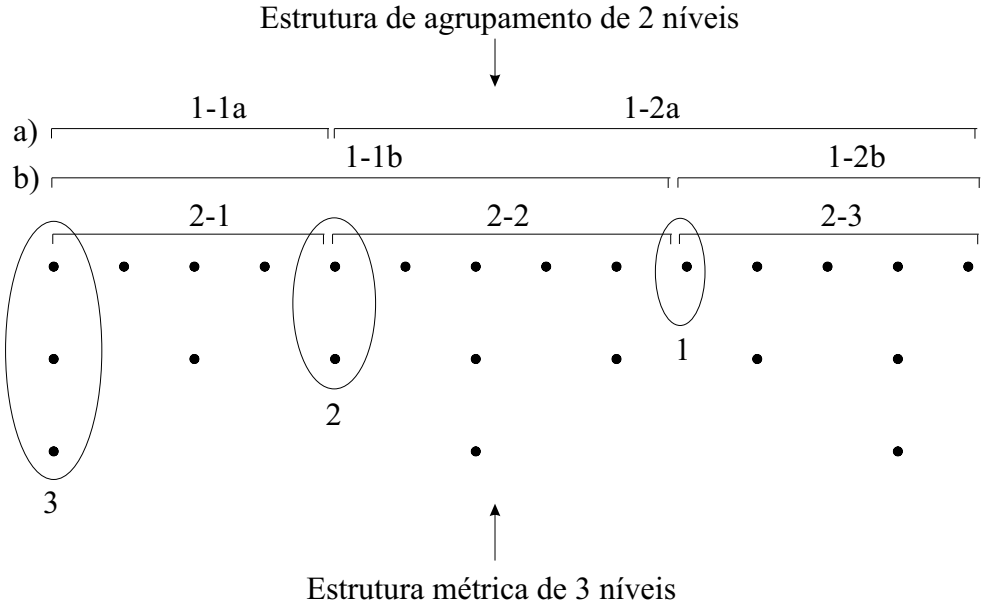

Figura 5.10: Interação entre Estrutura de Agrupamento e Estrutura Métrica.

A Estrutura Métrica deste exemplo possui 3 níveis. O primeiro nível é formado, como já foi dito anteriormente, por pulsos igualmente espaçados e separados temporalmente por um quantum de duração (menor duração presente na obra sendo analisada). O segundo nível possui metro binário (alternância entre pulsos fortes e fracos considerando o nível anterior) enquanto que o terceiro nível apresente metro ternário (pulsos fortes de três em três, considerando o nível anterior).

No que diz respeito à Estrutura de Agrupamento, trata-se de uma estrutura com dois níveis, sendo que o primeiro nível[7] mostrado como duas possibilidades (a e b) de absorção dos grupos do segundo nível quando de sua passagem para o primeiro nível. Na opção (a), o primeiro grupo fica intacto enquanto que o segundo e o terceiro unem-se em um único grupo. Na segunda opção, é a vez do terceiro grupo permanecer inalterado e o primeiro e segundo unirem-se em um único grupo. Como será constatado mais adiante, estas duas possibilidades apresentarão resultados diversos quando de sua interação com uma mesma Estrutura Métrica, tal como a da figura 5.10.

Neste exemplo serão calculados os erros na interação entre as estruturas de agrupamento e métrica mostradas na figura 5.10. Para isto serão inicialmente calculados os erros para o nível 2 da Estrutura de Agrupamento e, em seguida, os erros para o nível 1 (possibilidade (a) e depois (b)). Em seqüência, os erros serão somados e o erro final calculado como sendo o resultado da soma dividido pelo número de operações realizadas, o que totaliza três operações para o nível 2 mais uma operação para o nível 1 (para cada possibilidade do mesmo).

A seqüência de passos para o cálculo dos erros no nível 2 é:

1. Existem três pares de grupos: 2-1/2-2, 2-1/2-3 e 2-2/2-3. Assim, três valores de erro devem ser calculados.
2. Para 2-1/2-2 tem-se $min = 2$ e $max = 3$. Portanto, $e_{(2-1/2-2)} = 1 - (2/3) = 0,33$.
3. Para 2-1/2-3 tem-se $min = 1$ e $max = 3$. Portanto, $e_{(2-1/2-3)} = 1 - (1/3) = 0,67$.
4. Para 2-2/2-3 tem-se $min = 1$ e $max = 2$. Portanto, $e_{(2-2/2-3)} = 1 - (1/2) = 0,5$.
5. Número de operações realizadas: 3.

A seqüência de passos para o cálculo dos erros no nível 1 é:

1. Possibilidade (a):
   (a) Existe um par de grupos: 1-1a/1-2a. Assim, um valor de erro deve ser calculado.
   (b) Tem-se $min = 2$ e $max = 3$. Portanto, $e_{(1-1a/1-2a)} = 1 - (2/3) = 0,33$.
   (c) Número de operações realizadas: 1.
2. Possibilidade (b):
   (a) Existe um par de grupos: 1-1b/1-2b. Assim, um valor de erro deve ser calculado.
   (b) Tem-se $min = 1$ e $max = 3$. Portanto, $e_{(1-1b/1-2b)} = 1 - (1/3) = 0,67$.

[7]Na realidade, existe ainda um último nível, constituído de apenas um grupo, mas que não é mostrado aqui tanto para não sobrecarregar a figura quanto, motivo mais relevante, pelo fato de não ser necessário no contexto da ilustração sendo elaborada.

(c) Número de operações realizadas: 1.

Como resultados finais tem-se, para a possibilidade (a):

$$E_a = \frac{e_{(2-1/2-2)} + e_{(2-1/2-3)} + e_{(2-2/2-3)} + e_{(1-1a/1-2a)}}{4} = 0,4575$$

e para a possibilidade (b):

$$E_b = \frac{e_{(2-1/2-2)} + e_{(2-1/2-3)} + e_{(2-2/2-3)} + e_{(1-1b/1-2b)}}{4} = 0,5425$$

o que mostra que a Estrutura Métrica sendo avaliada é menos adequada à estrutura de agrupamento representada pela possibilidade (b) do que o é àquela representada pela possibilidade (a). Exemplos semelhantes podem ser construídos para testar as implementações das regras preferenciais que são expostas a seguir.

RPM 2 – Pulso Forte Antes

A RPM 2 diz que pulsos fortes devem, preferencialmente, aparecer relativamente cedo em um grupo. Como no caso anterior, para a implementação e aplicação desta regra é necessário que a Estrutura de Agrupamento do trecho musical a ser analisado já tenha sido previamente calculada e armazenada em um arquivo que será, então, lido pelo programa encarregado do cálculo da estrutura métrica.

Para quantificar esta regra, foi empregado o seguinte algoritmo:

1. Inicializar a variável $i = 0$.
2. Para cada nível i da Estrutura de Agrupamento:
   (a) Inicializar a variável $j = 0$.
   (b) Para cada grupo j do nível atual da Estrutura de Agrupamento:
      i. Calcular o maior nível métrico n e sua posição p_n.
      ii. Calcular o erro de densidade métrica do grupo:

$$e_d = 1 - \frac{n}{N}$$

      onde N é o número de níveis da Estrutura Métrica.
      iii. Calcular o erro de posição do grupo:

$$e_p = \frac{p_n + 1}{l}$$

      onde l é o comprimento do grupo em quanta de duração.
      iv. Calcular e armazenar o erro total para o grupo: $e_j = (e_d + e_p)/2$.
      v. $j = j + 1$.

(c) Calcular o erro do nível atual da Estrutura de Agrupamento

$$EG_i = \frac{\sum_{j=0}^{N} G_i e_j}{NG_i}$$

onde NG_i é o número de grupos do i-ésimo nível.

3. Se existem níveis da Estrutura de Agrupamento a serem processados, fazer $i = i+1$ e voltar ao item 2.

4. Senão:

(a) Somar todos os erros encontrados (de todos os níveis):

$$E_t = \frac{\sum_{i=0}^{NL} EG_i}{NL}$$

onde NL é o número de níveis da estrutura de agrupamento.

(b) Retornar o erro total E_t.

RPM 3 – Evento

Esta regra diz que, preferencialmente, pulsos fortes devem coincidir com ataques de eventos-altura. O erro, portanto, é calculado através do número de eventos-altura cujos ataques não coincidem com um pulso forte. Na medida em que o menor nível da Estrutura Métrica é constituído de pulsos igualmente espaçados de um quantum de duração, é evidente que todos os eventos-altura existentes no trecho a ser analisado coincidirão com alguns de seus pulsos. Assim, para efeito de cálculo do erro, somente serão considerados os níveis acima do primeiro. Além disso, como este trabalho trata de um sistema a ser empregado em música homofônica, somente a linha superior é considerada para efeitos de análise.

O algoritmo empregado para esta regra é bem mais simples do que os anteriores e está a seguir:

1. $e = 0$.

2. Para cada evento-altura do trecho considerado:

(a) Calcular o nível métrico l no momento do ataque do evento-altura.

(b) Se $l > 1$:

$$e = e + 1 - \frac{l}{NL}$$

onde NL é o número de níveis da estrutura métrica.

3. Calcular e retornar o erro final $E_f = e/NA$, onde NA é o número de eventos-altura.

RPM 4 – Ênfase

Esta regra tem sua operação semelhante à anterior, porém, ao invés de verificar momentos de ataque com pulsos fortes, verifica a coincidência destes com eventos acentuados (do ponto de vista da intensidade). Para isto é necessário definir-se esta expressão como sendo um evento que possui um valor de intensidade 30% maior do que o evento que o precede e aquele que o segue.

Como na regra precedente e pelo mesmo motivo, a RPM 4 somente utiliza a parte superior do trecho a ser analisado. O algoritmo está a seguir:

1. Inicializar a variável $e = 0$.
2. Para cada evento-altura do trecho considerado:
   (a) $l = 1$
   (b) Se a intensidade for 30% maior do que aquelas dos eventos precedente e subseqüente, calcular o nível métrico l do evento-altura.
   (c) Se $l > 1$:
   $$e = e + 1 - \frac{l}{NL}$$
   onde NL é o número de níveis da estrutura métrica.
3. Calcular e retornar o erro final $E_f = e/NA$, onde NA é o número de eventos-altura.

RPM 5 – Comprimento

Esta regra trata da importância do surgimento (momento de ataque) de eventos de relativamente longa duração ocorrerem em pulsos fortes. Antes de mais nada é necessário definir *evento de relativamente longa duração*.

Ao invés de, como na regra anterior, estabelecer um valor fixo que seria considerado como de longa duração, é utilizado aqui um sistema de pesos com o objetivo de classificar as durações umas em relação às outras. Feita esta classificação é possível fazer com que os diversos valores de duração encontrados no trecho a ser analisado influam de modo diverso no erro final.

Esta regra também utiliza somente a parte superior do trecho a ser analisado e seu algoritmo é o seguinte:

1. Inicializar a variável $e = 0$.
2. Calcular o vetor de pesos para o trecho a ser analisado e considerando a duração de cada evento:
   $$\mathbf{w} = \mathbf{dur}/dq$$
   onde $\mathbf{w}$ é o vetor de pesos, $\mathbf{dur}$ é o vetor com as durações dos eventos e dq é o quantum de duração.

3. Fazer $\mathbf{w} = \mathbf{w}/max(\mathbf{w})$ (normalização).
4. Inicializar a variável $i = 0$.
5. Para cada i-ésimo evento-altura do trecho considerado:
   (a) $i = 0$
   (b) $l = 1$
   (c) Calcular o nível métrico l do evento-altura.
   (d) Se $l > 1$:
   $$e = e + 1 - \frac{n}{NL} * w_i$$
   onde NL é o número de níveis da estrutura métrica.
   (e) i = i + 1
6. Calcular e retornar o erro final $E_f = e/NA$, onde NA é o número de eventos-altura.

RPM 6 – Baixo

A RPM 6 diz que o baixo de um trecho musical deve ocupar, preferencialmente, posições metricamente estáveis. Em outras palavras, o baixo deve, preferencialmente, ocupar posições onde existam vários níveis métricos. Naturalmente, para o cálculo desta regra somente o baixo do trecho a ser analisado é levado em consideração.

O algoritmo empregado é o seguinte:

1. Calcular a altura mais grave (p_{min}) e a mais aguda (p_{max}) do baixo[8].
2. Calcular $n_{div} = p_{max} - p_{min}$.
3. Inicializar um vetor de pesos $\mathbf{w}$ de tamanho n_{div} realizando a interpolação linear no intervalo $[1, 0]$, onde zero corresponda à p_{max} e um a p_{min}.
4. $e = 0$.
5. Para cada evento-altura do trecho considerado:
   (a) $l = 1$
   (b) Calcular o nível métrico l do evento-altura.
   (c) Se $l > 1$:
   $$e = e + 1 - \frac{l}{NL} * w[p - p_{min}]$$
   onde NL é o número de níveis da estrutura métrica e p é a altura do evento atual.
6. Calcular e retornar o erro final $E_f = e/NA$, onde NA é o número de eventos-altura.

[8]Como sempre, são utilizados os valores de altura representados no protocolo MIDI, os quais ocupam o intervalo $[0, 127]$, tendo o dó central o valor 60.

RPM 7 – Cadências

A implementação desta regra necessita que um arquivo com as cadências encontradas no trecho a analisar já tenha sido gravado. O algoritmo calcula, então, para cada uma das cadências listadas, os números de níveis encontrados nos instantes de ataque do *penult* e do *final*, calculando o erro médio[9]. Após o processamento de todas as cadências o erro médio total é encontrado. O algoritmo está a seguir:

1. Carregar o arquivo com os dados das cadências.
2. $i = 0$.
3. Para cada i-ésima cadência:
   (a) Calcular o erro para o *penult*:
   $$e_p = \frac{l_p}{NL}$$
   onde l_p é o numero de níveis métricos no instante do ataque do *penult* e NL é o número de níveis métricos.
   (b) Calcular o erro para o final:
   $$e_f = \frac{l_f}{NL}$$
   onde l_f é o numero de níveis métricos no instante do ataque do final e NL é o número de níveis métricos.
   (c) Calcular o erro da cadência i: $c_i = (e_p + e_f)/2$
   (d) $i = i + 1$
4. Calcular e retornar o erro total:
   $$E_t = \frac{\sum_{i=0}^{N_c-1}(1 - c_i)}{N_c}$$
   onde N_c é o número de cadências.

RPM 8 – Suspensão

Esta Regra Preferencial é, de longe, a que possui a mais difícil implementação, já que a detecção de suspensões é um processo que, por si mesmo, já envolve o conhecimento da Estrutura Métrica. Sendo assim, já que o objetivo da regra é auxiliar na detecção desta última, e tentando viabilizar uma implementação, elaborou-se um algoritmo que possui como entrada, considerando duas vozes quaisquer, os instantes de ataque, as durações e as alturas de dois eventos. Com estes parâmetros o algoritmo é capaz de detectar suspensões. Este está a seguir:

[9]Na teoria de Lerdahl & Jackendoff (1996) *penult* e *final* são definidos, respectivamente, como o penúltimo e o último acordes de uma cadência. Assim, numa cadência perfeita o *penult* é o V e o final é o I, numa cadência interrompida o *penult* é o V e o final é o VI e numa meia-cadência não existe *penult* e o final é o V.

1. n = 0.
2. Ao percorrer a matriz de eventos, duas a duas vozes:
3. Se ($ev_1.ataque < ev_2.ataque$ E $ev_1.ataque + e_1.dur > ev_2.ataque$) OU ($ev_2.ataque < ev_1.ataque$ E $ev_2.ataque + e_2.dur > ev_1.ataque$) E $Dissonante(ev_1.pitch, ev_2.pitch)$, então

   $e_n = 1 - \frac{d}{L}$
   $n = n + 1$
4. Ao terminar de percorrer a matriz de eventos, calcular $E_{total} = \sum_0^{n-1} e_n/n$

onde n é o contador de suspensões encontradas, ev_1 e ev_2 são dois eventos com seus parâmetros de ataque, duração e altura, $Dissonante()$ é uma função boolenana que retorna verdadeiro quando as alturas dos dois eventos em questão formam um intervalo dissonante, e_n é o erro encontrado para a n-ésima suspensão, d é a profundidade na Estrutura Métrica sendo investigada no instante de ataque do evento com maior ataque, L a quantidade de níveis métricos da Estrutura Métrica sendo investigada e E_{total} o erro total encontrado na RPM 8 para a estrutura métrica sob análise. A implementação da função $Dissonante()$, a qual determina se duas alturas são dissonantes, é mostrada a seguir:

```
boolean Dissonante(altura1,altura2)
intervalo = (abs(altura1 - altura2))mod12
tipo = FALSO
Se intervalo ==  1 OU
   intervalo ==  2 OU
   intervalo ==  5 OU
   intervalo ==  6 OU
   intervalo == 10 OU
   intervalo == 11 OU
   tipo = VERDADEIRO
retornar tipo
```

RPM 9 – Interação Temporal

Na medida em que esta regra trabalha com a interação entre uma Estrutura Métrica e as questões relativas à Redução Temporal faz-se necessário o acesso ao código deste componente. Como este é implementado separadamente como parte do sistema, fez-se necessário o emprego de recursos que possibilitassem a comunicação entre os dois processos relativos ao dois componentes. Assim, foi implementado um recurso de *memória compartilhada* o qual funciona da maneira descrita a seguir.

O processo resume-se no envio dos dados de uma Estrutura Métrica para o processo encarregado da Redução Temporal, sua avaliação por este último e o retorno de um valor de erro (adaptação). Ao ser chamada, a função destinada a avaliar RPM 9 inicializa uma estrutura composta de três elementos:

- Um campo do tipo *booleano* que indica o sentido da mensagem.
- Um campo do tipo *real* destinado a armazenar o erro.
- Um campo do tipo *MetricalGrid* (ver acima em 5.6.1) destinado a armazenar os dados da Estrutura Métrica sendo avaliada.

Ao receber a Estrutura Métrica, RPM 9 a coloca, juntamente com o erro (inicializado em zero) e a direção do processo (inicializado em *verdadeiro*), no segmento de memória compartilhada previamente alocado. Os dados tornam-se, então, visíveis pelo processo encarregado da Redução Temporal e o processo de cálculo do erro torna-se possível. O resultado, em seguida, é colocado no campo do erro e a direção do processo é alterada para *falso* com o objetivo de que a RPM 9 possa ler o valor de erro no segmento de memória compartilhada e retorná-lo.

Na medida em que diferentes *race conditions* estão em jogo, faz-se necessário o uso de um sincronismo, o que foi realizado através de semáforos. Além, disto, com o objetivo de separar no interior do processo de Redução Temporal as tarefas que lhe são próprias e aquela em que ele auxilia o processo encarregado da Estrutura Métrica, o cálculo do erro da RPM 9 foi implementado como sendo uma *thread*, o que possibilita um paralelismo de processamento no interior da Redução Temporal. No que diz respeito aos critérios acerca do cálculo do erro, estes serão descritos no texto relativo à Redução Temporal, já que fazem uso das Regras de Boa–Formatividade e das Regras Preferenciais deste componente da TGMT. Quanto às referências ao emprego de *threads*, podem ser encontradas em Stevens (1999), Brown (1994) e Glass (1993).

Como a integridade do processo somente pode ser compreendida levando-se em conta tanto os passos presentes na Estrutura Métrica quanto aqueles que compõem a Redução Temporal, são dados, em seguida, os dois algoritmos relativos à comunicação entre estes dois componentes. Inicialmente, o algoritmo de RPM 9:

```
/* Algoritmo de RPM 9 */
/*
 * Começa a seção crítica.
 */
Lock(semáforo);
TransfereEstruturaMétrica(); // grade métrica para a memória compartilhada.
Erro = 0;
Direção = VERDADEIRO;
Unlock(semáforo);
/*
 * Fim da seção crítica.
 */
/*
 * Loop que verifica se o cálculo já foi realizado
 * (valor de Direção alterado para FALSO).
 */
```

```
Enquanto(Direção); // somente espera.
/*
 * Começa a seção crítica.
 */
Lock(semáforo);
TransfereErro(ErroLocal); // memória compartilhada para variável ErroLocal.
Unlock(semáforo);
/*
 * Fim da seção crítica.
 */
Retorna(ErroLocal);
```

Segue agora o algoritmo presente na Redução Temporal e que atua juntamente com anterior:

```
/* Algoritmo para comunicação com a Redução Temporal */
/*
 * Inicializa o erro em zero.
 */
Lock(semáforo);
Erro = 0.;
Unlock(semáforo);
/*
 * Loop que verifica se o cálculo do erro foi demandado pela
 * Estrutura Métrica.
 */
Enquanto(Erro > -1.) {
   Se(Direção) {
      /*
       * Começa a seção crítica.
       */
      Lock(semáforo);
      /*
       * Calcula o erro.
       */
      Erro = CalculaErro();
      /*
       * Muda a direção.
       */
      Direção = FALSO;
      /*
       * Fim da seção crítica.
       */
      Unlock(semáforo);
```

```
    }
  }
  ThreadExit(0);
```

Quando, finalmente, chegar a termo o processo encarregado da Estrutura Métrica, as seguintes instruções são executadas:

```
Lock(semáforo);
Erro = -1;
Unlock(semáforo);
```

as quais desativam a *thread* responsável pelo calculo do erro.

RPM 10 – Regularidade Binária

A RPM 10, de modo diverso das demais, utiliza somente a Estrutura Métrica que deve ser avaliada. Para isto verifica, em todos os níveis (exceto o primeiro, naturalmente), quais deles possuem regularidade binária. Para o cálculo do erro é dividido o número de níveis não-binários pelo número de níveis menos 1. Segue o algoritmo:

1. Inicializar a variável $e = 0$.
2. Na Estrutura Métrica, para cada um dos níveis a partir do segundo,

   Se a regularidade é ternária fazer $e = e + 1$.
3. Calcular e retornar o erro total $E_t = e/(n_l - 1)$, onde n_l é o número de níveis menos 1.

5.7 Resultados

5.7.1 Testes de Validação

Os testes de validação para a Estrutura Métrica propõem a análise de dois dos temas da Sinfonia nº 40 de W.A. Mozart, aqueles do primeiro e do último movimento. No caso do primeiro destes temas, sua Estrutura de Agrupamento já foi apresentada no capítulo 4, página 37 e pode ser vista na figura 4.4. Portanto, não será repetida aqui. Entretanto, tal não é o caso do segundo dos temas e sua estrutura de agrupamento será mostrada conjuntamente com sua Estrutura Métrica. Os resultados com as respectivas estruturas métricas de cada um dos temas são mostrados nas figuras 5.11 e 5.12.

Levando em conta a importância que as questões harmônicas possuem para a compreensão da Estrutura Métrica, os dois temas citados antes também são analisados, cada um deles, considerando a parte da orquestra reduzida. Desta forma os resultados poderão ser confrontados, permitindo, assim, a verificação de até qual ponto a harmonia influenciou em suas diferenças. Os resultados com as reduções de orquestra para os temas citados acima podem ser vistos nas figuras 5.13 e 5.14.

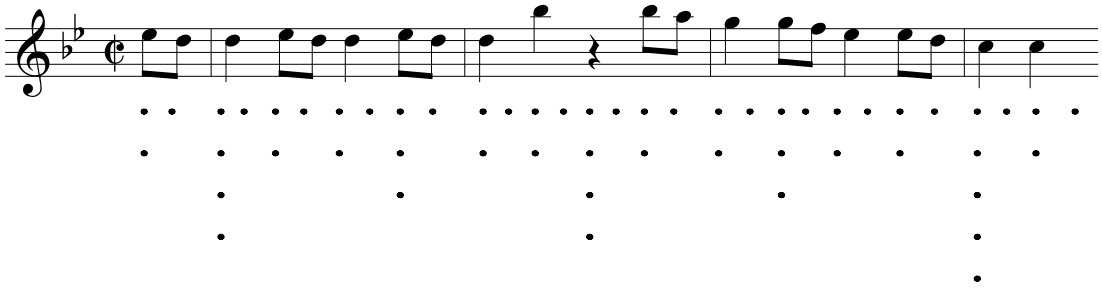

Figura 5.11: Estrutura Métrica para o primeiro tema do primeiro movimento da Sinfonia nº 40 de W.A. Mozart.

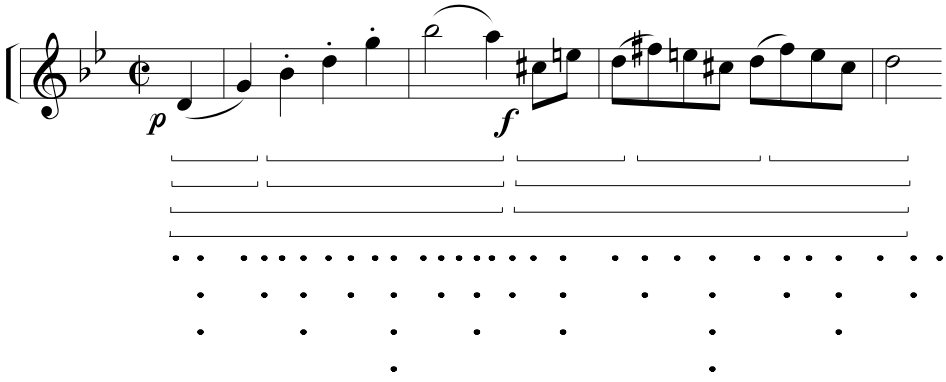

Figura 5.12: Estrutura Métrica para o primeiro tema do quarto movimento da Sinfonia nº 40 de W.A. Mozart.

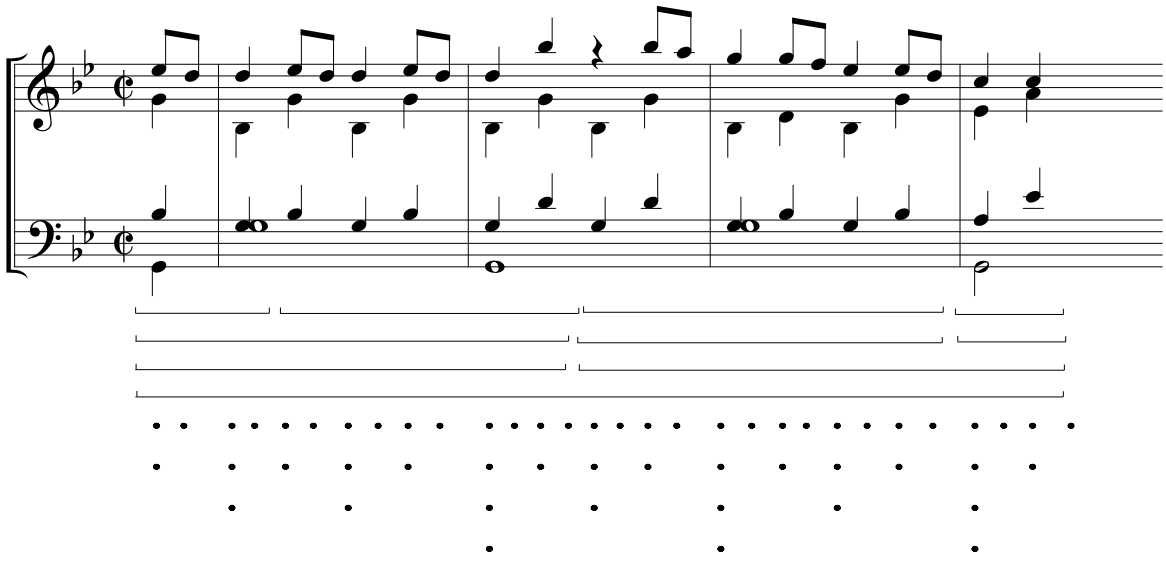

Figura 5.13: Estrutura Métrica para o primeiro tema do primeiro movimento da Sinfonia nº 40 de W.A. Mozart (com redução orquestral).

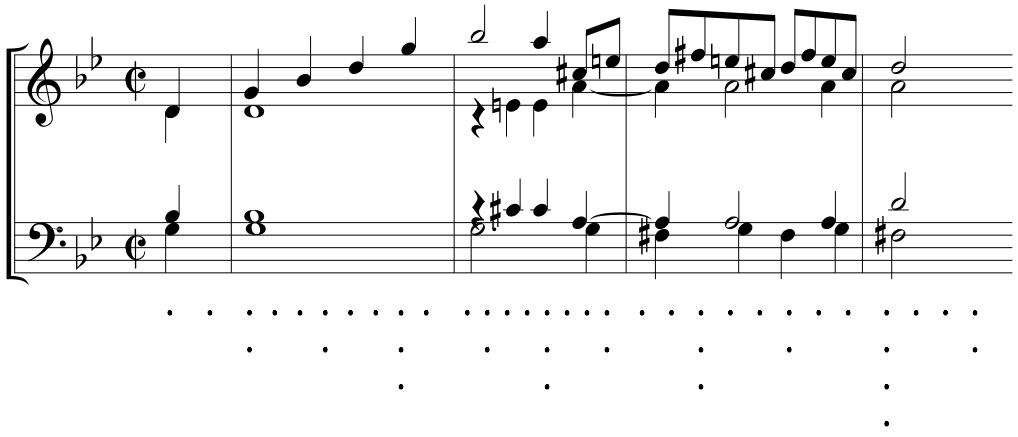

Figura 5.14: Estrutura Métrica para o primeiro tema do quarto movimento da Sinfonia nº 40 de W.A. Mozart (com redução orquestral).

5.7.2 Análises

Das obras selecionadas como objeto de análise do sistema duas delas já foram abordadas no capítulo 4 (Estrutura de Agrupamento). São elas o coral *Christus, der ist mein Leben* (BWV 95) de Johann Sebastian Bach e o tema do 3º movimento do Quarteto de Cordas Op. 18 nº 5, de Ludwig van Beethoven. Na seção de resultados do capítulo 4 há ainda uma terceira obra, a primeira das *Três Peças (para clarinete solo)*, de Igor Stravinsky, a qual não é incluída aqui por dois motivos. Inicialmente, trata-se de uma obra que ostenta uma métrica assimétrica e irregular, muito diferente da métrica dos períodos Barroco e Clássico na qual as outras obras foram escritas. Em segundo lugar, é uma obra monofônica, ou seja, uma obra escrita unicamente para um instrumento melódico, e que, em outras palavras, não possui a informação harmônica necessária para muitas das regras do componente Estrutura Métrica[10]. Estes obstáculos, entretanto, não se mostram como intransponíveis, não obstante sendo o primeiro deles de natureza mais complexa.

Inicialmente, trata-se de poder realizar no componente Estrutura Métrica, tanto uma ampliação do conjunto de regras (de Boa–Formatividade e Preferenciais) quanto uma reformulação das que já existem de modo a permitir a percepção de métricas não presentes na Música Clássica Ocidental. Trata-se, portanto, de uma reformulação da teoria de Lerdahl & Jackendoff (1996) visando a incorporação de métricas não-européias, tais como as empregadas na música popular e na música contemporânea em geral[11].

O segundo dos obstáculos citados, ou seja, a ausência de informação harmônica, pode

[10]Na realidade, somente o componente Estrutura de Agrupamento é capaz de prescindir de informações harmônicas. Para os demais componentes sua falta traduz-se em resultados incompletos ou equivocados.

[11]Trata-se de um processo semelhante e paralelo àquele de permitir aos componentes Redução Temporal e, principalmente, Redução Prolongacional a inclusão e percepção de harmonias diversas àquelas empregadas na Música Clássica Ocidental. Em outras palavras, isto traduz-se como uma ampliação dos dois conjuntos de regras (de Boa–Formatividade e Preferenciais) e também em uma ampliação da base de dados de acordes e funções possíveis (classificados).

ser ultrapassado através de um procedimento de harmonização automático para melodias escritas dentro do sistema tonal. Neste trabalho, um passo neste sentido já foi dado através da implementação de um algoritmo capaz de detectar, em uma melodia tonal, a seqüência de tonalidades que a compõem. Uma descrição deste algoritmo pode ser encontrada no apêndice D, página 217.

Ao lado das duas obras já citadas, nesta seção de análises está também incluído o coral nº 52 da *Paixão Segundo São João*, de J.S. Bach. Na medida em que, de modo diverso das outras duas obras, esta última não foi abordada em uma seção ou capítulo anteriores, antes de apresentar sua Estrutura Métrica são mostradas as demais análises das quais depende o entendimento da métrica, nominalmente, sua Estrutura de Agrupamento (figura 5.15) e sua análise harmônica (figura 5.16), esta acompanhada do seu prévio pré-processamento. A seguir, nas figuras 5.17 a 5.19, estão as obras analisadas e suas respectivas estruturas métricas.

5.8 Discussão dos Resultados

5.8.1 Testes de Validação

Primeiro Tema do 1º Movimento da Sinfonia nº 40, de W.A. Mozart

Este tema apresenta-se sob duas formas: uma unicamente melódica e outra acompanhada de sua redução orquestral, as quais podem ser vistas, respectivamente, nas figuras 5.11 e 5.13 (páginas 89 e 89).

Tratando inicialmente da primeira amostra, ao examinar-se a estrutura métrica resultante, nota-se que, enquanto o segundo nível (de cima para baixo), segue exemplarmente a regularidade binária do trecho, no terceiro acontece uma periodicidade ternária, em princípio estranha ao caráter do trecho. Entretanto, ao observar-se sua atuação conjunta com o quarto nível, o qual volta a ser binário, percebe-se claramente os grandes pontos de articulação do trecho, a saber, o primeiro tempo do primeiro compasso, o segundo tempo do segundo compasso e o primeiro tempo do quarto compasso.

No caso da amostra com redução orquestral, a regularidade binária é alcançada em todos os níveis, resultando em uma análise academicamente correta, mas carente de uma orientação mais voltada ao fraseado, bastando para alicerçar esta informação o fato de que o segundo tempo do segundo compasso, nesta análise, perdeu o acento que foi indicado na análise anterior e que é importante estruturalmente no trecho em estudo. Não obstante, o sentido de quadratura presente nesta análise também pode ser considerado correto e, sem dúvida, foi alcançado devido a harmonia do trecho ter sido levada em consideração.

Primeiro Tema do 4º Movimento da Sinfonia nº 40, de W.A. Mozart

De modo semelhante ao tratamento anterior, este tema apresenta-se sob duas formas: uma unicamente melódica e outra acompanhada de sua redução orquestral, as quais podem ser vistas, respectivamente, nas figuras 5.12 e 5.14 (páginas 89 e 90).

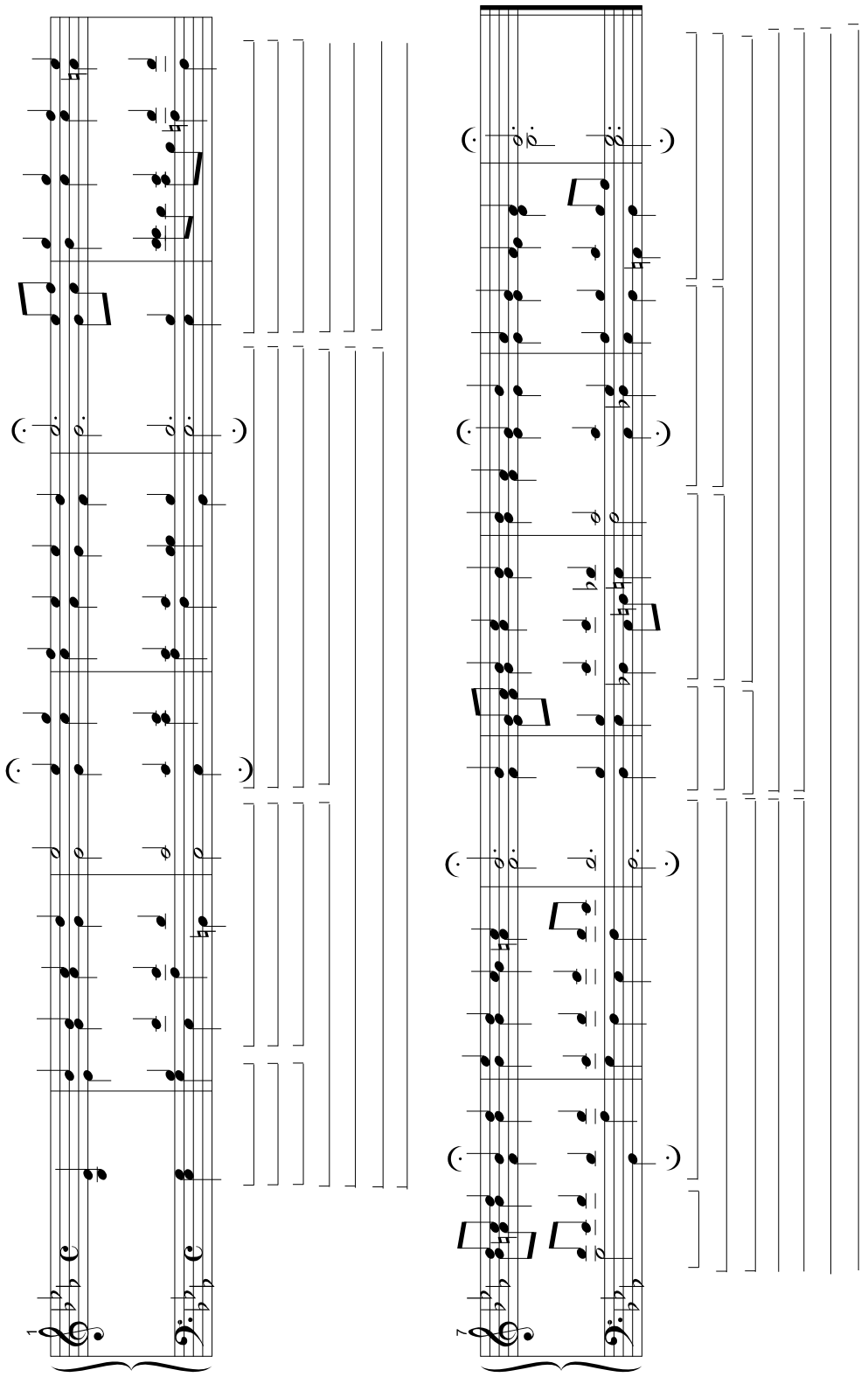

Figura 5.15: Texto integral e Estrutura de Agrupamento do Coral nº 52 da *Paixão Segundo São João*, de J.S. Bach.

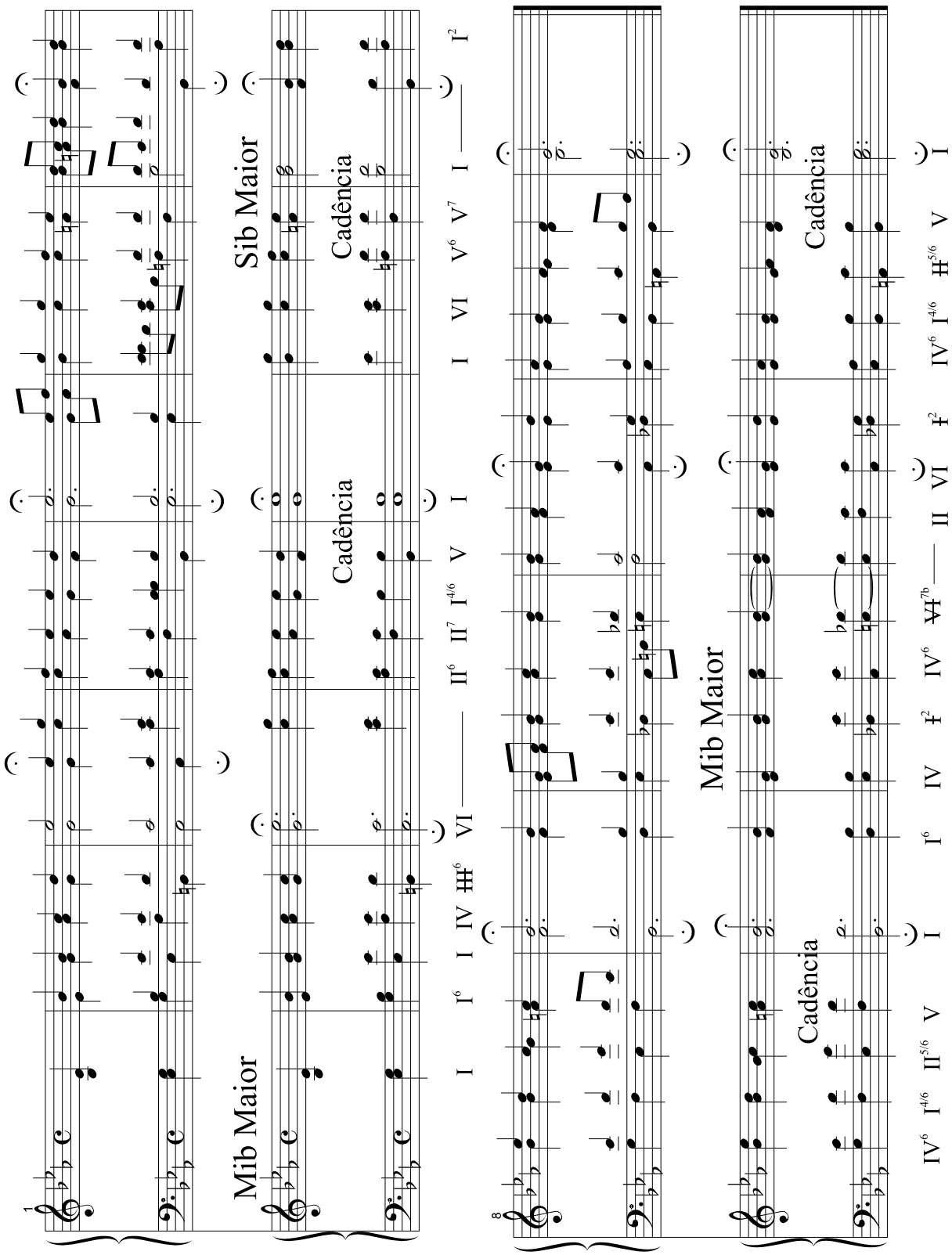

Figura 5.16: Pré-processamento e análise harmônica do Coral nº 52 da *Paixão Segundo São João*, de J.S. Bach.

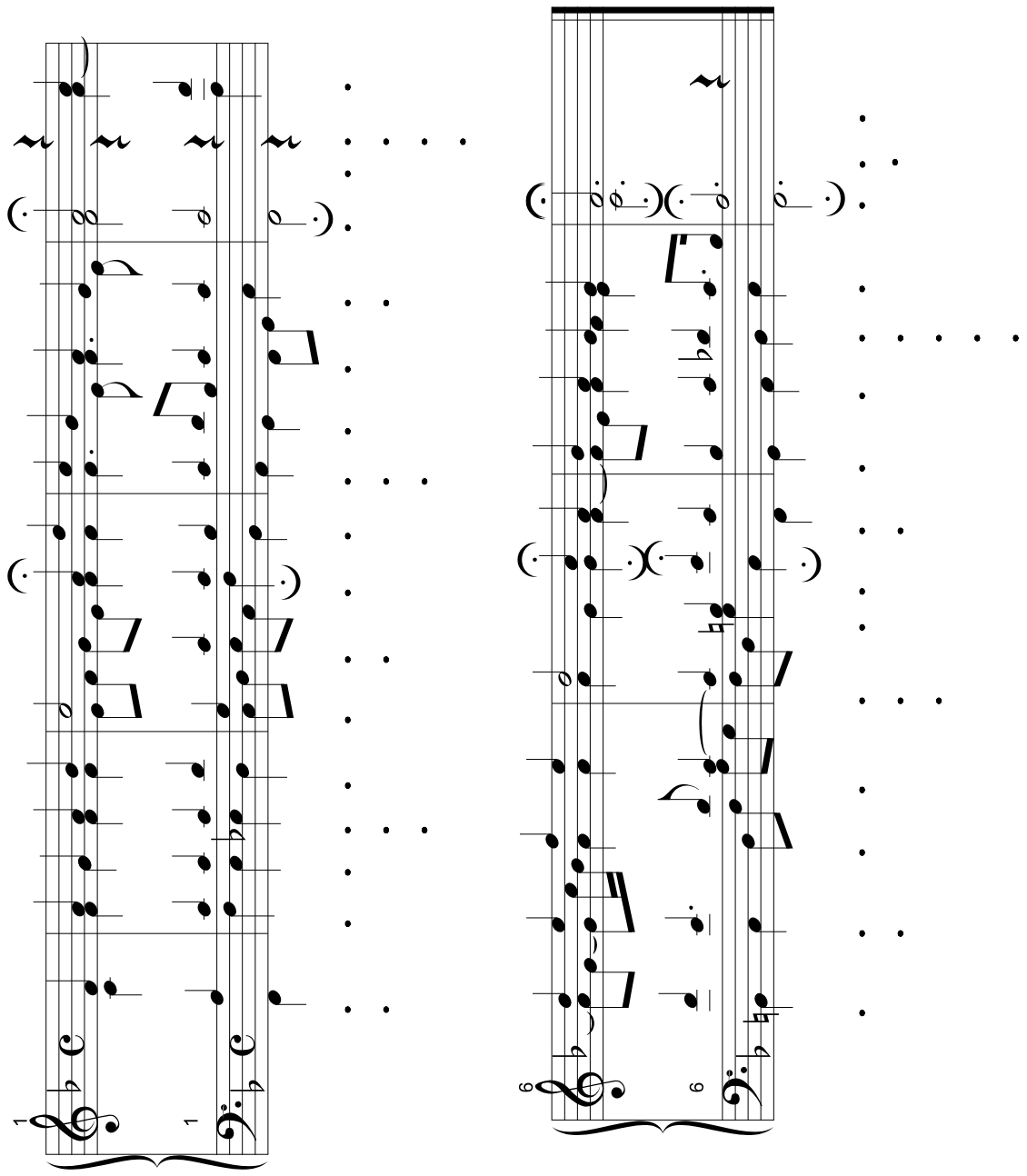

Figura 5.17: Estrutura Métrica do Coral *Christus, der ist mein Leben* (BWV 95) de J.S. Bach.

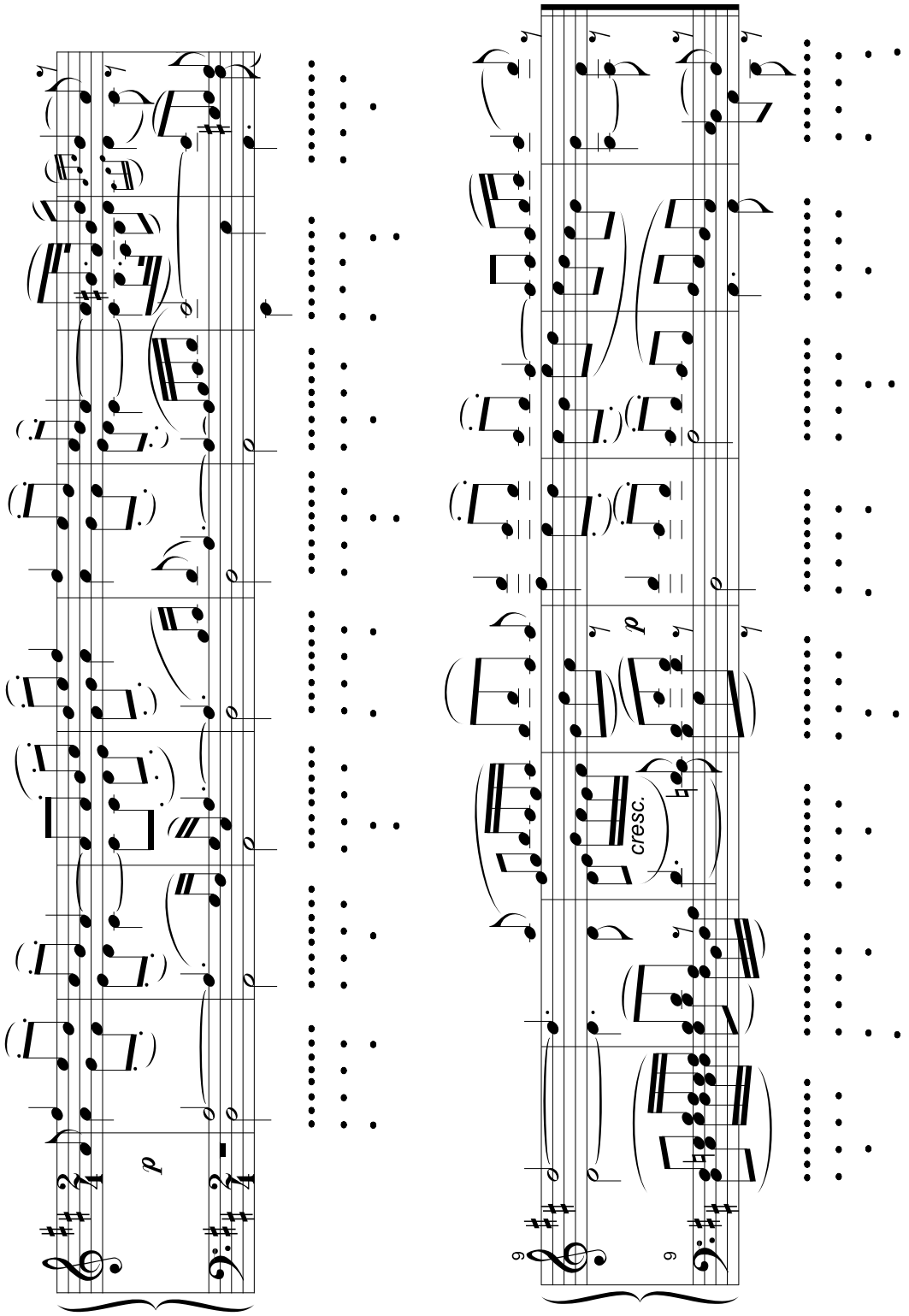

Figura 5.18: Estrutura Métrica do tema do 3º movimento do Quarteto de Cordas Op. 18 nº 5, de L.van Beethoven.

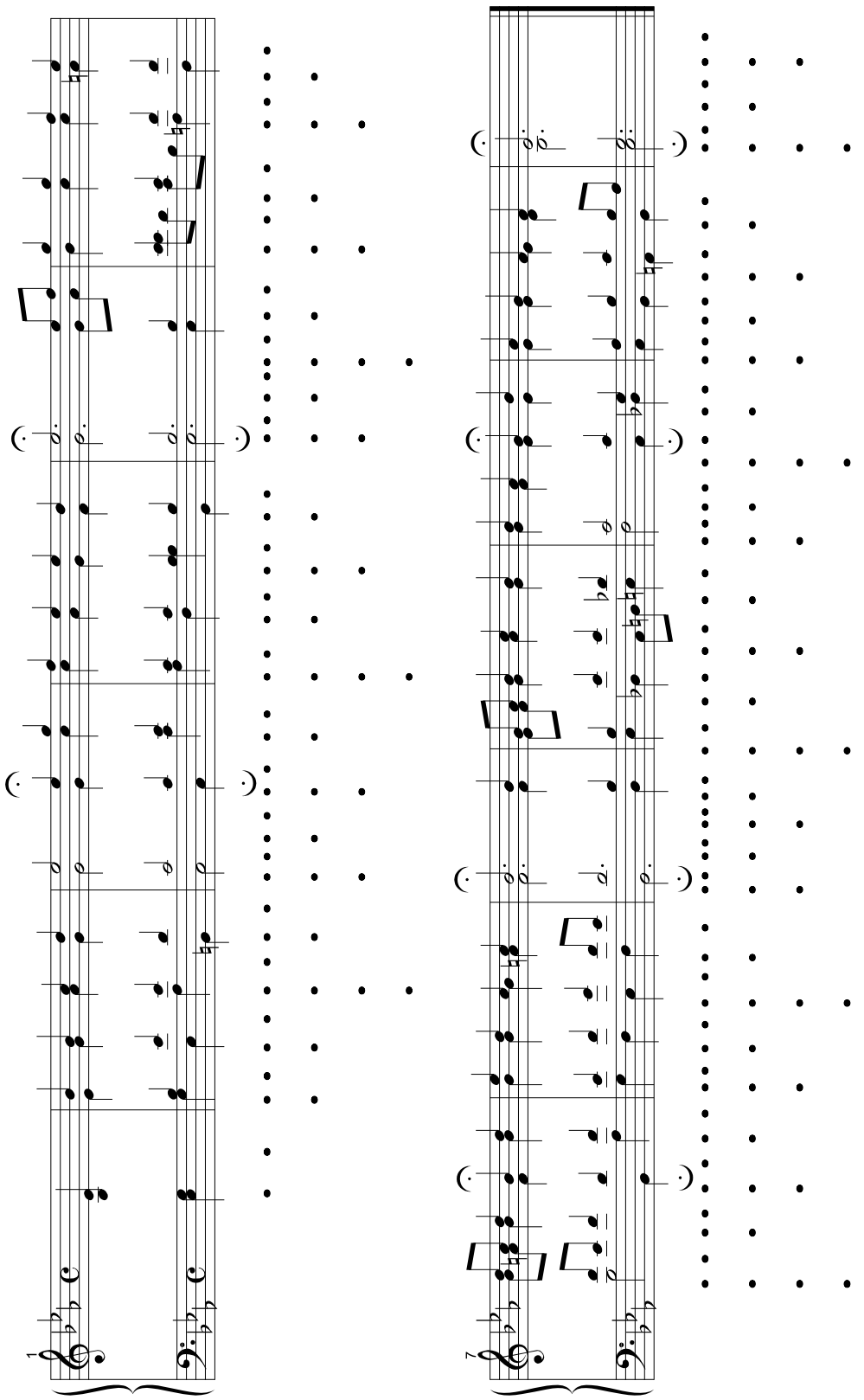

Figura 5.19: Estrutura Métrica do Coral nº 52 da *Paixão Segundo São João*, de J.S. Bach.

De modo diverso àquele encontrado na análise da primeira amostra melódica realizada anteriormente, neste caso apresenta-se claramente um erro resultante de uma defasagem, a partir do segundo nível, equivalente ao atraso de uma colcheia. Este atraso é devido à estimação errônea do *offset* do segundo nível, o que não é difícil de ser comprovado, bastando deslocar o conjunto formado pelo segundo, terceiro e quarto níveis de uma colcheia adiante. Assim, poder-se-á observar toda a regularidade métrica (binária) presente no trecho.

Abordando-se a amostra acompanhada da redução orquestral, observa-se que existe, já do primeiro para o segundo nível, uma intromissão de um metro ternário subjacente, o qual desaparece no terceiro, retornando no quarto nível. Uma possível causa para esta aparição do metro ternário pode ser devido à influência do terceiro grupo, do primeiro nível da estrutura de agrupamento, e sua continuação. Porém, antes de tudo, não se deve esquecer que o sistema, em sua análise, procede como se esta amostra fosse uma obra inteira. Em outras palavras, isto significa que, mais importante que os detalhes locais, a visão do todo deve ter um peso maior, exemplificado pelo fato, importante de ser notado, do último acorde da amostra coincidir com o ponto de maior acentuação métrica.

5.8.2 Análises

Coral nº 52 da Paixão Segundo São João, de J.S. Bach

Na medida em que as outras duas obras analisadas nesta seção já tiveram comentadas suas estruturas de agrupamento, seus pré-processamentos e suas análises harmônicas, para a presente obra serão realizados, antes daqueles sobre a estrutura métrica, comentários sobre estas facetas analíticas prévias.

Estrutura de Agrupamento. A estrutura de agrupamento do presente coral pode ser vista na figura 5.15 (página 92). A estrutura calculada pelo sistema mostra-se bastante razoável, sendo que em apenas três pontos mostra-se questionável e necessita uma discussão. Em primeiro lugar, no compasso 2, o sistema coloca a fronteira final do segundo grupo, primeiro nível, entre os dois mi♭ e não após o segundo, como esperado. Entretanto, o sistema não considera o segundo compasso como o local de uma cadência[12] e, por isso, a melhor segmentação foi selecionada. Em seguida, nos compassos 5 e 6, existe uma cadência, vista pelo sistema, mas que a estrutura de agrupamento não levou em consideração ao realizar a segmentação do trecho. Duas fronteiras aparecem como equivocadas do ponto de vista de um analista humano, a saber, a fronteira após o quarto tempo do compasso 5 e aquela entre o segundo e o terceiro tempos do compasso 6, pois ambas invadem o interior da cadência, fragmentando-a. Finalmente, e de modo semelhante ao primeiro caso, no compasso 9 existe uma cadência plagal que também, pelas

[12]No analisador harmônico implementado, uma cadência somente é detectada se a relação funcional é de *dominante–tônica*, se o baixo sobe uma quarta ou desce uma quinta (ambas justas) e se o soprano sobe um semitom desce um grau conjunto (tom ou semitom). Se alguma destas restrições não for satisfeita, o sistema rejeita o encadeamento como uma cadência.

mesmas razões expostas anteriormente, não foi vista pelo sistema e, conseqüentemente, segmentações alternativas foram empregadas.

Pré-Processamento e Análise Harmônica. O pré-processamento e a análise harmônica do coral sendo comentado podem ser vistos na figura 5.16 (página 93). Quanto ao pré-processamento, este realizou corretamente todas as reduções rítmicas necessárias para a realização da análise harmônica do trecho, sendo que apenas a expansão do último acorde do compasso 9, o qual invade o compasso 10, merece ser aqui discutida. Ao alcançar o compasso 10, durante o pré-processamento, o sistema encontra um acorde não-classificado. Os dois outros acordes que o cercam, entretanto, o são. Assim, o sistema opta por fazer o acorde da esquerda (por ser modulante e possuir mais peso) ocupar o tempo do não-classificado. No que tange à análise harmônica, essa foi também corretamente processada, detectando todas as funcionalidades dos acordes, cadências (com exceção das já discutidas) e modulações (com exceção daquela à tonalidade relativa, nos compassos 1 e 2, por não perceber o movimento cadencial).

Estrutura Métrica. A estrutura métrica deste coral é mostrada na figura 5.19 (página 96). Pode ser notado que o segundo e terceiro níveis mantém uma regularidade binária. O quarto nível, entretanto, introduz uma regularidade ternária (em mínimas), o que faz com que as articulações metricamente mais fortes fiquem se alternando entre o primeiro e o terceiro tempos do compasso. Essa alternância entre estes dois pontos do compasso pode ter sua origem na altenância das cadências entre estes mesmos dois momentos do compasso. Ainda que o sistema não tenha percebido todas as cadências, isto, possivelmente não o impediu de registrar esta regularidade ternária dentro da grade métrica.

Christus, der ist mein Leben (BWV 95), de J.S. Bach

Na figura 5.17 (página 94), é mostrada a estrutura métrica do coral. Nesta figura estão indicados somente os níveis cujo pulso é igual ou maior do que uma semínima. De modo diverso ao caso anterior, na grade métrica deste coral, já no segundo nível, existe uma regularidade ternária que contrasta com os demais níveis (terceiro, quarto e quinto), os quais são binários. A aparição de uma ternaridade tão próxima à superfície teve como resultado o atraso de uma mínima na grade métrica (dos três níveis mais baixos), o que é fácil de ver ao observar-se o penúltimo e o último compasso. Esta defasagem pode também ser a causa da escolha da altura *sol* como evento-cabeça, durante a Redução Temporal deste mesmo trecho, do último agrupamento cadencial, pois é exatamente sobre ele que fica situado o ponto de maior profundidade métrica[13].

Tema do 3º Movimento do Quarteto de Cordas Op. 18 nº 5, de L.van Beethoven)

A estrutura métrica do trecho é mostrada na figura 5.18 (página 95). Na figura estão indicados somente os níveis cujo pulso é igual ou maior do que uma semicolcheia. Dife-

[13]Ver na figura 6.6 (página 123) a Redução Temporal deste mesmo coral.

rentemente dos casos anteriores, onde foram geradas pelo sistema estruturas métricas de qualidade razoável, o uso desta amostra, muito possivelmente devido à sua complexidade rítmica, não apresentou resultados satisfatórios. Basta olhar que, já desde o segundo nível até o quarto todas as regularidades são ternárias. Uma das causas deste problema pode ser a já apontada complexidade da própria amostra, com seus constantes deslocamentos de acentos e síncopes. Outro fator pode ser a presença na amostra de um grande leque de valores de duração, dificultando o calculo da métrica subjacente.

Capítulo 6

Redução Temporal

6.1 Introdução

Nesta seção é apresentado um resumo do componente Redução Temporal. Assim como nos casos anteriores, isto é realizado através da apresentação de um Sistema Analítico e de uma Gramática Formal. A estes seguem as exposições de seu Objetivo Específico, da Metodologia utilizada e sua respectiva Implementação. Finalmente, são apresentados os Resultados obtidos sob a forma de Testes de Validação e Análises de obras. Para uma abordagem mais detalhada da matéria, consultar o capítulo 2 de Carvalho (2001) ou, preferencialmente, o texto original de Lerdahl & Jackendoff (1996).

6.1.1 Sistema Analítico

Para que seja feita uma análise reducional de um trecho de música é necessário que sejam considerados como dados tanto o sistema tonal, tal como é conhecido nos livros de teoria musical, quanto uma escala de estabilidade entre as configurações de altura que dele são derivadas. Isto permite uma abordagem das reduções como um fenômeno cujo fundamento está baseado no critério de relativa estabilidade entre alturas.

Entretanto, esta hipótese, apesar de necessária, não é suficiente, já que eventos-altura estáveis podem não ocupar dentro da frase posições que lhes confiram um relativo grau de importância. A expressão "ocupar posições dentro da frase" remete imediatamente à estrutura métrica e, através desta, aos *acentos estruturais*, que são critérios importantes na realização de uma Redução Temporal.

A Redução Temporal, portanto, repousa sobre uma integração apropriada de estabilidade de alturas com critérios rítmicos baseados nos componentes de agrupamento e métrico. Estes dois componentes permitem a segmentação de uma peça em domínios de elaboração de diversos níveis, formando uma hierarquia de intervalos-temporais. Nos níveis mais locais, o componente métrico particiona a peça em pulsos de intervalos-temporais iguais; nos níveis maiores, o componente de agrupamento divide a peça em motivos, frases, períodos, seções e assim por diante. A união analítica dos dois componentes permite que seja realizada na peça uma segmentação por intervalos-temporais, as-

sumindo os componentes uma dupla função na construção da redução. Depois de realizada a segmentação da peça em intervalos-temporais, é escolhido, para cada nível, um evento mais importante. Este evento é chamado de *evento-cabeça* do nível em questão. Após determinadas todas os eventos-cabeça dos intervalos-temporais de um determinado nível, estas formam um *nível reducional*, que é a seqüência dos eventos-cabeça dos intervalos-temporais naquele nível.

Na medida em que nem a estabilidade de alturas nem segmentação métrica podem, isoladamente, realizar a segmentação temporal, é possível concluir-se:

1. A segmentação com uso exclusivo do metro somente funciona para passagens em fase, o que não é o caso da maioria das músicas tonais clássicas;

2. A percepção do metro esvanece nos níveis maiores (de maior duração), enquanto que percepção de grupos é muito mais persistente;

3. O metro, tendo necessidade de ser periódico, não permite a segmentação de irregularidades, enquanto que os grupos, não possuindo a restrição da periodicidade, não apresentam aquela dificuldade.

Os eventos mais importantes de uma frase são seus *acentos estruturais*. Os acentos estruturais dividem-se em dois tipos: o *início estrutural* da frase e sua *cadência*. As cadências, por dois motivos, são bem mais difíceis de serem tratadas do que os inícios estruturais:

1. Tanto a meia-cadência quanto a cadência interrompida podem não emergir como estruturalmente importantes nas frases se analisadas puramente sob a ótica da estabilidade de alturas;

2. As cadências perfeita e interrompida possuem dois membros que são sentidos e percebidos como uma unidade.

As cadências, portanto, devem ser tratadas como signos ou fórmulas convencionais. Como na música tonal clássica o repertório de signos cadenciais é pequeno, estes podem ser facilmente etiquetados.

Após o tratamento das cadências como um signo complexo de dois eventos, os acentos estruturais podem, então, ser considerados como os eventos-cabeça do nível imediatamente anterior ao nível de frase e serem "reduzidos", daí por diante, a níveis cada vez maiores, até alcançar os acentos estruturais da peça, ou seja, seu início estrutural e sua cadência final.

6.1.2 Gramática Formal

Antes da apresentação das Regras de Boa–Formatividade referentes à Redução Temporal, é necessário que sejam formuladas duas regras acerca da segmentação de um trecho musical em intervalos-temporais e mais um pequeno conjunto de definições que possibilitem uma formalização mais refinada das primeiras.

A primeira das regras de segmentação opera nos níveis maiores da redução enquanto que a segunda nos níveis intermediários e menores.

Regra de Segmentação 1. Todo grupo numa peça é um na segmentação temporal da peça.

Regra de Segmentação 2. Na estrutura subjacente de agrupamento:

(a) cada pulso P do menor nível métrico determina um T_P estendendo-se de P até o próximo pulso do menor nível (mas não incluindo este);

(b) cada pulso P do nível métrico L_i determina um *intervalo-temporal regular* T_P, o qual é a união (ou soma) dos intervalos-temporais de todos os pulsos de nível L_{i-1} (o nível menor seguinte) de P até (mas não incluindo)

(i) o próximo pulso P' de nível L_i ou

(ii) uma fronteira de grupo, ou

(iii) qualquer dos dois que venha antes; e

(c) se uma fronteira de grupo G intervém entre P e o pulso precedente do mesmo nível, P determina um *intervalo-temporal aumentado* T'_P, o qual é o intervalo de G até o fim do regular T_P.

Além das definições e observações já feitas anteriormente, restam algumas outras que são fundamentais para o perfeito enunciado das Regras de Boa–Formatividade da Redução Temporal.

Em primeiro lugar, deve ser feita a distinção, dentro de um , entre *eventos-cabeça* e eventos *subordinados*. As relações de subordinação, por sua vez, devem ser transitivas, o que, por um lado, torna possível uma hierarquia estritamente ordenada e, por outro, permite a notação em árvore dos diferentes níveis da redução. Em seguida, ainda em relação à subordinação entre eventos, deve ser definido o conceito de eventos *diretamente subordinados*.

Com exceção do ramo mais longo da árvore, cada ramo termina em outro ramo e a cada um deles está conectado um evento-altura. O evento e_1, conectado ao ramo r_1, é dito *diretamente subordinado* ao evento e_2, conectado ao ramo r_2, quando, em algum passo da Redução Temporal, o evento e_1 for eliminado em favor de e_2.

Finalmente, faz-se necessária a definição de *inclusão imediata*. Diz-se que o T_i contém imediatamente outro T_j se T_i contém T_j e se não existe um T_k tal que T_i contém T_k e T_k contém T_j. É possível, agora, a formulação das Regras de Boa–Formatividade da Redução Temporal.

Com base no que foi exposto, é possível apresentar uma versão preliminar das *Regras de Boa–Formatividade de Redução Temporal* (RBFRT):

RBFRT 1. Para todo T existe um evento e (ou uma seqüência de eventos e_1e_2) que é o evento-cabeça de T.

RBFRT 2. Se T não contém outros intervalos-temporais (isto é, se T está no menor nível de intervalos-temporais), então e_k é qualquer evento que ocorra em T.

RBFRT 3. Se T contém outros intervalos-temporais e $e_1, \cdots, e_n$ seus respectivos eventos-cabeça. Então:

(a) (Redução ordinária) O evento-cabeça de T pode ser um dos eventos $e_1, \cdots, e_n$.

(b) (Fusão) Se $e_1, \cdots, e_n$ não são separados por uma fronteira de grupo (condição da "localidade") o evento-cabeça de T pode ser a superposição de dois ou mais eventos $e_1, \cdots, e_n$.

(c) (Transformação) Se $e_1, \cdots, e_n$ não são separados por uma fronteira de grupo o evento-cabeça de T pode ser alguma combinação mutuamente consonante de alturas escolhidas dentre $e_1, \cdots, e_n$.

(d) (Retenção cadencial) O evento-cabeça de T pode ser uma cadência cujo final é e_n (o evento-cabeça de T_n, o último imediatamente contido em T) e cujo *penult*, se existe um, é o evento-cabeça de um precedendo imediatamente T_n, apesar de que não necessariamente no mesmo nível.

RBFRT 4. Se uma cadência de dois elementos é diretamente subordinada ao evento-cabeça e de um T, o final é diretamente subordinado a e e o *penult* é diretamente subordinado ao final.

As *Regras Preferenciais da Redução Temporal* (RPRT) dividem-se em três categorias:

1. As *regras locais*, que dizem respeito, exclusivamente, à estrutura rítmica e ao conteúdo de alturas dos eventos dentro do .

2. As *regras não-locais*, que colocam em jogo o conteúdo de alturas de outros intervalos-temporais (essencialmente através de condução de vozes e paralelismo).

3. As *regras de acento estrutural*, que referem-se à articulação de fronteiras de grupos.

As regras locais são em número de três, abordando questões de metro, harmonia e tessitura:

RPRT 1. (Posição Métrica) Dentre as possíveis escolhas para o evento-cabeça de um T, prefira aquela que está numa posição métrica relativamente forte.

RPRT 2. (Harmonia Local) Dentre as possíveis escolhas para o evento-cabeça de um T, prefira aquela que é

(a) relativamente consonante do ponto de vista intrínseco, e

(b) relativamente próxima em sua relação à tônica local.

RPRT 3. (Registros Extremos) Dentre as possíveis escolhas para o evento-cabeça de um T, prefira aquela que tem

(a) a altura melódica mais aguda, e

(b) o baixo mais grave.

As influências não-locais também compreendem três regras, abrangendo, desta vez, questões de paralelismo, estabilidade métrica e estabilidade prolongacional.

RPRT 4. (Paralelismo) Se dois ou mais intervalos-temporais podem ser interpretados como motivicamente e/ou ritmicamente paralelos, preferivelmente a eles devem ser atribuídos eventos-cabeça paralelos.

RPRT 5. (Estabilidade Métrica) Na escolha do evento-cabeça de um T, prefira aquela que resulte numa escolha de uma estrutura métrica mais estável.

RPRT 6. (Estabilidade Prolongacional) Na escolha do evento-cabeça de um T, prefira aquela que resulte numa escolha de uma redução prolongacional mais estável.

As regras de acento estrutural, tratam dos dois tipos mais importantes de acento estrutural, nominalmente, o início estrutural e a cadência (final estrutural).

RPRT 7. (Retenção Cadencial) Se as seguintes condições existirem para um T:

(i) Existe um evento ou uma seqüência de eventos e_1e_2 formando a progressão para uma cadência perfeita, uma cadência interrompida ou uma meia-cadência.

(ii) O último elemento desta progressão está no fim de T ou é prolongado até o fim de T.

(iii) Existe um grupo maior G contendo T para o qual a progressão pode funcionar como final estrutural.

classifique a progressão como uma cadência e prefira escolhê-la como evento-cabeça.

RPRT 8. (Início Estrutural) Se, para um T, existe um grupo maior G contendo T para o qual o evento-cabeça de T pode funcionar como início estrutural, então prefira como evento-cabeça de T um evento relativamente próximo ao início de T (bem como, por isso, ao início de G).

Finalmente, resta a apresentação de uma regra preferencial, de âmbito global, que garante a discursividade presente em toda música tonal.

RPRT 9. Na escolha do evento-cabeça de uma peça, prefira o final estrutural ao início estrutural.

6.2 Objetivo Específico

O objetivo específico desta seção consiste no emprego dos resultados obtidos no contexto das estruturas de agrupamento e métrica dentro daquele da Redução Temporal. A partir da integração desta com os dois componentes anteriores, e de seu posterior processamento, seus resultados deverão ser aproveitados como elementos de entrada para o quarto e último componente, a Redução Prolongacional.

6.3 Metodologia

De modo diverso aos dois outros componentes já descritos da TGMT, este componente não possui um conjunto de ferramentas empregadas para o processamento de suas regras. Ao contrário, para este processamento são aproveitados resultados previamente calculados pelos demais componentes e, a partir desses, chega-se aos resultados do componente atual.

A única exceção a este cenário é a exposição neste texto (e utilização no sistema) da formalização do espaço tonal de alturas realizada por Lerdahl (2001), a qual é utilizada também pela Redução Prolongacional.

A metodologia para este capítulo, portanto, consistirá da exposição da teoria do espaço de alturas, sendo seguida da descrição da implementação do componente e de suas Regras Preferenciais. Finalmente, são apresentados os Resultados encontrados através de Testes de Validação e de Análises de trechos musicais reais.

6.4 Sobre o Espaço de Alturas

6.4.1 Generalidades

Para uma implementação fiel tanto da Redução Temporal quanto da Redução Prolongacional é importante uma eficiente representação do espaço de alturas visando sua distribuição no interior de uma hierarquia tonal.

A representação de alturas empregada neste trabalho é baseada naquela de Lerdahl (2001), a qual emprega três principais níveis hierárquicos, a saber, os níveis de altura, de acordes e de regiões.

Na medida em que diversas operações são realizadas sobre estes dados (alturas, acordes e regiões) pode-se pensar em todo o sistema como um conjunto de classes que, ordenadas, podem representar a estrutura tonal de uma peça.

Antes de explicitar mais detalhadamente cada um dos níveis é importante definir o chamado *espaço básico*, o qual também possui uma estrutura hierárquica, mostrando cada um de seus níveis internos.

O espaço básico, ponto de partida para o cálculo de todos os demais espaços empregados neste trabalho, possui cinco níveis:

1. Nível de oitava,

2. Nível de quinta,
3. Nível triádico,
4. Nível diatônico, e
5. Nível cromático.

Os elementos que estão ativados em cada nível podem ser representados por uma matriz 5x12 em que as linhas representam cada um dos níveis e as colunas as 12 classes de alturas. No tonalidade de Dó maior, isto assume a seguinte forma:

```
0
0                 7
0         4       7
0    2    4  5    7    9      11
0  1  2  3  4  5  6  7  8  9  10  11
```

na qual se pode ver que a classe de altura 0, ou seja, Dó, é única ativada em todos os níveis hierárquicos (primeira coluna). Nas linhas primeira, segunda e terceira, respectivamente os níveis de oitava, de quinta e triádico, cujo acorde é o correspondente ao I grau da tonalidade, ou seja, o acorde Dó–Mi–Sol. Na quarta linha, o nível diatônico, está representada uma região através de sua escala e, na quinta linha está representado o nível cromático.

Dois conceitos importantes neste tipo de representação do espaço de alturas são aqueles de *passo* e de *salto*. Define-se como passo o movimento de um elemento de um espaço para outro que lhe seja contíguo, *em um mesmo nível hierárquico*. Por exemplo, podem ser considerados como equivalentes, em termos de dimensão, sete passos no nível cromático, quatro passos no nível diatônico, dois passos no nível triádico e um passo no nível de quinta. De modo complementar, realiza-se um salto quando o movimento se faz *em um mesmo nível hierárquico*, de um elemento do espaço para outro que não lhe seja contíguo. Um salto acontece, portanto, quando, em um certo nível hierárquico do espaço, se realiza de uma só vez um movimento de dois ou mais passos. Desta forma, por exemplo, partindo-se da classe de altura 0, um passo no nível de quinta corresponde a saltos nos níveis mais baixos, correspondentes a dois passos para o nível triádico, quatro para o nível diatônico e sete para o nível cromático.

6.4.2 Transformações Realizadas Sobre o Espaço Básico

Tal como definido, o espaço básico representa a tonalidade de Dó maior, incluindo sua quinta, seu acorde de tônica, sua escala diatônica e suas possibilidades cromáticas. Entretanto, para que seja utilizado numa representação musical válida, tal espaço deve sofrer transformações visando sua maior generalidade. Estas transformações incluem:

1. Geração das tríades de uma região, e
2. Geração das demais regiões.

Geração das tríades de uma região

Na medida em que o espaço básico somente fornece o primeiro grau (tônica) de uma região, faz-se necessária uma transformação visando obter os demais graus. Na medida que esta transformação deverá ser feita dentro de uma única região, ela não será isomorfa, pois, neste caso, ela conduziria fatalmente a resultados que seriam alheios aos desejados, já que o nível diatônico do espaço em questão tem uma partição não–periódica. A esta transformação dá-se o nome de *transposição diatônica* e pode ser descrita assim:

(a) Calcula-se a distância diatônica d do I grau até o grau que se deseja representar.

(b) A cada um dos elementos dos níveis 1, 2 e 3 do espaço básico soma-se d passos (mod 7).

Como exemplo, mostra-se abaixo o resultado do algoritmo acima para o cálculo do III grau em Dó maior:

```
          4
          4                   11
          4       7           11
0     2   4 5     7     9     11
0 1 2 3 4 5 6 7 8 9 10 11
```

Nota-se que não houve alterações no quinto nível (cromático) nem no quarto (diatônico), mas somente naqueles níveis onde é definido o acorde.

Geração das demais regiões

A transformação do espaço básico de uma região naquele de outra é muito mais simples do que a geração interna de seus acordes, já que a primeira opera no nível cromático (transposição cromática). Assim:

(a) Calcula-se a distância cromática d da fundamental do espaço de partida (nível de oitava) até fundamental do espaço de destino (mod 12),

(b) Para cada um dos elementos dos níveis primeiro até o quarto do espaço de partida soma-se d passos (mod 12),

Como exemplo, mostra-se abaixo o resultado de uma transformação até a região de Mi maior. Considerando-se a região de partida como sendo Dó maior, calcula-se a distância cromática entre a fundamental da região de Dó maior (classe de altura 0) e aquela da região de Mi maior (classe de altura 4). Às posições dos níveis primeiro até quarto do espaço em Dó maior soma-se o valor encontrado ($d = 4$) (mod 12), ativa-se as novas posições e desativa-se as anteriores. O resultado é o espaço básico da região de Mi maior.

```
          4
          4                  11
          4          8       11
  1    3  4    6     8  9     11
0 1  2  3  4  5  6  7  8  9  10 11
```

Nota-se que o único nível onde não houve alterações é o quinto (cromático), já que este é comum a todas as regiões, diferindo apenas na interpretação e grafia de seus elementos.

6.4.3 Tratamento do Modo Menor

Na música tradicional ocidental, o modo menor, diverso do maior, apresenta-se de formas variadas e não uniformes. Dentre estas possibilidades escalares as três mais conhecidas são as escalas *natural*, *harmônica* e *melódica*. Seguindo a lição de Schönberg (1978), a qual diz que, apesar do nome, a escala melódica é melhor adaptada a questões harmônicas, e rejeitando aquela de Lerdahl (2001), a qual prefere a utilização da escala harmônica[1], é possível a formalização do espaço menor de maneira coerente e sem dificuldades. Assim, para lá menor, tem-se

```
                    9
          4         9
0         4         9
0    2    4    6    8  9     11
0 1  2  3  4  5  6  7  8  9  10 11
```

como a escala ascendente e

```
                    9
          4         9
0         4         9
0    2    4  5    7    9     11
0 1  2  3  4  5  6  7  8  9  10 11
```

como escala descendente. Para armazenar tal tipo de espaço pode-se pensar numa matriz tridimensional na qual as duas primeiras dimensões definam um espaço como já mostrado e a terceira dimensão o tipo de escala empregado (ascendente ou descendente).

6.4.4 Operações Realizadas Sobre o Espaço Básico

Depois de construído o espaço de alturas é necessário que seja definido um conjunto de operações que atuem sobre ele e que possibilitem seu emprego eficiente como definido na Redução Temporal e na Redução Prolongacional. As operações são:

[1] Ainda que em seu texto tal escolha reflita na criação de um conjunto de restrições necessárias para que a escala seja utilizada. Outras reflexões deste autor sobre o assunto podem ser vistas em Schönberg (1977b, 1984, 1980, 1977a).

1. Distância entre duas alturas em uma mesma região,
2. Distância entre dois acordes (tríades) numa mesma região,
3. Distância entre dois acordes (tríades) pertencentes a regiões diferentes (generalização do caso anterior), e
4. Distância entre duas regiões.

Todas estas operações resumem-se em encontrar o caminho mais curto entre dois elementos. Do modo em que estão enumeradas acima, estão em ordem de complexidade, ordem na qual também serão apresentadas aqui.

Distância entre Duas Alturas em uma Mesma Região

Esta operação determina a distância entre qualquer altura ativa em qualquer nível do espaço e o elemento ativo no nível de oitava. Na medida em que, em alguns casos, podem existir várias rotas, é necessário um método para otimizar o processo e calcular o caminho mais curto entre as duas alturas.

O algoritmo[2] (recursivo) para realizar esta tarefa está a seguir:

1. Inicializa a variável $td = 0$.
2. Entre com o pc do qual se quer calcular a distância. Se $pc = 0$, retorna td. Senão, calcula a distância vertical do pc: pcv.
3. Verifica o vizinho da esquerda e faz $td = td + 1$.
4. Se o vizinho for $pc0$, faz $td = td + pcv$ e retorna o resultado. Senão, calcula a distância vertical do vizinho nv e faz $td = td + abs(pcv - nv)$ e volta recursivamente ao passo 2 com o td e nv.
5. Repete o mesmos procedimentos com o vizinho da direita de pc.
6. Compara os dois resultados (esquerda e direita) e armazena o menor valor.

Distância entre dois Acordes em uma Mesma Região

Como já foi mostrado acima, o cálculo da distância entre dois acordes divide-se entre aquele dedicado a dois acordes pertencentes a uma mesma região e aquele que se ocupa de dois acordes pertencentes a regiões diferentes. Inicialmente, devido à sua menor complexidade, será abordado o caso de acordes pertencentes a uma mesma região.

Dados dois espaços (duas matrizes) representando, em seu nível triádico, dois acordes diferentes mas em uma região, ou seja, com nível diatônico igual, o cálculo de sua distância

[2]É ilustrado aqui o algoritmo sob sua forma mais simples, ou seja, aquela que calcula a distância de qualquer altura até a altura Dó. A forma completa envolve mais um certo número de passos, porém funciona fundamentalmente do mesmo modo.

envolve dois passos primordiais, a saber, a determinação das fundamentais de cada um dos acordes e, em seguida, a sua distância no círculo diatônico de quintas[3]. Um algoritmo simples para a determinação das fundamentais pode ser:

1. Colocar o nível triádico em estado natural, e

2. Considerar a altura mais grave, que fica sendo a fundamental.

Depois de determinadas as fundamentais de cada um dos acordes, basta seguir a regra apresentada por Lerdahl (2001, página 55) para o cálculo da distância entre os dois acordes e que pode ser definida como

$$\delta(x \rightarrow y) = j + k$$

onde x representa o primeiro acorde, y representa o segundo, j é o número de aplicações da regra do círculo-de-quintas de acordes[4] (Lerdahl, 2001) necessária para desviar o primeiro acorde no segundo acorde e k é o número de classes-de-altura diferentes entre o espaço básico que suporta o primeiro acorde e o espaço básico que suporta o segundo acorde.

Distância entre dois acordes em Regiões Diferentes

Para a distância entre acordes pertencendo a regiões diferentes Lerdahl (2001) sugere a inclusão de uma terceira parcela na soma que caracteriza a regra mostrada na seção anterior. Assim, tem-se

$$\delta(x \rightarrow y) = i + j + k$$

onde j e k possuem o mesmo significado anterior, enquanto i é o número de aplicações da regra do círculo-de-quintas regional[5] (Lerdahl, 2001) necessária para desviar a coleção diatônica que suporta o primeiro acorde para aquela que suporta o segundo acorde.

Este algoritmo, por possuir caráter mais geral do que o anterior, pode, naturalmente, ser empregado também para o cálculo da distância entre dois acordes em uma mesma região, apresentando, entretanto, a desvantagem de caracterizar-se por um maior custo computacional.

---

[3]O círculo diatônico de quintas é construído baseado no nível diatônico do espaço considerado. Assim, em Dó maior, o círculo diatônico de quintas é composto pelas alturas dó–sol–ré–lá–mi–si–fá. O círculo diatônico de quintas é também chamado círculo-de-quintas de acordes.

[4]Esta regra diz que, tomando-se como base o nível diatônico (quarto nível), o desvio de um acorde em um outro que lhe é contígüo no círculo pode ser realizado movendo-se todas as classes-de-alturas do primeiro ao terceiro nível quatro passos à direita (ascendente no círculo) ou quatro passos à esquerda (desdendente no círculo), sendo ambas as operações realizadas em módulo 7.

[5]Esta regra diz que, tomando-se como base o nível cromático (quinto nível), o desvio de uma região em uma outra que lhe é contígüa no círculo pode ser realizado movendo-se as classes-de-alturas do quarto nível sete passos à direita (ascendente no círculo) ou sete passos à esquerda (desdendente no círculo), sendo ambas as operações realizadas em módulo 12.

Distância entre duas Regiões

Esta operação, tal como descrita por Lerdahl (2001), é um procedimento intrincado no qual tonalidades *pivot* (centros de referência tonal) são deslocados em meio a um mapa geral de tonalidades através de movimentos restritos e que permitem, assim, calcular a distância entre duas regiões. Trata-se de um procedimento geral e bastante complexo, muito além, sob este aspecto, daquele necessário para este trabalho. Assim, é proposta aqui uma alternativa simples e que se constitui em calcular a distância entre duas regiões como equivalente à distância entre os dois acordes das tônicas destas regiões. Desta forma, o problema reduz-se a calcular a distância entre dois acordes pertencendo a duas regiões diferentes (já visto na seção anterior).

6.5 Implementação da Redução Temporal

Na sua implementação, a Redução Temporal, tal como já foi mostrado no capítulo 3, página 9, emprega dados que são gerados pelos outros três componentes. Nisto, não difere dos demais. Não obstante, possui duas diferenças em relação à implementação dos dois componentes da teoria mostradas até agora.

Inicialmente, as Regras de Boa–Formatividade da Redução Temporal, ao invés de possuírem um caráter de *checkers*, são integradas no próprio projeto das estruturas de dados utilizadas[6].

Em segundo lugar, as Regras Preferenciais da Redução Temporal não possuem uma ferramenta exclusiva para seu cálculo e sim operam a partir dos dados resultantes dos outros três componentes.

Sendo assim, inicialmente explicar-se-á a operação do algoritmo principal e depois, separadamente, cada uma das Regras Preferenciais do componente.

6.5.1 Algoritmo Principal

O algoritmo principal da Redução Temporal possui a tarefa de calcular três resultados:

1. Retornar ao componente Estrutura Métrica um valor numérico que represente o conflito interno da Redução Temporal para uma determinada Estrutura Métrica. Isto é realizado através de memória compartilhada e o processo de cálculo do erro (conflito) será mostrado adiante.

2. Calcular e retornar como resultado, a partir de uma população de estruturas métricas, aquela que resultar em menor conflito para a Redução Temporal. A população de estruturas métricas é aquela calculada pelo componente Estrutura Métrica.

[6]Isto, de certa forma, já havia sido feito durante a implementação da Estrutura de Agrupamento e da Estrutura Métrica. Entretanto, nestes dois casos, a própria arquitetura interna dos componentes não permitia a nova e simplificadora abordagem proposta aqui.

3. Retornar os eventos-cabeça para cada e uma representação clara da hierarquia e dependência entre os demais eventos, em cada nível de redução. Estes dados compõem o resultado final da Redução Temporal.

Considerando que tais resultados devem ser calculados simultaneamente, ao chamar-se o programa do componente três processos concorrentes e de mesmo nome surgem no ambiente sendo utilizado. A seguir, mostra-se o algoritmo principal e explica-se-lhe os detalhes.

O algoritmo principal baseia-se no fato de que todo grupo é um . Assim, é possível (e, na prática, necessário) basear-se na estrutura de agrupamento já calculada para servir como referência dos níveis de redução e da segmentação interna de cada nível. Segue o algoritmo:

1. Inicializar as variáveis $C = 0$ e $N = 0$.
2. Para cada nível da estrutura de agrupamento:
   (a) Para cada grupo do nível atual:
      i. Calcular o número n de eventos do grupo atual;
      ii. A partir da quantidade encontrada no item anterior alocar três vetores **w** (vetor de pesos), **idx** (vetor de índices) e **c** (vetor de eventos candidatos e evento-cabeça);
      iii. Copiar os eventos do grupo atual para **c**, inicializar **w** com zeros e **idx** com qualquer valor menor do que zero;
      iv. Percorrer cada uma das Regras Preferenciais selecionando, para cada uma delas, um evento-cabeça que esteja dentro do conjunto de eventos **c**. A cada evento escolhido, incrementar o vetor **w**, na mesma posição, de um valor real positivo qualquer;
      v. Calcular o número eq de elementos iguais em **w**.
      vi. Calcular o conflito fazendo $C = C + 1 - \frac{n-eq}{n}$ e $N \leftarrow N + 1$;
      vii. Armazenar o evento-cabeça do grupo.
      viii. Se for o $3^{\underline{o}}$ passo, ou seja, se for o momento de cálculo do resultado final da Redução Temporal:
         A. Calcular a posição m_p do valor máximo de **w**.
         B. Fazer o calculo das dependências entre os eventos e suas hierarquias através do algoritmo $CalcDepend(\mathbf{idx}, mp, \mathbf{w}, n)$;
         C. Armazenar o resultados, eventos-cabeça e dependência entre os eventos, em arquivo.
      ix. Liberar memória alocada para **w**, **idx** e **c**.
3. Retorna o conflito resultante $C = \frac{C}{N}$.

Algoritmo de Cálculo de Dependências

Um elemento importante do algoritmo que se irá agora apresentar é o processo no qual se estabelece a dependência e a hierarquia entre os eventos de um conjunto de candidatos e o evento-cabeça.

Antes de seguir e mostrar o algoritmo de cálculo das dependências e hierarquias dentro de um conjunto de eventos candidatos a evento-cabeça, é necessário mostrar como é realizada a representação de ambas.

O processo é ilustrado através de um exemplo. Suponhamos que o conjunto de candidatos a evento-cabeça seja constituído de cinco eventos e que, após o processamento de todas as Regras Preferenciais tenha-se chegado ao seguinte resultado para o vetor **w**:

```
[4 5 1 2 3]
```

o que diz que o primeiro elemento do conjunto de candidatos **c** foi escolhido por quatro regras, o segundo por cinco regras, o terceiro por uma regra, o quarto por duas regras e o último por três regras. Duas características saltam aos olhos neste resultado. Inicialmente, torna-se claro que o evento-cabeça é o segundo elemento de **c**, por possuir em **w** a mesma posição e o maior valor. Em segundo lugar, nota-se que a soma de todas as posições de **w** é quinze, mas o número de Regras Preferenciais na Redução Temporal é igual a nove. Isto demonstra que tanto uma mesma regra pode pontuar dois eventos diferentes quanto um único evento pode ser pontuado por todas as regras. Assim, os dois casos limite existentes são quando nenhum evento é pontuado (não existe evento-cabeça, o que é impossível por transgredir as Regras de Boa–Formatividade do componente) ou todos são pontuados por todas as regras (todos os eventos são eventos-cabeça, o que também transgride as Regras de Boa–Formatividade). Para solucionar este último problema e todos os demais casos de empate, em caso de escolha de dois eventos-cabeça utiliza-se um critério ordinal, ou seja, fica-se com o primeiro encontrado (o que é reforçado pela RPM 8). Resta agora verificar como tal representação pode ser transposta visualmente para uma árvore prolongacional.

Em primeiro lugar, a colocação do vetor **w** sob uma forma normalizada auxilia grandemente a visualização. Assim, tem-se:

```
[0.8 1.0 0.2 0.4 0.6]
```

no qual pode ver-se que a hierarquia entre os eventos w_i fica demonstrada através de uma escala no intervalo $0 \leq w_i \leq 1$, onde $0 \leq i < n$ e onde n é o número de eventos no vetor de candidatos (neste caso, cinco)[7].

[7]Uma outra representação possível é aquela em que os elementos do vetor representem a probabilidade do evento ocupando aquela posição ser um evento-cabeça. Esta representação poderia ser de grande relevância ao ter-se a intenção de gerar novos trechos a partir de análises já realizadas. Ainda que aqui não seja este o caso, é importante chamar a atenção para esta possibilidade já que um sistema complementar ao atual, porém em engenharia reversa, poderia ser projetado com o intuito de gerar novos textos musicais a partir de uma massa de dados previamente obtidos. Continuando com o exemplo dado anteriormente, obtém-se o vetor [0.27 0.33 0.07 0.13 0.20], o qual contém as probabilidades de cada evento de **c** ser um evento-cabeça. Vê-se claramente que o segundo elemento é o mais provável (33%) de ser selecionado como evento-cabeça para o atual.

Em segundo lugar, devem ser unidos os ramos da árvore baseando-se nos valores contidos em cada posição de cada evento. Será agora descrito o algoritmo em questão. O procedimento mais básico deste algoritmo é que valores menores são ramos (dependem) de valores maiores e sua única restrição é que ramos não devem se cruzar. Assim:

Algoritmo $CalcDepend(\mathbf{idx}, mp, \mathbf{w}, n)$.

1. Se o valor máximo não ocupa e primeira nem a última posição:
   (a) Percorrer **w** do início ao valor máximo marcando as dependências.
   (b) Percorrer **w** do final ao valor máximo marcando as dependências.
2. Senão, se o valor máximo ocupa a primeira posição:

   Percorrer **w** do final ao início marcando as dependências.
3. Senão, se o valor máximo ocupa a última posição:

   Percorrer **w** do início ao final marcando as dependências.

O que se denomina aqui como marcação de dependências é o preenchimento de uma lista indicando como os elementos de **c** dependem uns dos outros em função de seus respectivos pesos. O algoritmo retorna o vetor de índices **idx**, o qual contém a informação sobre qual é o evento-cabeça e quais são as dependências entre os eventos. No exemplo sendo dado, tem-se:

```
idx     ->[1 -1  3  4  1]
posição -> 0  1  2  3  4
```

o que deve ser lido como o primeiro evento dependendo do segundo, o segundo sendo o evento-cabeça, o terceiro evento dependendo do quarto, o quarto evento dependendo do quinto e este dependendo do segundo.

Desta forma, considerando-se ainda o exemplo dado, tem-se a seguinte hierarquia entre os eventos do conjunto **c** (a seta marcando a dependência de um evento em relação a outro):

- Percorrendo do início à posição do valor máximo:
  $e_0 \rightarrow e_1$
- Percorrendo do final à posição do valor máximo:
  $e_3 \rightarrow e_4$
  $e_2 \rightarrow e_3$
  $e_4 \rightarrow e_1$

o que, visualmente, corresponde à árvore reducional temporal mostrada na figura 6.1.

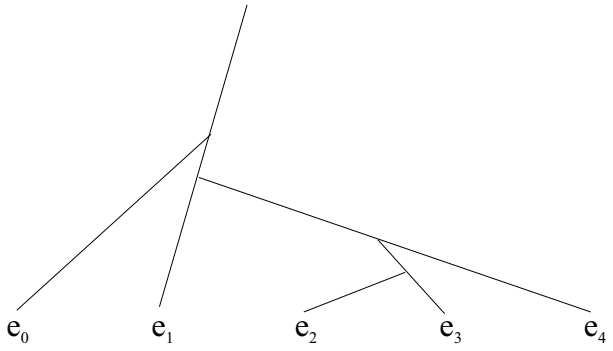

Figura 6.1: Exemplo de construção de uma árvore prolongacional temporal.

6.5.2 Implementação das Regras Preferenciais

Nesta seção estão descritas as implementações das RPRT. Como já foi anteriormente explicado, a função destas regras é selecionar, dentre um conjunto de candidatos a evento-cabeça de um , aquele que melhor se adapte às suas respectivas restrições. Assim, cada regra pontua o melhor candidato para um dado conjunto e, no final, este é selecionado como evento-cabeça do .

RPRT 1 – Posição Métrica

Nesta regra, a posição de cada candidato, para uma dada estrutura métrica, é comparada aos demais e é pontuado aquele candidato que possuir mais níveis métricos, ou seja, possuir uma posição métrica mais forte.

RPRT 2 – Harmonia Local

Esta regra é dividida em duas partes. A primeira delas seleciona e pontua os candidatos mais consonantes com a harmonia vigente no instante de sua aparição (em qual acorde o candidato está inserido) e a segunda seleciona e pontua os candidatos que possuem maior proximidade com a tonalidade vigente durante sua aparição. Assim, o que esta regra realmente realiza é a verificação de quais candidatos são partes integrantes do acorde vigente e quais deles fazem parte da coleção diatônica na qual estão inseridos. Obviamente não se aplica para trechos não-polifônicos.

RPRT 3 – Registros Extremos

Ainda que na regra original sejam consideradas as duas vozes extremas – maior altura da parte superior e menor altura da parte grave –, no próprio corpo do texto da teoria é dito pelos autores (Lerdahl & Jackendoff, 1996, página 162) que a utilização como regra preferencial tanto da parte aguda quanto da grave somente teria sentido se houvesse coincidência de ambas numa mesma posição métrica. Sendo assim e considerando que na música tradicional a altura mais aguda da parte superior tem importância formal

inquestionável, somente é considerada na implementação da regra a parte superior e sua altura mais aguda, a qual, nesta regra, é pontuada como candidata preferencial a evento-cabeça do atual.

RPRT 4 – Paralelismo

Como já foi dito em outra ocasião, no contexto da TGMT o conceito de paralelismo é utilizado de maneira extremamente flexível e variada. Isto, em outras palavras, significa que o conceito deve adaptar-se ao contexto do componente ao qual ele deve ser aplicado. Assim, na Estrutura de Agrupamento, pelo menos no caso do presente trabalho, "paralelismo" significa reiteração de motivos melódicos, enquanto que na Estrutura Métrica, significa reiteração de perfis de mesma densidade métrica. No caso da Redução Temporal, o termo paralelismo significa a atribuição de eventos-cabeça paralelos a intervalos-temporais que possuem a mesma segmentação motívica e/ou rítmica. Não obstante a Redução Temporal possuir acesso aos resultados da Estrutura de Agrupamento e àqueles da Estrutura Métrica, uma avaliação que, baseada nestas duas, tornasse perceptível para o sistema que dois intervalos-temporais comportam-se como eventos paralelos transcende as possibilidades deste mesmo sistema, já que alguns dados necessários para esta avaliação não são disponibilizados pelas ferramentas analíticas atualmente empregadas. Sendo assim, abriu-se mão da implementação desta regra, deixando-a para futuras melhorias no sistema.

RPRT 5 – Estabilidade Métrica

Esta regra é, de certa forma, o complemento da RPM 9, a qual, como foi visto, tem por objetivo a escolha de uma estrutura métrica que minimize os conflitos entre as Regras Preferenciais da Redução Temporal. Sendo assim, sua implementação coincide com a da Regra Preferencial de Métrica já citada, pois uma das conseqüências daquela implementação é também a seleção final de uma estrutura métrica mais adequada tanto ao componente Estrutura Métrica quanto ao componente Redução Temporal[8].

RPRT 6 – Estabilidade Prolongacional

Esta regra divide-se em duas partes, a saber, *Estabilidade Linear*[9] e *Progressão Harmônica*.

A primeira parte da regra pontua eventos candidatos que possuem melhor encadeamento linear com os intervalos-temporais adjacentes. Com este objetivo, é feita a quantificação, do ponto de vista da linearidade (quanto maior o intervalo menor a linearidade), de cada candidato com cada evento dos intervalos-temporais anterior e posterior. Faz-se depois a média e o candidato com melhor resultado é pontuado.

[8]Deve ser recordado aqui que uma das funções do módulo do sistema responsável pela Redução Temporal é dar como saída a estrutura métrica final do trecho sendo analisado (2º passo dos três descritos no texto sobre a implementação do componente).

[9]Não deve ser esquecido que o termo *linear*, em Música, diz respeito a conexões melódicas de pequenos intervalos, preferencialmente segundas e terças.

Quanto à segunda parte da regra, o procedimento utilizado é semelhante, porém agora o que é quantificado é o encadeamento harmônico do evento candidato com as harmonias (acordes) dos intervalos-temporais adjacentes. Sendo assim, tem-se que levar em conta não somente estas, mas também as harmonias presentes durante o tempo de vida de cada candidato, pois tal relação é que caracterizará o tipo de progressão harmônica presente entre o no qual está inserido o candidato e aquele que lhe é adjacente.

RPRT 7 – Retenção Cadencial

A implementação da RPM 7 é simples já que ela simplesmente pontua os eventos candidatos que fazem parte de uma cadência. Para isto, utiliza-se o mapa de cadências calculado pelo componente Redução Prolongacional e faz-se a verificação se o evento candidato está ou não inserido em uma cadência. Na medida em que vários eventos do

atual podem estar no interior de uma cadência, a RPM 7 é um exemplo típico de regra que pode, dentro de um mesmo , pontuar mais de um candidato a evento-cabeça.

RPRT 8 – Início Estrutural

Esta regra diz que, até o penúltimo nível, devem ser pontuados os candidatos mais próximos do Início Estrutural. Este define-se aqui como as localizações temporais onde possam existir estabilidade e que estejam o mais próximo possível do início do . Em outras palavras, trata-se de pontuar o evento candidato mais próximo do início do

e que também seja consonante com a harmonia que lhe sustenta.

RPRT 9 – Final Estrutural

Baseando-se na definição dada acima de Início Estrutural, define-se Final Estrutural como um intervalo temporal estável possível e mais próximo do final. O evento candidato que estiver imerso neste intervalo deve ser pontuado e ser preferencialmente escolhido em relação ao Início Estrutural. Como a RPM 9 diz respeito à escolha do evento-cabeça da peça, ou seja, o evento estruturalmente mais importante do trecho, sua ativação somente acontece no último nível de análise da Redução Temporal, definindo, portanto, o resultado final.

6.6 Resultados

6.6.1 Testes de Validação

Assim como foi realizado para o componente Estrutura Métrica são realizados, para a Redução Temporal Testes de Validação tendo como amostras os temas de Mozart já empregados no componente anterior. De modo semelhante ao que foi realizado antes, ambos os temas apresentam-se, inicialmente, como melodias sem acompanhamento e, depois, com suas respectivas reduções orquestrais. Os resultados podem ser vistos nas figuras 6.2 a 6.5.

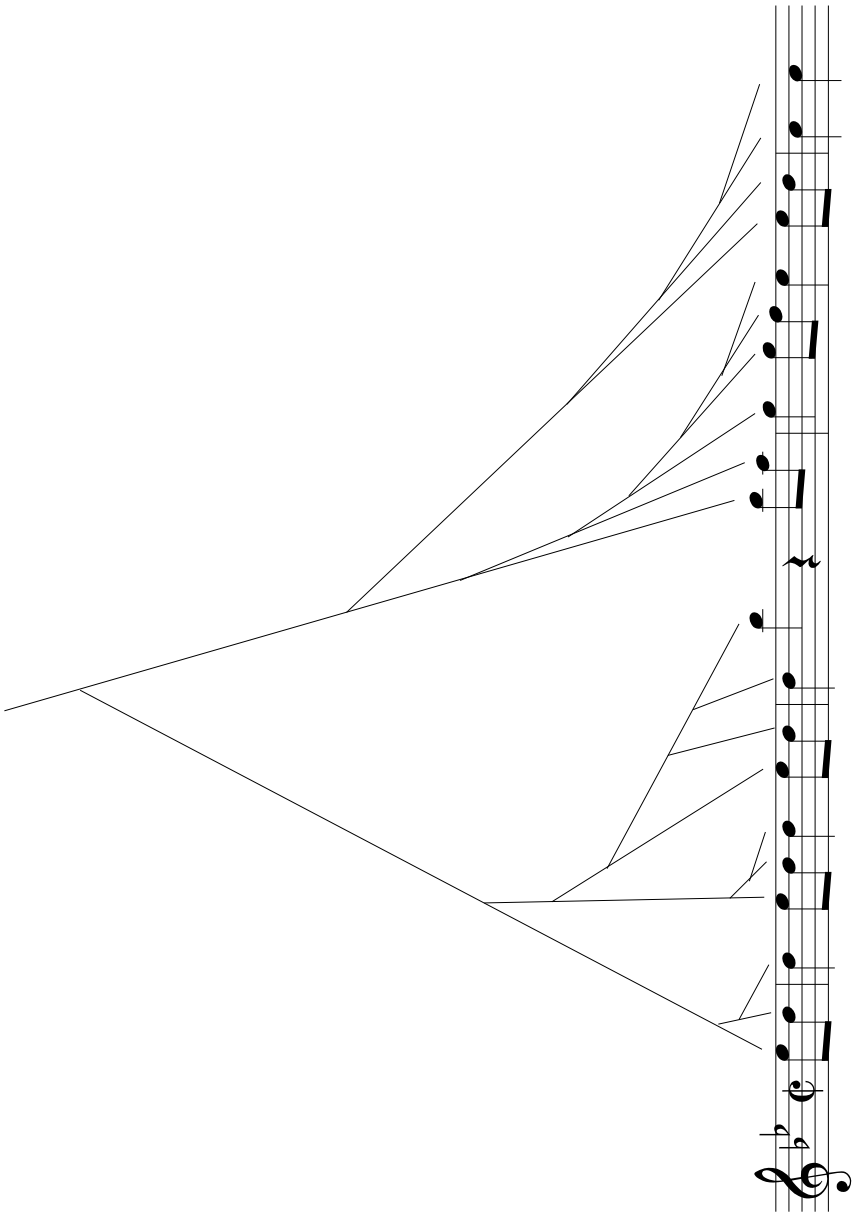

Figura 6.2: Redução Temporal para o primeiro tema do primeiro movimento da Sinfonia nº 40, de W.A. Mozart.

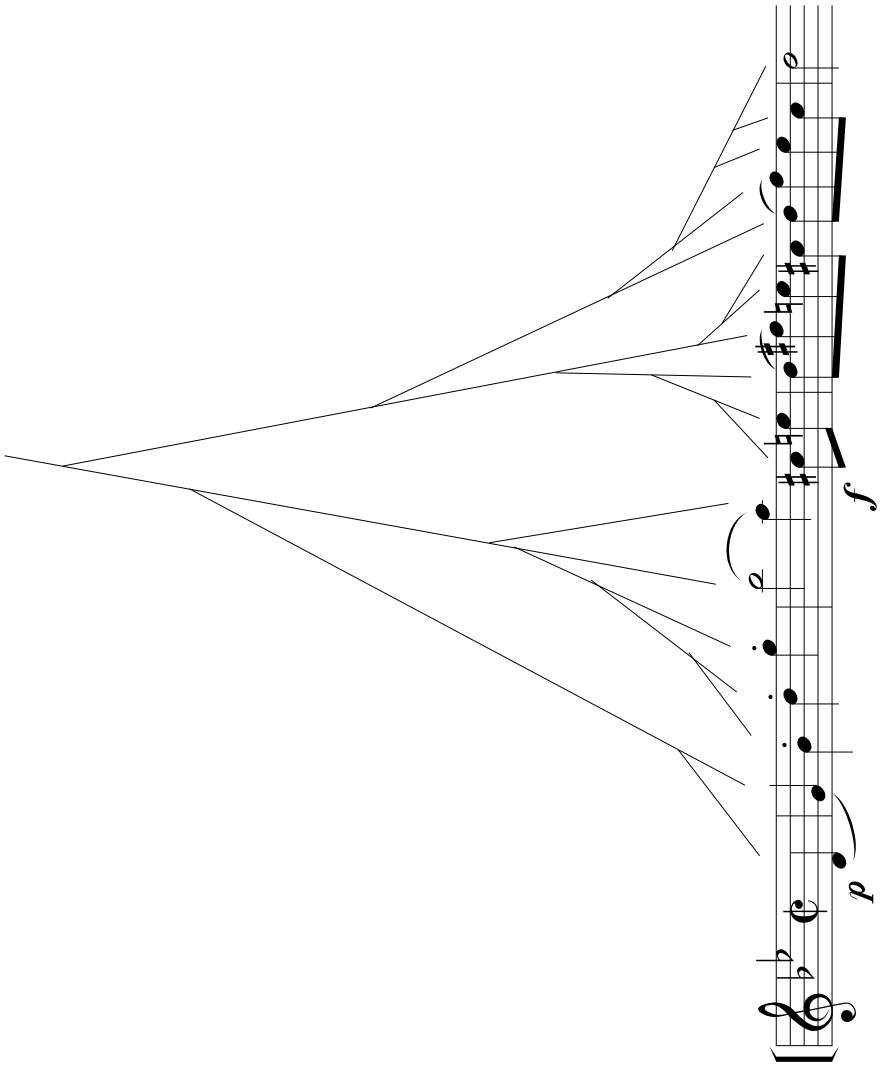

Figura 6.3: Redução Temporal para o primeiro tema do quarto movimento da Sinfonia nº 40, W.A. Mozart.

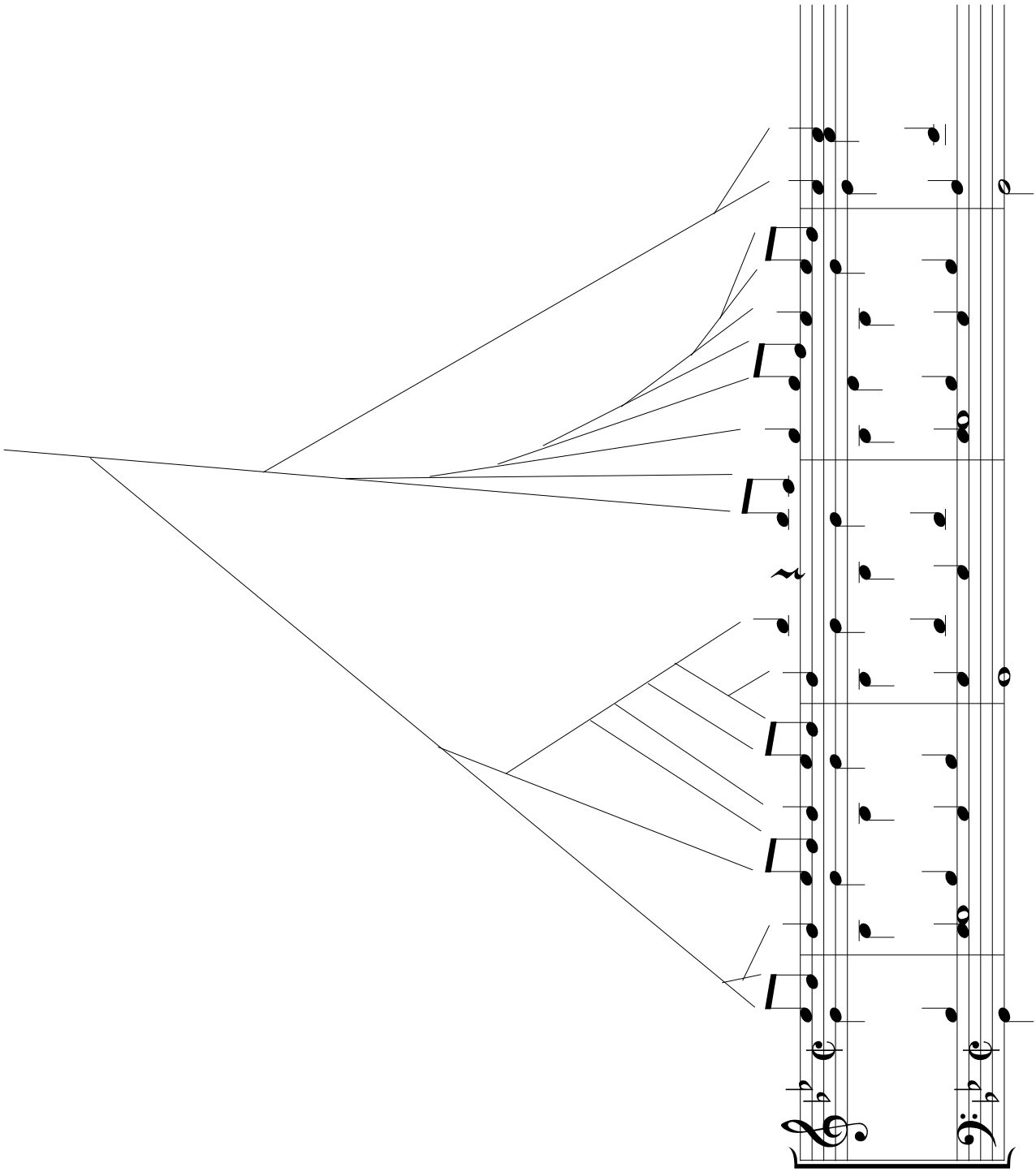

Figura 6.4: Redução Temporal para o primeiro tema do primeiro movimento da Sinfonia nº 40, W.A. Mozart (com redução orquestral).

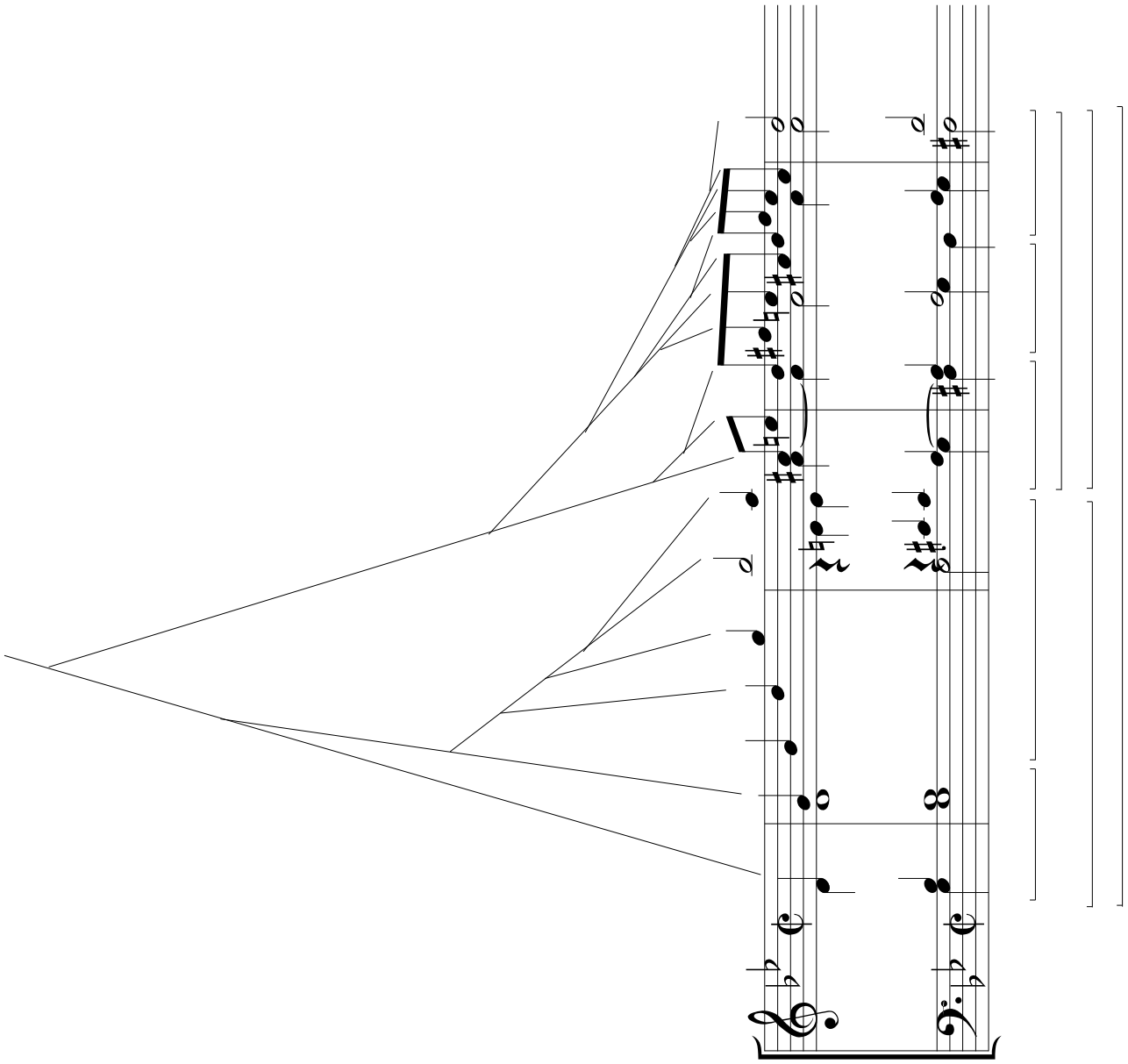

Figura 6.5: Redução Temporal para o primeiro tema do quarto movimento da Sinfonia nº 40, W.A. Mozart (com redução orquestral).

6.6.2 Análises

As obras analisadas são também as mesmas apresentadas no componente anterior, a Estrutura Métrica. É mantida também a mesma ordem de apresentação, ou seja, inicialmente o Coral *Christus, der ist mein Leben* (BWV 95), de J.S. Bach, seguido do tema do 3º movimento do Quarteto de Cordas Op. 18 nº 5, de Ludwig van Beethoven e, finalizando, com o Coral nº 52 da *Paixão Segundo São João*, de J.S. Bach. Tanto no Coral nº 52 quanto no tema do quarteto de Beethoven foram indicados também os níveis de redução, devido à sua maior complexidade.

6.7 Discussão dos Resultados

6.7.1 Testes de Validação

Primeiro Tema do 1º Movimento da Sinfonia nº 40, de W.A. Mozart

A Redução Temporal das formas melódica e com redução orquestral deste tema podem ser vistas, respectivamente, nas figuras 6.2 e 6.4 (páginas 118 e 120). É interessante notar que, nas duas análises, o evento- cabeça da peça foi o si♭ da segunda metade do segundo tempo do segundo compasso, ainda que, no segundo caso, questões harmônicas tenham sido levadas em conta. Interessante também é o fato de, mais uma vez em ambos os casos, a nota inicial mi♭ ser diretamente subordinada ao evento-cabeça da peça. Tal relação de quinta justa não é casual, mesmo levando-se em contas as relações harmônicas em jogo, ou talvez mesmo por causa delas[10]. A sucessividade dos blocos harmônicos sol–si♭–ré e sol–si♭–mi♭, existente no primeiro compasso, é altamente instável. Isto abre terreno para uma possível interpretação do mi♭ como uma altura proeminente na amostra sendo analisada. A mudança de harmonia mais notável ocorre somente no último compasso, ainda com um acorde contendo mi♭, e gera uma nota dó, também subordinada diretamente ao evento-cabeça. Finalmente, pode-se concluir que a presença de dados harmônicos na análise desta amostra não influi nos resultados de maneira notória, sendo estes, aparentemente, mais influenciados por características melódicas.

Primeiro Tema do 4º Movimento da Sinfonia nº 40, de W.A. Mozart

A Redução Temporal das duas formas deste tema podem ser vistas, respectivamente, nas figuras 6.3 e 6.5 (páginas 119 e 121). De modo muito diverso ao que ocorreu quando da análise do tema anterior, no presente caso a harmonia apresenta-se como fator importantíssimo no que tange à qualidade dos resultados da análise. Ao considerar-se inicialmente o caso unicamente melódico, tem-se como evento-cabeça o si♭ do segundo

[10]Deve se ter presente que o analisador harmônico implementado neste projeto foi direcionado para obras completas, ou seja, que terminem cadencialmente com uma sucessão dominante–tônica. O emprego de amostras incompletas do ponto de vista formal, como é o caso do tema da sinfonia acima, não garante nem assegura corretas análises harmônicas.

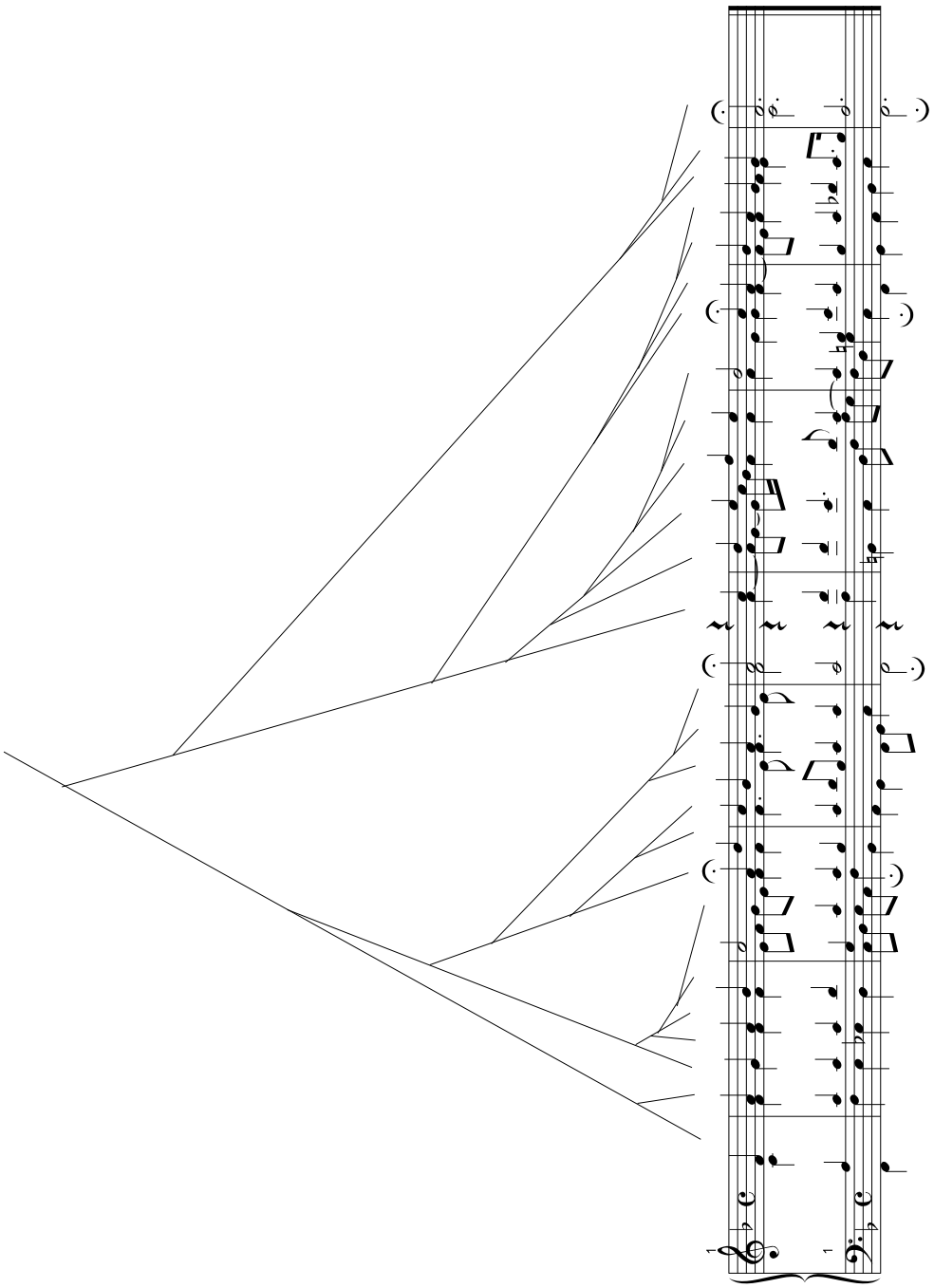

Figura 6.6: Redução Temporal do Coral *Christus, der ist mein Leben* (BWV 95), de J.S. Bach.

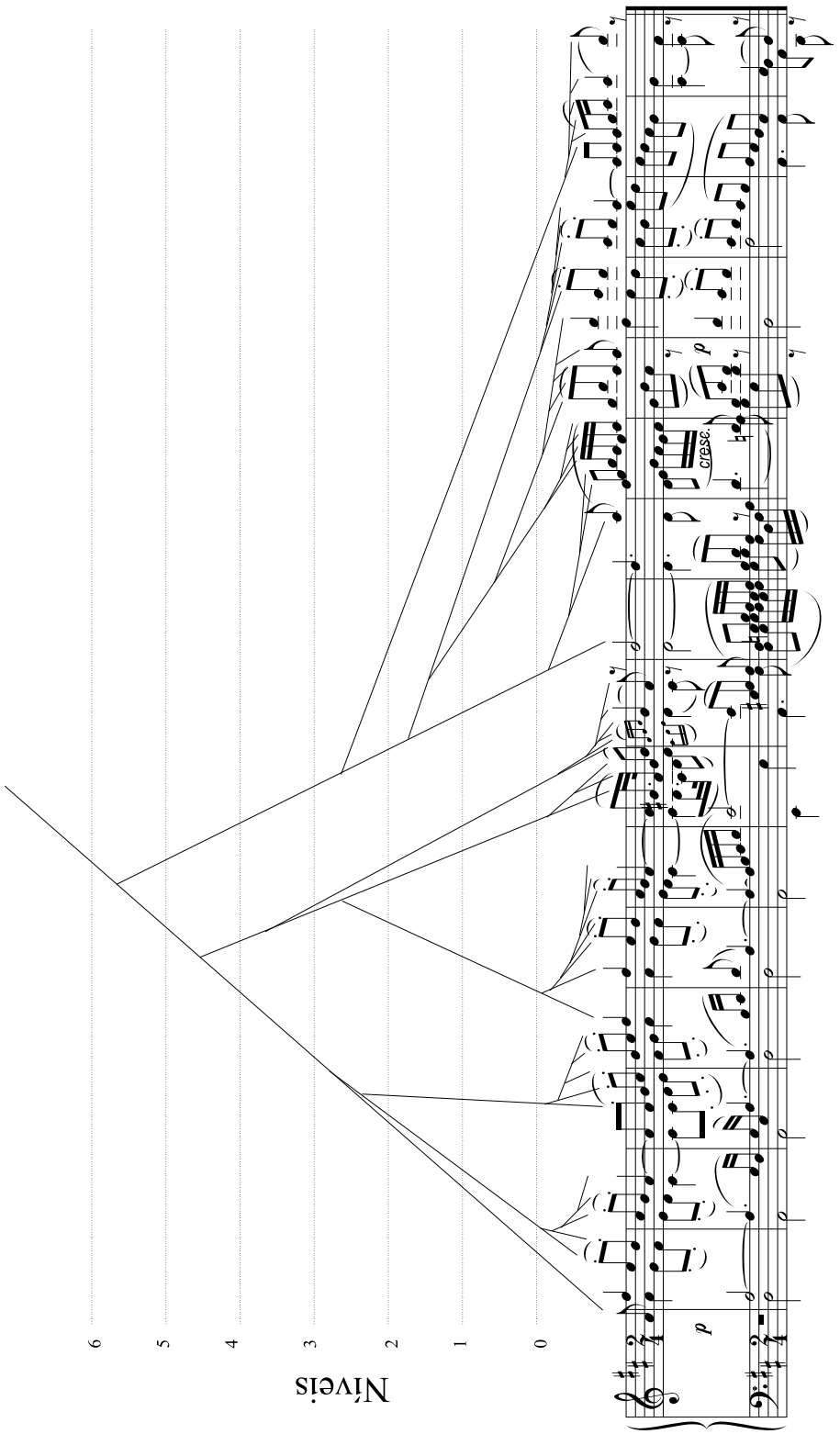

Figura 6.7: Redução Temporal do tema do 3º movimento do Quarteto de Cordas Op. 18 nº 5, de L.van Beethoven.

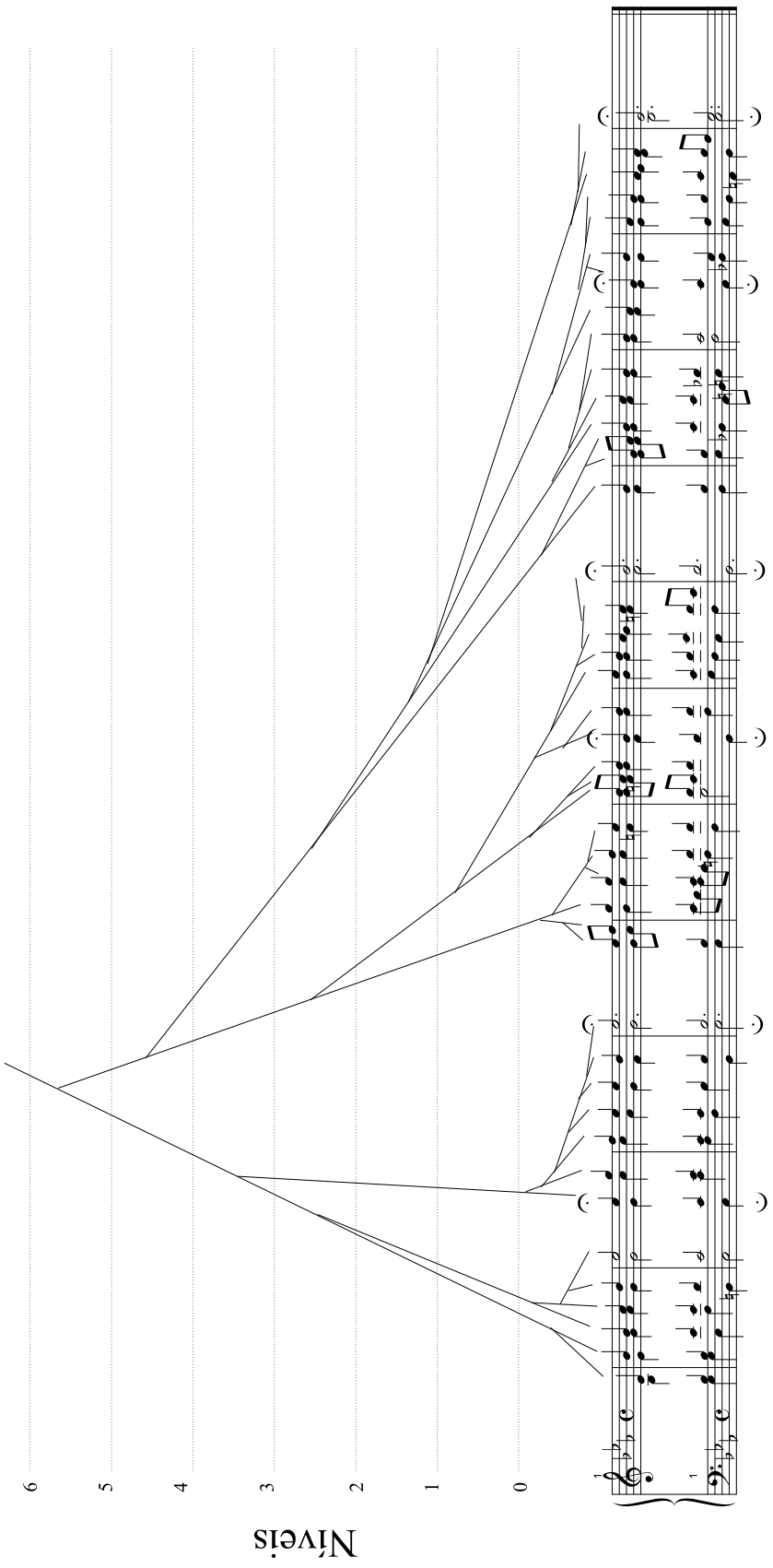

Figura 6.8: Redução Temporal do Coral nº 52 da *Paixão Segundo São João*, de J.S. Bach.

compasso e a árvore reducional não apresenta quaisquer indícios de uma tonalidade subjacente. Ao comparar-se este cenário com o da mesma amostra acompanhada de sua harmonia orquestral, pode-se ver que, neste caso, uma cadência a um acorde com fundamental ré (evento-cabeça da amostra) é claramente delineada e é refletida com nitidez na árvore reducional.

6.7.2 Análises

Christus, der ist mein Leben (BWV 95), de J.S. Bach

Na figura 6.6 (página 123) pode ser vista a Redução Temporal da presente obra. A análise alcançada mostra-se como inteiramente plausível, tendo como evento cabeça a altura fá e como evento diretamente subordinado a altura dó. Considerando que a tonalidade do coral é Fá maior, pode-se afirmar que os resultados refletem bem este contexto tonal. Uma questão apontada anteriormente, entretanto, merece ser revisitada devido à sua importância analítica (ver seção 5.8.2). Devido a uma defaseagem na grade métrica durante seu cálculo, o terceiro tempo do compasso anterior ao compasso final ficou sobrevalorizado, acarretando, durante a Redução Temporal, que o evento-cabeça do último grupo cadencial seja a altura sol e não a altura fá (tônica da região). Este problema pode servir de ilustração de o quanto os componentes da TGMT são interdependentes e influem de forma decisiva e veemente nos resultados dos demais.

Tema do 3º Movimento do Quarteto de Cordas Op. 18 nº 5, de L.van Beethoven)

Na figura 6.7 (página 124) pode ser vista a Redução Temporal do tema do quarteto. Na árvore reducional pode claramente ser vista a estrutura harmônica da peça, inclusive sua modulação central à dominante e o respectivo retorno à tônica. O evento-cabeça é a altura inicial do trecho (dominante da escala) e a altura diretamente subordinada a ele é a tônica da escala, delineando claramente a tonalidade de Ré maior. Possivelmente esta análise não foi ainda mais precisa devido à deficiente estrutura métrica que resultou da análise desta mesma amostra.

Coral nº 52 da Paixão Segundo São João, de J.S. Bach

A Redução Temporal do coral em questão pode ser vista na figura 6.8 (página 125). Ainda que não se possa afirmar que a análise resultante da Redução Temporal do coral em jogo seja equivocada, alguns pontos deixam margens a dúvidas e ficam em aberto. Em primeiro lugar, não existe na árvore reducional nenhum delineamento de um contexto tonal, parecendo, antes, uma forma monódica e baseada em um conceito unicamente escalar. Apesar das alturas diretamente subordinadas ao evento-cabeça (si♭) formarem uma tríade da tônica da tonalidade principal, esta característica não é recorrente, e tal fato, isoladamente, não garante uma presença da tonalidade principal na árvore reducional. Finalmente, é importante ressaltar o perfil de escalar descendente da melodia do coral, o que talvez tenho contribuído com vigor para o perfil da árvore reducional.

Capítulo 7

Redução Prolongacional

7.1 Introdução

Nesta seção, tratando do quarto e último componente da teoria, é apresentado um resumo da Redução Prolongacional. Assim como nos três casos anteriores, é apresentado um conjunto de considerações gerais, denominado Sistema Analítico, seguido de uma Gramática Formal contendo as Regras de Boa–Formatividade e Preferenciais que caracterizam este componente. Em seguida, são apresentados seu Objetivo Específico, sua Metodologia, a respectiva Implementação e, finalmente, os Resultados alcançados, sob forma de Testes de Validação e Análise de obras. Para uma abordagem mais detalhada da matéria pode-se consultar o capítulo 2 de Carvalho (2001) ou, prioritariamente, o texto original de Lerdahl & Jackendoff (1996).

7.1.1 Sistema Analítico

Um dos fatores mais marcantes da música tonal ocidental é o sentido de tensão e relaxamento presentes no desenrolar de qualquer peça escrita dentro de seus paradigmas. Dentro dos componentes já abordados da teoria (Estrutura de Agrupamento, Estrutura Métrica e Redução Temporal) nada existe para expressar a característica formalizada através da Redução Prolongacional, que é a incessante respiração de uma obra musical e o produto da justaposição de fatores rítmicos e de alturas.

O primeiro passo para que isto seja feito é conceber tensão e relaxamento como termos relativos definidos com respeito a eventos particulares. Como exemplo, pode-se pensar no fim de uma frase como um ponto de relativo relaxamento e, a partir deste evento ter-se-á um relativo incremento de tensão.

A partir desta noção simples é possível pensar em árvores prolongacionais nas quais tensão e relaxamento são definidos, respectivamente, em termos de ramificação à direita e ramificação à esquerda.

Para a construção da árvore, existem três tipos possíveis de conexão entre os eventos:

1. Uma *progressão* é a conexão entre dois eventos cujas fundamentais harmônicas são

diferentes, ou, em outras palavras, é a conexão entre dois eventos que não possuem a mesma harmonia;

2. Uma *prolongação fraca* é a conexão entre dois eventos de mesma harmonia, mas de consonância intrínseca diferente, ou seja, dois eventos de mesma harmonia, mas com o baixo (prioritariamente) ou a nota mais aguda (secundariamente) diferentes.

3. Uma *prolongação forte* é a conexão existente entre dois eventos cujas harmonias, baixos e notas mais agudas são idênticos.

Considerando-se que a construção da árvore prolongacional processa-se no sentido do global para o local, faz-se necessário o desenvolvimento de uma *forma básica* prolongacional a partir da qual toda a árvore seja construída.

Os eventos presentes na forma básica são os primeiros disponíveis para a análise prolongacional. Também é digna de nota a integridade gramatical da forma básica, expressando esta a prolongação da tônica e a resolução cadencial em praticamente todos os níveis de agrupamento.

Depois da forma básica, surge como elemento importante na construção da árvore prolongacional a *preparação cadencial*. Por esta expressão entende-se o evento ou grupo de eventos que conduzem até uma cadência. Esta preparação é feita através da chamada "função de subdominante"[1], que faz com que um evento seja dependente do evento seguinte, o qual, por sua vez, é dependente do último evento em consideração.

Finalmente, após a apresentação da forma básica e da preparação cadencial surge como elemento importante a "estrutura prolongacional normativa". Esta estrutura visa garantir as condições mínimas de ramificação para a representação dos padrões de tensão/relaxamento em música tonal. Os pré-requisitos para a estrutura prolongacional normativa são:

1. Não há necessidade de conexão direta entre os dois eventos prolongacionalmente mais importantes;

2. O núcleo cadencial (a preparação da cadência seguida da própria cadência) é construído através de uma dupla ramificação à esquerda;

3. É necessário pelo menos uma ramificação à direita antes do núcleo cadencial.

A combinação da forma básica com a estrutura prolongacional normativa forma uma estrutura que é mais completa que qualquer uma das duas tomadas isoladamente.

7.1.2 Gramática Formal

Visando a formalização da Redução Prolongacional são necessárias três hipóteses específicas. A primeira delas, a Hipótese Forte da Redução, afirma que é possível construir em árvore uma estrutura de natureza prolongacional, lançando, assim, a semente

[1] A função de subdominante pode ser representada por qualquer variante do II ou do IV graus.

para a formulação das Regras de Boa–Formatividade da Redução Prolongacional. A segunda hipótese trata da questão de determinar, dentre todas as estruturas em árvores atribuídas a uma peça pelas Regras de Boa–Formatividade, qual representa uma audição real. Esta hipótese, portanto, será incorporada pelas Regras Preferenciais da Redução Prolongacional.

Hipótese Forte da Redução A Hipótese Forte da Redução divide-se em duas partes:

a. Eventos-altura são ouvidos no contexto de uma hierarquia estrita (ou seja, recursiva e sem sobreposição entre os componentes do mesmo nível).

b. Eventos estruturalmente menos importantes não são ouvidos simplesmente como inserções, mas dentro de uma relação específica com os eventos mais importantes que os circundam.

Hipótese Prolongacional 1. Intuições sobre tensão/relaxamento derivados de alturas numa peça podem ser expressas em termos de uma segmentação estritamente hierárquica em *regiões prolongacionais*, tais que:

a. Cada região representa um todo de tensão ou relaxamento na progressão de seu início a seu fim; e

b. Tensões e relaxamentos internos a cada região representam estágios subordinados e não-sobrepostos na progressão total.

Hipótese Prolongacional 2. A escolha de eventos que definem regiões prolongacionais é fortemente influenciada pela importância relativa de eventos na Redução Temporal.

Quatro regras são necessárias para a definição da estrutura prolongacional em árvore. Estas quatro regras devem garantir as seguintes condições:

a. Uma árvore deve conter um único evento mais importante, o evento-cabeça.

b. Devem ser especificados os modos como os eventos elaboram outros eventos (como vimos antes, tratam-se da progressão e das prolongações fracas e fortes).

c. Todo evento numa peça deve estar conectado na árvore.

d. Os ramos da árvores não devem cruzar-se.

Antes da formulação das regras, entretanto, são necessárias, ainda, mais duas definições referentes ao tipo de elaboração de eventos:

a. Um evento e_i é uma *elaboração direta* de outro evento e_j se o ramo de e_i termina no ramo de e_j; e

b. Um evento e_i é uma *elaboração recursiva* de outro evento e_j se ele é uma elaboração direta de e_j ou se seu ramo procede ascendentemente através de uma seqüência de elaborações diretas até o ramo de e_j.

Tendo em mãos estas definições, é possível a formulação das *Regras de Boa–Formatividade da Redução Prolongacional* (RBFRP):

RBFRP 1. Em toda peça, existe um único evento na estrutura subjacente de agrupamento que funciona como evento-cabeça prolongacional.

RBFRP 2. Um evento e_i pode ser uma elaboração direta de outro evento e_j em qualquer do seguintes modos:

a. e_i é uma *prolongação forte* de e_j se as fundamentais, baixos e notas melódicas dos dois eventos são idênticas;

b. e_i é uma *prolongação fraca* de e_j se as fundamentais dos dois eventos são idênticas, mas os baixos e/ou notas melódicas diferem; e

c. e_i é uma *progressão para ou de* e_j se as fundamentais dos dois eventos são diferentes.

RBFRP 3. Todo evento na estrutura subjacente de agrupamento é, ou o evento-cabeça prolongacional, ou uma elaboração recursiva do evento-cabeça prolongacional.

RBFRP 4 (Não Cruzamento de Ramos). Se um evento e_i é uma elaboração direta de um evento e_j, todo evento entre e_i e e_j deve ser uma elaboração direta de e_i, e_j ou algum evento entre eles.

Antes de passar à formulação das *Regras Preferenciais da Redução Prolongacional* (RPRP), é necessário apresentar a definição de região prolongacional:

> Uma *região prolongacional* é uma seqüência de eventos (e_i–e_j), (#–e_i) ou (e_j–#) tal que todos os eventos dentro da seqüência são elaborações recursivas de e_i ou de e_j.

O signo # é uma representação abstrata das fronteiras da peça. Ele é necessário por dois motivos:

a. O início estrutural de uma peça pode não coincidir os eventos iniciais;

b. Pode existir uma prolongação da cadência final.

As duas primeiras Regras Preferenciais estabelecem a relação entre a Redução Temporal e a Redução Prolongacional.

RPRP 1 (Importância Temporal). Na escolha do evento prolongacionalmente mais importante e_k de uma região prolongacional (e_i–e_j), prefira uma escolha na qual e_k seja relativamente importante do ponto de vista temporal (ou seja, do *intervalo-temporal*).

RPRP 2 (Segmentação Temporal). Considere e_k como o evento prolongacionalmente mais importante na região prolongacional (e_i–e_j). Se existe um intervalo-temporal que contém e_i e e_k, mas não e_j, prefira uma redução prolongacional na qual e_k é uma elaboração de e_i; similarmente para a inversão de papéis entre e_i e e_j.

RPRP 3 (Conexão Prolongacional). Na escolha do evento prolongacionalmente mais importante e_k na região (e_i–e_j), prefira um e_k que se ligue com um dos extremos da região de modo a proporcionar a maior estabilidade de conexão prolongacional possível.

Um dos pontos centrais da presente teoria da Redução Prolongacional é a interação das RPM 1 e 3, tratando a primeira da importância temporal e a segunda, como acabamos de ver, da estabilidade das conexões. Para que sejam minimizados os possíveis conflitos entre as duas regras foi flexibilizada a escolha da importância temporal para dois níveis imediatos de intervalos-temporais. Esse processo aparece formalizado como:

Princípio da Interação Com o objetivo de fazer uma conexão prolongacional suficientemente estável, e_k deve ser escolhido dentre os eventos dos dois mais importantes níveis da Redução Temporal representada em (e_i–e_j).

RPRP 4 (Importância Prolongacional). Considere e_k como o evento prolongacionalmente mais importante na região prolongacional (e_i–e_j). Prefira uma redução prolongacional na qual e_k é uma elaboração do extremo mais importante, do ponto de vista prolongacional.

RPRP 5 (Paralelismo). Prefira uma redução prolongacional na qual passagens paralelas recebam análises paralelas.

Finalmente, a última regra preferencial regula a aplicação da estrutura prolongacional normativa.

RPRP 6 (Estrutura Prolongacional Normativa). Um grupo cadencial contém, preferencialmente, quatro (ou cinco) elementos em sua estrutura prolongacional:

a. um início prolongacional;

b. um final prolongacional consistindo de um elemento da cadência;

(c. uma prolongação por ramificação à direita como a mais importante elaboração direta do início prolongacional);

d. uma progressão por ramificação à direita como a (próxima) elaboração direta mais importante do início prolongacional; e

e. uma progressão "subdominante" por ramificação à esquerda como a mais importante elaboração do primeiro elemento da cadência.

O item entre parênteses refere-se ao caso da existência de reprises.

7.2 Objetivo Específico

O objetivo específico da Redução Prolongacional inclui o emprego dos resultados fornecidos pelos componentes anteriores na construção de uma árvore reducional. Para a construção de tal árvore faz-se necessário, além disso, uma correta interpretação dos fenômenos harmônicos presentes no trecho que se deseja analisar. A árvore reducional, na medida em que representa a integração de todos os componentes da teoria, é também o coroamento de todo o processo analítico, permitindo observar as relações hierárquicas e recursivas do discurso musical que está sendo submetido à análise.

7.3 Metodologia

Na medida em que, para a realização da Redução Prolongacional, além da resolução dos três componentes anteriores da teoria, é necessária uma eficiente metodologia de análise harmônica, foi dedicada especial atenção para o estudo dos possíveis algoritmos a serem empregados neste tópico.

Além disto, ao considerar-se que nos outros componentes da teoria – Estrutura de Agrupamento, Estrutura Métrica e Redução Temporal – faz-se necessária uma correta análise dos fenômenos harmônicos, a implementação de tal analisador mostra-se fundamental para uma boa continuidade do trabalho. Entretanto, o analisador, devido às suas características de implementação, necessita que sua entrada sofra um pré-processamento afim de que sejam eliminados elementos que estejam fora de seu âmbito de análise. Assim, para a descrição do trabalho realizado seguir-se-á o seguinte roteiro:

a. Algoritmo de pré-processamento.

b. Contextualização do algoritmo do analisador.

c. Representação musical em emphConstraint Satisfaction Problem (CSP).

d. Descrição do algoritmo.

e. Melhorias realizadas.

f. Implementação.

7.3.1 Pré-processamento do Trecho

O analisador harmônico utilizado pelo sistema possui duas restrições importantes que direcionam seu modo de utilização. Elas são a limitação das análises a trechos com quatro vozes (partes) e com escritura homofônica (todas as vozes com o mesmo ritmo).

A primeira restrição determina que, em princípio, somente trechos a quatro vozes poderão ser analisados. Para trechos de uma a três vozes novos algoritmos apresentam-se como necessários para a inserção das vozes ausentes tornando, assim, possível a análise

de trecho com densidades diversas. O desenvolvimento de tais algoritmos, entretanto, apresenta-se fora do escopo deste trabalho.

A segunda restrição diz que o trecho a ser analisado deve estar isento de notas melódicas (notas estranhas à harmonia) e de notas com valores menores do que a unidade de tempo do compasso sendo utilizado, ainda que estas sejam harmônicas. Devido à esta natureza do analisador harmônico e também ao fato de que numa obra musical real é certa a presença de notas melódicas, faz-se necessário o pré-processamento do trecho musical a ser analisado. A proposta deste processamento prévio é a de retirar do trecho todas as notas não harmônicas de modo que a entrada do analisador somente contenha acordes classificáveis.

A idéia básica do algoritmo utilizado para o pré-processamento é a interpolação de um trecho musical, a partir do menor valor de duração encontrado e, a partir do conjunto de acordes encontrado, compará-los com estruturas interválicas armazenadas numa base de dados. Tais estruturas representam as possibilidades interválicas dos acordes classificados da música tradicional. São incluídos os diferentes estados dos acordes, os tipos possíveis de dobramentos e os acordes de sétima, também com seus diferentes estados e dobramentos. Na medida em que existe uma restrição concernente à densidade dos acordes, a base de dados foi projetada e construída para uma densidade de quatro partes (vozes polifônicas).

Abaixo estão enumerados os passos do algoritmo de pré-processamento harmônico:

1. Fazer a varredura do trecho e armazenar o menor valor de duração. A partir deste valor, fazer a interpolação em todas as quatro vozes definindo todos os demais valores de duração como múltiplos do menor valor encontrado.

2. Dentro do conjunto de acordes encontrado (resultado da interpolação) remover os que são repetidos somando os valores de duração de forma a obter a duração correspondente a cada acorde em questão.

3. Remover os acordes que não estão na base de dados.

4. Remover os acordes com menos do que um tempo.

5. Remover os acordes repetidos residuais. Este passo (repetição do segundo) é necessário porque acordes repetidos podem surgir durante a operação dos passos anteriores.

A seguir estão descritos detalhadamente cada um dos passos. O primeiro consiste em:

1.1 Fazer a varredura dos valores de duração de cada uma das vozes armazenando seu *quantum* (menor valor para todas as vozes).

1.2 Fazer a interpolação das durações para cada uma das vozes, ou seja, transformar cada uma das durações de cada uma das vozes numa soma de *quanta* de duração.

O passo seguinte consiste na remoção, dentro da lista calculada no passo anterior, dos acordes repetidos (acordes idênticos). Para isto segue-se a seqüência:

2.1 Alocar memória para uma lista temporária de acordes de mesmo tamanho da lista de entrada (com os acordes interpolados). A dimensão igual entre as duas listas é necessária porque, no pior caso, todos os acordes serão diferentes.

2.2 Ler o primeiro acorde da lista de entrada.

2.3 Enquanto o acorde seguinte for igual a ele, somar sua duração com a do seguinte.

2.4 Colocar o acorde com sua duração na lista temporária.

2.5 Incrementar contador da lista temporária.

2.6 Se não existem mais acordes na lista de entrada, passar para o próximo item. Senão, ler o próximo acorde da lista de entrada e voltar ao item 2.3.

2.7 Copiar resultados para a lista de entrada (lista original).

Para implementar o terceiro passo, emprega-se a lista de tarefas abaixo:

3.1 Alocar memória para uma lista temporária de acordes de mesmo tamanho da lista de entrada (com os acordes resultantes do passo anterior). A dimensão igual entre as duas listas é necessária porque, no pior caso, nenhum acorde será removido.

3.2 A partir da lista de entrada calcular uma matriz com os intervalos entre a quarta voz (baixo) e cada uma das outras vozes.

3.3 Varrer a lista de entrada até achar uma acorde cuja estrutura interválica exista na base de dados de acordes classificados e colocá-lo na lista temporária.

3.4 Enquanto o acorde seguinte não existir na base de dados, somar sua duração com aquela do acorde atual.

3.5 Colocar o acorde na lista temporária.

3.6 Incrementar contador da lista temporária.

3.7 Ler o próximo acorde da lista de entrada.

3.8 Se for o último acorde, finalizer, senão, ir para o passo 3.4.

No quarto passo são removidos os acordes com menos de um tempo, mas que estão na base de dados. O pré-processamento leva em conta pesos para cada tipo de acorde, considerando sua função e seu estado. Importante é considerar os instantes de ataques, principalmente nos casos onde pode existir anacrusis, pois isto poderá ser relevante para a análise harmônica. Para alcançar os resultados a unidade de tempo é normalizada em 1 (o que é realizado previamente pelos outros módulos do programa). Levando em conta a complexidade do algoritmo deste quarto passo, sua descrição será feita em pseudo-código ao invés de linguagem comum, visando sua melhor compreensão:

```
count = 0      ; contador da lista original
finalcount = 0 ; contador da lista final
Enquando(count < tam_lista_original) {
   Aponta para um acorde e verifica seu peso.
   Se (acorde for no tempo) {
      Armazena no acorde temporário.
      Enquanto (próximo acorde for fora do tempo) {
         Verifica seu peso.
         Se (peso for maior do que o peso do acorde temporário) {
            Armazena o segundo no acorde temporário com o ataque original
                                    do temporário e soma das durações.
         }
         Senão, Se (peso for menor do que o peso do temporário) {
            Soma as durações do temporário e do segundo e armazena
                                                 no temporário.
         }
          count = count + 1
          Vai para o próximo acorde.
      }
      Armazena o acorde temporário na lista final.
      finalcount = finalcount + 1
   }
   Se (acorde for fora do tempo) {
      Armazena no acorde temporário.
      Enquanto (próximo acorde for fora do tempo) {
          Verifica seu peso.
          Se (peso for maior do que o peso do acorde temporário) {
              Armazena o segundo no acorde temporário com o ataque original
              do temporário e soma das durações.
           }
           Senão, Se (peso for menor do que o peso do temporário) {
              soma as durações do temporário e do segundo e deixa os
              demais parâmetros intactos.
           }
           count = count + 1
           Vai para o próximo acorde.
      }
      Se (peso do temporário for maior do que o do acorde no tempo) {
         Coloca temporário na lista final com o ataque e duração do
         acorde no tempo.
      }
      Senão, se (peso do temporário for menor do que o do acorde no tempo){
         Coloca acorde no tempo na lista final com o ataque e duração
```

```
        originais.
      }
      Armazena o acorde temporário na lista final.
      finalcount = finalcount + 1
    }
    count = count + 1
 }
```

Finalmente, o quinto passo faz a remoção dos acordes repetidos residuais, ou seja, repetições de acordes que podem surgir durante o processamento dos passos terceiro e quarto.

Antes de prosseguir com a exposição de um exemplo de pré-processamento serão ditas algumas palavras acerca da elaboração da base de dados e do sistema de pesos nela empregado.

Como já foi dito anteriormente, a base de dados empregada é uma matriz cujas linhas representam os estados (disposições ou dobramentos) possíveis de cada um dos acordes classificados presentes na música tradicional. Cada coluna representa, então, os intervalos de cada uma das partes superiores em relação ao baixo. Por exemplo, para o acorde perfeito maior, tem-se[2]:

```
Estado fundamental:

{7,4,0}  -> fundamental dobrada
{4,4,0}  -> fundamental dobrada, sem quinta
{7,7,0}  -> fundamental dobrada, sem terça
{4,0,0}  -> fundamental triplicada, sem quinta
{7,0,0}  -> fundamental triplicada, sem terça
{7,4,4}  -> terça dobrada e quinta
{7,7,4}  -> quinta dobrada e terça

1a. inversão:

{8,8,3}  -> fundamental dobrada, terça e quinta
{8,8,0}  -> fundamental dobrada, terça dobrada
{8,3,3}  -> quinta dobrada
{8,3,0}  -> terça dobrada, fundamental e quinta
{8,8,8}  -> fundamental triplicada

2a. inversão:
```

[2]Acerca desta representação devem ser observadas duas características. Inicialmente, que os elementos destes conjuntos representam intervalos ordenados de classes de alturas tais como definidos em Straus (1990). Isto significa que estão representadas na base de dados as reais estruturas interválicas dos acordes e não sua representação em alturas. Em segundo lugar, que os intervalos estão ordenados de forma decrescente. Isto por motivos de implementação, sem qualquer conseqüência sobre os resultados.

```
{9,5,5}  -> fundamental dobrada com terça
{5,5,0}  -> fundamental dobrada, quinta dobrada
{9,9,5}  -> terça dobrada
{9,5,0}  -> quinta dobrada, fundamental e terça
{5,5,5}  -> fundamental triplicada
```

A ordem em que são apresentadas cada uma das possibilidades para cada acorde é importante porque define a hierarquia de categorias, estados e instâncias. Em outras palavras, define uma hierarquia entre os diferentes tipos de acordes, a nota que está no baixo e os dobramentos possíveis. Esta ordem também é importante para o cálculo dos pesos que definem a importância relativa de cada um dos acordes. Mais à frente, esse cálculo será descrito detalhadamente.

Portanto, na base de dados estão representados os seguintes acordes, respeitando a ordem colocada:

a. Acorde perfeito maior.

b. Acorde perfeito menor.

c. Acorde de sétima de dominante.

d. Acorde de quinta diminuta.

e. Acorde perfeito maior com sétima maior.

f. Acorde perfeito menor com sétima menor.

g. Acorde de quinta diminuta com sétima menor.

h. Acorde de sétima diminuta.

i. Acorde de quinta aumentada.

j. Acorde de quinta aumentada com sétima menor.

k. Acorde de quinta aumentada com sétima maior.

Na figura 7.1 é possível observar cada uma destas tipologias (colunas) tendo como fundamental cada uma das possibilidades cromáticas (linhas).

É importante notar que a representação presente na base de dados diz respeito à estrutura interna dos acordes sem conter nenhuma indicação acerca de sua categoria funcional dentro de uma tonalidade. Esta diferenciação é feita no processamento realizado pelo analisador harmônico, o qual é o passo seguinte a análise que emprega esta base de dados. Em seqüência, é descrito o sistema de pesos empregado para o cálculo da importância relativa de cada acorde.

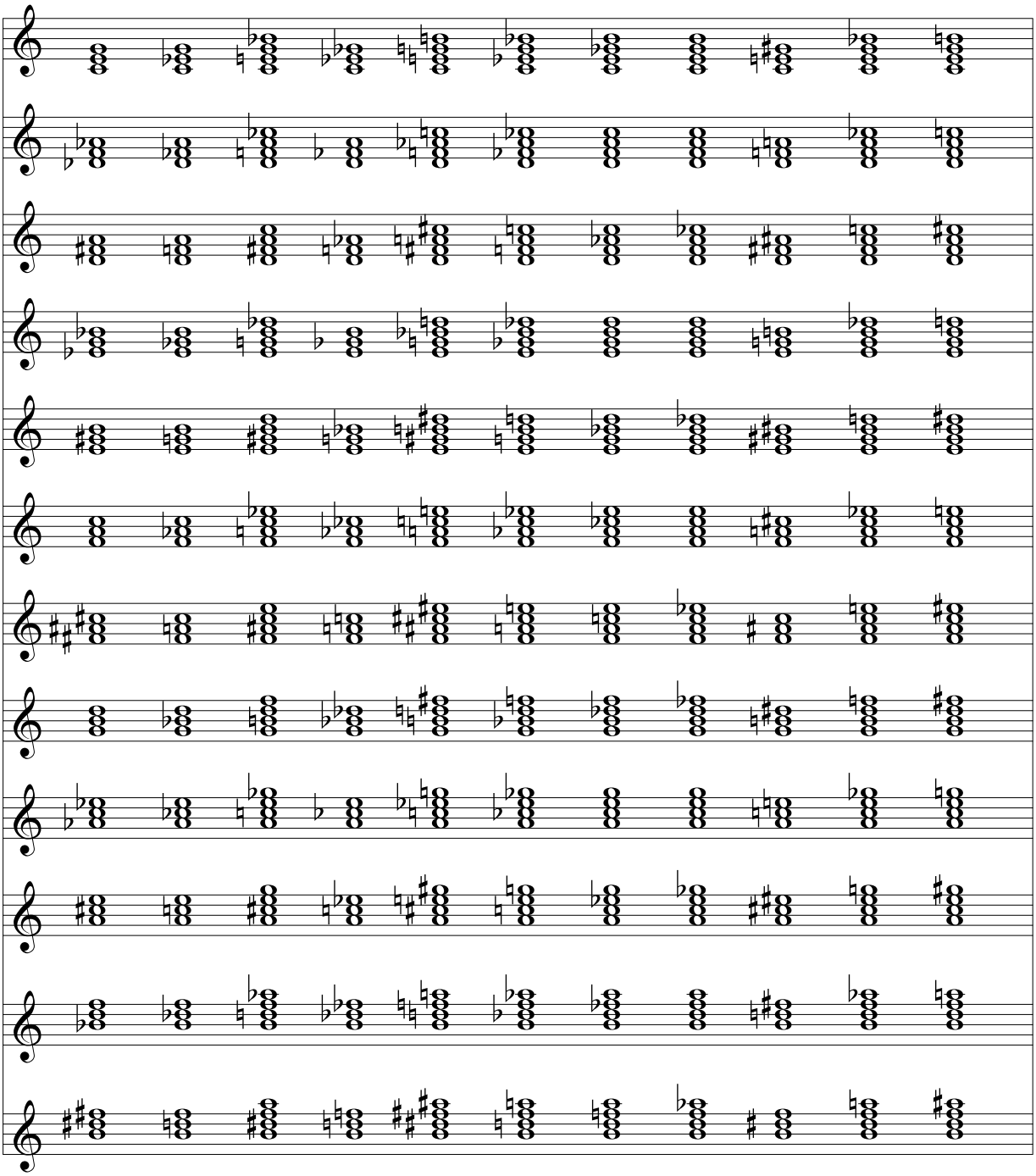

Figura 7.1: Tipologias de acordes presentes na base de dados.

Apesar de não ser um requisito do analisador harmônico empregado que os acordes tenham um ou mais tempos do compasso, é interessante uma redução da densidade temporal dos mesmos quando existirem dois ou mais acordes por tempo, pois isto conduzirá a uma compreensão harmônica mais clara do trecho em análise. Um modo objetivo de realizar esta tarefa é a introdução de pesos para cada um dos acordes presentes na base de dados.

Para isto, considera-se as onze categorias (tipos de acordes) presentes na base de dados e divide-se o intervalo [1,0] em onze parte iguais, correspondendo cada uma das partes a uma das categorias listadas acima (na ordem apresentada). Em seguida, faz-se uma interpolação linear de n valores em cada um destes onze sub-intervalos, sendo n o número de instâncias presentes para cada categoria (incluindo os diferentes estados e dobramentos). Os acordes perfeito maior e perfeito menor, por exemplo, possuem dezessete instâncias cada um. Desta forma, considerando que as categorias estão hierarquicamente ordenadas, e que dentro delas também os estados e as instâncias, os resultados destas interpolações representam pesos numéricos que correspondem à importância relativa de cada categoria dentro do conjunto de acordes possíveis, de cada estado dentro de uma determinada categoria e de cada instância dentro de um determinado estado. Abaixo pode ser vista a lista de pesos para o acorde perfeito maior (considerado neste caso como o mais importante):

```
Estado Fundamental

7 4 0  (1.000000)
4 4 0  (0.994118)
7 7 0  (0.988235)
4 0 0  (0.982353)
7 0 0  (0.976471)
7 4 4  (0.970588)
7 7 4  (0.964706)

1a. Inversão

8 8 3  (0.958824)
8 8 0  (0.952941)
8 3 3  (0.947059)
8 3 0  (0.941176)
8 8 8  (0.935294)

2a. Inversão

9 5 5  (0.929412)
5 5 0  (0.923529)
9 9 5  (0.917647)
9 5 0  (0.911765)
```

```
5 5 5  (0.905882)
```

As primeiras três colunas representam o acorde e a última, entre parêntesis, mostra o peso do acorde. Na figura 7.2 podem ser vistos, em grafia musical, os estados do acorde e seus possíveis dobramentos. Finalmente, duas últimas palavras de caráter complementar. A primeira acerca da atribuição de pesos aos acordes classificados e a outra sobre a própria base de dados.

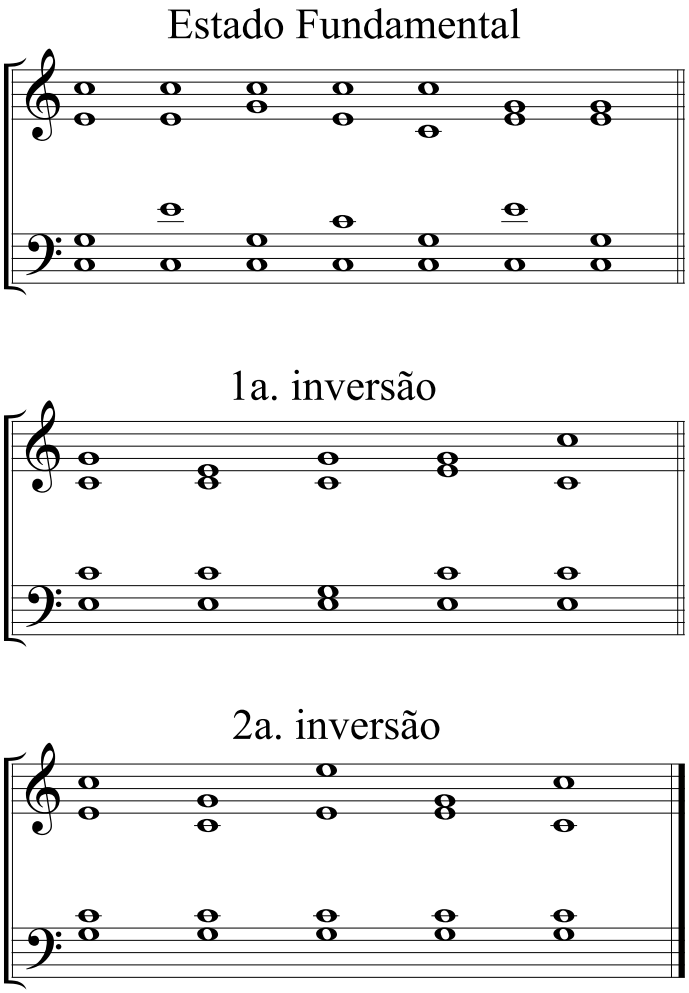

Figura 7.2: Exemplo de um acorde com suas inversões e dobramentos.

Levando em conta a importância dos acordes perfeitos na teoria e prática musicais e também a existência de tonalidades maiores e menores, cujas tônicas são, respectivamente, um acorde perfeito maior e um acorde perfeito menor, estas duas categorias de acordes receberam, apesar da interválica diversa, os mesmos pesos hierárquicos. Tal foi também o procedimento adotado nos casos dos acordes perfeitos maiores com sétima maior relativamente aos acordes perfeitos menores com sétima menor.

Ainda que na música tradicional ocorram acordes que não estão presentes na base de dados, tais formações assemelham-se muito mais a produtos da condução das vozes do que realmente a acordes com função estrutural mais nítida. Tal é o caso, por exemplo, de acordes como o acorde perfeito menor com sétima maior, no qual a sétima pode ser vista, muito mais apropriadamente, como um componente melódico do que como parte integrante do acorde. Com os meios apresentados aqui, tal acorde seria catalogado como não classificado e sua posição no discurso ocupada por outro acorde de função estrutural mais definida, eventualmente, o mesmo acorde sem sétima ou algum outro no qual a resolução ascendente da sétima fosse viável.

Para finalizar, é apresentado um exemplo de pré-processamento de um trecho especialmente elaborado para esta seção. Visando ilustrar ainda melhor cada um dos passos do pré-processamento e os procedimentos neles empregados, são mostradas as formas numérica e simbólica (partitura), o que, além do mais, possibilita uma avaliação mais apurada dos resultados. O trecho musical empregado pode ser visto na figura 7.3.

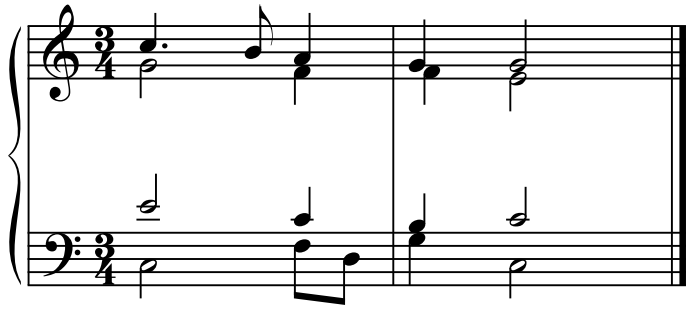

Figura 7.3: Trecho original empregado como exemplo de pré-processamento.

Após o processamento do primeiro passo, obtém-se um conjunto de 12 acordes com a mesma duração (a menor duração, neste caso corresponde a uma colcheia), como pode ser visto na figura 7.3):

```
72 72 72 71 69 69 67 67 67 67 67 67
67 67 67 67 65 65 65 65 64 64 64 64
64 64 64 64 60 60 59 59 60 60 60 60
48 48 48 48 53 50 55 55 48 48 48 48
```

Sua representação musical está na figura 7.4.

Lá pode-se ver que do primeiro ao terceiro acorde, depois no sétimo e oitavo e, finalmente, do nono ao décimo-segundo existem repetições. Estas são eliminadas no segundo passo:

```
72 0.00 1.50
67 0.00 1.50
64 0.00 1.50
48 0.00 1.50
```

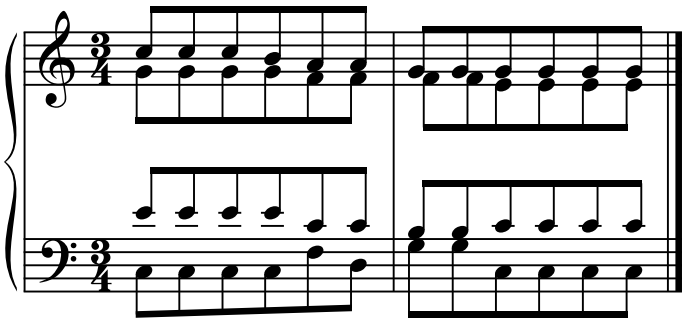

Figura 7.4: Resultado do passo 1 do pré-processamento.

```
71 1.50 0.50
67 1.50 0.50
64 1.50 0.50
48 1.50 0.50

69 2.00 0.50
65 2.00 0.50
60 2.00 0.50
53 2.00 0.50

69 2.50 0.50
65 2.50 0.50
60 2.50 0.50
50 2.50 0.50

67 0.00 1.00
65 0.00 1.00
59 0.00 1.00
55 0.00 1.00

67 1.00 2.00
64 1.00 2.00
60 1.00 2.00
48 1.00 2.00
```

Como aqui as durações passam a ser diferentes, cada acorde é mostrado como uma matriz em que cada linha é uma altura, seu instante de ataque e sua duração. O instante de ataque é representado em função modular do número de tempos do compasso, o que significa que o início do compasso é sempre 0 (zero). A representação simbólica do passo 2 pode ser vista na figura 7.5.

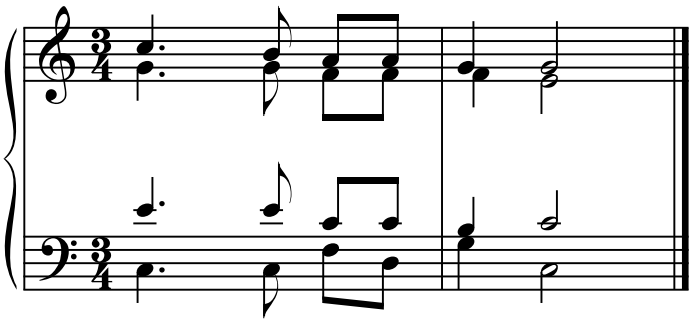

Figura 7.5: Resultado do passo 2 do pré-processamento.

O passo 3 trata da remoção dos acordes que não estão na base de dados. No caso deste exemplo todos os acordes são classificados, o que significa que nenhum deles será removido.

Os resultados presentes na saída do processamento do passo 4 são os seguintes:

```
O peso do acorde 0 é 104.000.
Acorde no tempo.

O peso do acorde 1 é 51.000.
Peso do segundo menor ou igual ao do primeiro.

O peso do acorde 2 é 104.000.
Acorde no tempo.

O peso do acorde 3 é 35.000.
Peso do segundo menor ou igual ao do primeiro.

O peso do acorde 4 é 63.000.
Acorde no tempo.

O peso do acorde 5 é 104.000.
Acorde no tempo.
```

Através deles pode-se observar que os únicos acordes fora do tempo são o segundo e o quarto e que seus pesos relativos são menores do que aqueles de seus antecessores e predecessores. Assim, eles passam a ser absorvidos por aqueles acordes que os precedem. A representação simbólica do resultado do processamento do passo 4 está na figura 7.6.

Na medida em que não surgiram novas repetições de acordes após o processamento do último passo, a remoção de repetições residuais (passo 5) torna-se desnecessária. Assim, o resultado presente na figura 7.6 é aquele que será enviado ao analisador.

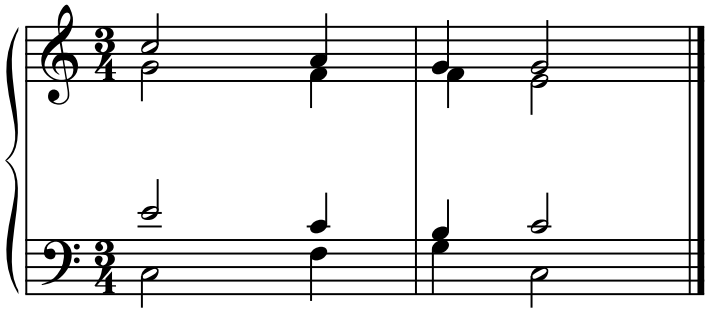

Figura 7.6: Resultado do passo 4 do pré-processamento.

7.3.2 Contextualização do Algoritmo do Analisador

O algoritmo do analisador harmônico é sugerido por Hoffman & Birmingham (2000) no contexto da técnica conhecida como CSP (*Constraint Satisfaction Problem*), a qual é descrita a seguir.

Descrição básica do CSP

O CSP pode ser definido como tendo os seguintes componentes:

a. Um conjunto de *variáveis* $\mathbf{V} = \{V_1, \ldots, V_i, \ldots, V_n\}$.

b. Um *domínio* para cada variável $\mathbf{D_i} = \{d_1, \ldots, d_d\}$, o qual é o conjunto de valores que podem ser atribuídos à variável V_i. Os domínios podem ser conjuntos discretos ou contínuos. Cada elemento do domínio pode ser, também, formado por um conjunto de atributos.

c. Um conjunto de *restrições* (*constraints*) $\mathbf{C} = \{C_{1,1,1}, \ldots, C_{i,j,k}\}$ que restringem os possíveis valores das variáveis. A notação $C_{i,j,k}$ diz respeito à k-ésima restrição que possui as variáveis V_i e V_j como argumento. Como auxiliares ao conjunto de restrições existem:

   – Um conjunto de funções de propagação de restrições $h_{i,j}(d_i.atributos)$ que restringe os atributos do domínio de V_i.

   – Uma função de avaliação de pré-condição $c_{i,j}^{pre}$ que define quando a restrição $C_{i,j}$ está ativa.

   – Um conjunto de avaliação de restrições $C_{i,j}(V_i, V_j)$ que retornam `VERDADEIRO` se os argumentos satisfazem a restrição $C_{i,j}$.

d. Um grafo de restrições $G(V, C)$ o qual descreve as relações entre variáveis e restrições, sendo essas os arcos e aquelas os nodos.

Uma referência clássica de CSP, na qual podem ser encontrados numerosos detalhes omitidos aqui, é Tsang (1993).

Propriedades do CSP

Diz-se que se tem uma solução para um CSP quando a todas as variáveis em jogo são atribuídos valores que satisfazem a todos as restrições. Um método para melhorar o desempenho do algoritmo em sua busca (possivelmente combinatorial) por soluções é a chamada aferição por *consistência de arco*, a qual assegura que, para cada variável, todos os elementos do domínio satisfazem as restrições para esta mesma variável. Isto pode ser representado assim:

$$\forall\ V_i : \{\forall\ d_i \in D_i : C_{i,j}(V_i, V_j)\}$$

Três aspectos, não obstante, devem ser apontados em relação a este método:

1. A consistência de arco não garante que todas as restrições serão satisfeitas simultâneamente, o que significa, em outras palavras, que não é garantida uma solução.

2. Se o grafo for decomponível – consistência-n – uma solução é garantida.

3. Para melhorar a busca, devem ser empregadas regras *preferenciais*, as quais guiam a análise em direção a soluções preferenciais.

Um Exemplo Numérico

Um exemplo ilustrativo encontrado em Hoffman & Birmingham (2000) ajuda a entender as definições expostas. Neste exemplo, é mostrado um grafo representando três variáveis (nodos) ligadas por duas restrições (arcos). Isto pode ser visto na figura 7.7. Os respectivos domínios são mostrados nas tabelas ao lado de cada uma das variáveis e tem como atributos a tupla `<largura,altura>`. Como já foi dito antes, existem somente dois tipos de restrições:

1. $C_{i,1}$ o qual diz que $V_{i+1}.largura = V_i.largura + 1$

2. $C_{i,2}$ o qual diz que $V_{i+1}.altura = V_i.altura$

Os passos para a obtenção de uma solução são os seguintes:

1. Empregando consistência de arco confere-se a restrição $C_{1,1}$ e verifica-se que o segundo elemento do domínio de V_2 (a tupla `<3,3>`) pode ser eliminado. Atribuir $V_2.largura = 3$ viola a restrição $C_{1,1}$ porque não existe nenhum elemento no domínio de V_1 cuja largura seja 2.

2. Continua-se o processo de consistência de arco conferindo-se a restrição $C_{2,2}$, após o que, elimina-se do domínio de V_3 as tuplas `<9,3>` e `<2,10>` (porque a tupla `<3,3>` foi eliminada do domínio de V_2 e não existe neste mais nenhum elemento com altura 3). Após estes dois primeiros passos tem-se o resultado mostrado na figura 7.8.

3. O próximo passo é a instanciação das variáveis a partir do(s) ponto(s) onde o domínio tenha sido reduzido a apenas um elemento. No nosso caso, isto acontece em V_3. Procede-se a instanciação, portanto, em ordem reversa.

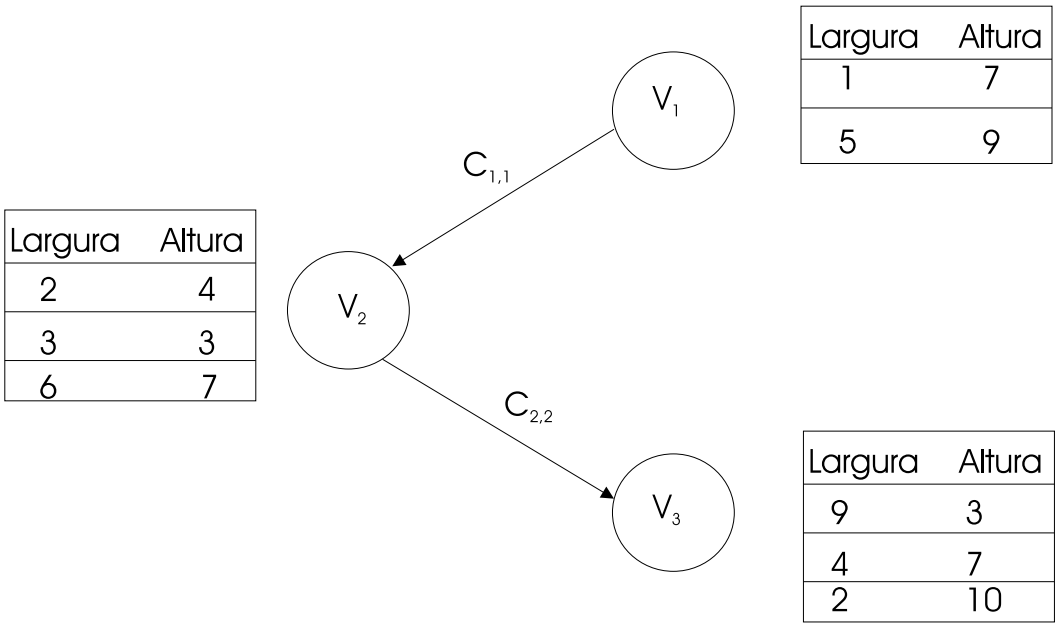

Figura 7.7: Grafo inicial de restrições.

4. Para satisfazer $C_{2,2}$, sendo V_3 = <4,7>, é necessário que V_2 = <6,7>.

5. Para satisfazer $C_{1,1}$, sendo V_2 = <6,7>, é necessário que V_1 = <5,9>.

A solução para o problema, portanto, é:

V_1 = <5,9>

V_2 = <6,7>

V_3 = <4,7>

Antes de abordar um exemplo estritamente musical, o qual terá o objetivo de ilustrar com maior clareza o presente trabalho, é necessário que seja exposto qual tipo de representação musical em CSP foi utilizada.

7.3.3 Representação Musical em CSP

Antes de ilustrar como foi realizado o mapeamento de uma análise harmônica para um CSP é necessário expor o conjunto de pressupostos referentes às peças a serem analisadas:

a. A tonalidade de uma peça é aquela da fundamental do primeiro e último acordes.

b. O modo é aquele do primeiro acorde.

c. Todas as modulações são para tonalidades maiores.

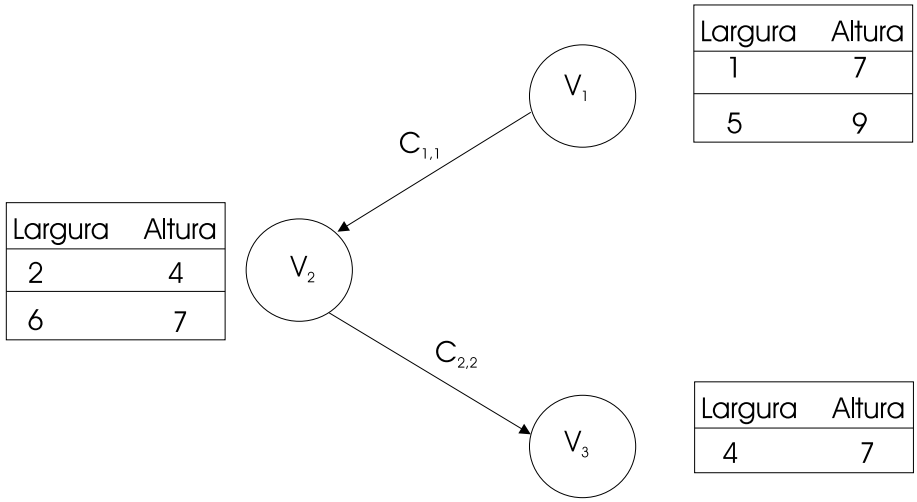

Figura 7.8: Grafo de restrições com consistência de arco.

d. As modulações seguem o círculo de quintas.

e. Suspensões são resolvidas na entrada dos dados.

f. Notas vizinhas e notas de passagem são omitidas.

sendo que os dois últimos itens são resolvidos durante o pré-processamento e os três primeiros são restrições ligadas à amostra sendo analisada. Com este conjunto de pressupostos garante-se a eficiência das heurísticas que compõem o algoritmo. A seguir é mostrado o mapeamento entre os componentes de um CSP e os parâmetros musicais em jogo:

a. Variável:

   - Acorde (e suas características)

b. Domínio:

   - Tonalidade
   - Modo (da tonalidade)
   - Função
   - Cadencial (booleano)
   - Índice (da variável)
   - Fundamental
   - Inversão
   - Acorde (vetor com quatro alturas)

c. Funções de Propagação de Restrições:

- $V_1.tonalidade = V_n.fundamental$
- $V_n.tonalidade = V_1.tonalidade$
- $V_i.modo = maior$
- $V_{i+1}.tonalidade = V_i.tonalidade$
- $V_{i+1}.tonalidade = (V_i.tonalidade + 7) mod 12$
- $V_{i+1}.tonalidade = (V_i.tonalidade + 5) mod 12$

d. Funções de Avaliação de Pré-condição:

- `TRUE`
- `TRUE`
- `TRUE`
- $V_{i+1}.tonalidade = V_i.tonalidade$
- $V_{i+1}.tonalidade = (V_i.tonalidade + 7) mod 12$
- $V_{i+1}.tonalidade = (V_i.tonalidade + 5) mod 12$

e. Preferências de Busca:

- Se $Tam(D_i) > 1$ e
  $V_{i+1}.fundamental = V_1.fundamental + 7$ e
  $V_i.tonalidade = V_{i+1}.tonalidade$
  Prefira
  $V_i.func = V/V$ a
  $V_i.fun = II$.
- Se $Tam(D_i) > 1$ e
  $V_{i+1}.fundamental = V_1.fundamental + 6$ e
  $V_i.tonalidade = V_{i+1}.tonalidade$
  Prefira
  $V_i.fun = vii/V$ a
  $V_i.fun = \#iv$.

Um Exemplo Musical

Tendo em mãos a representação musical em CSP dada anteriormente, é possível agora a apresentação de um exemplo musical. Neste, é analisada uma seqüência de três acordes, cada um deles sendo uma variável de um CSP e cujos domínios possuem três atributos, a saber, `<tonalidade,modo,função>`. A seqüência de acordes e sua representação em CSP podem ser vistas na figura 7.9. A seqüência de passos para a análise é a que segue:

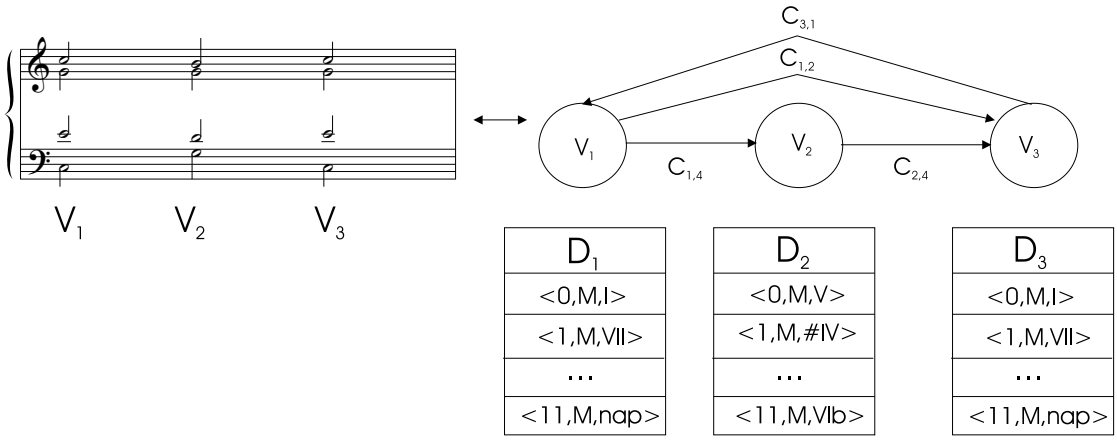

Figura 7.9: Exemplo musical e sua representação CSP, sendo que V_n é uma variável, sendo n seu índice, e $C_{m,n}$ uma restrição, sendo m o índice da variável e n o índice da restrição.

- Iniciando com a variável V_3 e através da restrição 1 ($C_{3,1}$) o domínio de V_1 fica reduzido a um elemento: `<0,maior,I>`.
- Considerando-se agora a variável V_1 e a restrição 2 ($C_{1,2}$) reduz-se também o domínio de V_3 a um único elemento: `<0,maior,I>`.
- Finalmente, por meio da restrição 4 aplicada às variáveis V_1 e V_2, tem-se, respectivamente, a redução do domínio de V_2 e a confirmação do elemento ativo em V_3.

7.3.4 Descrição do Algoritmo

Nesta subseção é descrito sucintamente o algoritmo do analisador harmônico e seus principais componentes. Para maiores detalhes consultar a fonte já citada.

O principal objetivo do algoritmo é a identificação de centros tonais. Para este fim são identificadas as fundamentais e o estado de cada acorde (variável no CSP) e estabelecidos seus respectivos domínios. Em seguida é determinada a tonalidade da peça. O próximo passo é a busca por padrões cadenciais e a criação de uma lista de cadências. Esta lista, finalmente, é utilizada na busca por modulações. Para se chegar à análise final é empregada a consistência de arco e, se após todo o processo alguma variável tiver em seu domínio mais de um elemento, é utilizada uma busca preferencial para determinar o melhor elemento e desativar os demais.

Para realizar seu trabalho, o algoritmo conta com as rotinas listadas abaixo:

FIND_ROOT(V_i) Este componente encontra a fundamental e o estado (inversão) de uma variável.

CREATE_DOMAIN(V_i)
Este componente cria o domínio de uma variável.

IS_CADENCE(V_i)
Este componente verifica se uma variável é cadencial. Se for, a coloca numa lista de cadências.

MODULATE(Z_i,Z_{i+1})
Este componente verifica se existe modulação entre dois elementos da lista de cadências. Caso exista modulação retorna o índice da variável onde a mesma se inicia.

EVALUATE_PRECONDITIONS(Z_k,Z_{k+1},mod_index)
Ajusta os domínios das variáveis presentes entre duas cadências levando em conta se existe ou não modulação no intervalo entre elas.

O algoritmo empregado possui melhorias, ampliações e correções em relação às descrições fornecidas por Hoffman & Birmingham (2000). O algoritmo completo é dado abaixo:

```
/*
 * Acha as fundamentais e os estados.
 * Cria os domínios.
 */
1. Cria uma lista V de acordes a partir da entrada.
2. Para i <- 1 até Tam(V)
3.    FIND_ROOT(V_i)
4.    CREATE_DOMAIN(V_i)
5. Fim do Para(i)
/*
 * Cria uma lista de cadências como vazia e
 * coloca a primeira variável na lista.
 */
6. Z <- 0
7. Z <- V_1
/*
 * Verifica se uma variável é cadencial.
 * Se for, a coloca na lista de cadências.
 */
8. Para i <- 1 até Tam(V)
9.    Se IS_CADENCE(V_i)
10.      Para j <- 1 até Tam(D_i)
11.         v_{i,j}.cadencial = VERDADEIRO
12.      Fim do Para(j)
13.      Z <- V_i
14.   Fim do Se(IS_CADENCE)
15. Fim do Para(i)
/*
```

```
 * Verifica se existe modulação entre duas variáveis
 * consecutivas na lista de cadências e ajusta seus
 * domínios levando em conta as restrições e os campos
 * tonais encontrados.
 */
16. mod_index <- 0
17. Para i <- 1 até Tam(Z)-1
18.    mod_index = MODULATE(Z_i,Z_{i+1})
19.    EVALUATE_PRECONDITIONS(Z_i,Z_{i+1},mod_index)
20. Fim do Para(i)
/*
 * Percorre a lista de variáveis e verifica se algum
 * elemento dos respectivos domínios violou alguma
 * das restrições 4, 5 ou 6. Se for o caso, retira
 * o elemento.
 */
21. Para i <- 1 até Tam(V)
22.    Para j <- 4 até 6
23.       Se C_{i,j}^pre = VERDADEIRO
24.          Para k <- 1 até Tam(D_i)
25.             Se v_{i,k} viola C_{i,j}
26.                retira v_{i,k} de D_i
27.             Fim do Se(v_{i,k})
28.          Fim do Para(k)
29.        Fim do Se(C_{i,j}^pre)
30.    Fim do Para(j)
31. Fim do Para(i)
/*
 * Se restar algum dos domínios com mais de um elemento,
 * utiliza uma busca preferencial para encontrar o mais
 * adequado e retirar os demais.
 */
32. Para i <- 1 até Tam(V)
33.    Se Tam(D_i) > 1
34.       imprime v_i preferencial
35.    Senão
36.        imprime v_i
37.    Fim do Se(Tam(D_i))
38. Fim do Para(i)
39. Fim do algoritmo
```

7.3.5 Implementação

O analisador recebe como entrada uma matriz 4 x n, onde 4 é o número de componentes de cada acorde e n é o número de acordes. As vozes dos acordes devem ser dadas em ordem invertida, ou seja, a primeira linha da matriz corresponde ao baixo e a última ao soprano.

7.4 Implementação da Redução Prolongacional

A implementação da Redução Prolongacional emprega, de maneira direta ou indireta, os resultados dos três outros componentes descritos anteriormente. Isto pode ser melhor compreendido ao observar-se a figura 3.1, na qual se pode visualizar esta dependência.

De modo semelhante à Redução Temporal, da qual depende diretamente, as Regras de Boa–Formatividade da Redução Prolongacional são integradas no próprio projeto das estruturas de dado utilizadas, enquanto que suas Regras Preferenciais não possuem, igualmente como no caso anterior, uma ferramenta exclusiva para seu cálculo.

7.4.1 Algoritmo Principal

O algoritmo para a implementação da Redução Prolongacional baseia-se em dois pilares principais:

1. Criação de vetor de pesos para cada evento da superfície. Para o cálculo dos pesos utiliza-se as Regras Preferenciais da Redução Prolongacional)[3].

2. Mecanismo de recursão atuando sobre o vetor citado no item anterior e visando a criação da árvore prolongacional.

Vetor de Pesos

Para o cálculo do vetor de pesos empregou-se o algoritmo a seguir:

1. Inicializar com zeros o vetor de pesos w, de dimensão igual à superfície de eventos.

2. Enquanto houver um evento e_i na superfície:

   (a) Percorrer o conjunto de Regras Preferenciais R_p tendo como argumento e_i;

   (b) A cada regra que o evento e_i satisfizer fazer $w_i = w_i + c$, onde c é uma constante qualquer.

[3]Este procedimento tem semelhanças com sua contrapartida equivalente da Redução Temporal. A principal diferença é que, nesta última, o conjunto de eventos a ser avaliado corresponde aos eventos no interior de um agrupamento de um determinado nível e, no caso da Redução Prolongacional, trabalha-se com toda a superfície do trecho a ser analisado.

Construção da Árvore Prolongacional

Esta parte do algoritmo da Redução Prolongacional tem como entradas o evento-cabeça resultante da Redução Temporal e o vetor de pesos w descrito acima. Seu funcionamento baseia-se, em primeiro lugar, em achar a posição do evento- cabeça na superfície da amostra, separando-a assim, em duas regiões prolongacionais[4]. Em seguida, a partir do evento-cabeça faz recursivamente a varredura à esquerda e depois à direita de todas as regiões prolongacionais subjacentes, selecionando seus eventos- cabeça através do vetor de pesos calculado anteriormente. O algoritmo, que tem como saída a árvore prolongacional, é mostrado a seguir:

1. Algoritmo ControiArvoreProlongacional()
2. Inicializar o vetor **s** com a superfície de eventos.
3. Inicializar l com a dimensão da superfície.
4. Inicializar h com a posição do evento-cabeça da amostra.
5. Inicializar com zeros os vetores **m** e **d**, ambos de dimensão l.
6. Fazer $m_h = 1$.
7. PesquisaEsquerda(s,h,m,d,l)
8. PesquisaDireita(s,h,m,d,l)
9. ImprimeDependencias()

A seguir, estão os algoritmos de varredura à esquerda e à direita, respectivamente:

```
PesquisaEsquerda(s,h,m,d,l)
{
  Se(h && !m[h-1]) {
     max = -1;
     i = h-1;
     Enquanto( i >= 0 && !m[i]) {
Se(max < s[i]){
   max = s[i];
   hd = i;
}
i--;
  }
  m[hd] = 1;
```

[4]Os casos-limite nos quais o evento-cabeça coincide com o primeiro ou com o último evento não alteram o funcionamento do algoritmo, pois este considera que são regiões prolongacionais vazias, respectivamente à esquerda ou à direita.

```
  Se(hd)
     PesquisaEsquerda(s,hd,m,d);
  Senão
     retorna;
  d[hd] = h;
  PesquisaDireita(s,hd,m,d,l);
}

PesquisaDireita(s,h,m,d,l)
{
  Se( h < l-1 && !m[h+1]) {
     max = -1;
     i = h + 1;
     Enquanto (i <= l-1 && !m[i]) {
Se(max < s[i]){
   max = s[i];
   hd = i;
}
i++;
  }
  m[hd] = 1;
  Se(hd < l-1)
     PesquisaDireita(s,hd,m,d,l);
  Senão
     retorna;
  d[hd] = h;
  PesquisaEsquerda(s,hd,m,d,l);
}
```

Para a montagem do resultado final da análise emprega-se o mesmo algoritmo de cálculo de dependências já mostrado e descrito no capítulo 6, página 100.

Para ilustrar o funcionamento do algoritmo, suponhamos que se tenha uma superfíce constituída de seis eventos, com o evento-cabeça na posição 3 e cujos pesos formam o vetor `[2 3 1 x 3 2]` (o valor `x`, na posição do evento-cabeça, não é utilizado). Partindo-se do evento-cabeça tem-se a seqüência de regiões prolongacionais:

- Região à esquerda[3]: intervalo 0–2, onde é escolhida posição 1.
- Região à esquerda[1]: único elemento na posição 0.
- Região à esquerda[0]: não existe.
- Região à direita[0]: não existe.
- Região à direita[1]: único elemento na posição 2.

- Região à direita[3]: intervalo 4–5, onde é escolhida posição 4.
- Região à direita[4]: único elemento na posição 5.
- Região à esquerda[5]: não existe.
- Região à esquerda[4]: não existe.

Ao construir-se uma árvore de dependências entre os eventos utilizando-se este processo recursivo, obtém-se a figura 7.10.

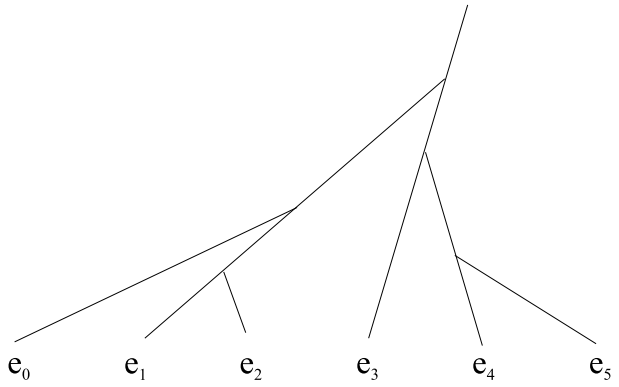

Figura 7.10: Exemplo de árvore prolongacional.

7.4.2 Implementação das Regras Preferenciais

Para o cálculo do vetor de pesos foi visto que é necessário que cada evento da superfície seja avalidado por cada uma das Regras Preferenciais. A seguir, é descrita a implementação de cada uma delas.

RPRP 1 – Importância na Redução Temporal

Levando-se em conta que o arquivo de comunicação `timespan.tmp` tem por conteúdo os eventos-cabeça de cada grupo de cada nível na Redução Temporal, foi utilizado um método simples para avaliar a importância de um dado evento na Redução Temporal. Considerando-se L o número de níveis da Redução Temporal, c uma constante qualquer (válida para todos os eventos ao calcular seus pesos relativos) e l o nível do evento, o peso w pode ser calculado como:

$$w = c\frac{l}{L-1}$$

É possível observar-se que, quanto mais próximo o evento estiver do nível contendo o evento-cabeça da amostra, maior será o seu peso e, daí, sua importância na Redução Temporal.

RPRP 2 – Segmentação Temporal

Devido ao sistema do vetor de pesos adotado, a RPM 2 está embutida na regra anterior, já que a importância relativa de cada evento é calculada para todos os níveis da Redução Temporal. Portanto, ao realizar o processo recursivo e construir a árvore prolongacional os pesos relativos de cada evento se encarregam de organizar suas elaborações (dependências).

RPRP 3 – Conexão Prolongacional

Esta regra preferencial é dividida em duas partes. Na primeira parte é verificado se o evento atual é consonante com todas as alturas do acorde em vigor no momento de seu início. Em caso positivo o evento é pontuado. A segunda parte diz respeito à distância do evento atual até a tônica local. Neste caso o critério de pontuação é definido pela equação abaixo:

$$w_i = \begin{cases} w_i + c & \text{para } d = 0 \\ w_i + \frac{c}{d} & \text{para } d \neq 0 \end{cases}$$

onde w_i é o peso do i-ésimo evento, d é a distância até a tônica local e c é uma constante qualquer de incremento nos pesos. A equação mostra claramente que quanto maior a distância de um evento até sua tônica local, menor será o incremento em seu peso.

RPRP 4 – Importância Prolongacional

A presente regra recai no mesmo caso da RPM 2, já que trata da importância entre dois eventos que formam uma região prolongacional. Sendo assim, o pré-processamento do vetor de pesos já garante a aplicação de seus princípios.

RPRP 5 – Paralelismo

Devido aos mesmos motivos já expostos no capítulo 6, página 116, a RPM 5 não é abordada nesta implementação. Além disso, como é possível ver nos resultados tanto da Redução Temporal quanto da Redução Prolongacional, a conjunção das regras e o caráter automático de seu processamento de alguma forma garantem análises paralelas de passagens paralelas.

RPRP 6 – Estrutura Normativa Prolongacional

Esta regra realiza a conferência se o evento atual faz parte de uma cadência. Para isto, verifica no arquivo de cadências `cadences.tmp` se o evento atual está no interior de alguma delas. Se estiver, pontua o evento e, em caso contrário, o deixa inalterado.

7.5 Resultados

7.5.1 Testes de Validação

Neste capítulo sobre a Redução Prolongacional, os testes de validação apresentam-se divididos em duas partes distintas, a saber, testes de validação do Analisador Harmônico e testes de validação empregando amostras musicais.

Analisador Harmônico

Para testar o analisador harmônico foram empregados os dois trechos[5] mostrados nas figuras 7.11 e 7.12, ambos com caráter modulatório. A seqüência de passos para a realização de uma análise é a seguinte:

1. Ler um arquivo MIDI transferindo seus dados para uma lista na memória.
2. A partir dos eventos presentes na lista criada anteriormente, montar uma seqüência de acordes que constituirá a entrada para o analisador.

   Por exemplo, considerando o trecho musical representado na figura 7.11, tem-se a matriz (seqüência de acordes) abaixo[6]:

```
48  53  50  55  48  45  50  43  55  53  52  53  55  48
60  62  65  65  64  64  62  62  62  59  60  62  62  64
67  69  69  67  67  67  66  67  65  67  67  69  67  67
76  74  72  71  72  72  72  71  71  74  72  72  71  72
```

3. Entrar com a matriz no analisador, o qual devolve o trecho analisado.

A seguir são mostrados os resultados das análises dos dois trechos musicais já citados.

Resultado a análise do trecho na figura 7.11

```
Acorde 0: [ 48 60 67 76 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: I
Cadencial: Não
Fundamental: Dó
Estado: Est. Fundamental

Acorde 1: [ 53 62 69 74 ]
Tonalidade e modo: Dó maior
```

[5] Como pode ser observado, estes trechos foram escritos para serem diretamente lidos pelo analisador, não necessitando, assim, de pré-processamento.

[6] Observar que a primeira linha da matriz representa o baixo e a última o soprano, como já explicado na subseção 7.3.5.

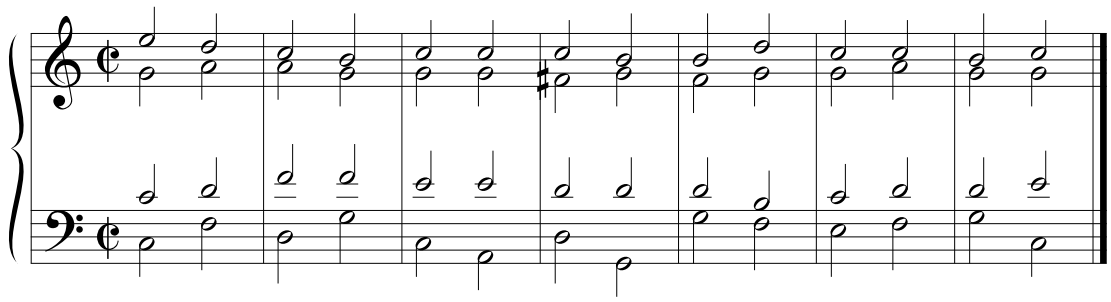

Figura 7.11: Exemplo musical de CSP número 1.

```
Tipo: menor
Função: II
Cadencial: Não
Fundamental: Ré
Estado: Primeira Inv.

Acorde 2: [ 50 65 69 72 ]
Tonalidade e modo: Dó maior
Tipo: menor_7m
Função: II
Cadencial: Não
Fundamental: Ré
Estado: Est. Fundamental

Acorde 3: [ 55 65 67 71 ]
Tonalidade e modo: Dó maior
Tipo: maior_7m
Função: V
Cadencial: Não
Fundamental: Sol
Estado: Est. Fundamental

Acorde 4: [ 48 64 67 72 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: I
Cadencial: Sim
Fundamental: Dó
Estado: Est. Fundamental

Acorde 5: [ 45 64 67 72 ]
```

```
Tonalidade e modo: Dó maior
Tipo: menor_7m
Função: VI
Cadencial: Não
Fundamental: La
Estado: Est. Fundamental

Acorde 6: [ 50 62 66 72 ]
Tonalidade e modo: Sol maior
Tipo: maior_7m
Função: V
Cadencial: Não
Fundamental: Ré
Estado: Est. Fundamental

Acorde 7: [ 43 62 67 71 ]
Tonalidade e modo: Sol maior
Tipo: maior
Função: I
Cadencial: Sim
Fundamental: Sol
Estado: Est. Fundamental

Acorde 8: [ 55 62 65 71 ]
Tonalidade e modo: Dó maior
Tipo: maior_7m
Função: V
Cadencial: Não
Fundamental: Sol
Estado: Est. Fundamental

Acorde 9: [ 53 59 67 74 ]
Tonalidade e modo: Dó maior
Tipo: maior_7m
Função: V
Cadencial: Não
Fundamental: Sol
Estado: Terceira Inv.

Acorde 10: [ 52 60 67 72 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: I
```

```
Cadencial: Não
Fundamental: Dó
Estado: Primeira Inv.

Acorde 11: [ 53 62 69 72 ]
Tonalidade e modo: Dó maior
Tipo: menor_7m
Função: II
Cadencial: Não
Fundamental: Ré
Estado: Primeira Inv.

Acorde 12: [ 55 62 67 71 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: V
Cadencial: Não
Fundamental: Sol
Estado: Est. Fundamental

Acorde 13: [ 48 64 67 72 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: I
Cadencial: Sim
Fundamental: Dó
Estado: Est. Fundamental
```

Resultado a análise do trecho na figura 7.12

```
Acorde 0: [ 60 60 64 67 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: I
Cadencial: Não
Fundamental: Dó
Estado: Est. Fundamental

Acorde 1: [ 59 60 64 67 ]
Tonalidade e modo: Dó maior
Tipo: maior_7M
Função: I
Cadencial: Não
```

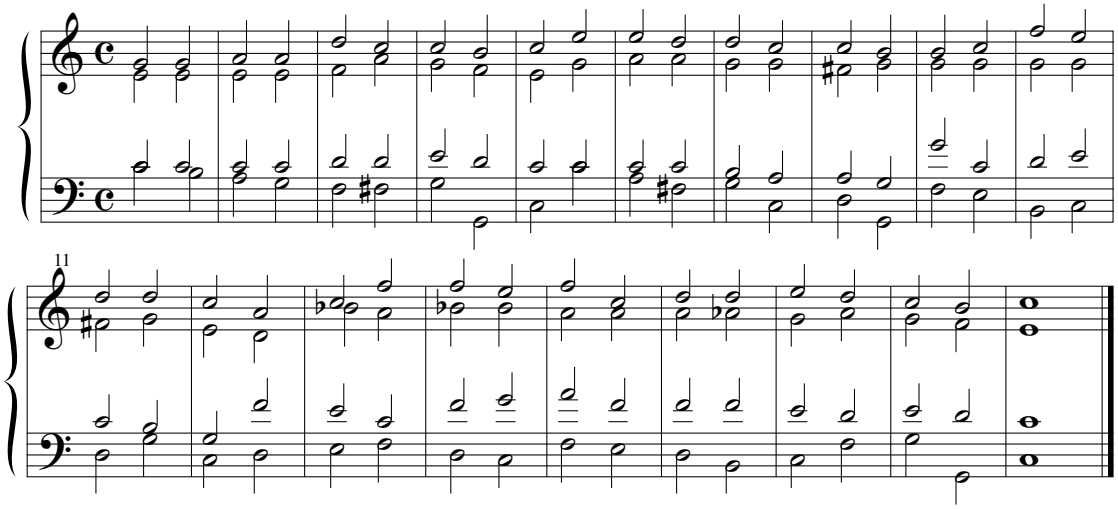

Figura 7.12: Exemplo musical de CSP número 2.

```
Fundamental: Dó
Estado: Terceira Inv.

Acorde 2: [ 57 60 64 69 ]
Tonalidade e modo: Dó maior
Tipo: menor
Função: VI
Cadencial: Não
Fundamental: Lá
Estado: Est. Fundamental

Acorde 3: [ 55 60 64 69 ]
Tonalidade e modo: Dó maior
Tipo: menor_7m
Função: VI
Cadencial: Não
Fundamental: Lá
Estado: Terceira Inv.

Acorde 4: [ 53 57 65 74 ]
Tonalidade e modo: Dó maior
Tipo: menor
Função: II
Cadencial: Não
Fundamental: Re
Estado: Primeira Inv.
```

```
Acorde 5: [ 54 62 69 72 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: I
Cadencial: Não
Fundamental: Dó
Estado: Segunda Inv.

Acorde 6: [ 55 64 67 72 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: I
Cadencial: Não
Fundamental: Dó
Estado: Segunda Inv.

Acorde 7: [ 43 62 65 71 ]
Tonalidade e modo: Dó maior
Tipo: maior_7m
Função: V
Cadencial: Não
Fundamental: Sol
Estado: Est. Fundamental

Acorde 8: [ 48 60 64 72 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: I
Cadencial: Sim
Fundamental: Dó
Estado: Est. Fundamental

Acorde 9: [ 60 60 67 76 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: I
Cadencial: Não
Fundamental: Dó
Estado: Est. Fundamental

Acorde 10: [ 57 60 69 76 ]
Tonalidade e modo: Dó maior
```

```
Tipo: menor
Função: VI
Cadencial: Não
Fundamental: Lá
Estado: Est. Fundamental

Acorde 11: [ 54 60 69 74 ]
Tonalidade e modo: Sol maior
Tipo: maior
Função: IV
Cadencial: Não
Fundamental: Dó
Estado: Segunda Inv.

Acorde 12: [ 55 59 67 74 ]
Tonalidade e modo: Sol maior
Tipo: maior
Função: I
Cadencial: Não
Fundamental: Sol
Estado: Est. Fundamental

Acorde 13: [ 48 57 67 72 ]
Tonalidade e modo: Sol maior
Tipo: diminuto_7m
Função: II
Cadencial: Não
Fundamental: Lá
Estado: Primeira Inv.

Acorde 14: [ 50 57 66 72 ]
Tonalidade e modo: Sol maior
Tipo: maior_7m
Função: V
Cadencial: Não
Fundamental: Re
Estado: Est. Fundamental

Acorde 15: [ 43 55 67 71 ]
Tonalidade e modo: Sol maior
Tipo: maior
Função: I
Cadencial: Sim
```

```
Fundamental: Sol
Estado: Est. Fundamental

Acorde 16: [ 53 67 67 71 ]
Tonalidade e modo: Dó maior
Tipo: maior_7m
Função: V
Cadencial: Não
Fundamental: Sol
Estado: Terceira Inv.

Acorde 17: [ 52 60 67 72 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: I
Cadencial: Não
Fundamental: Dó
Estado: Primeira Inv.

Acorde 18: [ 47 62 67 77 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: IV
Cadencial: Não
Fundamental: Fá
Estado: Segunda Inv.

Acorde 19: [ 48 64 67 76 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: I
Cadencial: Não
Fundamental: Dó
Estado: Est. Fundamental

Acorde 20: [ 50 60 66 74 ]
Tonalidade e modo: Dó maior
Tipo: maior_7m
Função: V_V
Cadencial: Não
Fundamental: Re
Estado: Est. Fundamental
```

```
Acorde 21: [ 55 59 67 74 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: V
Cadencial: Não
Fundamental: Sol
Estado: Est. Fundamental

Acorde 22: [ 48 55 64 72 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: I
Cadencial: Sim
Fundamental: Dó
Estado: Est. Fundamental

Acorde 23: [ 50 65 62 69 ]
Tonalidade e modo: Dó maior
Tipo: menor
Função: II
Cadencial: Não
Fundamental: Re
Estado: Est. Fundamental

Acorde 24: [ 52 64 70 72 ]
Tonalidade e modo: Fá maior
Tipo: maior
Função: IV
Cadencial: Não
Fundamental: Sib
Estado: Segunda Inv.

Acorde 25: [ 53 60 69 77 ]
Tonalidade e modo: Fá maior
Tipo: maior
Função: I
Cadencial: Não
Fundamental: Fá
Estado: Est. Fundamental

Acorde 26: [ 50 65 70 77 ]
Tonalidade e modo: Fá maior
Tipo: maior
```

```
Função: IV
Cadencial: Não
Fundamental: Sib
Estado: Primeira Inv.

Acorde 27: [ 48 67 70 76 ]
Tonalidade e modo: Fá maior
Tipo: maior_7m
Função: V
Cadencial: Não
Fundamental: Dó
Estado: Est. Fundamental

Acorde 28: [ 53 69 69 77 ]
Tonalidade e modo: Fá maior
Tipo: maior
Função: I
Cadencial: Sim
Fundamental: Fá
Estado: Est. Fundamental

Acorde 29: [ 52 65 69 72 ]
Tonalidade e modo: Fá maior
Tipo: maior_7M
Função: I
Cadencial: Não
Fundamental: Fá
Estado: Terceira Inv.

Acorde 30: [ 50 65 69 74 ]
Tonalidade e modo: Fá maior
Tipo: menor
Função: VI
Cadencial: Não
Fundamental: Re
Estado: Est. Fundamental

Acorde 31: [ 47 65 68 74 ]
Tonalidade e modo: Dó maior
Tipo: diminuto_7
Função: Desconhecido.
Cadencial: Não
Fundamental: Fá
```

```
Estado: Segunda Inv.

Acorde 32: [ 48 64 67 76 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: I
Cadencial: Não
Fundamental: Dó
Estado: Est. Fundamental

Acorde 33: [ 53 62 69 74 ]
Tonalidade e modo: Dó maior
Tipo: menor
Função: II
Cadencial: Não
Fundamental: Re
Estado: Primeira Inv.

Acorde 34: [ 55 64 67 72 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: I
Cadencial: Não
Fundamental: Dó
Estado: Segunda Inv.

Acorde 35: [ 43 62 65 71 ]
Tonalidade e modo: Dó maior
Tipo: maior_7m
Função: V
Cadencial: Não
Fundamental: Sol
Estado: Est. Fundamental

Acorde 36: [ 48 60 64 72 ]
Tonalidade e modo: Dó maior
Tipo: maior
Função: I
Cadencial: Sim
Fundamental: Dó
Estado: Est. Fundamental
```

Validação de Amostras

Conforme os capítulos precedentes, os testes de validação realizados aqui compõem-se dos dois temas do primeiro e do quarto movimentos da Sinfonia nº 40 de Mozart. Entretanto, desta vez não são apresentados exemplos unicamente monódicos já que a harmonia na Redução Prolongacional é fator de imprescindível na análise de trechos musicais. Assim, somente as amostras com redução orquestral são apresentadas.

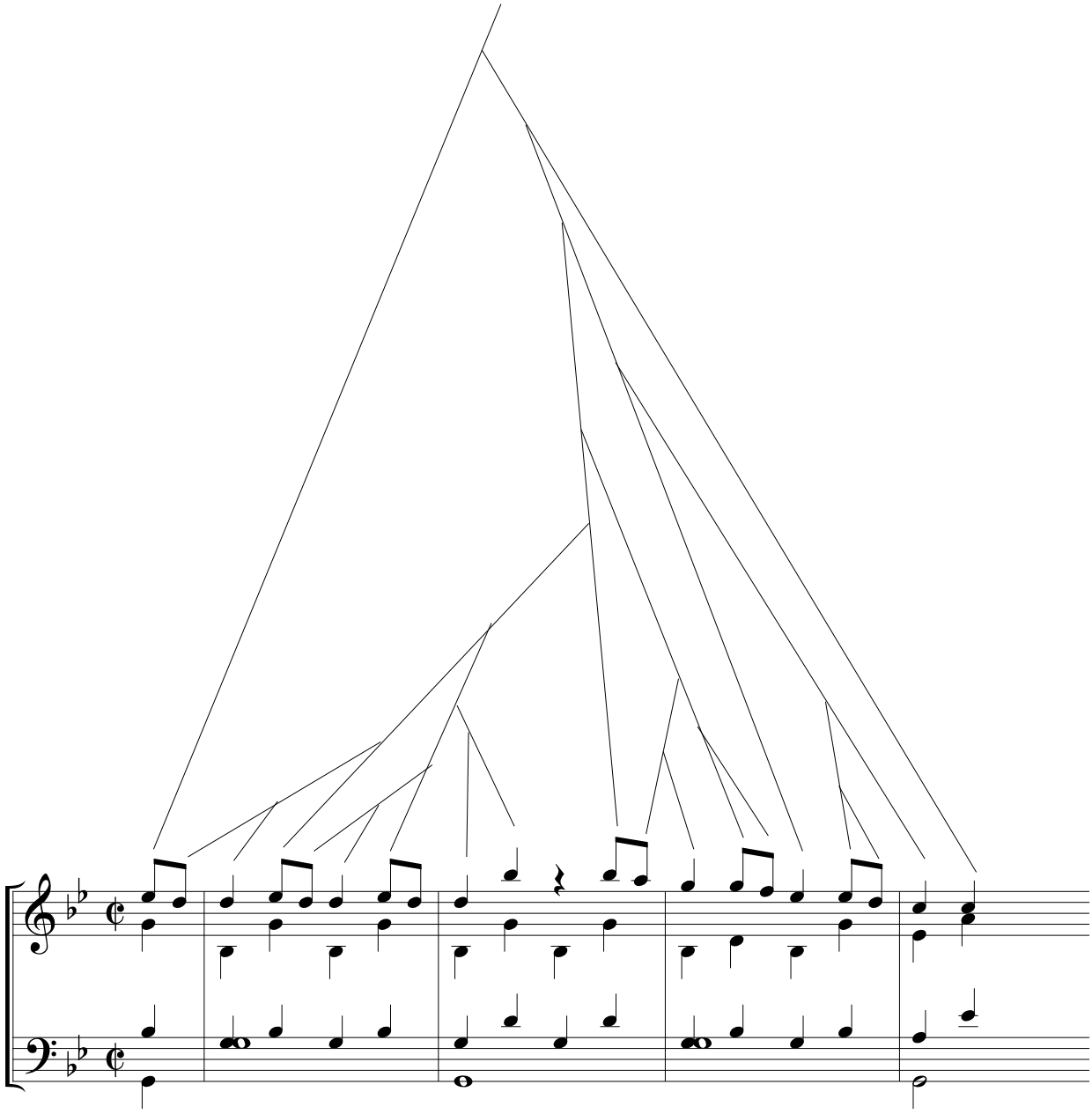

Figura 7.13: Redução Prolongacional para o primeiro tema do primeiro movimento da Sinfonia nº 40 de W.A. Mozart.

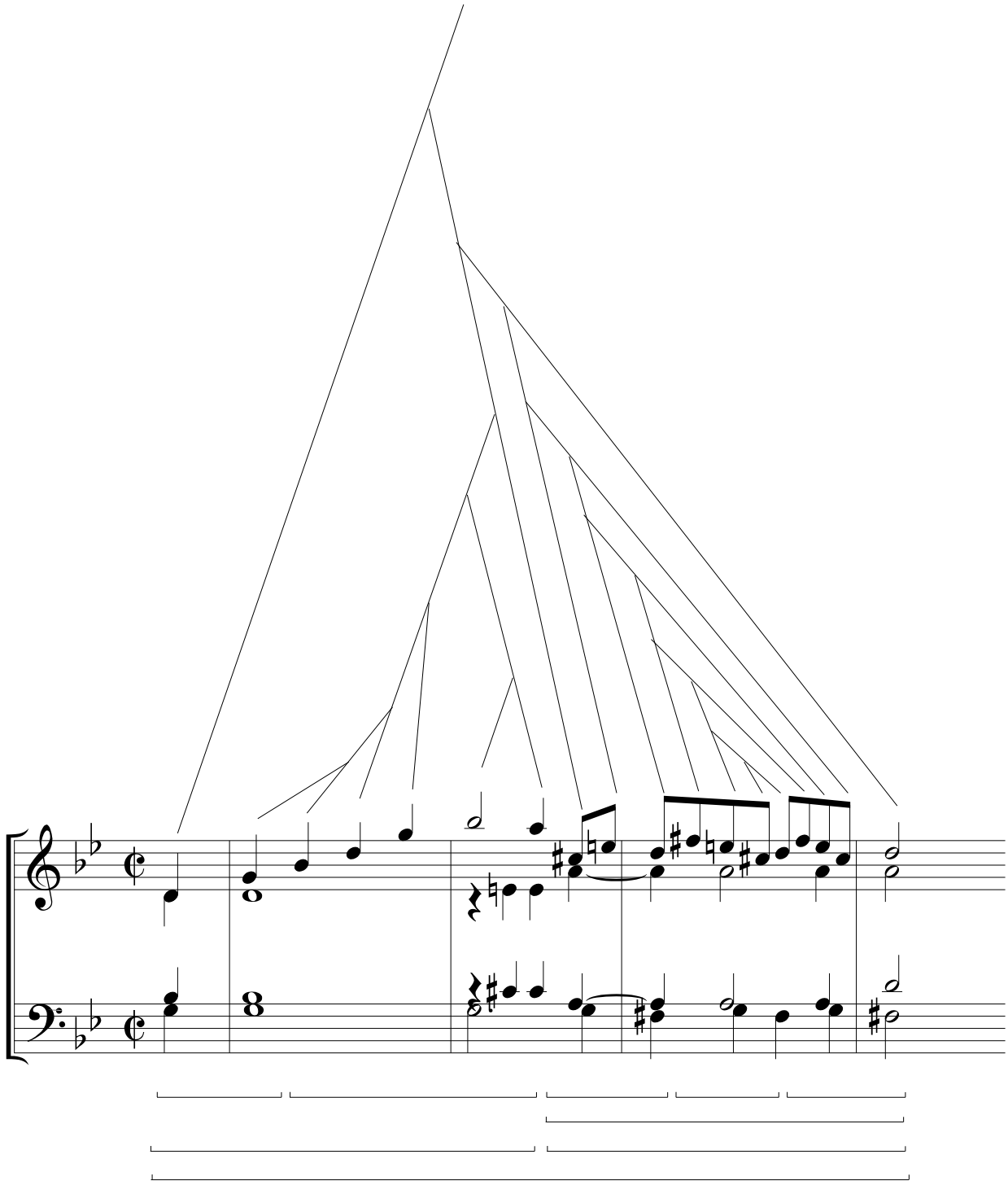

Figura 7.14: Redução Prolongacional para o primeiro tema do quarto movimento da Sinfonia nº 40 de W.A. Mozart.

7.5.2 Análises

As obras analisadas neste capítulo são as mesmas utilizadas no componente precedente, a Redução Prolongacional. A ordem de apresentação também é mantida, ou seja, inicialmente o Coral *Christus, der ist mein Leben* (BWV 95), o qual pode ser visto na figura 7.15, seguido do Coral nº 52 da *Paixão Segundo São João*, apresentado na figura 7.16, ambos de Johann Sebastian Bach, e finalizando com do tema do 3º movimento do Quarteto de Cordas Op. 18 nº 5, de Ludwig van Beethoven, presente na figura 7.17. Ao manter esta ordem procura-se facilitar a comparação entre os diversos estágios da análise de uma obra.

7.6 Discussão dos Resultados

7.6.1 Testes de Validação

Resultados do Analisador Harmônico

Para tornar possível um melhor olhar sobre os resultados obtidos com os dois exemplos apresentados, serão construídas duas tabelas, uma para cada caso, com os índices de acerto e erro para cada um dos parâmetros resultantes da análise. Desta forma, torna-se mais fácil a visualização e o estudo do comportamento do algoritmo. Para facilitar a leitura, os parâmetros resultantes da análise são relembrados abaixo:

Tonalidade: A tonalidade na qual está o acorde.

Modo: O modo da tonalidade (maior ou menor).

Tipo: A qualidade do acorde (perfeito maior, perfeito menor, diminuto, etc).

Função: Função harmônica do acorde dentro da tonalidade na qual está inserido.

Cadencial: Variável boolena indicando se o acorde é o termo de uma cadência ou não.

Fundamental: Fundamental do acorde.

Estado: Estado do acorde (estado fundamental e inversões).

Exemplo 1 Para o exemplo 1 não houve nenhum erro em qualquer dos parâmetros a serem analisados em cada um dos 14 acordes do trecho.

Exemplo 2 O exemplo 2, ao contrário do anterior, apresentou alguns erros quando de sua análise. Apesar de que foram relativamente poucos (6 acordes apresentando erros num conjunto de 37, representando pouco mais de 16% do conjunto), em qualquer tipo de análise harmônica o erro deve ser próximo de zero, pois, em caso contrário, poderá comprometer seriamente os resultados da análise. Dentre o conjunto dos acordes que apresentaram erros[7], somente um deles, o de número 31, não pertence à categoria sétima de dominante. Além disso, todos eles, excetuando-se o de número 31, estão na primeira inversão. Tais resultados podem ser vistos na tabela 7.1.

[7]Os acordes que apresentaram erros de análise foram os de número 5, 11, 13, 18, 24 e 31.

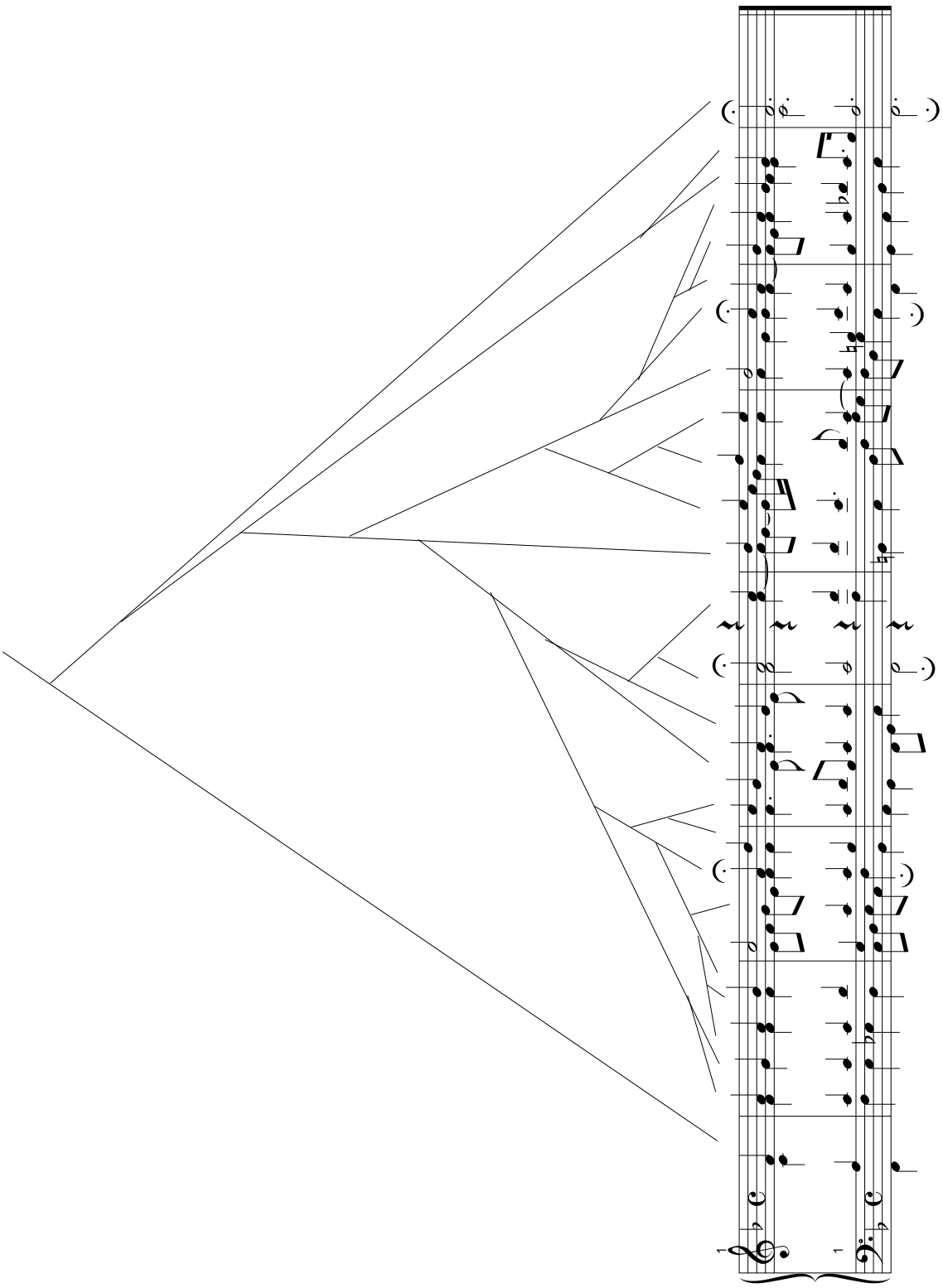

Figura 7.15: Redução Temporal do Coral *Christus, der ist mein Leben* (BWV 95), de J.S. Bach.

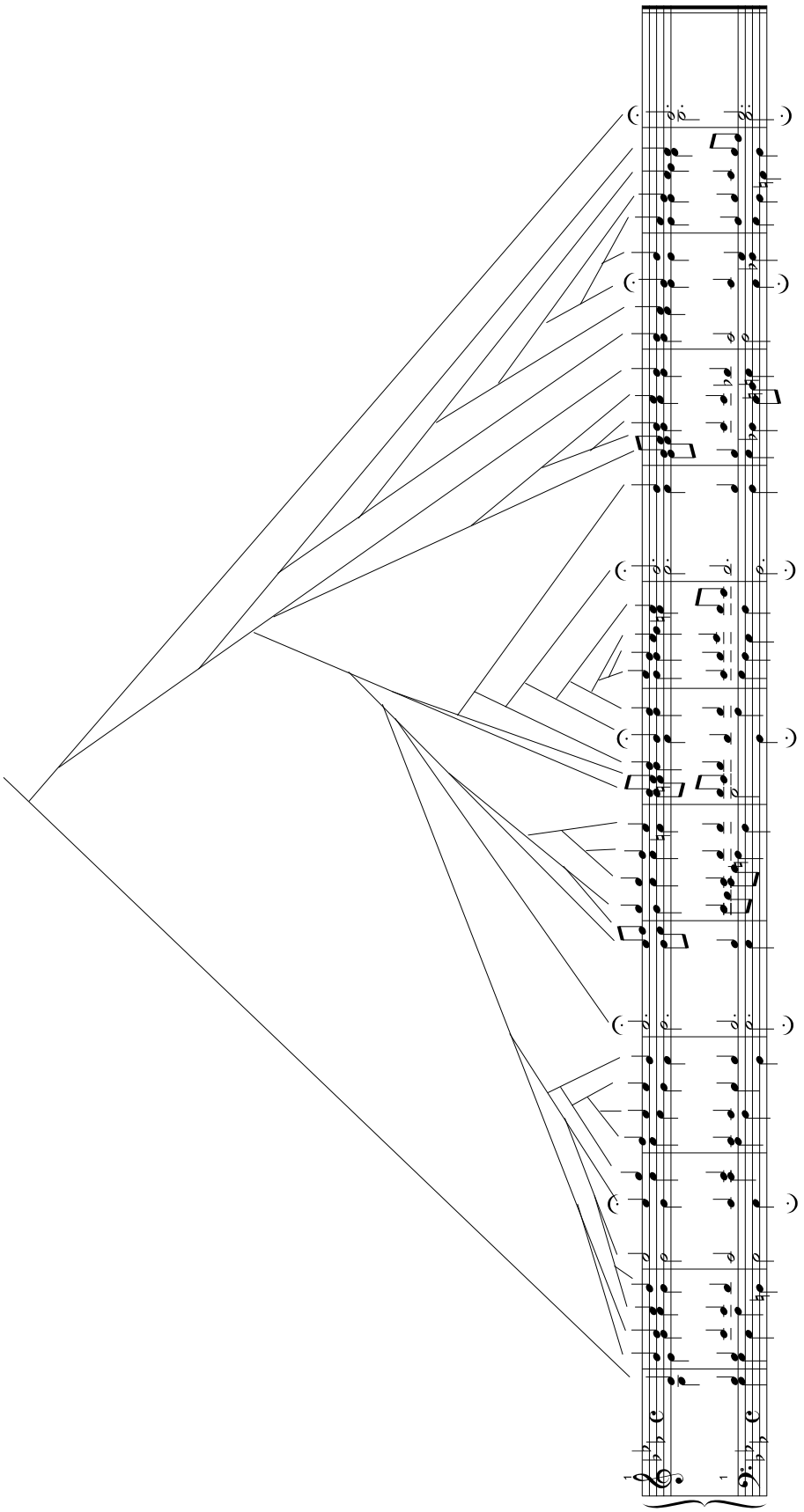

Figura 7.16: Redução Temporal do Coral nº 52 da *Paixão Segundo São João*, de J.S. Bach.

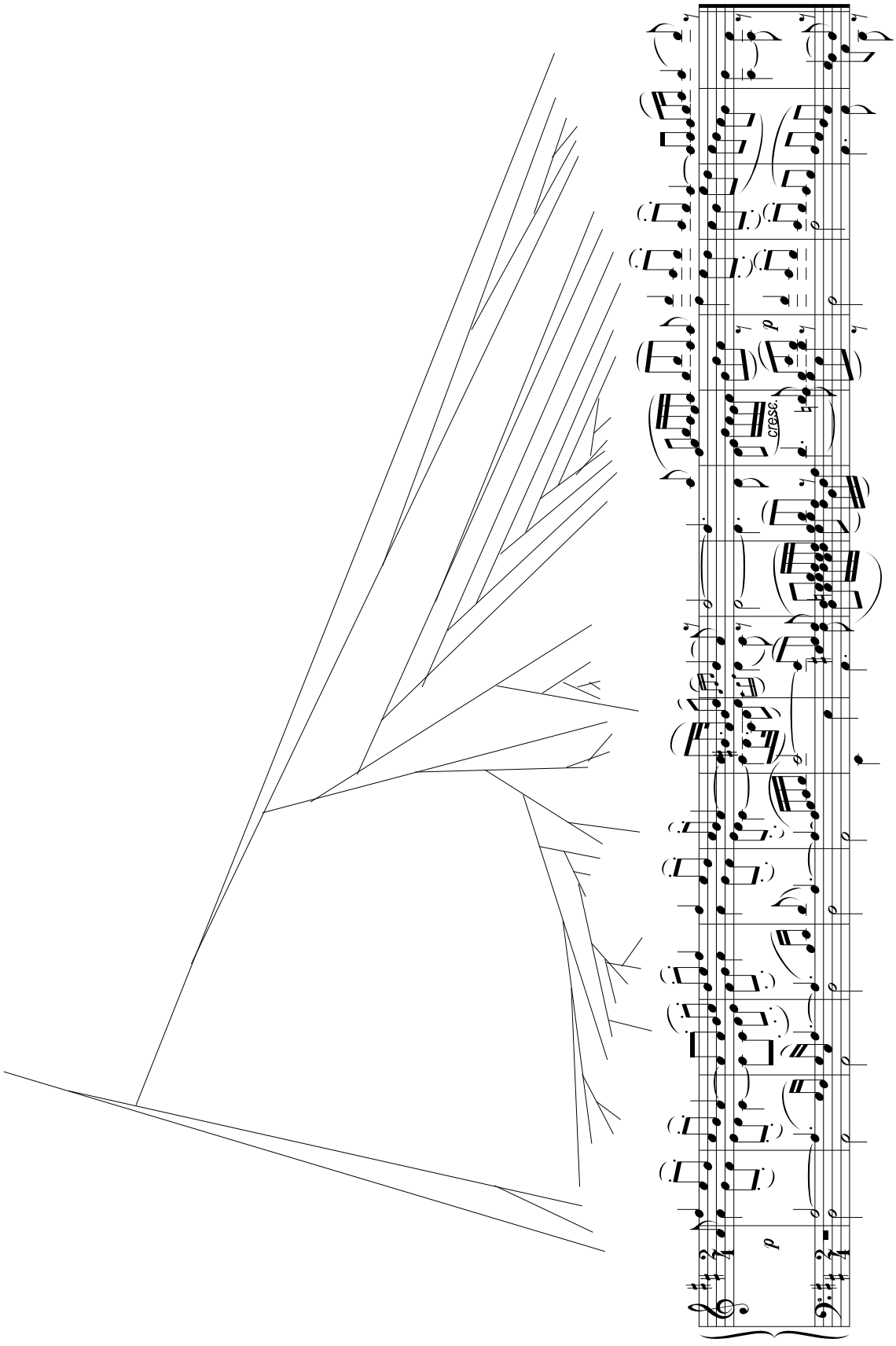

Figura 7.17: Redução Prolongacional do tema do 3º movimento do Quarteto de Cordas Op. 18 nº 5, de L.van Beethoven.

Acorde	Tonalidade	Modo	Tipo	Função	Cadencial	Fundamental	Estado
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	1	1	0	1	1
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0
11	0	0	1	1	0	1	1
12	0	0	0	0	0	0	0
13	0	0	1	0	0	0	0
14	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0
18	0	0	1	1	0	1	1
19	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0
24	0	0	1	1	0	1	1
25	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0
31	0	0	0	1	0	1	1
32	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0
Total	0	0	5	5	5	0	5

Tabela 7.1: Dados da análise do exemplo 2. Para cada parâmetro de cada acorde, 0 significa acerto e 1 significa erro.

Validação de Amostras

Em relação à Redução Prolongacional do primeiro tema do primeiro movimento da Sinfonia nº 40 de W.A. Mozart, mostrada na figura 7.13 (página 168), pode ser observado que, de modo semelhante ao que aconteceu com o comportamento da mesma amostra na Redução Temporal, a carência de completude na harmonia até o instante do final da amostra, ou seja, a falta de uma cadência, fez com que a análise resultante valorizasse excessivamente questões melódicas e motívicas. Isto demonstra mais uma vez que, para uma correta realização da Redução Prolongacional é necessária uma clara constituição da seção harmônica da amostra.

No que diz respeito à outra amostra, o primeiro tema do quarto movimento da mesma sinfonia do mesmo autor e que pode ser vista na figura 7.14 (página 169), por estar a harmonia claramente indicando uma cadência, a Redução Prolongacional resultante mostra esta delimitação harmônica, indicando a altura Ré como evento-cabeça da amostra e mostrando as dependências de cada evento e, conseqüentemente, suas direcionalidades.

7.6.2 Análises

***Christus, der ist mein Leben* (BWV 95), de J.S. Bach**

Este coral, visto na figura 7.15 (página 172), apresenta uma Redução Prolongacional coerente com sua estrutura, realçando as cadências e mostrando as dependências dos seus elementos constituintes. Pode-se observar ainda, na figura correspondente, o caráter relevante dado à cadência final, constituindo todo o corpo do coral uma elaboração em sua direção, fato corroborado por esta mostrar-se como uma prolongação do evento-cabeça da peça.

Tema do 3º Movimento do Quarteto de Cordas Op. 18 nº 5, de L.van Beethoven)

De modo semelhante ao ocorrido com os componentes Estrutura Métrica e Redução Temporal, esta peça, a qual pode ser vista na figura 7.17 (página 174), foi a que apresentou os resultados mais problemáticos. O mais notável fato é aquele no que se percebe que, sendo a tonalidade principal do trecho aquela de Ré maior, o último evento da superfície, que deveria ser subordinado ao evento-cabeça (evento inicial), na verdade o é a um outro evento, segundo na hierarquia. Este evento pode ter adquirido relevância devido à modulação à dominante na metade da peça, já que o citado evento é a fundamental da função *dominante-da-dominante*. Entretanto, a mais provável causa da deficiência analítica na Redução Prolongacional desta peça seja devido à propagação do erro proveniente do calculo de sua Estrutura Métrica (ver seção 5.8.2, página 98). Este erro prejudicou o cálculo da Redução Temporal e, finalmente, propagou-se até a Redução Prolongacional. Não obstante estes problemas expostos, a análise da peça apresenta pontos positivos, tais como a correta detecção da modulação à dominante (no meio da peça) e a importância da altura lá durante toda a peça, por ser esta o evento-cabeça, do qual todos os outros eventos são

prolongações, e também pelo grande número de prolongações desta altura no corpo da peça.

Coral nº 52 da Paixão Segundo São João, de J.S. Bach

Este coral pode ser visto na figura 7.16 (página 173), sendo que sua Redução Prolongacional apresenta todas as características comumente presentes neste tipo de análise, tais como, detecção das cadências, evento final como prolongação do evento-cabeça e diversos níveis de prolongação entre os eventos da peça. Como geralmente acontece, é possível ver-se na figura correspondente como todo o corpo da peça mostra-se com uma elaboração em direção à cadência final. Em outras palavras, é possível observar-se como todos os eventos da peça, com exceção do evento-cabeça, são dependentes do evento final, sendo este dependente do evento-cabeça.

Capítulo 8

Conclusões

8.1 Momento Atual

No decorrer deste trabalho foram feitas análises de diversas amostras e análises completas (peças inteiras) de quatro obras. Destas, uma foi analisada apenas pelo primeiro componente, Estrutura de Agrupamento, pelas razões já expostas na seção 5.7.2 (página 90). Trata-se da primeira das Três Peças (para clarinete solo) de I. Stravinsky. Nos componentes seguintes da teoria esta peça foi substituída pelo coral nº 52 da Paixão Segundo São João, de J.S. Bach.

No tocante ao desempenho de cada componente, a Estrutura Métrica mostrou-se o mais delicado no que diz respeito à qualidade dos seus resultados, demandando, talvez, uma revisão na implementação de suas regras e um procedimento automático através dos quais os parâmetros do algoritmo genético utilizado sejam otimizados. No outro extremo, a Redução Prolongacional foi o componente que apresentou resultados mais fiéis àqueles que poderiam ser produzidos por um ser humano. Seus resultados somente não foram melhores, provavelmente, devido aos erros propagados desde a Estrutura Métrica, através da Redução Temporal, e que influenciam, ainda que indiretamente, nos resultados da Redução Prolongacional. Entre estes dois extremos, aparecem a Estrutura de Agrupamento e a Redução Temporal, produzindo ambos resultados satisfatórios.

Finalmente, considerando-se o conjunto total de amostras utilizado neste trabalho e também os resultados obtidos através dele pode-se concluir que o sistema como um todo é capaz de produzir análises de qualidade razoável, entendendo-se esta expressão como análises cuja qualidade é assegurada por comparação à análises subjetivas das mesmas obras.

8.2 Possibilidades Futuras

Este trabalho, tal como foi apresentado, procurou cobrir de forma mais ampla possível, as possibilidades analíticas computacionais presentes na TGMT. Entretanto, devido à sua própria natureza, a qual é uma implementação computacional de uma teoria analítica pro-

posta independentemente de qualquer emprego ou utilização de computadores, o trabalho enfrentou as dificuldades costumeiras em tal tipo de aplicação. Dentre elas as mais importantes são a falta de uma formalização rigorosa das regras da teoria e, em segundo lugar, a não-adeqüabilidade destas às ferramentas existentes de Inteligência Computacional. É claro que a primeira causa motiva a segunda.

Outro aspecto importante a ser considerado é aquele da transcrição dos resultados oriundos do sistema implementado. Neste trabalho, esta foi inteiramente realizada manualmente, isto é, os resultados numéricos foram lidos (ver apêndice C) e realizada sua transcrição em figuras cujo conteúdo é um misto de grafia musical e signos gráficos. O maior problema com este processo, ao lado da temporalmente dispendiosa tarefa, é a possibilidade, nem pequena e nem remota, do surgimento de erro humano na passagem do universo numérico para o universo gráfico. Tendo isto em mente, torna-se claro que uma transcrição automática dos resultados seria de grande utilidade.

Finalmente, com a massa de dados resultantes de uma determinada análise é-se tentado a imaginar sua possível utilização num processo de engenharia reversa, ou seja, realizar com aqueles um determinado tipo de processamento que possa gerar uma outra composição, semelhante àquela que foi analisada, no mesmo estilo e com as mesmas características.

Diante de todo este contexto, portanto, gera-se um cenário indicando claramente as vias nas quais o trabalho demanda uma continuação.

8.2.1 Formalização Mais Rigorosa

De um modo geral, na teoria de Lerdahl & Jackendoff (1996) tanto a formulação das Regras de Boa–Formatividade quanto aquela das Regras Preferenciais carecem de uma formulação mais objetiva e mais rigorosa. Isto é devido, indubitavelmente, ao fato de tratar-se de uma teoria destinada a ser utilizada por seres humanos e que, por isso mesmo, conta-se com a intuição do analista que a emprega.

Porém, isto que pode até ser considerado uma vantagem quando utilizado por um analista humano, já que lhe permite liberdades de escolha baseadas em sua intuição, transforma-se em grande dificuldade ao tentar-se implementar a teoria tal como ela foi originalmente idealizada, já que o elemento "intuição" deve ser descartado.

A solução, portanto, mostra-se como sendo a realização de uma revisão da teoria, mais especificamente uma revisão na formulação de suas regras, tornando-as mais *formais* e, desta monta, possibilitando uma implementação mais direta e eficiente. Em outras palavras, esta tarefa significaria uma *formalização rigorosa* das regras de teoria, evitando, sem exceções, quaisquer tipos de subjetividades ou intuições.

8.2.2 Leitura Automática dos Resultados

Conforme indicado no na nota de rodapé 1 da página 180, já foi realizada uma tentativa de processar os resultados oriundos da Estrutura de Agrupamento visando sua integração

com o programa *LilyPond*[1]. Ainda que as figuras resultantes apresentassem desajustes e desalinhamentos quando as estruturas de agrupamento possuíssem mais do que dois níveis (o que é o usual), esta possibilidade de transcrição automática apresenta-se como extremamente valiosa e como um recurso a ser integrado à operacionalidade do sistema. Neste sentido, o trabalho a ser realizado incluiria o estudo dos fontes e das estruturas de dados do *LilyPond* visando a construção de interfaces entre ele e o sistema descrito neste trabalho. Um mínimo de quatro programas (interfaces) seriam necessários, um para cada componente da teoria[2].

8.2.3 Engenharia Reversa

Uma terceira possibilidade de continuação deste trabalho apresenta-se através do emprego dos dados capturados durante um determinado processo analítico como base para a geração de novas amostras musicais. Ainda que somente sob a forma de especulações e sem nenhum experimento concreto ainda realizado, duas possibilidades principais mostram-se como viáveis. A primeira diz respeito ao emprego direto dos dados disponíveis e, através de engenharia reversa, gerar novos textos musicais que possuíssem as mesmas características daquele de onde os dados são originários. Em seguida, existe a possibilidade da utilização dos dados como conjunto de treinamento de uma ferramenta de Inteligência Computacional (uma rede neural, por exemplo) que, por sua vez, faria a mediação entre os dados obtidos e a geração de novos textos. Numa primeira avaliação, o segundo método parece mais promissor, pois enquanto que o primeiro teria que contar com as ferramentas de natureza estocástica utilizadas no sistema aqui implementado (sistema *fuzzy* e algoritmos genéticos) para garantir alguma variedade na reversão do processo, o método utilizando treinamento através dos dados poderia garantir fidelidade aos dados originais sem perder a necessária generalidade.

---

[1] *LilyPond* é um programa que utiliza uma linguagem de marcação de mesmo nome e que tem por objetivo a edição automática de partituras musicais (Nienhuys & Nieuwenhuizen, 2007). Seu núcleo principal é escrito em C++ e a interface com o usuário utiliza o *Scheme*, o qual é um dialeto de *Lisp*. Devido ao fato do *LilyPond* possuir uma sintaxe bem definida e incluir uma ferramenta (no bojo de suas possibilidades educacionais) dedicada à marcação de grupos em uma partitura, foi tentada sua integração no presente sistema. Isto foi realizado através de um programa que, tendo como entrada um arquivo no formato *LilyPond* e outro com a indexação dos inícios e finais de cada agrupamento, faz o cruzamento das informações entre ambos e gera um novo arquivo *LilyPond* idêntico ao primeiro, porém com as marcações dos agrupamentos. Essa proposta produziu resultados corretos somente para as Estruturas de Agrupamento de apenas um ou dois níveis. Para estruturas mais profundas os resultados produzidos pelo *LilyPond* apresentaram diversos problemas de ajuste gráfico entre os grupos de um mesmo nível. Este é o motivo pelo qual os resultados das Estruturas de Agrupamento presentes neste trabalho não são apresentados em transcrição automática. De qualquer modo, a tendência natural deste projeto é sua futura integração com o *LilyPond*, visando a transcrição automática não apenas das Estruturas de Agrupamento, mas também dos resultados dos demais componentes do sistema.

[2] No caso da Estrutura de Agrupamento o programa, como dito antes, já foi implementado, restando, entretanto, para este componente, a realização de melhorias no modo através dos quais o LilyPond faz a marcação de grupos.

Apêndice A

Matrizes Esparsas

A.1 Generalidades

Na medida em que algumas das ferramentas empregadas são implementadas a partir de matrizes nas quais um pequeno número de elementos são válidos, faz-se necessário o emprego de operações que possam utilizar matrizes esparsas.

Vale lembrar que o emprego de um método de matrizes esparsas é justificado por duas razões, as quais nem sempre operam sem conflito. Estas são economia de tempo e economia de espaço em memória. Embora a primeira delas seja um benefício para qualquer sistema computacional, neste trabalho a preocupação é principalmente a economia de memória. Além disto, considerando que, neste trabalho, todas as ferramentas que operam com matrizes esparsas trabalham com matrizes quadradas, somente este caso será considerado ao longo desta seção.

Desde que as matrizes que devem ser representadas com o método escolhido não possuem um padrão definido no que tange suas características como esparsas, dos métodos mostrados em Press et al. (1997) optou-se por um que pode ser empregado em matrizes esparsas em geral e que é conhecido como *armazenagem esparsa por indexação de linhas*. Além da possibilidade de poder ser empregado com quaisquer tipos de matrizes esparsas, o método também apresenta a vantagem de requerer pouco espaço de armazenamento, sendo este igual a:

$$S = N + n + 1$$

onde S é o espaço necessário, N é o número de linhas (e de colunas) da matriz e n é o número de elementos válidos fora da diagonal (*off-diagonal*). Isto significa que, em uma matriz 10×10 e com cinco elementos fora da diagonal principal, somente 16 elementos da matriz necessitarão ser armazenados contra os 100 elementos da armazenagem da matriz em forma ordinária.

A forma de representação constando da referência citada é a seguinte:

- Considerar dois vetores `sa` e `ija`, sendo o primeiro do mesmo tipo de dado da matriz e o segundo de inteiros.
- Alocar S posições de memória para ambos.

- As primeiras N posições de `sa` armazenam, em ordem de linhas, os elementos da diagonal principal (mesmo quando forem zeros[1]).
- Cada uma das primeiras N posições de `ija` armazena o índice de `sa` que contém o primeiro elemento fora da diagonal da linha correspondente, sendo que a diferença entre os valores da posição atual e da seguinte é o número de elementos fora da diagonal que possui a linha correspondente.
- A posição 0 de `ija` é sempre igual a $N+1$. Esta posição pode ser lida para determinar N quando se tem como dados somente `sa` e `ija`.
- A posição N de `ija` é o valor da posição $N-1$ mais o número de elementos fora da diagonal presentes na última linha. Esta posição pode ser lida para determinar o número de elementos fora da diagonal

  $$n = ija[N] - (N+1)$$

  ou o tamanho dos vetores `sa` e `ija`, ou seja

  $$S = ija[N].$$

- A posição N de `sa` não é utilizada e pode ter um valor *dummy*.
- As posições de `sa` maiores do que N contêm os valores fora da diagonal ordenados por linhas e, dentro de cada linha, por colunas.
- As posições de `ija` maiores do que N contêm as colunas dos elementos correspondentes em `sa`.

Como exemplo da representação matricial compactada proposta por este algoritmo, pode-se considerar a matriz quadrada onde $N = 5$:

```
3 0 1 0 0
0 4 0 0 0
0 7 5 9 0
0 0 0 0 2
0 0 0 6 5
```

a qual, em sua forma compactada, torna-se:

k	0	1	2	3	4	5	6	7	8	9	10
ija[k]	6	7	7	9	10	11	2	1	3	4	3
sa[k]	3	4	5	0	5	x	1	7	9	2	6

[1]Esta pequena ineficiência do método é minimizada pelo fato de que, na maioria das aplicações reais, os elementos da diagonal principal são diferentes de zero.

onde `x` é o valor *dummy* citado anteriormente e que pode ser arbitrariamente escolhido.

Ainda que sendo útil ao sistematizar a passagem de uma matriz na representação tradicional para uma representação compactada, este algoritmo não aborda o principal motivo que conduz à necessidade de representações especiais para matrizes esparsas, o qual é exatamente a possibilidade de trabalhar-se diretamente com as formas compactadas.

Sendo assim, foi implementado um conjunto de funções para trabalhar diretamente com as formas compactadas de matrizes esparsas de tipo genérico. Como base foi empregada a forma de representação mostrada no algoritmo descrito acima. Deste conjunto, quatro funções fazem o trabalho mais importante, ou seja, aquele referente à manipulação das matrizes esparsas em sua forma compacta:

- Criação de uma matriz em representação compacta
- Inserção de um elemento numa matriz em representação compacta.
- Achar um elemento, dadas sua linha e coluna, numa matriz em representação compacta.
- Destruição de uma matriz em representação compacta.

A.2 Criação de uma Matriz em Representação Compacta

A primeira função simplesmente cria, dado N, uma matriz somente de zeros em representação compacta. Para isto, realiza os passos seguintes:

1. Aloca $N + 1$ posições de memória para os vetores `sa` e `ija`.
2. Preenche todo o vetor `ija` com o valor $N + 1$. Isto quer dizer que não existem elementos válidos fora da diagonal.
3. Preenche as N primeiras posições de `sa` com zeros. Isto quer dizer que todos os elementos da diagonal são zeros.
4. Insere o *dummy* valor escolhido na posição $sa[N]$.
5. Fim.

O resultado impresso da saída da função para uma matriz com $N = 5$ é o seguinte:

```
ija[0] = 6 sa[0] = 0.000000
ija[1] = 6 sa[1] = 0.000000
ija[2] = 6 sa[2] = 0.000000
ija[3] = 6 sa[3] = 0.000000
ija[4] = 6 sa[4] = 0.000000
ija[5] = 6 sa[5] = 0.000000
```

Observar que o *dummy* escolhido é também zero.

A.3 Inserção de um Elemento

A segunda função tem por objetivo a inserção de um valor v na linha i, coluna j. Para isto considera uma matriz em representação compacta criada através do algoritmo citado na seção A.2. Quatro casos surgem imediatamente como possíveis:

1. O valor a inserir é diferente de zero e a posição a ser inserida não está ocupada.
2. O valor a inserir (zero ou diferente de zero) deverá ocupar um lugar na diagonal principal.
3. O valor a inserir é diferente de zero e deverá ocupar uma posição que já é ocupada por um outro valor diferente de zero.
4. O valor a inserir é igual a zero, mas a posição a ser inserida já é ocupada por um valor diferente de zero.

Pode-se ver que, enquanto o primeiro e o último caso envolvem realocação de memória (alteração do tamanho de `sa` e de `ija`) a segunda e a terceira não necessitam de tal procedimento.

O caso padrão acontece quando se quer inserir um valor que é diferente de zero e a posição a ser inserida não está ocupada. Em outras palavras, isto significa que os vetores `sa` e `ija` devem ser incrementados em tamanho (se a posição na qual o valor será inserido não está na diagonal). O algoritmo abaixo pode ser utilizado para isto:

1. Realoca a memória ocupada por `sa` e `ija` de mais um elemento.
2. Para $(i-1) < k \leq N$ fazer $ija[k] = ija[k] + 1$
3. Para $(i-1) < k < N$ mover o conteúdo de $sa[ija[k]-1]$ e $ija[ija[k]-1]$ uma posição adiante.
4. Fazer $sa[ija[j-1]-1] = v$ e $ija[ija[j-1]-1] = i-1$.
5. Fim.

Este algoritmo será tomado como base para o outro, de caráter mais genérico, que está a seguir. Entretanto, antes de abordar esse algoritmo, deve ser comentado o caso de quando o valor a inserir está na diagonal principal. Quando isto acontece e na medida em que toda a diagonal está representada nas N primeiras posições de `sa`, basta inserir o valor (zero ou não-zero) na posição correspondente. Em outras palavras, basta fazer $sa[i-1] = v$.

Para o tratamento dos dois últimos casos citados anteriormente pode ser utilizado um algoritmo que trate conjuntamente de ambos. Este é descrito a seguir:

1. Calcular quantos elementos fora da diagonal válidos possui a linha i, ou seja:

$$n = ija[i] - ija[i-1]$$

2. Se $n = 0$ e $v \neq 0$ insere-se como mostrado no algoritmo exposto anteriormente (ou seja, é necessária a realocação de memória). Se n e v são ambos iguais a zero não é necessária nenhuma ação.

3. Se $n \neq 0$

   (a) Se $v \neq 0$ e na linha i a coluna j não é válida (quer dizer, não possui um elemento válido), inserir como mostrado no algoritmo anterior.

   (b) Senão, se $v \neq 0$ e existe uma coluna na linha i que possui um elemento válida e que é igual a j, fazer $sa[p] = v$, onde p é a posição da coluna j na linha i.

   (c) Se $v = 0$ e na linha i a coluna j não é válida (quer dizer, não possui um elemento válido), nenhuma ação é necessária.

   (d) Senão, se $v = 0$ e existe uma coluna na linha i que possui um elemento válida e que é igual a j:

      i. Para $(i-1) < k \leq N$ fazer $ija[k] = ija[k] - 1$

      ii. Para $(i-1) < k < N$ mover o conteúdo de $sa[ija[k]-1]$ e $ija[ija[k]-1]$ uma posição para trás.

      iii. Realocar memória de `ija` e de `sa` de menos um elemento.

A.4 Achar um Elemento

Para encontrar um elemento v, dadas sua linha i e sua coluna j, na representação compactada mostrada na seção A.2, seguiu-se o método abaixo:

1. Se $i = j$ (o elemento está na diagonal), fazer $v = sa[i]$.

2. Senão, para $ija[i] \leq i \leq ija[i+1]-1$, verificar se $ija[i] = j$. Se verdadeiro, $v = sa[i]$, senão, $v = 0$.

A.5 Destruição de uma Matriz

Para a destruição de uma representação matricial compactada basta liberar a memória ocupada pelos vetores `ija` e `sa`. A importância desta função reside, naturalmente, no fato de que seu emprego correto impede que, após sucessivas utilizações dos algoritmos apresentados acima, principalmente em matrizes de grande N, passe a existir um acúmulo de lixo na memória que poderia terminar por prejudicar o funcionamentos dos programas que os utilizassem.

A.6 Avaliação do Método Empregado

Para avaliar o desempenho, sob o ponto de vista de economia de memória, do algoritmo proposto, foi adotado o método descrito a seguir.

Inicialmente, para maior clareza, serão exemplificados os cálculos os quais tornam possível uma comparação entre o espaço requerido por uma matriz em sua forma ordinária e sua representação compactada. Para isto, é considerado que os elementos da matriz (logo, também aqueles armazenados em `sa`) são armazenados em precisão simples (4 *bytes*) e que o tipo de `ija` é de inteiros curtos não sinalizados (2 *bytes*). Como base para esta demonstração é tomada a matriz dada como exemplo na página 182. Nesta matriz, pode ser visto que $N = 5$, $n = 5$ e $S = 11$. Portanto, tem-se (todos os resultados estão em *bytes*):

- Tamanho da matriz original: $T_m = N \times N \times 4 = 100$.
- Tamanho de `sa`: $T_{sa} = S \times 4 = 44$.
- Tamanho de `ija`: $T_{ija} = S \times 2 = 22$.

A partir dos dados acima é possível ver que a forma compactada ocupa 66% do espaço ocupado pela matriz original. Resta agora automatizar o processo para valores diferentes de N e de n para tornar possível a observação do comportamento do algoritmo.

Uma abordagem inicial pode ser aquela de considerar vários valores de N, mas mantendo fixa entre o quadrado deste e n uma mesma proporção afim de aferir o comportamento do algoritmo. Considerando-se três proporções fixas entre n e $N \times N$, a saber, metade, um quinto e um décimo é possível uma visualização do percentual ocupado pelos vários valores de N dentro destas proporções. Considerando que $5 \leq N \leq 50$, este resultado pode ser visto na figura A.1. Pode ser visto claramente que a eficiência do algoritmo cresce com o tamanho das matrizes e também que que, para matrizes de tamanho 7 ou menores, a ocupação de metade dos elementos por valores *off-diagonal* faz com que a representação compactada seja *maior* do que a representação ordinária. Naturalmente, isto faz com que a utilização do algoritmo seja completamente desnecessária. Não obstante, no que tange às demais proporções e tamanhos das matrizes mostrados no exemplo, o emprego do algoritmo mostra-se satisfatório.

Uma visão mais completa do comportamento do algoritmo pode ser alcançada fazendo variar tanto N quanto n. Isto pode ser realizado tomando-se como exemplo matrizes que possuem $5 \leq N \leq 50$ e na qual o número de seus elementos válidos *off-diagonal* variem como $0 \leq n \leq (N \times N/2)$. É possível, desta forma, construir uma superfície que represente o comportamento do algoritmo para matrizes de diferentes tamanhos e com diferentes números de elementos válidos *off-diagonal*.

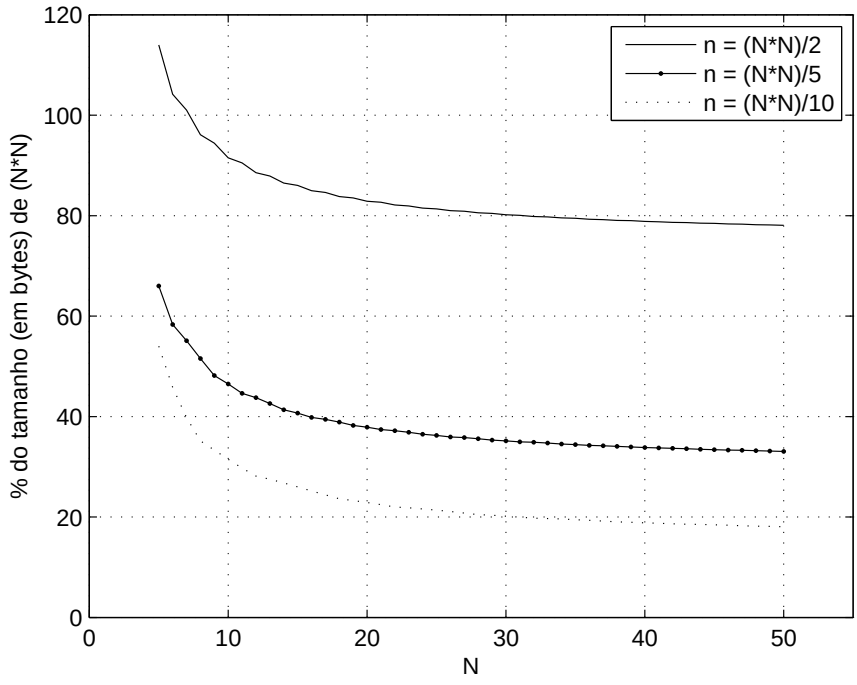

Figura A.1: Evolução do espaço ocupado empregando proporções fixas entre N^2 e n. As proporções escolhidas são $\frac{N^2}{2}$, $\frac{N^2}{5}$ e $\frac{N^2}{10}$ no intervalo $5 \leq N \leq 50$.

Tal superfície pode ser vista na figura A.2 e nela pode ser confirmado o que já havia sido constatado na figura anterior, a saber, que a eficiência do algoritmo é diretamente proporcional ao tamanho da matriz a ser representada e inversamente proporcional ao número de elementos *off-diagonal* presentes[2].

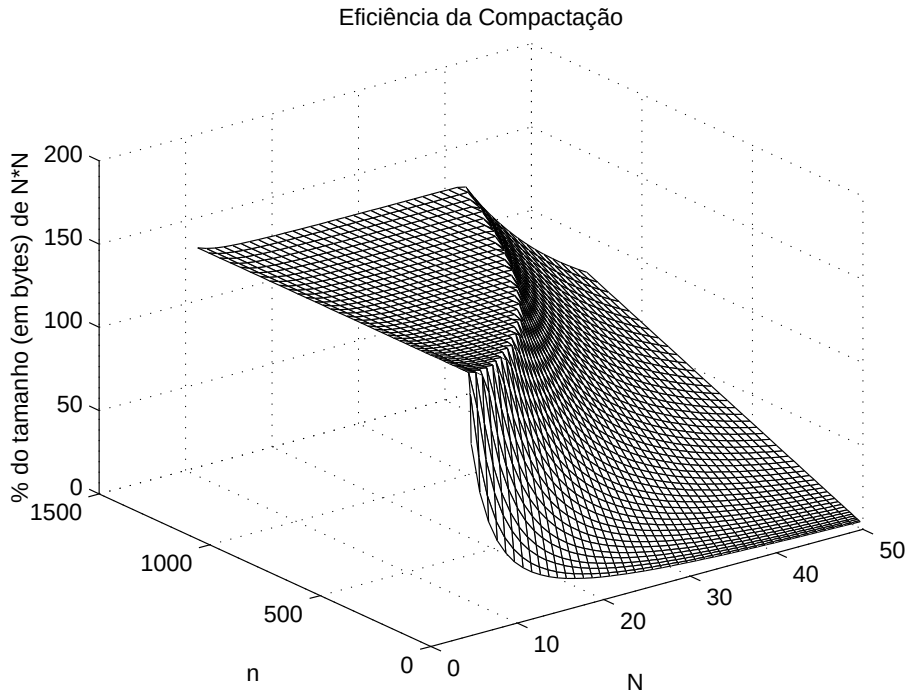

Figura A.2: Superfície representando a evolução do espaço ocupado variando-se simultaneamente $5 \leq N \leq 50$ e seus elementos válidos fora da diagonal como $0 \leq n \leq (N \times N)/2$.

O custo computacional foi avaliado para três valores de N, iguais a 200, 600 e 1000, e para um número de inserções variando entre 0 e 500. Pode ser observado na figura A.3 que, enquanto matrizes com $N \leq 200$ apresentam um comportamento estável, matrizes com $N = 600$ já apresentam um custo computacional razoável, caso se pense na operação do sistema como um todo, já que este apresenta também outros pontos de custo computacional alto, como o cálculo da Estrutura Métrica, por exemplo. Considerando $N \geq 1000$ o custo torna-se já muito algo e a solução poderá ser a utilização de uma biblioteca otimizada de funções para manipulação de matrizes esparsas.

Entretanto, é importante lembrar que, no caso deste sistema, o valor de N representa o número de eventos da superfície musical sendo analisada, o que significa uma partitura cuja parte superior é composta de mil eventos, valor bastante razoável considerando o repertório clássico e barroco da música tonal. Porém, no caso deste trabalho, o qual utiliza sempre exemplos curtos visando a facilidade na verificação dos resultados, o custo

[2]O platô superior foi mantido para facilitar a visualização dos máximos alcançados pelas matrizes onde $5 \leq N \leq 35$.

computacional dos algoritmos utilizados pode ser desprezado, o que pode ser comprovado através da figura A.3 para os casos com $N \leq 200$.

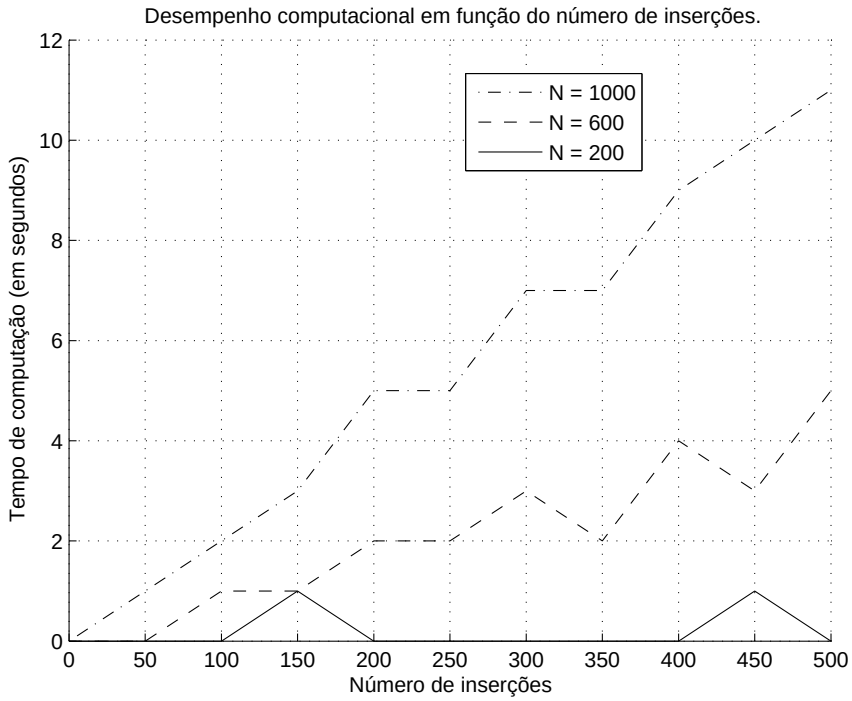

Figura A.3: Custo computacional (em segundos) em função do tamanho da matriz ($N \times N$) e do número de inserções. N assumes os valores de 200, 600 e 1000 e $0 \leq n \leq 500$, onde N é o número de linhas (colunas) da matriz n é o número de inserções.

Apêndice B

Representação em Árvores com Restrições

Um dos problemas na integração dos quatro componentes da TGMT diz respeito à representação dos resultados. Trata-se de um ponto importante porque, devido à natureza hierárquica da teoria, há uma necessidade de transferência de resultados de um componente para os outros. Assim, é notória a necessidade de uma representação dos resultados que possa ser lida por todos os componentes.

Uma proposta que viabiliza este objetivo, e que apresenta-se viável de ser empregada no primeiro componente da teoria, a Estrutura de Agrupamento, é sugerida por Curry et al. (2000) e é descrita a seguir[1].

B.1 Proposta

Curry et al. (2000) propõem a utilização de restrições (*constraints*) para a representação de uma classe específica de árvores que possuem os seguintes componentes e propriedades:

Raiz – cada árvore possui um nodo identificado como a raiz.

Ordenação – os filhos de cada nodo são diferentes e não podem ser reordenados sem mudar o que é representado pela árvore.

Profundidade Constante – todas as folhas tem a mesma distância da raiz.

Rigor – a cada profundidade, cada nodo tem, no mínimo, dois descendentes.

Na medida em que o número de folhas cresce, o número de árvores pertencentes a esta classe alcança valores muito altos, o que torna necessário o emprego de algum método para a redução destas possibilidades. No presente caso, os autores empregaram a programação

[1]Na realidade, esta abordagem somente chegou a ser implementada visando sua utilização na RPA 5 do primeiro componente da teoria, a Estrutura de Agrupamento. Mesmo assim, foi posteriormente abandonada em favor da implementação mostrada na seção 4.6.3, página 36.

lógica com restrições (*constraint logic programming*) para possibilitar a utilização prática das árvores da classe exposta.

B.2 Motivação

Em seu artigo original, o objetivo dos autores é a criação de performances musicais expressivas baseadas em dados obtidos de performances humanas reais. Estes dados seriam obtidos principalmente através da segmentação da obra musical realizada pela *Estrutura de Agrupamento*, tal como definida em Lerdahl & Jackendoff (1996). A figura B.1, baseada em Curry et al. (2000), ilustra como a classe de árvore mostrada anteriormente pode representar uma segmentação. Nela é mostrado um trecho musical, sua segmentação

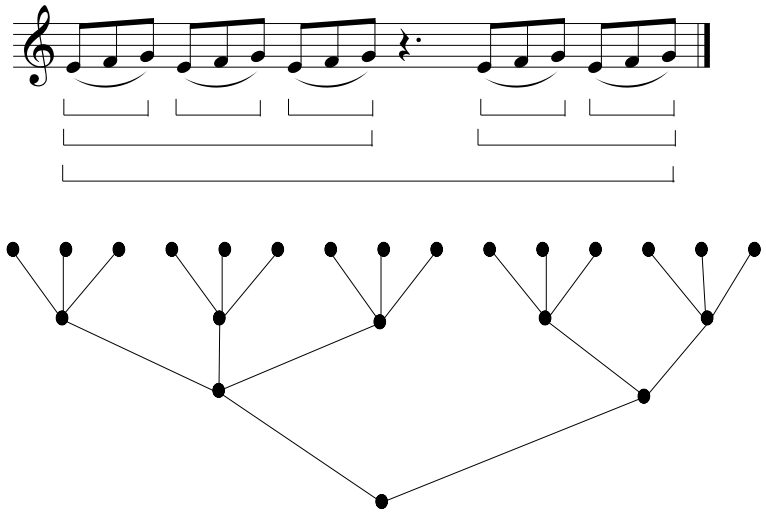

Figura B.1: Trecho musical e a representação em árvore de seu agrupamento.

em agrupamentos e a árvore representando tal segmentação (notar que a árvore está invertida para facilitar a visualização e a relação entre as duas formas de representação). É importante observar que, enquanto as folhas representam as notas do trecho musical, todos os demais nodos representam agrupamentos de notas. Uma outra maneira de observar esta questão seria considerar as folhas como grupos de apenas um elemento que, apesar de não considerados na TGMT, possibilitariam uma homogeneidade no tratamento dos nodos.

B.3 Programação Lógica com Restrições

A programação lógica com restrições sobre domínios finitos possibilita a especificação de problemas em termos de variáveis (que atuam sobre um conjunto finito de valores formado por um intervalo fechado, o *domínio* da variável) e equações que especificam relações entre estas variáveis (Hoffman & Birmingham, 2000; Curry et al., 2000; Tsang, 1993).

Como exemplo, consideremos a situação mostrada abaixo:

$$\begin{aligned} x &\in \{1 \ldots 4\} \\ y &\in \{3 \ldots 6\} \\ x + y &\geq 9 \\ x \in \{3 \ldots 4\} &\wedge y \in \{5 \ldots 6\} \end{aligned}$$

Considerando que as duas primeiras linhas representam, respectivamente, os domínios de x e y e a terceira linha representa uma restrição aos valores assumidos por estas duas variáveis, então a quarta linha representa o resultado da aplicação da restrição aos domínios das duas variáveis.

B.4 Representação

Uma característica importante da classe de árvores descrita é que ela é monotonicamente decrescente das folhas até a raiz. Isto permite que uma árvore desta classe seja representada por entrelaçamento triangular de nodos. Cada um dos nodos de uma árvore possui as variáveis abaixo relacionadas:

id.
A variável *id* é um identificador único para cada nodo e é especificada como as coordenadas (x, y) no entrelaçamento de nodos (x crescendo da direita para a esquerda e y crescendo de baixo para cima).

uplink.
A variável *uplink* estabelece uma conexão com o próximo nível acima. É especificada como um inteiro que representa a coordenada x do nodo no nível acima ao qual um nodo está conectado, ou seja, $(uplink, y + 1)$.

downlink.
A variável *downlink* representa todos os nodos no nível abaixo aos quais um nodo está conectado. Trata-se de um vetor de duas posições composto de um limite inferior (dl – *lowerbound*) e um limite superior (du – *upperbound*), sendo que todos os valores dentro do intervalo fechado entre estes limites referem-se aos nodos do nível abaixo que podem ser conectados a um nodo, ou seja, os nodos $(dl, y-1) \ldots (du, y-1)$.

B.5 Restrições

No tipo de representação utilizado existem cinco tipos de restrições, cada uma delas tratando de uma característica comportamental da árvore. São elas:

- Restrições de Nodo

- Restrições de Nível
- Restrições de Consistência
- Restrições de Largura
- Restrições de Arco

A seguir serão examinadas cada uma delas.

Restrições de Nodo

As restrições de nodo dizem respeito aos domínios das variáveis (*uplink* e *downlink*) de cada nodo. Estas podem ser sumarizadas nas equações:

$$\begin{aligned} domain(uplink) &= \{0 \ldots x\} \\ domain([du, dl]) &= \{0 \ldots n\} \\ (dl = 0) &\vee (dl \geq x) \\ du &\geq dl \\ ((dl = 0) \Leftrightarrow (du = 0)) &\wedge ((dl = 0) \Leftrightarrow (uplink = 0)) \end{aligned}$$

A primeira linha define o domínio para as variáveis *uplink* de cada nodo, enquanto que a segunda define aquele das variáveis *downlink*. Por estas equações é possível observar que, devido ao formato triangular do entrelaçamento de nodos, uma variável *uplink* é restrita a uma coordenada x igual ou menor (à esquerda) do nodo corrente, enquanto que uma variável *downlink* é restrita de zero a n nodos abaixo ou abaixo e à direita do nodo corrente. No caso das variáveis *uplink* o valor zero significa que o nodo corrente não é conectado ao nível superior, enquanto que para as variáveis *downlink* ele significa que não existe conexão com o nível inferior. As duas outras restrições representadas pela terceira e quarta linhas dizem, respectivamente, que dl deve ser igual a zero ou maior ou igual a x e que du deve ser maior ou igual a dl. A quinta e última restrição diz que, se um nodo não é utilizado em uma árvore tanto sua variável *uplink* quanto sua variável *downlink* devem ser zero.

Restrições de Nível

As restrições de nível, em número de duas, asseguram a inexistência de cruzamentos nas conexões entre dois níveis. A primeira delas diz que, para um par de nodos A e B, estando A diretamente à esquerda de B, a variável $uplink_B$ deve apontar ou para o mesmo nodo que $uplink_A$, ou para o nodo à direita dele ou, no caso de B não ser empregado na árvore, ser igual a zero. Esta restrição pode ser representada pela equação:

$$(uplink_B = uplink_A) \vee (uplink_B = uplink_A + 1) \vee (uplink_B = 0)$$

A outra restrição impede a existência de nodos não conectados no meio de nodos conectados. Diz que, se um dos *uplinks* de um determinado nível tornar-se igual a zero, então todos os *uplinks* à direita dele também se tornarão iguais a zero, ou seja:

$$(uplink_A = 0) \Rightarrow (uplink_B = 0)$$

Restrição de Consistência

A única restrição de consistência diz que, se o *uplink* de um nodo é diferente de zero, então a coordenada x deste nodo deve estar dentro do intervalo representado pela variável *downlink* do nodo do nível acima apontado pelo *uplink*. Isto pode ser representado como:

$$(x_{acima} = uplink_{este}) \Leftrightarrow ((x_{este} \geq dl_{acima}) \wedge (x_{este} \leq du_{acima}))$$

Restrição de Largura

Considerando a largura de um nível como o número de nodos neste nível que possuem uma variável *uplink* diferente de zero, a restrição agora mostrada determina que haja um decrescimento de largura da árvore à medida que se caminha das folhas até a raiz. A representação desta restrição é mostrada a seguir:

$$(largura_i > 1) \Rightarrow (largura_j < largura_i)$$

Como já foi dito antes, cada nodo deve ter no mínimo dois descendentes. Isto implica na precondição de que a largura seja maior do que um e mostra que a raiz da árvore é o nodo onde começa a ramificação e não necessariamente o ponto mais alto do entrelaçamento.

Restrição de Arco

A restrição de arco assegura que o nodo mais à direita em um nível aponte diretamente para cima ou para cima e à esquerda. Isto pode ser representado assim:

$$S = x : id(x, y) \text{ com } uplink \neq 0$$

$$uplink \leq max(S)$$

Considerando S o conjunto de todas as coordenadas x dos nodos do nível acima que possuem *uplink* diferente de zero, escolhe-se o máximo deles e faz-se o *uplink* do nodo mais à direita no nível corrente apontar para ele.

B.6 Emprego da Representação com Restrições

Para que se possa construir uma árvore pertencente à classe mostrada e que represente uma segmentação em grupos de um trecho musical é necessário que exista, para cada evento em relação aos que o circundam, um peso referente à sua possibilidade de ser uma fronteira de grupo[2]. Assim, através dos pesos relativos e de sua transferência através dos diferentes níveis, é possível obter-se uma árvore que represente fielmente a estrutura de grupos do trecho a ser analisado.

[2]No método empregado na presente tese, entretanto, os pesos são dados a fronteiras que estão *entre* dois eventos, e não aos próprios eventos.

Representação de Árvore com Restrições

Neste exemplo é realizada a segmentação de uma seqüência melódica através do emprego de uma árvore da classe descrita anteriormente. A única regra para o cálculo do peso de cada elemento é a diferença intervalar entre duas alturas. Isto corresponde a uma variação da RPA 3a, relativa à Estrutura de Agrupamento (página 19). Como valor mínimo de representação de uma fronteira considerou-se o valor 3, correspondendo à $3^{\underline{a}}$ menor. A entrada para o sistema, portanto, não é o conjunto de alturas, mas sim as diferenças absolutas entre aquelas que são contíguas. Duas restrições ainda se fazem necessárias para este caso.

A primeira delas diz respeito ao peso da primeira altura, a qual tem que ser colocada como zero para evitar possíveis agrupamentos de apenas um elemento. Em seguida, o peso da última altura deve ser ajustado para um valor fictício e maior do que qualquer valor que possa ser encontrado no vetor de entrada. Isto assegura que a última altura sempre será tratada como uma fronteira de grupo. A escolha de um valor impossível de ser encontrado segundo a regra imposta no cálculo dos pesos recaiu sobre o valor 64.

Na figura B.2 pode ser vista a seqüência de alturas, seus valores numéricos no protocolo MIDI[3], seus pesos relativos em valores absolutos e seu agrupamento em quatro diferentes níveis. Observar, consoante com o que foi dito antes, que o primeiro e o último elementos tem seus pesos tratados de forma especial. Assim, o peso da primeira altura é zero e o da última um valor muito grande (a macro `HUGE` vale 64, valor impossível de ser encontrado em outra posição do vetor de entrada).

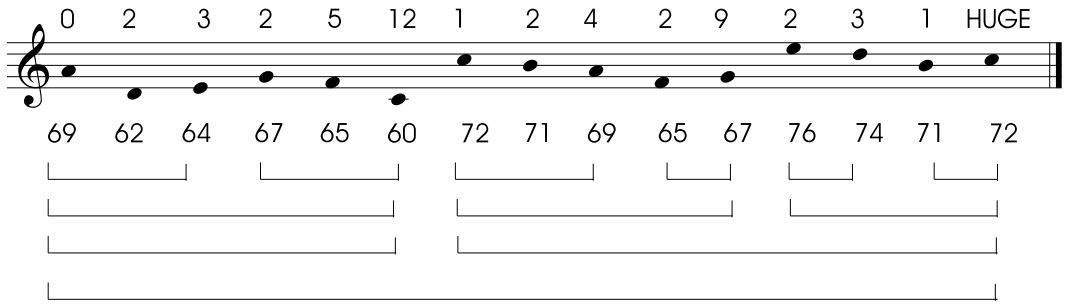

Figura B.2: Exemplo de agrupamento.

Abaixo pode ser vista a matriz resultante do processamento da seqüência de alturas mostrada antes.

A partir de sua penúltima linha, em ordem decrescente, pode-se ver os diferentes níveis do entrelaçado correspondentes aos diferentes níveis de agrupamento. Os elementos com valores iguais a `-1` são nodos não utilizados na árvore.

[3]No protocolo MIDI o valor 60 corresponde ao dó central do piano e incremento ou decremento de um corresponde, respectivamente, a uma elevação ou abaixamento de um semitom.

```
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
64 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
12 64 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
12  9 64 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 3 12  4  9  3 64 -1 -1 -1 -1 -1 -1 -1 -1 -1
 0  2  3  2  5 12  1  2  4  2  9  2  3  1 64
```

Antes de prosseguir, duas observações fazem-se necessárias.

Em primeiro lugar, é importante lembrar que todos os níveis de agrupamento iniciam-se pela primeira altura do trecho. Desta forma, seu peso relativo não precisa estar representado na matriz. Assim, no nível 4 o primeiro grupo compreende desde a primeira altura (peso zero) até a altura com peso 3; no nível 3, assim como no nível 2, o primeiro grupo compreende desde a primeira altura até a altura com peso 12; finalmente, no nível 1 existe um único grupo com todas as alturas.

O outro ponto importante a notar é que o processo utilizado conduz a uma matriz altamente esparsa. Sendo assim, um tratamento específico se faz necessário na medida em que o número de elementos (nodo) pode facilmente chegar a centenas. Na medida em que o emprego de métodos para o tratamento de matrizes esparsas se faz necessário, aqueles descritos no apêndice A, página 181 podem ser utilizados com facilidade.

Abaixo está mostrada uma outra forma de saída do programa, na qual estão explicitados os níveis:

```
NÍVEL: 1
Inicio: 0 Fim: 1
Father 0 Node Value 2:
0 1
NÍVEL: 2
Inicio: 0 Fim: 0
Father 0 Node Value 1:
0
Inicio: 1 Fim: 2
Father 1 Node Value 3:
1 2
NÍVEL: 3
Inicio: 0 Fim: 1
```

```
Father 0 Node Value 2:
0 1
Inicio: 2 Fim: 3
Father 1 Node Value 4:
2 3
Inicio: 4 Fim: 5
Father 2 Node Value 6:
4 5
NÍVEL: 4
Inicio: 0 Fim: 2
Father 0 Node Value 3:
0 1 2
Inicio: 3 Fim: 5
Father 1 Node Value 6:
3 4 5
Inicio: 6 Fim: 8
Father 2 Node Value 9:
6 7 8
Inicio: 9 Fim: 10
Father 3 Node Value 11:
9 10
Inicio: 11 Fim: 12
Father 4 Node Value 13:
11 12
Inicio: 13 Fim: 14
Father 5 Node Value 15:
13 14
```

Apêndice C

Resultados sob Forma Numérica

Neste apêndice são mostrados os formatos dos arquivos gerados pelo sistema. Eles são de dois tipos:

- Arquivos temporários que permitem a comunicação interna entre os quatro componentes; e
- Arquivos com os resultados das análises realizadas por cada componente.

A coordenação e atuação dos dois grupos de arquivos pode ser vista no capítulo 3, figura 3.1. Nela podem ser vistos as saídas, os direcionamentos e utilização dos arquivos por cada um dos quatro componentes. Nesta figura é ainda mostrado o arquivo `func.dat`, o qual é gerado previamente (ver apêndice E) e que somente faz parte da figura devido à sua alta relevância na processamento da Redução Prolongacional.

C.1 Arquivos de Comunicação entre Componentes

Os arquivos de comunicação entre os componentes são arquivos temporários gerados para cada análise e apagados ao início da seguinte. De modo diverso aos arquivos com os resultados de cada componente, os quais possuem o mesmo nome do trecho sendo analisado sendo variada somente sua extensão, os arquivos temporários de comunicação possuem nomes fixos indicando sua função dentro do conjunto. Além disto, os arquivos temporários são arquivos ASCII, o que facilita a manutenção e depuração dos programas. A seguir estão os arquivos temporários e seus formatos.

C.1.1 GROUPING.TMP

Este arquivo carrega os resultados temporários encontrados no componente *Estrutura de Agrupamento* e os disponibiliza para os demais componentes. Seu formato pode ser descrito como sendo:

```
NL
Li
LiNG
LiGj.begin LiGj.end
```

onde NL é o número de níveis da estrutura de agrupamento, Li é o i-ésimo nível, $LiNG$ é o número de grupos do i-ésimo nível, $LiGj.begin$ é o início do j-ésimo grupo do i-ésimo nível e $LiGj.end$ é o final do `j`-ésimo grupo do `i`-ésimo nível, sendo que $0 \leq i < NL$ e $0 \leq j < LiNG$ Tanto o início quanto o final de cada grupo, em cada nível, são dados em segundos.

C.1.2 MSPOP.TMP

O conteúdo deste arquivo, gerado pelo componente *Estrutura Métrica*, é uma população de estruturas métricas, todas igualmente válidas de serem aplicadas no trecho sendo analisado. Trata-se do front de Pareto encontrado através da minimização multi-objetivo do erro de cada regra do componente. Este arquivo temporário deve ser lido pelo componente *Redução Temporal*, o qual, então, seleciona uma estrutura que mais se adapte à sua própria redução interna de conflitos entre suas regras. O formato do arquivo é como segue:

```
Pop
PhiNL
Phin Ip
Phi_n S
```

onde Pop é o número de fenótipos (indivíduos da população), $PhiNL$ é o número de níveis do fenótipo i, $Phin$ é o n-ésimo nível do fenótipo i e Ip e S sua posição inicial (*offset* em relação ao nível anterior) e seu espaçamento (regular), respectivamente, sendo que $0 \leq i < Pop$ e $0 \leq n < PhiNL$.

C.1.3 TIMESPAN.TMP

O objetivo deste arquivo é levar à Redução Prolongacional os eventos-cabeça encontrados pela Redução Temporal através da interação entre a Estrutura de Agrupamento e a Estrutura Métrica. Como cada grupo de cada nível da Estrutura de Agrupamento original possui um evento-cabeça, isto deve ser indicado no arquivo, de modo que a Redução Prolongacional possa calcular adequadamente as regiões prolongacionais. O formato do arquivo é o seguinte:

```
NL
Li
LiNG
LiGjH
```

onde NL é o número de níveis da estrutura de agrupamento, Li é o i-ésimo nível, $LiNG$ é o número de grupos do i-ésimo nível, $LiGjH$ é o evento-cabeça do j-ésimo grupo no i-ésimo nível, sendo que $0 \leq i < NL$ e $0 \leq j < LiNG$. Notar que o evento-cabeça é arquivado com todos os dados pertinentes à caracterização do evento, tal como definido pela estrutura `CELULA` no capítulo 3, a saber, seu canal MIDI, sua altura, sua intensidade, seu ataque relativo, seu ataque absoluto e sua duração.

C.1.4 PREPROC.TMP

Este arquivo, assim como o da próxima sub-seção, é gerado pelo componente *Redução Prolongacional* e dá as coordenadas, momentos inicial e final, dos acordes encontrados ao longo do trecho sendo analisado. O conteúdo deste arquivo é o resultado do pré-processamento realizado visando à análise harmônica do trecho. Sendo assim, este arquivo somente é gerado para trechos polifônicos. Seu formato é:

```
Nc
Aci Dci
Cipj
```

onde Nc é o número de acordes, Aci e Dci são, respectivamente, o ataque e a duração do i-ésimo acorde e $Cipj$ é o conjunto de alturas pj do i-ésimo acorde (todas as alturas do acorde possuindo o mesmo ataque e a mesma duração, ou seja, uma homofonia), sendo que $0 \leq i < Nc$ e $0 \leq j < 4$.

C.1.5 CADENCES.TMP

Este arquivo, assim como o da sub-seção anterior, é gerado pelo componente *Redução Prolongacional* e dá as coordenadas, momentos inicial e final, das cadências encontradas ao longo do trecho sendo analisado. Para isto, considera uma cadência como sendo constituída dos elementos *penult* e *final*, cada um deles tendo um instante de ataque e uma duração. Assim como no caso anterior, somente é calculado para analises de trechos polifônicos. Seu formato é o que segue abaixo:

```
N
Api Dpi Afi Dfi
```

onde N é o número de cadências e Api, Dpi, Afi e Dfi são, respectivamente, o ataque do *penult*, a duração do *penult*, o ataque do *final* e a duração do *final*, todos da i-ésima cadência, sendo que $0 \leq i < N$.

C.1.6 TONSEQ.TMP

Este arquivo, como nos casos anteriores, é gerado no componente *Redução Prolongacional* e somente é calculado para análises polifônicas. Seu conteúdo é o mapeamento de cada um dos acordes encontrados no pré-processamento com a tonalidade na qual está inserido. Seu formato é:

```
Nc
Aci Tci
```

onde Nc é o número de acordes (resultantes do pré-processamento), Aci e Tci são, respectivamente, o ataque e a tonalidade do i-ésimo acorde, sendo que $0 \leq i < Nc$.

C.2 Arquivos com Resultados

Como resultado de seu processamento, cada programa correspondente a um componente da TGMT gera automaticamente um arquivo no formato `nome.ext`, onde `nome` é o mesmo nome do arquivo MIDI que possui a amostra sendo analisada e `ext` é uma das extensões abaixo:

1. `gs` ← Estrutura de Agrupamento: saída para usuário.
2. `gly` ← Estrutura de Agrupamento: saída LilyPond (ver adiante).
3. `ms` ← Estrutura Métrica.
4. `tsr` ← Redução Temporal.
5. `har` ← Análise Harmônica.
6. `prl` ← Redução Prolongacional.

Estes arquivos, na medida em que se destinam a ser lidos pelo usuário do sistema, possuem um formato descritivo, no qual os dados possam ser facilmente identificados e a análise realizada possa ser, sem dificuldade, transcrita para sua forma usual em grafia musical tradicional.

Visando um melhor entendimento dos formatos destes arquivos e levando em conta que eles possuem muitas informações verbais, ao invés de serem descritos seus respectivos formatos, são dados exemplos de arquivos reais gerados pelo sistema.

C.2.1 Estrutura de Agrupamento

Na Estrutura de Agrupamento existem dois formatos de arquivos de saída. O primeiro, com extensão `gs`, é um arquivo texto que mostra os componentes (alturas da parte superior) de cada agrupamento, em cada nível estrutural.

Um exemplo deste tipo de saída, tal como gerado pelo sistema pode ser (as alturas seguem a numeração MIDI)[1]:

[1]Para este e todos os exemplos seguintes é utilizada a primeira parte do primeiro tema do quarto movimento da Sinfonia nº 40 de W.A. Mozart.

```
Número de níveis = 4

Nível 0:
Grupo #0:
Elemento: 62
Elemento: 67

Grupo #1:
Elemento: 70
Elemento: 74
Elemento: 79
Elemento: 82
Elemento: 81

Grupo #2:
Elemento: 73
Elemento: 76
Elemento: 74
Elemento: 78
Elemento: 76
Elemento: 73

Grupo #3:
Elemento: 74
Elemento: 78
Elemento: 76
Elemento: 73
Elemento: 74

Nível 1:
Grupo #0:
Elemento: 62
Elemento: 67

Grupo #1:
Elemento: 70
Elemento: 74
Elemento: 79
Elemento: 82
Elemento: 81

Grupo #2:
```

```
Elemento: 73
Elemento: 76
Elemento: 74
Elemento: 78
Elemento: 76
Elemento: 73
Elemento: 74
Elemento: 78
Elemento: 76
Elemento: 73
Elemento: 74

Nível 2:
Grupo #0:
Elemento: 62
Elemento: 67
Elemento: 70
Elemento: 74
Elemento: 79
Elemento: 82
Elemento: 81

Grupo #1:
Elemento: 73
Elemento: 76
Elemento: 74
Elemento: 78
Elemento: 76
Elemento: 73
Elemento: 74
Elemento: 78
Elemento: 76
Elemento: 73
Elemento: 74

Nível 3:
Grupo #0:
Elemento: 62
Elemento: 67
Elemento: 70
Elemento: 74
Elemento: 79
Elemento: 82
```

```
Elemento: 81
Elemento: 73
Elemento: 76
Elemento: 74
Elemento: 78
Elemento: 76
Elemento: 73
Elemento: 74
Elemento: 78
Elemento: 76
Elemento: 73
Elemento: 74
```

Notar que os parâmetros indicados aqui como "elementos" são as alturas da parte superior (ou da única parte, caso seja um trecho monofônico) do trecho sendo analisado. Também estão claramente indicados os diferentes níveis (quatro neste caso) e o número de agrupamentos em cada nível.

O segundo formato é para ser empregado por um programa (o qual foi implementado separadamente) que faz a *interface* com o programa de editoração musical *LilyPond*. Na medida em que este arquivo de saída deverá ser lido por outro programa, os dados nele contidos possuem natureza puramente numérica.

Um exemplo de um arquivo, gerado com a mesma análise que forneceu o exemplo anterior, é o seguinte:

```
4
4
  0  1
  2  6
  7 12
 13 17
3
  0  1
  2  6
  7 17
2
  0  6
  7 17
1
  0 17
```

Este exemplo é lido como uma estrutura de agrupamento de quatro níveis, sendo que o primeiro nível tem quatro grupos, sendo o par de inteiros que se seguem os índices do primeiro e do último elemento de cada grupo. Para os demais níveis, lê-se da mesma forma.

C.2.2 Estrutura Métrica

O arquivo com o resultado encontrado para uma estrutura métrica é composto dos dados necessários para a construção da grade métrica. Este arquivo carrega uma codificação de uma estrutura métrica que contém o número de níveis seguido das posições iniciais e dos espaçamentos de cada nível. Estes dados são os necessários para a construção de uma grade métrica. Exemplo de um arquivo de saída com uma estrutura métrica:

```
Número de níveis: 5
Nível: 0 Posição inicial: 0 Espaçamento: 1
Nível: 1 Posição inicial: 2 Espaçamento: 2
Nível: 2 Posição inicial: 2 Espaçamento: 2
Nível: 3 Posição inicial: 3 Espaçamento: 3
Nível: 4 Posição inicial: 2 Espaçamento: 3
```

C.2.3 Redução Temporal

Os resultados finais da Redução Temporal são, não apenas os eventos-cabeça, mas também a árvore resultante da análise. Sendo assim, o arquivo de saída da Redução Temporal deve fornecer não somente os eventos-cabeça, mas também a dependência entre cada componente, de cada grupo em cada nível. Desta forma, foi pensado um formato no qual são mostrados, para cada grupo, o evento-cabeça e a dependência entre cada elemento de um grupo e os demais. A seguir está um exemplo completo de uma saída do componente Redução Temporal:

```
Nível 0 Grupo 0:
62[0] -> 67[1]
67[1] -> head

Nível 0 Grupo 1:
70[0] -> 74[1]
74[1] -> 79[2]
79[2] -> 82[3]
82[3] -> head
81[4] -> 82[3]

Nível 0 Grupo 2:
73[0] -> 76[1]
76[1] -> head
74[2] -> 76[1]

Nível 0 Grupo 3:
78[0] -> head
76[1] -> 78[0]
```

```
73[2] -> 76[1]
74[3] -> 73[2]

Nível 0 Grupo 4:
78[0] -> head
76[1] -> 78[0]
73[2] -> 76[1]
74[3] -> 73[2]

Nível 1 Grupo 0:
67[0] -> head

Nível 1 Grupo 1:
82[0] -> head

Nível 1 Grupo 2:
76[0] -> 78[1]
78[1] -> head
78[2] -> 78[1]

Nível 2 Grupo 0:
67[0] -> 82[1]
82[1] -> head

Nível 2 Grupo 1:
78[0] -> head

Nível 3 Grupo 0:
82[0] -> head
78[1] -> 82[0]
```

C.2.4 Redução Prolongacional

Árvore Prolongacional

Os resultados finais da Redução Prolongacional são semelhantes àqueles da Redução Temporal, constituíndo-se de uma lista mostrando a dependência (prolongação) de cada evento da superfície em relação a um outro, gerando, assim, a árvore prolongacional. Abaixo é mostrado um exemplo completo de uma saída do componente Redução Prolongacional:

```
62[0] -> head
67[1] -> 70[2]
70[2] -> 74[3]
74[3] -> 73[7]
```

```
79[4] -> 74[3]
82[5] -> 81[6]
81[6] -> 79[4]
73[7] -> 62[0]
76[8] -> 74[17]
74[9] -> 73[16]
78[10] -> 76[15]
76[11] -> 78[14]
73[12] -> 74[13]
74[13] -> 76[11]
78[14] -> 78[10]
76[15] -> 74[9]
73[16] -> 76[8]
74[17] -> 73[7]
```

Análise Harmônica

A análise harmônica realizada na Redução Prolongacional é construída a partir do pré-processamento da obra sendo analisada. Consta de nove parâmetros de análise:

1. `Acorde` ← Alturas constituintes do acorde.
2. `Ataque` ← Instante de ataque (em segundos).
3. `Duração` ← Duração (em segundos).
4. `Tonalidade e modo` ← Tonalidade e modo nos quais o acorde está inserido.
5. `Tipo` ← Tipo do acorde (classe).
6. `Função` ← Função do acorde dentro do contexto tonal no qual está inserido.
7. `Cadencial` ← Diz se é cadencial ou não (valor booleano).
8. `Fundamental` ← Altura fundamental do acorde.
9. `Estado` ← Estado do acorde.

A análise harmônica dada aqui como exemplo corresponde ao pré-processamento do trecho apresentado na figura 5.16, página 93.

```
Acorde 0: [ 51 55 58 63 ]
Ataque: 3.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (0): maior
Função[0]: I
```

```
Cadencial: Não
Fundamental: Mib
Estado: Est. Fundamental

Acorde 1: [ 55 58 63 70 ]
Ataque: 4.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (0): maior
Função[0]: I
Cadencial: Não
Fundamental: Mib
Estado: Primeira Inv.

Acorde 2: [ 51 63 67 70 ]
Ataque: 5.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (0): maior
Função[0]: I
Cadencial: Não
Fundamental: Mib
Estado: Est. Fundamental

Acorde 3: [ 56 63 68 72 ]
Ataque: 6.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (0): maior
Função[11]: IV
Cadencial: Não
Fundamental: Lab
Estado: Est. Fundamental

Acorde 4: [ 47 62 67 74 ]
Ataque: 7.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (0): maior
Função[10]: V_VI
Cadencial: Não
Fundamental: Sol
Estado: Primeira Inv.
```

```
Acorde 5: [ 48 60 67 75 ]
Ataque: 8.000
Duração: 3.000
Tonalidade e modo: Mib maior
Tipo (1): menor
Função[14]: VI
Cadencial: Não
Fundamental: Do
Estado: Est. Fundamental

Acorde 6: [ 60 63 72 79 ]
Ataque: 11.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (1): menor
Função[14]: VI
Cadencial: Não
Fundamental: Do
Estado: Est. Fundamental

Acorde 7: [ 56 60 72 77 ]
Ataque: 12.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (1): menor
Função[5]: II
Cadencial: Não
Fundamental: Fa
Estado: Primeira Inv.

Acorde 8: [ 53 60 68 75 ]
Ataque: 13.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (6): menor_7m
Função[5]: II
Cadencial: Não
Fundamental: Fa
Estado: Est. Fundamental

Acorde 9: [ 58 58 67 75 ]
Ataque: 14.000
```

```
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (0): maior
Função[0]: I
Cadencial: Não
Fundamental: Mib
Estado: Segunda Inv.

Acorde 10: [ 46 58 65 74 ]
Ataque: 15.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (0): maior
Função[13]: V
Cadencial: Não
Fundamental: Sib
Estado: Est. Fundamental

Acorde 11: [ 51 58 67 75 ]
Ataque: 16.000
Duração: 4.000
Tonalidade e modo: Mib maior
Tipo (0): maior
Função[0]: I
Cadencial: Sim
Fundamental: Mib
Estado: Est. Fundamental

Acorde 12: [ 63 63 70 79 ]
Ataque: 20.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (0): maior
Função[0]: I
Cadencial: Não
Fundamental: Mib
Estado: Est. Fundamental

Acorde 13: [ 60 63 72 79 ]
Ataque: 21.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (1): menor
```

```
Função[14]: VI
Cadencial: Não
Fundamental: Do
Estado: Est. Fundamental

Acorde 14: [ 57 65 72 77 ]
Ataque: 22.000
Duração: 1.000
Tonalidade e modo: Sib maior
Tipo (0): maior
Função[13]: V
Cadencial: Não
Fundamental: Fa
Estado: Primeira Inv.

Acorde 15: [ 53 65 69 75 ]
Ataque: 23.000
Duração: 1.000
Tonalidade e modo: Sib maior
Tipo (5): maior_7m
Função[13]: V
Cadencial: Não
Fundamental: Fa
Estado: Est. Fundamental

Acorde 16: [ 58 65 70 74 ]
Ataque: 24.000
Duração: 2.000
Tonalidade e modo: Sib maior
Tipo (0): maior
Função[0]: I
Cadencial: Sim
Fundamental: Sib
Estado: Est. Fundamental

Acorde 17: [ 46 62 65 70 ]
Ataque: 26.000
Duração: 1.000
Tonalidade e modo: Sib maior
Tipo (0): maior
Função[0]: I
Cadencial: Não
Fundamental: Sib
```

```
Estado: Est. Fundamental

Acorde 18: [ 56 65 70 74 ]
Ataque: 27.000
Duração: 1.000
Tonalidade e modo: Sib maior
Tipo (5): maior_7m
Função[3]: V_IV
Cadencial: Não
Fundamental: Sib
Estado: Terceira Inv.

Acorde 19: [ 55 63 70 75 ]
Ataque: 28.000
Duração: 1.000
Tonalidade e modo: Sib maior
Tipo (0): maior
Função[11]: IV
Cadencial: Não
Fundamental: Mib
Estado: Primeira Inv.

Acorde 20: [ 53 65 70 74 ]
Ataque: 29.000
Duração: 1.000
Tonalidade e modo: Sib maior
Tipo (0): maior
Função[0]: I
Cadencial: Não
Fundamental: Sib
Estado: Segunda Inv.

Acorde 21: [ 51 67 70 72 ]
Ataque: 30.000
Duração: 1.000
Tonalidade e modo: Sib maior
Tipo (6): menor_7m
Função[5]: II
Cadencial: Não
Fundamental: Do
Estado: Primeira Inv.

Acorde 22: [ 53 65 69 72 ]
```

```
Ataque: 31.000
Duração: 1.000
Tonalidade e modo: Sib maior
Tipo (0): maior
Função[13]: V
Cadencial: Não
Fundamental: Fa
Estado: Est. Fundamental

Acorde 23: [ 46 62 65 70 ]
Ataque: 32.000
Duração: 3.000
Tonalidade e modo: Sib maior
Tipo (0): maior
Função[0]: I
Cadencial: Sim
Fundamental: Sib
Estado: Est. Fundamental

Acorde 24: [ 50 58 65 70 ]
Ataque: 35.000
Duração: 1.000
Tonalidade e modo: Sib maior
Tipo (0): maior
Função[0]: I
Cadencial: Não
Fundamental: Sib
Estado: Primeira Inv.

Acorde 25: [ 51 58 63 67 ]
Ataque: 36.000
Duração: 1.000
Tonalidade e modo: Sib maior
Tipo (0): maior
Função[11]: IV
Cadencial: Não
Fundamental: Mib
Estado: Est. Fundamental

Acorde 26: [ 49 63 67 70 ]
Ataque: 37.000
Duração: 1.000
Tonalidade e modo: Mib maior
```

```
Tipo (5): maior_7m
Função[3]: V_IV
Cadencial: Não
Fundamental: Mib
Estado: Terceira Inv.

Acorde 27: [ 48 63 68 72 ]
Ataque: 38.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (0): maior
Função[11]: IV
Cadencial: Não
Fundamental: Lab
Estado: Primeira Inv.

Acorde 28: [ 52 61 67 70 ]
Ataque: 39.000
Duração: 2.000
Tonalidade e modo: Mib maior
Tipo (10): diminute_7
Função[15]: V_II
Cadencial: Não
Fundamental: Ref
Estado: Primeira Inv.

Acorde 29: [ 53 60 65 68 ]
Ataque: 41.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (1): menor
Função[5]: II
Cadencial: Não
Fundamental: Fa
Estado: Est. Fundamental

Acorde 30: [ 48 60 63 67 ]
Ataque: 42.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (1): menor
Função[14]: VI
Cadencial: Não
```

```
Fundamental: Do
Estado: Est. Fundamental

Acorde 31: [ 49 55 63 70 ]
Ataque: 43.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (5): maior_7m
Função[3]: V_IV
Cadencial: Não
Fundamental: Mib
Estado: Terceira Inv.

Acorde 32: [ 48 56 63 68 ]
Ataque: 44.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (0): maior
Função[11]: IV
Cadencial: Não
Fundamental: Lab
Estado: Primeira Inv.

Acorde 33: [ 46 58 63 67 ]
Ataque: 45.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (0): maior
Função[0]: I
Cadencial: Não
Fundamental: Mib
Estado: Segunda Inv.

Acorde 34: [ 45 60 63 65 ]
Ataque: 46.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (5): maior_7m
Função[6]: V_V
Cadencial: Não
Fundamental: Fa
Estado: Primeira Inv.
```

```
Acorde 35: [ 46 58 62 65 ]
Ataque: 47.000
Duração: 1.000
Tonalidade e modo: Mib maior
Tipo (0): maior
Função[13]: V
Cadencial: Não
Fundamental: Sib
Estado: Est. Fundamental

Acorde 36: [ 51 55 58 63 ]
Ataque: 48.000
Duração: 3.000
Tonalidade e modo: Mib maior
Tipo (0): maior
Função[0]: I
Cadencial: Sim
Fundamental: Mib
Estado: Est. Fundamental
```

Apêndice D

Um Algoritmo Dinâmico para Identificação Harmônica

D.1 Introdução

A identificação da tonalidade de um trecho melódico, longe de ser um problema trivial, apresenta dificuldades que demandam diferentes métodos para serem resolvidas. O problema se agrava ainda mais quando no trecho a ser analisado ocorrem modulações, ou seja, acontecem manifestações, temporárias ou não, de outras tonalidades no decorrer do mesmo. Não obstante estas dificuldades, trata-se de um problema que necessita ser atacado por ser a base tanto de aplicações teóricas, como aquela de possibilitar uma análise completa de um trecho puramente melódico, quanto de aplicações práticas, como o desenvolvimento de um harmonizador automático de melodias.

Uma abordagem visando a identificação da tonalidade de um trecho melódico escrito dentro do idioma tonal clássico é o algoritmo proposto por Krumhansl & Schmuckler e apresentado em Krumhansl (1990). Basicamente, este funciona fazendo a correlação entre dados previamente obtidos através de experimentos com ouvintes e os dados decorrentes da análise quantitativa, de natureza duracional, dos componentes do trecho melódico a ser estudado.

Entretanto, um problema encontrado na abordagem de Krumhansl é o caráter estático do algoritmo, ou seja, sua incapacidade de detectar modulações ao longo do trecho melódico. Trata-se de uma grave limitação na medida em que somente amostras muito simplificadas ou curtas podem ser corretamente analisadas desta forma. Em outras palavras, um trecho melódico não pode, deste modo, ser segmentado em função das regiões harmônicas pelas quais passa.

Uma melhoria no desempenho do algoritmo pode ser alcançada através dos valores propostos por Temperley (2001) e Temperley (1999). Trata-se de uma modificação dos valores originais de Krumhansl visando uma melhor abordagem dos contextos tonais em jogo. Esta contribuição, entretanto, não resolve o problema de identificação de modulações.

A solução apresentada neste trabalho emprega uma técnica baseada em janelas móveis,

muito utilizada em processamento de sinais, com o objetivo de analisar progressivamente cada porção do trecho principal, obtendo, assim, uma representação da sucessão de tonalidades. Para uma melhor resposta, foi implementada uma sobreposição entre as janelas, afim de evidenciar possíveis trechos comuns entre regiões assim como ambigüidades entre as tonalidades. Ainda com o objetivo de melhorar o desempenho, foram implementadas restrições que atuam sobre os resultados obtidos na saída do algoritmo. A primeira delas diz respeito a considerar somente as tonalidades vizinhas da tonalidade principal, o que impede possíveis ambigüidades, oriundas principalmente do tamanho da janela e do percentual de sobreposição, sejam consideradas como saídas válidas. A outra restrição é considerar como saída válida não somente o coeficiente de correlação mais alto encontrado, mas sim todos os valores que estiverem acima de um determinado limiar. Desta forma, não é obtida na saída uma única tonalidade, mas um conjunto de possibilidades.

D.2 O Algoritmo de Krumhansl & Schmuckler

Em seus trabalhos acerca da percepção e cognição de estruturas tonais, Krumhansl (1990) relata experimentos na área que foram realizados por ela isoladamente e também com alguns outros colaboradores. Tais experimentos abordam questões tais como relações entre tons simples e contextos tonais (Krumhansl & Shepard, 1979; Krumhansl & Kessler, 1982), a distância percebida entre tonalidades (Krumhansl & Kessler, 1982), correlação entre hierarquias tonais e consonância, correlação entre hierarquias tonais e distribuição estatística de tons e relações percebidas entre tons, relações percebidas entre acordes e tonalidades (Krumhansl et al., 1982) e relações percebidas de acordes dentro de tonalidades (Krumhansl, 1990).

Um dos coroamentos do conjunto destes trabalhos é o algoritmo de detecção de tonalidade (*key-finding algorithm*). Este algoritmo divide-se em duas partes principais:

1. Inicialmente, para cada classe de altura presente no trecho a ser analisado soma-se o total de seus valores de duração, definindo-se duração como a dimensão temporal de cada classe de altura;

2. Em seguida, calcula-se a correlação entre o vetor encontrado e aqueles correspondentes aos perfis de cada uma das 12 tonalidades maiores e 12 menores. Onde for obtida a maior correlação, esta corresponde à tonalidade percebida.

Os perfis citados na segunda parte do algoritmo foram obtidos através do experimento realizado por Krumhansl & Kessler (1982) acerca das relações entre tons simples e contextos tonais. Os valores obtidos para as tonalidades maiores e menores estão presentes na tabela D.1 e sua representação gráfica está presente na figura D.1. Notar que cada linha corresponde a uma classe de altura e seus correspondentes valores para as tonalidades em questão. Os valores mostrados aqui correspondem às tonalidades de Dó Maior e dó menor (duas últimas colunas).

Tabela D.1: Matriz com os perfis para as tonalidades de Dó maior e dó menor encontrados no experimento de Krumhansl & Kessler.

Classes de Alturas	Maior	Menor
0	6,35	6,33
1	2,23	2,68
2	3,48	3,52
3	2,33	5,38
4	4,38	2,60
5	4,09	3,53
6	2,52	2,54
7	5,19	4,75
8	2,39	3,98
9	3,66	2,69
10	2,29	3,34
11	2,88	3,17

Para ajustar-se estes valores a cada uma das tonalidades maiores e menores, e considerando a primeira coluna como um estrutura circular, basta realizar a rotação, no sentido ascendente, da primeira coluna da matriz pelo número correspondente de passos. Por exemplo, se quisermos saber quais são os valores correspondentes à Sol Maior, é suficiente fazer a rotação no sentido ascendente em 7 posições da primeira coluna da matriz. Isto pode ser visto na tabela D.2.

Como exemplo da operação do algoritmo pode ser visto na figura D.2 um trecho melódico analisado com o mesmo. Nela é possível observar-se que o resultado obtido corresponde à tonalidade correta do trecho. Como entrada para o algoritmo é dada uma matriz cuja primeira linha são as classes de altura presentes no trecho e a segunda a duração correspondente a cada uma delas. Assim, tem-se:

$$\mathbf{I} = \begin{bmatrix} 2 & 7 & 7 & 9 & 11 & 7 & 11 & 9 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Na medida em que todas as notas tem o mesmo valor de duração optou-se por representá-la pelo valor unitário.

Ao executar o primeiro passo do algoritmo, chega-se ao vetor abaixo:

$$\mathbf{D} = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 3\ 0\ 2\ 0\ 2]^T$$

Nele pode-se observar as somas das durações de cada uma das classes dee altura presentes na amostra.

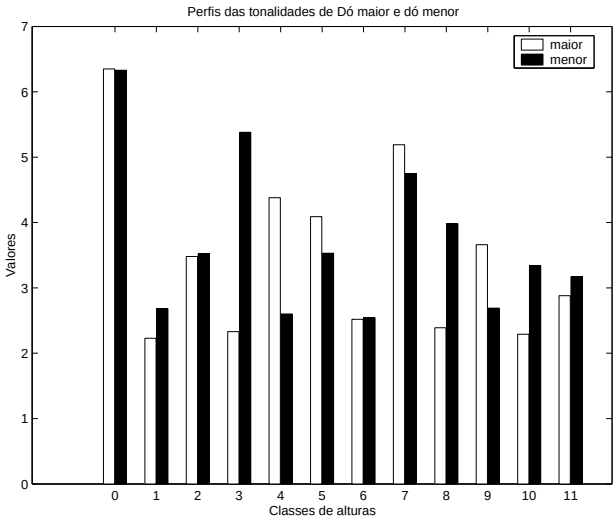

Figura D.1: Representação gráfica dos perfis da tabela D.1.

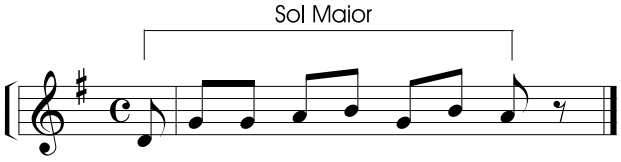

Figura D.2: Exemplo de Aplicação do algoritmo de Krumhansl & Schmuckler.

O próximo passo é realizar a correlação entre o vetor acima e os vetores correspondentes aos perfis de cada uma das tonalidades maiores e menores. Os valores encontrados podem ser vistos na tabela D.3, na qual também pode-se observar que o maior valor corresponde à tonalidade de Sol maior, o qual é o resultado correto.

Não obstante sua comprovada eficiência, o algoritmo, entretanto, apresenta dois problemas que dificultam ou impossibilitam, dependendo do caso, sua aplicação de forma mais abrangente. Inicialmente, o alto grau de subjetividade nos dados dos perfis, devido à própria natureza do experimento no qual foram obtidos, conduz, em trechos mais extensos a uma propagação de erro que tende a invalidar os resultados. Em segundo lugar, já comentado anteriormente, o caráter estático do algoritmo inviabiliza seu emprego em trechos melódicos onde ocorram modulações. No que diz respeito à primeira deficiência, já foram propostas por Temperley (2001) alterações nos valores dos perfis (tanto das tonalidades maiores quanto das menores) que provaram ser eficientes. Quanto à segunda, é proposto, neste trabalho, um método que generaliza o algoritmo para melodias com modulações.

Tabela D.2: Matriz com rotação da primeira coluna em 7 posições.

Classes de Alturas	Maior	Menor
7	6,35	6,33
8	2,23	2,68
9	3,48	3,52
10	2,33	5,38
11	4,38	2,60
0	4,09	3,53
1	2,52	2,54
2	5,19	4,75
3	2,39	3,98
4	3,66	2,69
5	2,29	3,34
6	2,88	3,17

D.3 Modificações Realizadas por Temperley

Temperley realiza em seu trabalho três modificações no algoritmo de identificação de tonalidades. Inicialmente, modifica os valores numéricos dos perfis originais, intensificando a diferença entre alturas cromáticas e diatônicas, diminuindo o valor do sétimo grau abaixado (que leva até à subdominante) e incrementando o valor do sétimo grau com função de sensível. Em seguida, emprega como entrada vetores de 12 posições que podem ser 1 ou 0, indicando ou não a presença de uma determinada classe de altura no trecho a ser analisado. Finalmente, adota em seu trabalho o que ele chama de "classes de alturas tonais" no lugar de "classes de alturas neutras" (como é o usual), visando melhores resultados. No caso do presente trabalho, entretanto, somente a primeira modificação apresenta-se como relevante contribuição, já que as demais não dizem respeito ao principal foco do mesmo, que é a identificação dinâmica de variações tonais. Os perfis modificados por Temperley são encontrados em (Temperley, 1999) e podem ser vistos na tabela D.4 e sua representação gráfica na figura D.3.

D.4 O Algoritmo Dinâmico

Este algoritmo foi pensado principalmente para permitir que linhas melódicas sem harmonização possam servir de entrada num sistema automático de análise baseado no Teoria Gerativa para Música Tonal, de Lerdahl & Jackendoff (1996), foco principal da presente tese. Na TGMT, a análise harmônica tem lugar capital na detecção da estrutura sintática de um trecho musical. Desta forma, na análise de uma amostra puramente melódica

Tabela D.3: Coeficientes de correlação do vetor de durações com os vetores correspondentes aos perfis de cada uma das tonalidades maiores e menores.

Tonalidade	Maior	Menor
Dó	0,274	-0,013
Dó#	-0,559	-0,332
Ré	0,543	0,149
Ré#	-0,130	-0,398
Mi	-0,001	0,447
Fá	0,003	-0,431
Fá#	-0,381	0,012
Sol	0,777	0,443
Sol#	-0,487	-0,106
Lá	0,177	0,251
Sib	-0,146	-0,513
Si	-0,069	0,491

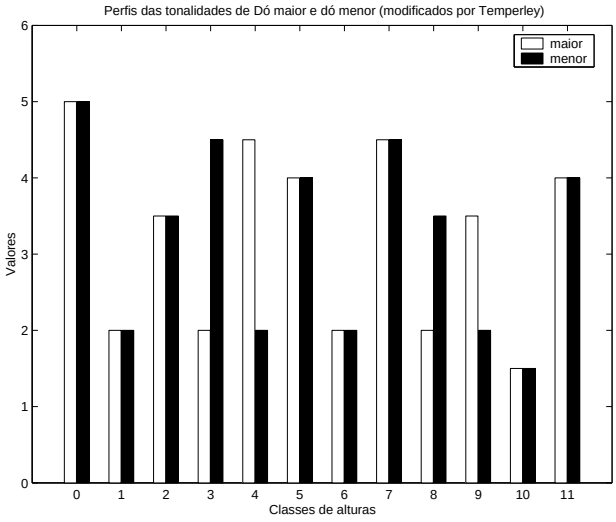

Figura D.3: Perfis modificados por Temperley para Dó maior e dó menor.

faz-se necessário o emprego de ferramentas auxiliares para, em primeiro lugar, mapear a seqüência de tonalidades subjacentes ao trecho melódico e, em seguida, propor uma seqüência harmônica plausível dentro deste conjunto de tonalidades que seja coerente com a melodia proposta. Para a solução da primeira parte do problema é proposto o algoritmo descrito neste apêndice. A segunda parte, deixada para o futuro, poderá ser resolvida, por exemplo, com a proposta descrita por Temperley & Sleator (1999) em seu artigo sobre metro e harmonia. O algoritmo aqui proposto é o seguinte:

Tabela D.4: Matriz com os perfis para as tonalidades de Dó maior e dó menor conforme modificados por Temperley.

Classes de Alturas	Maior	Menor
0	5,0	5,0
1	2,0	2,0
2	3,5	3,5
3	2,0	4,5
4	4,5	2,0
5	4,0	4,0
6	2,0	2,0
7	4,5	4,5
8	2,0	3,5
9	3,5	2,0
10	1,5	1,5
11	4,0	4,0

1. Em primeiro lugar, seleciona-se o tamanho da janela móvel e da taxa percentual de sobreposição;
2. Aplica-se o algoritmo de Krumhansl & Schmuckler aos elementos presentes na janela móvel;
3. Se o processo está no início, armazena-se numa lista a tonalidade encontrada no item anterior assim como um apontador para o primeiro elemento do trecho melódico;
4. Se não, verifica-se se a tonalidade encontrada é a mesma que a da janela anterior;
5. Se for, salta-se para o passo 7;
6. Se não, insere-se na lista a nova tonalidade e um apontador para o primeiro elemento da janela;
7. Se ainda existirem elementos para serem analisados, incrementa-se a janela e salta-se para o passo 2;
8. Se não, o algoritmo encerra-se.

Ainda que, nestes primeiros testes, o tamanho da janela e a sobreposição tenham sido escolhidos experimentalmente, um estudo que pudesse automatizar e otimizar estas escolhas apresenta-se como necessário e deverá ser realizado para as próximas versões do algoritmo.

Como saída do algoritmo, tem-se uma lista com as tonalidades subjacentes ao trecho melódico assim como uma lista com apontadores correspondentes aos elementos da

melodia onde cada uma delas tem início. As janelas móveis citadas no algoritmo foram implementadas de duas maneiras. Na primeira delas foram empregadas janelas temporais, utilizando um número fixo de unidades de análise de duração (o menor valor de duração presente no trecho). O problema desta abordagem é a realização de uma análise incorreta quando estão presentes no trecho durações muito longas. O outro método é o emprego de uma janela com número fixo de eventos, a qual apresenta o problema de ignorar as durações individuais de cada evento. Nos resultados apresentados neste trabalho foram empregadas janelas móveis temporais.

D.5 Resultados

Como amostras para os testes do algoritmo dinâmico foram empregadas quatro melodias modulantes e de curta extensão apresentando tanto exemplos no modo maior quanto no menor. Em todas as análises foi utilizada uma janela de 12 colcheias com um sobreposição de 2 colcheias (aproximadamente 17% da janela). Foram também empregados, para o cálculo, os valores dos perfis alterados por Temperley. As melodias, juntamente com os resultados encontrados para cada uma delas, são mostradas nas figuras D.4, D.5, D.6 e D.7.

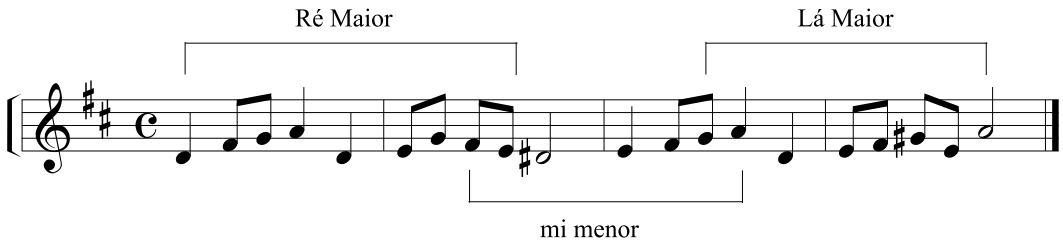

Figura D.4: Melodia modulante em Ré Maior.

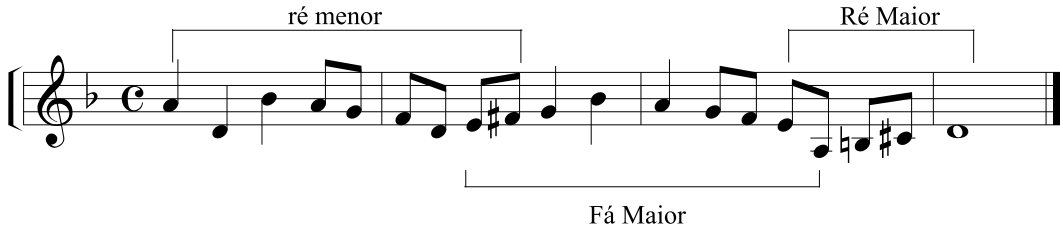

Figura D.5: Melodia modulante em ré menor.

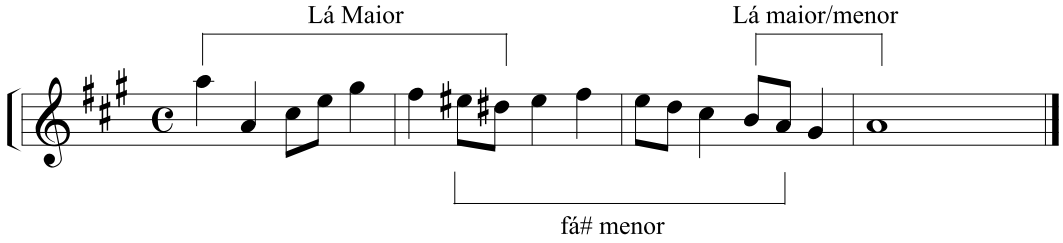

Figura D.6: Melodia modulante em Lá Maior.

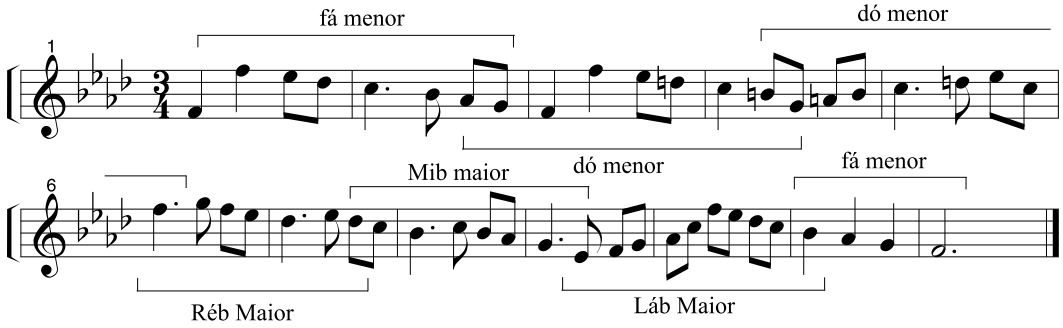

Figura D.7: Melodia modulante em fá menor.

D.6 Discussão

É possível observar-se que, de um ponto de vista geral, os resultados alcançados mostram-se satisfatórios no que diz respeito à análise do mapa de tonalidades, já que, em quase todos os casos, não apresentaram erros. Entretanto, alguns comentários fazem-se necessários para o caso da melodia em ré menor (figura D.5). Nesta, a modulação para a subdominante não foi detectada, possivelmente pelo tamanho excessivo da janela e pela pequena sobreposição. Além disto, nesta mesma melodia, o final em ré menor foi confundido com seu homônimo maior, equívoco causado pela ausência de uma nota característica que diferenciasse as duas tonalidades. No que diz respeito às demais melodias, o algoritmo não cometeu nenhum erro.

É possível notar-se, através deste trabalho, que o algoritmo original de Krumhansl & Schmuckler pode ser melhorado no sentido de realizar um mapeamento da seqüência tonal de um trecho melódico escrito segundo as normas da música clássica ocidental. Entretanto, esta abordagem poderá ser também estendida para outros contextos tonais, bastando para isto realizarem-se alterações nos dados das tabelas já mostradas, assim como desenvolvimentos no conjunto de restrições. O algoritmo proposto, desta forma, pode ser empregado também como um primeiro passo na harmonização automática de melodias.

Apêndice E

Utilização do Sistema

Neste apêndice é descrita, passo a passo, a maneira como deve-se utilizar o `Pierre`, nome do sistema descrito neste trabalho (ver seção 3.5, página 17).

É pressuposto que o usuário possua os executáveis do sistema, e não os códigos fonte. Assim, são omitidas todas as etapas de compilação e ligação com bibliotecas.

O pacote compilado do `Pierre` possui 11 arquivos:

1. `gpr`
2. `ms`
3. `tsr`
4. `prl`
5. `makebase`
6. `savebase`
7. `sfm`
8. `fbase.dat`
9. `config.par`
10. `prbase`
11. `pierre`

onde os primeiros quatro programas correspondem, na ordem em que estão dados, aos quatro componentes da teoria de Lerdahl & Jackendoff (1996). Os dois programas seguintes, respectivamente criam e arquivam as bases de dados para o sistema *fuzzy* empregado no primeiro componente da teoria. O programa `sfm` cria, a partir do arquivo texto `fbase.dat`, a base binária de funcionalidades harmônicas `func.dat`, a qual é utilizada pelo programa correspondente ao quarto componente da teoria. O arquivo texto `config.par`

contém uma lista de parâmetros que podem ser modificados pelo usuário a cada sessão de análise. Este arquivo, devido à sua importância, será separadamente abordado e explicado em uma seção exclusiva. O *script* `prbase` realiza todas as tarefas de preparação do ambiente citadas anteriormente e o *script* `pierre` é chamado com um arquivo MIDI, realizando a análise do arquivo, sendo que ambos os scripts estão mostrados a seguir. Inicialmente, o *script* `prbase`:

```
#
# Script para geração e arquivamento de bases de regras para o
# sistema fuzzy empregado no programa gpr.
# Deve ser chamado assim:
#
#     prbase num_pontos
#
# onde num_pontos é o número de pontos de treinamento.
#
#                       Gilberto Carvalho/2007
#
makebase $1
savebase trnpts.2a trnpts.2b trnpts.3a trnpts.3b trnpts.3c trnpts.3d
sfm fbase.dat

exit 0
#
# Fim
#
```

e em seguida o *script* `pierre`:

```
#
# Este script serve para executar em background cada um dos
# componentes da gramática.
# Também permite que sejam gravados logs da operação de cada um deles.
# Deve-se digitar na linha de comando
#
#        pierre arquivo
#
# onde 'arquivo' é o nome do arquivo MIDI sem extensão.
#
#                                 Gilberto Carvalho - 2007
#

echo ´Início do processamento...´
echo
```

```
#
# Apaga todos os arquivos temporários e os logs
# da sessão anterior
#
rm -f *.tmp
rm -f *.log

#
# Roda os quatro componentes em background
#
gpr $1 > grp.log &
ms  $1 > ms.log  &
tsr $1 > tsr.log &
prl $1 > prl.log &

#
# Espera os processos terminarem
#
wait

echo ´...Fim.´
echo

exit 0

#
# FIM
#
```

E.1 CONFIG.PAR

Este arquivo contém um conjunto de parâmetros (atualmente em número de 7) que podem ser modificados pelo usuário, influindo, desta forma, nos resultados alcançados. A cada etapa de manutenção do sistema, novos parâmetros deverão, possivelmente, ser acrescentados. A estrutura do arquivo é muito simples, pois basta seguir a cada nome de parâmetro seu valor correspondente e acrescentar comentários onde for necessário, sendo que estes devem estar em chaves e não podem ser aninhados. A versão atual do arquivo está a seguir.

```
{ ---> PARÂMETROS DE CONFIGURAÇÃO <--- }

{ Parâmetros da Estrutura de Agrupamento }
```

```
group_lower_threshold   2    { limiar inferior para tamanho de um grupo }
N                       2    { fator de vizinhança. Deve ser potência de 2 }
percep_threshold      0.2    { limiar de percepção }

{ Parâmetros da Estrutura de Métrica }
pop_size         15   { tamanho da população no algoritmo genético }
num_generations  10   { número de gerações }
crossover_rate   0.8  { taxa de crossover }
mutation_rate    0.1  { taxa de mutação }

{ ------------------- FIM----------------- }
```

Bibliografia

Aucouturier, J.-J. (2001). *Segmentation of Musicals Signals, and Applications to the Analisys of Musical Structure*, Master's thesis, Audio & Music Processing Lab., King's College, University Of London.

Brown, C. (1994). *UNIX Distributed Programming*, Prentice Hall, Hemel Hempstead, Hertfordshire.

Cambouropoulos, E. (1998). *Towards a General Computational Theory of Musical Structure*, PhD thesis, University of Edinburgh, Faculty of Music and Department of Artificial Intelligence.

Carvalho, A. G. M. d. (2001). *Implementação Computacional da Estrutura de Agrupamento da Teoria Gerativa de Lerdahl & Jackendoff para Música Tonal*, Master's thesis, Centro de Pesquisa e Desenvolvimento em Engenharia Elétrica (CPDEE) – Universidade Federal de Minas Gerais.

Coello, C. A. C. (2000). An Updated Survey of GA-Based Multiobjective Optimization Techniques, *ACM Computing Surveys* **32**(2): 109–143.

Curry, B., Wiggins, G. & Hayes, G. (2000). Representing Trees with Constraints, *Informatics Research Report EDI-INF-RR-0024*, University of Edinburgh – Institute of Perception, Action and Behaviour.

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation* **6**(2).

Dias, A. H. & Vasconcelos, João A, d. (2002). Multiobjective Genetic Algorithms Applied to Solve Optimization Problems, *IEEE Transactions On Magnetics* **38**(2): 1133–1136.

Eck, D. S. (2002). *Meter Through Synchrony: Processing Rhythmical Patterns With Relaxation Oscillators*, PhD thesis, Department of Computer Science – Indiana University.

Franz, D. M. (1998). *Markov Chains as Tools for Jazz Improvisation Analysis*, Master's thesis, Virginia Polytechnic Institute, USA.

Gerhard, D. (2002). Computer Music Analysis, *Technical Report CMPT TR 97-13*, Simon Fraser University, School of Computing Science, Canada.

Glass, G. (1993). *UNIX for Programmers and Users*, Prentice Hall, Upper Saddle River, New Jersey.

Goldberg, D. E. (1989). *Genetic Algorithm in Search, Optimizatioan and Machine Learning*, Addison-Wesley Publishing, Reading, Massachusetts.

Goldman, C. V., Gang, D., Rosenschein, J. S. & Lehmann, D. (1999). Negneg: A Connectionist-agent Integrated System for Representing Musical Knowledge, *Annals of Mathematics and Artificial Intelligence* **25**(1,2): 69–90.

Hamanaka, M. & Hirata, K. (2002). Applying Voronoi Diagrams in the Automatic Grouping of Polyphony, *Forum on Information Technology*, Tokyo, Japan, pp. 101–102.

Hamanaka, M., Hirata, K. & Satoshi, T. (2007). Implementing GTTM. Artigo em rascunho gentilmente cedido pelos autores.

Hamanaka, M., Hirata, K. & Tojo, S. (2004). Automatic Generation of Grouping Structure based on the GTTM, *Proceedings of the ICMC2004*, pp. 141–144.

Hamanaka, M., Hirata, K. & Tojo, S. (2005a). ATTA: Automatic Timespan Tree Analyzer based on Extended GTTM, *Proceedings of the ISMIR2005*, pp. 358–365.

Hamanaka, M., Hirata, K. & Tojo, S. (2005b). Automatic Generation of Metrical Structure based on the GTTM, *Proceedings of the ICMC2005*, pp. 53–56.

Hoffman, T. & Birmingham, W. P. (2000). A Constraint Satisfaction Approach To Tonal Harmonic Analysis, *Technical Report. CSE-TR-397-99*, Electrical Engineering And Computer Science Department – The University Of Michigan.

Holland, S. (1989). *Artificial Intelligence, Education and Music*, PhD thesis, The Open University, UK.

Holland, S. (2000). Artificial Intelligence in Music Education: A Critical Review, *in* E. Miranda (ed.), *Readings in Music and Artificial Intelligence*, Routledge, pp. 239–274.

Hsu, J.-l., Liu, C.-c. & Chen, A. L. P. (2001). Discovering Nontrivial Repeating Patterns in Music Data, *Ieee Transactions On Multimedia* **3**(3): 311–325.

Jang, J.-S., Sun, C. T. & Mizutani, E. (1997). *Neuro-Fuzzy and Soft Computing*, Prentice Hall, Upper Saddle River, New Jersey.

Kalyanmoy, D. (1998). Multi-objective genetic algorithms: Problem difficulties and construction of test problems. Technical Report.

Kohonen, T., Hynninen, J., Kangas, J. & Laaksonen, J. (1996). Som pak: The self-organizing map program package. Report A31, Helsinki University of Technology, Laboratory of Computer and Information Science, Jan. 1996.
*http://www.cis.hut.fi/research/som-research

Krumhansl, C., J.J., B. & Kessler, E. (1982). Perceived harmonic structure of chords in three related musical keys, *Journal of Experimental Psychology: Human Perception and Performance* **8**: 24–36.

Krumhansl, C. L. (1990). *Cognitive Foundations of Musical Pitch*, Oxford University Press, Oxford.

Krumhansl, C. L. & Kessler, E. (1982). Tracing the dynamic changes in perceived tonal organisation in spatial representation of musical keys, *Psychological Review* **89**: 334–368.

Krumhansl, C. & Shepard, R. (1979). Quantification of the hierarchy of tonal functions within a diatonic context, *Journal of Experimental Psychology: Human Perception and Performance* **5**(4): 579–594.

Large, E. W. & Kolen, J. F. (1999). Resonance And The Perception Of Musical Meter, *in* N. Griffith & P. Todd (eds), *Musical Networks: Parallel Distributed Perception and Performance*, MIT Press, Cambridge, Massachussets, pp. 279–312.

Lartillot, O. (2002). Musical Analysis by Computer Following Cognitive Model of Induction of Analogies. Paper on musical analysis.
*citeseer.ist.psu.edu/663115.html

Lerdahl, F. (1989). Analyse de "La Terrasse des Audiences du Clair de Lune" de Debussy, *Analyse Musicale* **16**: 54–60.

Lerdahl, F. (2001). *Tonal Pitch Space*, second ed., Oxford University Press, New York, New York.

Lerdahl, F. & Jackendoff, R. (1996). *A Generative Theory of Tonal Music*, second ed., MIT Press, Cambridge, Massachusetts.

Lew, A. & Mauch, H. (2007). *Dynamic Programming – A Computational Tool*, first ed., Springer, Berlin.

Lu, H. & Yen, G. G. (2003). Rank-Density-Based Multiobjective Genetic Algorithm and Benchmarck Test Function Study, *IEEE Transactions On Evolutionary Computation* **7**(4): 325–343.

Maidin, D. S. (1995). *A Programmer's Environment for Music Analysis*, PhD thesis, National University of Ireland at University College Cork.

Marco, N., Désidéri, J.-A. & Lanteri, S. (1999). Multi-Objective Optimization in CFD by Genetic Algorithms, *Technical Report 3686*, Institut National de Recherche en Informatique et en Automatique .

Meehan, J. R. (1979). An Artificial Intelligence Approach to Tonal Music Theory, *ACM Annual Conference – Proceedings of the 1979 Annual Conference*, New York, USA, pp. 116–120.

Meredith, D. (2004). Musical Metre. Survey on Musical Metre.
*http://www.titanmusic.com

Meudic, B. (2001). *Modélisation de structures rythmiques*, Master's thesis, Université d'Aix-Marseille II. Mémoire de Dea Atiam.

Monti, G. (2001). Signal Processing And Music Analysis, *Research work report.*, Universitá di Bologna, Italia.

Nienhuys, H.-W. & Nieuwenhuizen, J. (2007). LilyPond. Programa on musical engraving for automatec music edition.
*http://www.lilypond.org

Orife, I. F. O. (2001). *Riddim: A Rhythm Analysis And Decomposition Tool Based On Independent Subspace Analysis*, Master's thesis, Dartmouth College, Hanover, New Hampshire.

Port, R., Tajima, K. & Fred, C. (1998). Speech and Rhythmic Behavior. Studies concerning the rhythm inside the speech.

Press, W. H., Teukolsky, Saul A.and Vetterling, W. T. & Flannery, B. P. (1997). *Numerical Recipes in C: The Art of Scientific Computing*, second ed., Cambridge University Press, Cambridge, Massachusetts.

Sbalzarini, I. F., Mller, S. & Koumoutsakos, P. (2000). Multiobjective optmization using evolutionary algorithms, *Proceedings of the Summer Program 2000*, Institute of Computational Sciences (ETH Zrich, Switerland) and Center for Turbulence Research (NASA Ames Stanford University.

Schaffer, J. (1984). *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms*, PhD thesis, Vanderbilt University, Nashville (TN).

Schönberg, A. (1977a). *Preliminary Exercises in Counterpoint*, second ed., Faber and Faber, London.

Schönberg, A. (1977b). *Structural Functions of Harmony*, third ed., Norton, London.

Schönberg, A. (1978). *Theory of Harmony*, Faber and Faber, London.

Schönberg, A. (1980). *Fundamentals of Musical Composition*, fourth ed., Faber and Faber, London.

Schönberg, A. (1984). *Style and Idea*, Faber and Faber, London.

Seppánnen, J. (2001). *Computational Models Of Musical Meter Recognition*, Master's thesis, Tampere University Of Technology – Department of Information Technology.

Snyder, J. S. & Large, E. W. (2002). Neurophysiological Correlates Of Meter Perception: Evoked And Induced Gamma-band (20-60 Hz) Activity, *Proceedings of the 7th International Conference on Music Perception and Cognition*, Sydney, Australia.

Srinivas, N. & Deb, K. (1994). Multiobjective Optimization Using Nondominated Sorting Genetic Algorithms, *Evolutionary Computation* **2**(3): 221–248.

Stevens, W. (1999). *UNIX Networking Programming – Interprocess Communications*, Vol. second, second ed., Prentice Hall, Upper Saddle River, New Jersey.

Straus, J. N. (1990). *Introduction to Post-tonal Theory*, Prentice-Hall, Englewood Cliffs, New Jersey.

Temperley, D. (1996). Hypermetrical Ambiguity In Sonata Form Closing Themes. Presented at the 1996 Meeting of the Society for Music Theory.
*http://www.theory.esm.rochester.edu/temperley/hyp-amb-clo.pdf

Temperley, D. (1999). What's Key for Key? The Krumhansl-Schmuckler Key-Finding Algorithm Reconsidered, *Music Perception* **17**: 65–100.

Temperley, D. (2001). *The Cognition of Basic Musical Structures*, MIT Press, Cambridge.

Temperley, D. (2004). An Evaluation System for Metrical Models, *Computer Music Journal* **28**(3): 28–44.

Temperley, D. & Sleator, D. (1999). Modeling Meter an Harmony: A Preference-Rule Approach, *Computer Music Journal* **23**(1): 10–27.

Thom, B., Spevak, C. & Hthker, K. (2002). Melodic segmentation: evaluating the performance of algorithms and musical experts. Paper on musical segmentation.

Tsang, E. (1993). *Foundations of Constraint Satisfaction*, Academic Press, London.

Vasconcelos, J. A. d. (2003). Algoritmos Genéticos e Otimização de Sistemas em Engenharia. Notas de aula.

Wang, L.-X. (1994). *Adaptive Fuzzy Systems and Control*, Prentice Hall, Englewood Cliffs, New Jersey.

Wang, L.-X. & Mendel, J. M. (1992). Generating fuzzy rules by learning from examples, *IEEE Transactions on Systems, Man, and Cybernetics* **22**(6): 1414–1427.

Weyde, T. (2001). Grouping, Similarity and the Recognition of Rhythmic Structure, *Proceedings of the International Computer Music Conference 2001*, International Computer Music Conference, Havanna, Cuba, pp. 475–478.

Weyde, T. (2002). Knowldege and Learning Based Segmentation and Recognition of Rhythm Using Fuzzy-prolog. Paper presenting musical segmentation with computational intelligence.

Zitzler, E., Laumanns, M. & Bleuler, S. (2004). A Tutorial on Evolutionary Multiobjective Optimization. Swiss Federal Institute of Technology (ETH), Computer Engineering an Networks Laboratory (TIK), Zurich, Switzerland.

Zizler, E., Laumanns, M. & Thiele, L. (2002). Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization, *in* K. Giannakoglou, D. Tsahalis, J. Periaux, K. Papailiou & T. Fogarty (eds), *Evolutionary Methods for Design, Optimization and Control*, CIMNE, Barcelona, Spain.

Zizler, E. & Thiele, L. (1999). Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strengh Pareto Approach, *IEEE Transactions On Evolutionay Computation* **3**(4): 257–271.