CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

COMBINANDO REGRAS E PROPORÇÕES: UMA ABORDAGEM MULTIOBJETIVO PARA COMPOSIÇÃO MUSICAL ALGORÍTMICA

HENRIQUE BARROS LOPES

Orientador: Vinícius Fernandes dos Santos
Universidade Federal de Minas Gerais

Coorientador: Flávio Vinícius Cruzeiro Martins
Centro Federal de Educação Tecnológica de Minas Gerais

BELO HORIZONTE
JULHO DE 2017

HENRIQUE BARROS LOPES

COMBINANDO REGRAS E PROPORÇÕES: UMA ABORDAGEM MULTIOBJETIVO PARA COMPOSIÇÃO MUSICAL ALGORÍTMICA

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Matemática e Computacional do Centro Federal de Educação Tecnológica de Minas Gerais, como requisito parcial para a obtenção do título de Mestre em Modelagem Matemática e Computacional.

Área de concentração: Modelagem Matemática e Computacional

Linha de pesquisa: Sistemas Inteligentes

Orientador: Vinícius Fernandes dos Santos
Universidade Federal de Minas Gerais

Coorientador: Flávio Vinícius Cruzeiro Martins
Centro Federal de Educação Tecnológica de Minas Gerais

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL
BELO HORIZONTE
JULHO DE 2017

Lopes, Henrique Barros
L864c Combinando regras e proporções: uma abordagem multiobjetivo para composição musical algorítmica / Henrique Barros Lopes. – 2017.
76 f.

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional.
Orientador: Vinícius Fernandes dos Santos.
Dissertação (mestrado) – Centro Federal de Educação Tecnológica de Minas Gerais.

1. Música – Modelos matemáticos – Teses. 2. Composição musical por computador – Teses. 3. Algoritmos genéticos – Teses. 4. Inteligencia artificial – Teses. I. Santos, Vinícius Fernandes dos. II. Centro Federal de Educação Tecnológica de Minas Gerais. III. Título.

CDD 006.3

Elaboração da ficha catalográfica pela Biblioteca-Campus II / CEFET-MG

Agradecimentos

Agradeço em primeiro lugar aos meus pais José Leonardo e Maria Ruth, por me proporcionarem uma excelente educação que com certeza foi fundamental para a finalização deste trabalho. Agradeço a eles também por todo apoio emocional durante todo o momento em que estive estudando e por me apoiarem na decisão de ingressar neste curso de mestrado.

Agradeço aos meus irmãos, Sarah, César, Mateus e Cecília por todo carinho que sempre tiveram comigo, seu irmão mais velho, e por todo apoio, mesmo que inconsciente, nesse percurso que caminhei.

Aos meus orientadores Vinícius e Flávio, pessoas extremamente competentes com quem eu tive imenso prazer de trabalhar durante esses dois anos. Sem vocês não seria possível conseguir tudo que conseguimos.

Ao professor Rodrigo pela apresentação de nosso trabalho no CEC.

Ao professor Fábio e ao meu primo Matheus pela consultoria em estatística.

Agradeço aos meus amigos Luís Henrique, Isabella, Talhadas, Carlos, Góber, João, Waner, Jorge e Fernanda, por todo apoio e descontração que me proporcionaram durante essa jornada.

Aos meus colegas do MMC Túlio, Samara, Pedro, Rafaela, Alexandre, Daniel, Moisés, Bruno e Wagner pelo companheirismo e amizade durante nosso curso.

Aos professores e funcionários do MMC pelos serviços prestados que foram essenciais para minha formação.

Ao CEFET-MG e à Capes pela estrutura fornecida e pelo apoio financeiro.

Resumo

Composição musical é uma atividade da produção fonográfica de alta complexidade e que, até mesmo para os músicos mais experientes, demanda grande quantidade de tempo para ser concluída. Apesar de existirem vários trabalhos na área, a composição musical por um computador ainda é um tema bastante incipiente na literatura e os resultados obtidos ainda se encontram longe de serem utilizados como insumo na indústria do entretenimento. Este trabalho propõe quatro métodos para composição de melodias monofônicas, dois mono-objetivos e dois multiobjetivos, cuja abordagem combina regras de Fux, provenientes de teoria musical e regras de proporções baseadas na lei de Zipf, em uma implementação do algoritmo NSGA-II, e uma nova função de avaliação baseada na lei de Zipf. Propusemos também uma nova metodologia baseada em questionários capaz de avaliar a qualidade de um método de composição algorítmica. Nós utilizamos essa nova metodologia para validar nossos resultados e verificar a qualidade das melodias compostas pelos nossos métodos.

Palavras-chave: Composição Algorítmica, Sistemas Inteligentes, Música, Algoritmos Genéticos, Algoritmos Multiobjetivos, Inteligência Artificial, Composição Musical.

Abstract

Musical composition is an activity of high complexity in phonographic production and that, even for the most experient musicians, demands great amount of time to be finished. Although there are many works in the field, computer music composition is an incipient theme in literature and the results remain far from being used as input in the entertainment industry. This work proposes four new methods to compose monophonic melodies, two of which are mono-objectives approaches and two multi-objectives approaches that combine Fux's rules from music theory and proportions' rules based on Zipf's law in an implementation of the NSGA-II algorithm. We also propose a new fitness function based on Zipf's law. We also proposed a new methodology based on questionnaires that is capable to evaluate the quality of an algorithmic composition method. We used this new methodology to validate our results and to verify the melodies' quality composed by our methods.

Keywords: Algorithmic Composition, Intelligent Systems , Music, Genetic Algorithms, Multi-objective Algorithms, Artificial Intelligence, Music Composition.

Lista de Figuras

Figura 1 – Diagrama de estratégias de composição algorítmica. Extraído de (FERNÁNDEZ; VICO, 2013), traduzido para o português e adaptado para este trabalho. 7
Figura 2 – Gráfico do Logaritmo das Ocorrências pelo Logaritmo das Classificações 21
Figura 3 – Fluxograma de um algoritmo genético. 28
Figura 4 – Tipo de indivíduo implementado. 32
Figura 5 – Gráfico Função Gaussiana 33
Figura 6 – Diferença do erro quadrático médio para duas regressões com coeficientes angulares próximos de -1 35
Figura 7 – Execução do operador *crossOverBar*. 37
Figura 8 – Modificação do trigrama rítmico pelo operador mutateRhythmTrigram . . 38
Figura 9 – Regressão linear ($y = bx + a$) do fenômeno trigrama rítmico de uma música composta neste trabalho: $b = -0{,}72$ 38
Figura 10 – Modificação do trigrama melódico pelo operador mutateMelodicTrigram . 39
Figura 11 – Exemplo de conjunto pareto-ótimo de um problema de minimização, formado pelos pontos A, B e E. 41
Figura 12 – Avaliações de melodias aleatórias. 49
Figura 13 – Curva de convergência de 33 execuções dos métodos mono-objetivos. . 50
Figura 14 – Pareto combinado dos métodos 3 e 4. O método multiobjetivo 4 é representado pelos pontos em formato de cruz vermelha e o método multiobjetivo 3 é representado pelos pontos em formato de asteriscos azuis. As cruzes verdes e pretas representam os 33 melhores indivíduos dos métodos mono-objetivos 1 e 2 respectivamente. 53
Figura 15 – Pareto combinado dos métodos 3 e 4. Legenda análoga a Figura (14). . 53
Figura 16 – Pareto combinado dos métodos 3 e 4. Legenda análoga a Figura (14). . 54
Figura 17 – Pareto combinado dos métodos 3 e 4. Legenda análoga a Figura (14). . 54
Figura 18 – Respostas dos questionários por grupos. 58
Figura 19 – Resíduos amostrais das respostas do questionário para os métodos 1, 2, 3, e 4. 59
Figura 20 – Respostas dos questionários por grupos. 62
Figura 21 – Resíduos amostrais das respostas do questionário para melodias compostas em Dó maior e com diferentes operadores de mutação. 62
Figura 22 – Partitura da melodia composta utilizando o método 4 da capítulo 5, com taxa de cruzamento $0{,}7$ e mutação $0{,}3$. 75
Figura 23 – Partitura da melodia composta utilizando o método 4 da capítulo 5, com taxa de cruzamento $0{,}7$ e mutação $0{,}3$ em Dó maior. 76

Figura 24 – Partitura da melodia composta utilizando o método 4 da capítulo 5, com taxa de cruzamento $0{,}7$ e mutação $0{,}3$, em Dó maior e com a mutação *mutateCopyingPartMusic*. 76

Lista de Tabelas

Tabela 1 – Objetivos e áreas de CAC - Extraída de (HERREMANS, 2014), traduzido para o português e adaptada para este trabalho. 1
Tabela 2 – Tabela Resumo 17
Tabela 3 – Ocorrências de Intervalos Melódicos 20
Tabela 4 – Ocorrências de Intervalos Melódicos Ordenadas 20
Tabela 5 – Logaritmo de Ocorrências e Classificações 20
Tabela 6 – Estrutura de Dados de uma nota Musical 31
Tabela 7 – Tabela ANOVA do método 1 51
Tabela 8 – Tabela ANOVA do método 2 52
Tabela 9 – Músicas compostas por músicos avaliadas por Fux e ZipfEQM 55
Tabela 10 – Testes de Dunn para comparação múltipla dos grupos das respostas do questionário para os métodos 1, 2, 3 e 4. 59
Tabela 11 – Influência da instrução musical na avaliação de melodias dos métodos 1, 2, 3 e 4. 60
Tabela 12 – Teste de Dunn para comparação múltipla dos grupos das respostas dos questionários para melodias compostas em Dó maior e com diferentes operadores de mutação. 61
Tabela 13 – Influência da instrução musical na avaliação de melodias compostas em Dó maior e com diferentes operadores de mutação 64

Lista de Algoritmos

Algoritmo 1 – Algoritmo NSGA-II. Extraído de (DEB et al., 2002) e adaptado para este trabalho. 42

Algoritmo 2 – Procedimento *fast nondominated sorting*. Extraído de (DEB et al., 2002) e adaptado para este trabalho. 44

Algoritmo 3 – Procedimento *crowding distance assignment*. Extraído de (DEB et al., 2002) e adaptado para este trabalho. 45

Sumário

1 – Introdução 1
1.1 Contextualização do Problema 1
1.2 Justificativa 2
1.3 Objetivos 3
1.4 Metodologia 3
1.5 Organização do trabalho 4
1.6 Publicações 5

2 – Trabalhos Relacionados 6
2.1 Histórico 6
2.2 Estratégias de Inteligência Artificial para Composição Algorítmica 6
2.2.1 Gramáticas 8
2.2.2 Sistemas Simbólicos e Baseados em Regras 9
2.2.3 Cadeias de Markov 10
2.2.4 Redes Neurais Artificiais 11
2.2.5 Algoritmos Evolucionários 13
2.2.6 Abordagens Multiobjetivo 15
2.3 Considerações Finais 16

3 – Métricas de Avaliação Musical 18
3.1 Lei de Zipf 18
3.1.1 Aplicação da Lei de Zipf na Música 18
3.1.2 Cálculo da Lei de Zipf 19
3.2 Fux 21
3.2.1 Regras de Contraponto de Quinta Espécie 22
3.2.2 Funções de Regras de Fux 23
3.3 Considerações Finais 26

4 – Algoritmo Genético Aplicado a Composição Algorítmica 27
4.1 Algoritmos Genéticos 27
4.1.1 Definição do Indivíduo 27
4.1.2 Função de Avaliação 28
4.1.3 Operadores 29
4.1.3.1 Seleção 29
4.1.3.2 Cruzamento 29
4.1.3.3 Mutação 30

4.1.4 Definindo a Nova População . . . 30
4.2 Aplicação de Algoritmos Genéticos em Composição Algorítmica . . . 31
4.2.1 Codificação do Indivíduo . . . 31
4.2.2 Função de Avaliação Musical . . . 32
4.2.2.1 Funções de Avaliação baseadas na Lei de Zipf . . . 32
4.2.2.1.1 Função Gaussiana . . . 33
4.2.2.1.2 Função Erro Médio Quadrático . . . 34
4.2.2.2 Função de Avaliação baseada nas Regras de Fux . . . 36
4.2.3 Operadores de Cruzamento . . . 36
4.2.4 Operadores de Mutação . . . 37
4.2.5 Definição da Próxima Geração . . . 39
4.3 Abordagem Multiobjetivo . . . 40
4.3.1 Otimização Multiobjetivo . . . 40
4.3.2 Método NSGA-II . . . 41
4.3.3 Aplicação do NSGA-II na Composição Algorítmica . . . 43
4.4 Considerações Finais . . . 45

5 – Análise e Discussão dos Resultados . . . 47
5.1 Codificação e Simplificação Musical . . . 47
5.1.1 Formato de Arquivo e Biblioteca . . . 47
5.1.2 Simplificações Musicais . . . 48
5.2 Resultados Numéricos . . . 48
5.2.1 Experimentos com Métodos Mono-objetivos . . . 49
5.2.2 Experimentos com Métodos Multiobjetivos . . . 52
5.2.3 Avaliação de Músicas Reais . . . 54
5.3 Resultados Qualitativos . . . 55
5.3.1 Elaboração do Questionário . . . 56
5.3.2 Coeficiente de Criatividade . . . 57
5.3.3 Estatísticas de Questionário dos 4 Métodos Iniciais . . . 57
5.3.4 Estatísticas de Questionário com Outras Mutações e em Dó Maior . 60

6 – Conclusão . . . 65
6.1 Trabalhos Futuros . . . 66
6.2 Considerações Finais . . . 66

Referências . . . 67

Apêndices 74

APÊNDICE A – Partituras de Melodias Compostas no Trabalho 75

1 Introdução

Apesar da escassez de tempo que as pessoas enfrentam nos dias de hoje, a indústria do entretenimento produz cada vez mais atrações que ocupam cada vez mais nosso tempo livre. Essas atrações consistem em uma variedade enorme de produtos dessa indústria, como por exemplo: filmes, seriados, músicas, programas de TV e vídeo clipes. O volume de produção da indústria do entretenimento é tão grande e tão usado que não é de se assustar que grande parte do tráfego em toda internet consista de mídia. Isto torna interessante e motivador estudar e melhorar as diversas etapas dessa cadeia produtiva, em especial, a etapa de composição musical, intrínseca a diversos produtos dessa indústria.

1.1 Contextualização do Problema

Desde os anos 50, computadores são utilizados como ferramentas para auxiliar a composição musical (PINKERTON, 1956; BROOKS et al., 1957) e, ao longo dos anos, foram utilizadas diferentes formas para realizar tal atividade. Pearce, Meredith e Wiggins (2002) introduziram quatro objetivos considerados principais na composição automatizada por computador (CAC): expansão de repertório composicional, desenvolvimento de ferramentas para compositores, proposição e avaliação de teorias de estilos musicais e proposição e avaliação de teorias cognitivas para composição musical. Cada um desses objetivos está contido em sua área de atuação conforme Tabela 1.

Tabela 1 – Objetivos e áreas de CAC - Extraída de (HERREMANS, 2014), traduzido para o português e adaptada para este trabalho.

Domínio	**Atividade**	**Objetivos**
Composição	Composição algorítmica	Expansão de repertório composicional
Engenharia de software	Desenho de ferramentas composicionais	Desenvolvimento de ferramentas para compositores
Musicologia	Modelagem computacional de estilos musicais	Propostas e avaliações de teorias de estilos musicais
Ciência cognitiva	Modelagem computacional de cognição musical	Propostas e avaliações de teorias cognitivas da composição musical

A composição algorítimica (CA) é definida quando o compositor vê o computador como uma extensão do seu próprio trabalho de composição, uma ferramenta que lhe

fornece insumos musicais, fazendo com que ele possa enxergar mais possibilidades de composições durante o processo. Quando tratamos da atividade de desenvolvimento de ferramentas composicionais, estamos nos referindo a softwares que auxiliam o compositor a escrever suas músicas. Dois exemplos de softwares desse tipo são Guitar Pro e Encore. Por meio desses softwares é possível escrever uma partitura digitalmente podendo reproduzi-la a fim de escutar os sons que foram compostos. Modelagem computacional de estilos está atrelada à implementação de teorias musicais. A quarta e última atividade está relacionada a teorias cognitivas composicionais, ou seja, como a mente trabalha quando estamos compondo uma música.

As atividades contidas na Tabela 1 são extremamente correlacionadas, de forma que ao trabalhar com um objetivo específico, estamos indiretamente trabalhando com outros. Neste trabalho temos um pouco das quatro questões apresentadas, porém este trabalho possui está mais relacionado a primeira questão, composição algorítmica.

Uma consideração importante a se fazer quando estamos tratando de composição algorítmica é que o resultado final é geralmente uma música simbólica, ou seja, uma partitura que indica quais notas devem ser tocadas e qual a duração de cada nota, que é diferente de uma música real que é gravada em estúdio e distribuída no mercado em forma de CDs ou arquivos MP3. Muitos trabalhos de composição algorítmica geram arquivos MIDI *(Musical Instrument Digital Interface)*. MIDI é um sistema de codificação que é usado para representar, transferir e armazenar informações musicais. Ao invés de conter amostras de sons, arquivos MIDI armazenam instruções que podem ser executadas em um computador que reproduz uma série de sons a partir delas. McKay (2004) apresentou informações técnicas de como um arquivo MIDI é construído e interpretado pelo computador. Existem outros tipos de arquivos de música simbólica, Field (1997) forneceu uma visão geral de quais e como são esses arquivos.

A composição algorítmica consiste na utilização de um computador para realizar alguma das tarefas do processo de composição musical. Pode-se utilizar um algoritmo para compor harmonias de uma melodia pré-existente, podemos também compor melodias monofônicas, ou seja, melodias de um único instrumento em que apenas uma nota é tocada em um mesmo intervalo de tempo. Podemos compor melodias polifônicas, em que podemos ter vários instrumentos ou várias notas tocadas simultaneamente em um mesmo intervalo de tempo, entre outras tarefas. Neste trabalho comporemos apenas melodias monofônicas.

1.2 Justificativa

Composição musical é uma atividade da produção fonográfica de alta complexidade e que, até mesmo para os músicos mais experientes, demanda grande quantidade de tempo para ser concluída. Herremans (2014) sugeriu em sua tese intitulada *“Compose = Compute”*

que, desde que você consiga modelar matematicamente as regras de um determinado estilo musical, é possível fazer com que um computador componha novas músicas. Em seu trabalho, (HERREMANS, 2014) modelou matematicamente o estilo denominado música de contraponto a partir de regras propostas por Fux e Mann (1971).

A composição musical por um computador ainda é uma área bastante incipiente na literatura. Trabalhos atuais introduzem algoritmos para composição de músicas, porém, apesar dos resultados apresentarem sequências de sons mais audíveis que uma sequência aleatória e existir o uso, mesmo que escasso, dessas composições como ponto de partida para músicos, as composições geradas dessa forma não possuem, por exemplo, qualidade adequada para se obter um ganho considerável em uma produção fonográfica.

1.3 Objetivos

Nosso objetivo geral é criar um sistema capaz de compor melodias monofônicas sem nenhuma intervação humana, porém não necessariamente perfeitas do ponto de vista musical, combinando funções de avaliação baseadas em regras de teoria musical e de proporções para encontrar resultados melhores do que os obtidos com abordagens mono-objetivo. Especificamente deseja-se:

- Elaborar uma métrica capaz de avaliar o quanto um trecho de melodia ou uma melodia é considerado bom em relação à lei de Zipf;
- Implementar um algoritmo genético para composição de melodias monofônicas baseadas na função de avaliação proposta no item anterior;
- Adaptar o algoritmo genético para criar melodias monofônicas baseadas nas regras de Fux;
- Implementar um algoritmo multiobjetivo que utiliza regras de Fux e lei de Zipf como funções de avaliação de melodias;
- Elaborar um questionário para que pessoas possam avaliar as músicas compostas algoritmicamente;
- Validar o trabalho comparando as melodias geradas com melodias aleatórias e melodias compostas por compositores humanos a partir das respostas do questionário.

1.4 Metodologia

Serão executadas as seguintes atividades ao longo do projeto:

1. Definir uma função de avaliação a partir do erro quadrático médio da regressão linear dos fenômenos musicais da lei de Zipf.
2. Utilizar a métrica acima como função de avaliação em um algoritmo genético para compor melodias monofônicas. Será feita uma análise estatística baseada em análise

de variância para verificar a influência de diferentes taxas de mutação e cruzamento no valor final de avaliação do algoritmo. Gráficos de convergência para os diferentes parâmetros serão exibidos para visualizar o desempenho do algoritmo.

3. Utilizar as funções matemáticas derivadas das regras de Fux citadas em Herremans (2014) que se aplicam a este trabalho como função de avaliação em um algoritmo genético para compor melodias. A mesma análise estatística feita para o item acima será realizada para este algoritmo bem como os gráficos de convergência.
4. Implementar o algoritmo NSGA-II utilizando como funções de avaliação as duas funções definidas acima. Serão plotados gráficos que exibem o pareto combinado de várias execuções do método em experimentos com uma população inicial aleatória e com uma população inicial proveniente das otimizações dos métodos mono-objetivos.
5. Elaborar um questionário que será respondido por pessoas leigas, pessoas com conhecimentos musicais moderados e conhecimentos musicais avançados para verificar se as melodias compostas são agradáveis de se escutar.
6. Verificar a qualidade musical das melodias compostas pelos métodos de composição algorítmica a partir da análise estatística dos dados obtidos no questionário, comparando-as com melodias aleatórias e melodias compostas por compositores.

1.5 Organização do trabalho

O capítulo 2 deste trabalho apresentará uma revisão bibliográfica explicitando um breve histórico da composição algorítmica e explicará superficialmente as estratégias que já foram utilizadas para resolver o problema. Por ser um tema relevante para este trabalho, os algoritmos evolucionários serão abordados com maior profundidade. Por fim será discutido o potencial da abordagem multiobjetivo devido à escassez de trabalhos que utilizam esta estratégia.

O capítulo 3 definirá a lei de Zipf e explicará como podemos definir fenômenos musicais e como podemos calcular a reta de Zipf a partir das ocorrências dos fenômenos musicais. Também serão detalhadas cada uma das regras de Fux utilizadas neste trabalho e como elas podem ser traduzidas em funções matemáticas para avaliação de melodias.

No capítulo 4 será exibida a estrutura de um algoritmo genético, explicando como seus operadores de mutação, seleção e cruzamento e sua função de avaliação podem ser utilizadas para tratar problemas de otimização. Explicaremos como algoritmos genéticos foram utilizados em conjunto com as métricas do capítulo 3 para resolver o problema de composição algorítmica. Posteriormente explicaremos detalhadamente o algoritmo NSGA-II e como ele foi utilizado neste trabalho como uma abordagem multiobjetivo para composição algorítmica.

O capítulo 5 descreverá todas as abstrações musicais e todas as decisões tomadas

para realizar os experimentos desta pesquisa. Apresentará análises gráficas e estatísticas dos 2 métodos mono-objetivos e dos 2 métodos multiobjetivos implementados neste trabalho. Apresentaremos também um estudo qualitavo realizado baseado em respostas de questionários para validação das melodias compostas pelos métodos mencionados anteriormente e para investigação do uso de um operador de mutação diferente em uma abordagem multiobjetivo.

Por fim, o capítulo 6 apresentará toda contribuição deste trabalho para a área, uma crítica aos métodos implementados e possíveis trabalhos futuros a serem realizados.

1.6 Publicações

Durante o período de desenvolvimento desta dissertação foi apresentado no *IEEE Congress on Evolutionary Computation 2017* o artigo *Combining Rules and Proportions: A Multiobjective Approach to Algorithmic Composition*. Este artigo será publicado futuramente nos anais do congresso.

2 Trabalhos Relacionados

Este capítulo tem o objetivo de mostrar como surgiu a composição algorítmica no meio acadêmico e como ela foi convergindo até se tornar o que é hoje. Apresentaremos também exemplos das principais estratégias utilizadas para composição, bem como as principais aplicações computacionais nesta área. Ao final mostraremos a escassez de trabalhos nos quais a otimização multiobjetivo é utilizada para abordar o problema e porque sua utilização é uma direção de pesquisa promissora.

2.1 Histórico

O primeiro trabalho relacionado a composição algorítimica, apesar de não ter sido publicado, foi implementado em 1955 por Caplin e Prinz (ARIZA, 2011). Esse trabalho consiste em duas abordagens, implementação do jogo de dados de Mozart e um gerador de linhas melódicas utilizando probabilidade de transição estocástica para vários aspectos da composição.

Em 1956, Hiller e Isaacson (1958) compuseram a *Iliac Suite*, gerada por um sistema de regras e cadeias de Markov. Esse trabalho inspirou colegas da mesma universidade a realizar experimentos em composição algorítmica, utilizando uma biblioteca que implementava os métodos utilizados por Hiller. Além do uso em pesquisas, um renomado compositor chamado Iannis Xenakis utilizou algoritmos estocásticos para gerar músicas como matéria-prima, que depois eram ajustadas em suas composições finais. Em 1964, Xenakis implementou um algoritmo *(PROJECT1)* utilizando uma teoria musical chamada composição serial e cadeias de Markov na geração de músicas (AMES, 1987).

Outros trabalhos relacionados a composição algortímica foram desenvolvidos, como o *framework* composicional encontrado em (PADBERG, 1964), que foi a primeira tese de doutorado no tema e, à medida que os computadores foram se tornando populares, novas aplicações de inteligência artificial em composição algorítimica foram surgindo.

2.2 Estratégias de Inteligência Artificial para Composição Algorítmica

O problema da composição algorítmica vem acompanhando a evolução de áreas relativas a desempenho computacional e inteligência artificial desde que essas foram criadas. Sempre que a capacidade de processamento de um computador evolui ou uma nova técnica de inteligência artificial é criada, abrem-se inúmeras lacunas a serem exploradas para

tentar obter melhores resultados na composição algorítmica. Como o problema é abordado praticamente desde a criação da inteligência artificial, não é surpresa que inúmeros métodos e técnicas já tenham sido utilizadas para tentar resolver a questão. Fernández e Vico (2013) escreveram um *survey* englobando trabalhos que utilizaram diversas técnicas de inteligência artificial na composição algorítmica. Nesse trabalho, Fernandez e Vico classificaram o uso das técnicas em: gramáticas, sistemas simbólicos e baseados em conhecimento, cadeias de Markov, redes neurais artificiais e algoritmos evolucionários e baseados em populações. Apesar de estratégias como autômatos celulares e autossimilaridade serem utilizadas na composição algorítmica, seu uso é bastante restrito e essas técnicas são mais bem aplicadas à sintetização de sons. Portanto, neste trabalho, seu uso não será discutido. A Figura 1 mostra o diagrama construído por Fernández e Vico (2013) para classificar as estratégias de inteligência artificial na composição algorítmica. Apesar da classificação proposta, é comum encontrar muitos trabalhos que utilizam mais de uma técnica específica.

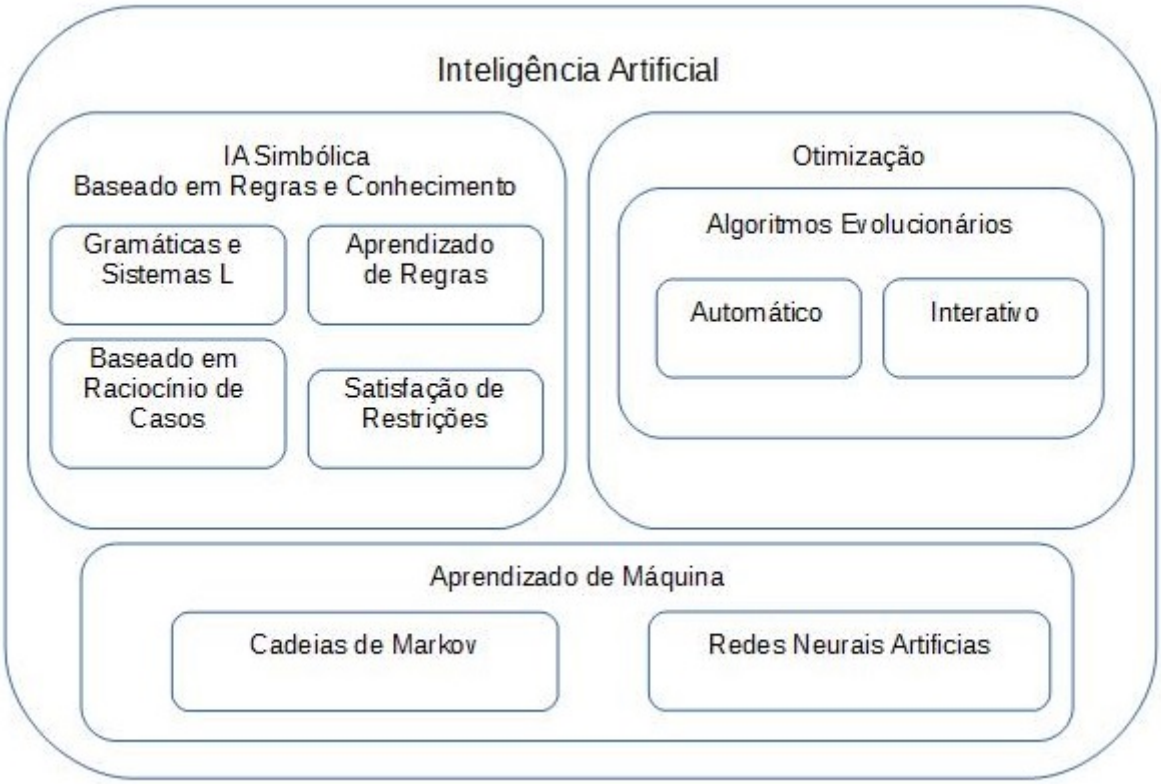


Figura 1 – Diagrama de estratégias de composição algorítmica. Extraído de (FERNÁNDEZ; VICO, 2013), traduzido para o português e adaptado para este trabalho.

2.2.1 Gramáticas

Gramáticas formais possuem diversas utilizações na ciência da computação. Linguagens de programação são definidas por gramáticas formais, onde aplicando regras de escrita/derivação, derivamos os símbolos para construir palavras (no caso das linguagens de programação, palavras são programas).

A utilização de gramáticas requer que regras de derivação sejam definidas para que as músicas sejam compostas. Essas regras podem ser geradas a mão ou extraídas de uma base de dados de composições. Os primeiros trabalhos utilizando gramáticas datam da década de 70. Os trabalhos de Lidov e Gabura (1973) e Rader (1974) apresentaram gramáticas definidas a mão para compor ritmos e conceitos musicais simples respectivamente. Já Ulrich (1977) e Baroni e Jacoboni (1978) utilizaram o paradigma de derivação por base de dados para gerar uma sequência de acordes de jazz e melodias de música clássica respectivamente. Trabalhos recentes utilizando gramáticas são vistos em (QUICK, 2010), tomando teorias musicais existentes e organizando-as hierarquicamente com uma análise Schenkeriana para formação da gramática e gerando música de contraponto em três vozes. Gillick, Tang e Keller (2009) fizeram uso de uma base de dados musicais para inferir uma gramática estocástica para compor improvisações de jazz.

Um outro tipo de gramática, sistemas de Lindenmayer (sistemas-L), também são utilizados na composição musical. A principal diferença dos sistemas-L para gramáticas convencionais é que todas as regras de derivação são aplicadas simultaneamente. Nos anos 90 os sistemas-L foram usados por compositores como Nelson (1996) e Mason e Saffle (1994) para compor a música *Summer Song* e compor música de contraponto respectivamente. Recentemente, encontramos trabalhos que geram acompanhamentos com sistemas-L (PESTANA, 2012) e arpejos (BULLEY; JONES, 2011).

Existem também usos híbridos de gramáticas e algoritmos evolucionários. O algoritmo de Bulley e Jones (1999) evoluiu gramáticas a partir de uma função objetivo interativa (avaliada por um usuário). Khalifa et al. (2007) utilizaram uma aborgadem em que a gramática faz parte de uma função objetivo para compor melodias. Houve também uso de algoritmos evolucionários para criação de sistemas-L (PECK, 2011), que foram utilizados posteriormente para derivar melodias.

Os exemplos citados consistem apenas em uma pequena parte das tentativas de utilização de gramáticas para composição algorítmica. Como o objetivo aqui é apenas mostrar um breve resumo sobre a técnica em questão, vários exemplos que a utilizam não foram citados. Mais referências podem ser encontradas em (FERNÁNDEZ; VICO, 2013).

2.2.2 Sistemas Simbólicos e Baseados em Regras

Dado que teorias musicais são eventualmente descritas como regras formais para manipular símbolos musicais (ANDERS; MIRANDA, 2011), sistemas baseados em conhecimentos também foram amplamente utilizados na tentativa de composição algorítmica. Um exemplo antigo de seu uso foi a tese de doutorado de Rothgeb (1968), em que uma série de regras extraídas de músicas do século XVIII foram utilizadas para harmonizar uma trilha de baixo. Ele descobriu que regras clássicas eram incompletas e incoerentes para uma certa dimensão. Esse problema foi encontrado em outros trabalhos como o algoritmo de harmonização implementado em Lisp por Thomas (1985).

Há também trabalhos baseados em aprendizado de regras. Schwanauer (1993) implementou um sistema baseado em regras chamado MUSE, que possuía um conjunto estático de regras e mais uma série de restrições para guiar o processo de composição. Ao passo que a composição era executada, o próprio sistema priorizava dinamicamente as restrições a serem seguidas. Morales e Morales (1993) implementaram um sistema que aprendia regras de contraponto de exemplos de música clássica.

Assim como com gramáticas, sistemas baseados em regras também possuem abordagens híbridas com algoritmos evolucionários. McIntyre (1994) produziu um algoritmo para harmonizar música barroca, em que regras eram codificadas em uma pontuação e a função objetivo era um somatório ponderado dessa pontuação. Gwee (2002) estudou a complexidade computacional de problemas relacionados à criação de música de contraponto e implementou um algoritmo evolucionário em que a função objetivo era um conjunto de regras *fuzzy*.

Com o passar do tempo, algoritmos de composição baseados em regras convergiram para utilização de programação lógica. Entretanto, a maioria dos trabalhos encontra-se dentro do paradigma de satisfação de restrições *(constraint satisfaction problems)*. As referências mais importantes para este trabalho dentro desse paradigma são (HERREMANS; SORENSEN, 2012), que modelaram matematicamente regras de contraponto de primeira espécie e utilizaram a meta-heurística *variable neighborhood search (VNS)* para compor melodias e, posteriormente Herremans e Sorensen (2013) estenderam o algoritmo de composições modelando as regras de contraponto de quinta espécie. Esse trabalho consiste em compor uma trilha denominada *cantus firmus* (uma melodia de contraponto de primeira espécie) e, a partir do cantus firmus gerado, compor uma segunda melodia chamada contraponto (de quinta espécie). A importância dessas citações dá-se pela modelagem detalhada das regras de contraponto de quinta espécia que serão utilizadas neste trabalho para composição de melodias. Outra abordagem que utilizou satisfação de restrições como paradigma foi o algoritmo de Davismoon e Eccles (2010), que utilizaram a heurística *simulated annealing* para otimizar regras combinadas com processos de Markov.

Existem trabalhos, como o de Pereira et al. (1997), cuja composição algorítmica foi baseada em raciocínio de casos *(case-based reasoning)*. Esse paradigma consiste em construir uma base de dados de como resolver um problema e, a menos que o problema de entrada seja idêntico a algum presente no banco de dados, o sistema deve selecionar um caso similiar e adaptar a solução do banco para o problema. Um algoritmo recente baseado em raciocínio de casos foi encontrado em (EIGENFELDT; PASQUIER, 2010) que gera uma cadeia de Markov para compor progressões de acordes de jazz.

Uma abordagem curiosa, porém dentro do paradigma baseado em regras foi encontrada em (DELGADO; FAJARDO; MOLINA-SOLANA, 2008), em que foi desenvolvido um sistema especialista, chamado *Inmamusys*, de duas camadas (uma de compositores e uma de geradores de voz) capaz de compor músicas de acordo com parâmetros inseridos pelo usuário. O sistema fornecia parâmetros sentimentais explícitos (como alegria, tristeza, preocupação, medo, etc.) para uma camada de agentes compositores no sistema especialista. Essa camada, por sua vez, escolhia que tipo de sequência de notas musicais seriam compostas para gerar o sentimento escolhido pelo usuário. O trabalho de Delgado, Fajardo e Molina-Solana (2008) foi o único encontrado que utilizou questionários para validação de suas composições e, mesmo que o método de utilização seja diferente do nosso, esse trabalho serviu de inspiração para que pudéssemos elaborar o nossa própria metodologia de validação baseada em respostas de questionários.

2.2.3 Cadeias de Markov

Cadeias de Markov têm sido usadas desde o surgimento da composição algorítmica até os dias de hoje para compor músicas. Elas consistem na idéia de que podemos transitar de um estado para vários outros com uma certa probabilidade. Cadeias de Markov podem assumir ordens, uma cadeia de Markov de primeira ordem nos diz que o próximo estado depende apenas do estado atual, uma cadeia de segunda ordem depende do estado atual e do anterior, uma cadeia de n-ésima ordem depende do estado atual e dos $n-1$ estados anteriores a ele.

Na composição algorítmica (de melodias por exemplo) notas podem representar os estados e uma sequência de notas podem representar um caminho entre esses estados. Cadeias de Markov foram muito populares nos primórdios da composição algorítmica, podendo ser treinadas a partir de músicas existentes (como na maioria dos trabalhos), ou definidas à mão ou por tentativa e erro. Um fato importante observado por Moorer (1972) é que cadeias de Markov de baixas ordens produzem músicas dissonantes e estranhas, enquanto cadeias de ordem maior tendem a repetir fragmentos das músicas utilizadas no treinamento da cadeia.

Trabalhos utilizando essa simples metodologia de treinamento e geração de caminhos em uma cadeia de Markov foram encontrados em (TIPEI, 1975), (JONES, 1981),

(NORTH, 1991) e (AMES; DOMINO, 1992). Trabalhos mais recentes utilizam cadeias de Markov associadas a alguns métodos diretos de pré e pós-processamento. Ponsford, Wiggins e Mellish (1999) treinaram uma cadeia de Markov com peças de Sarabande (músicas simples de dança) porém, diferente do estilo tradicional, existia um método de pré-processamento que definia a estrutura das músicas geradas e um de pós-processamento que garantia que as composições resultantes eram minimamente aceitáveis. Verbeurgt, Fayer e Dinolfo (2004) utilizaram cadeias de Markov para gerar padrões básicos de melodia, que eram então refinados por uma rede neural artificial. Lo e Lucas (2006) treinaram uma cadeia de Markov com músicas clássicas e posteriormente utilizaram a cadeia treinada em um algoritmo evolucionário para compor melodias. Pachet (2002) implementou um sistema chamado *Continuator* que mesclava cadeias de Markov de várias ordens. Martin, Jin e Bown (2012) implementaram um sistema de acompanhamento musical em tempo real utilizando também cadeias de Markov de ordem variável. Simon, Morris e Basu (2008) treinaram uma cadeia de Markov com arquivos MIDI para gerar acompanhamento em tempo real para uma voz cantada. Herremans, Sorensen e Conklin (2014) treinaram uma cadeia de Markov com músicas do tipo Bagana e a utilizaram como função objetivo em seu algoritmo VNS (o mesmo utilizado para compor músicas de contraponto) para compor músicas desse estilo. Também em geração de acompanhamentos, Manaris, Hughes e Vassilandonakis (2013) utilizaram um algoritmo genético, com função objetivo baseada em leis de Zipf, para gerar diversos acompanhamentos em tempo real, e então uma cadeia de Markov era utilizada para decidir qual acompanhamento escolher.

Modelos escondidos de Markov *(Hidden Markov Models)* também são utilizados para tarefas de composição algorítmica. A principal diferença entre modelos escondidos de Markov e cadeias de Markov é, que além da matriz de transição, modelos escondidos de Markov precisam treinar uma matriz de probabilidades de saída para cada estado. (PAIEMENT; ECK; BENGIO, 2006) e (RACZYNSKI; FUKAYAMA; VINCENT, 2012) são exemplo de trabalhos recentes que utilizam essa estratégia para compor melodias.

Conforme a revisão indica, cadeias de Markov foram utilizadas antigamente de forma solitária para tarefas de composição algorítmica. Apesar de sozinha não obter bons resultados musicais, sua utilização em conjunto com outras técnicas tem sido amplamente investigada, gerando resultados satisfatórios.

2.2.4 Redes Neurais Artificiais

Redes neurais artificiais (RNAs) são modelos matemáticos inspirados em redes neurais biológicas. Assim como em nosso cérebro possuímos neurônios e sinapses, existe também o conceito de neurônios e sinapses nas RNAs. A abstração feita é que, a partir de conexões com outros neurônios, se determinado limiar é atingido, aquele neurônio é ativado. Uma combinação de vários neurônios artificiais forma uma RNA e, a partir do

treinamento adequado da rede, obtemos a característica de inteligência artificial. RNAs são uma importante estratégia no problema de composição algorítmica, sendo que empresas como a Google, em seu projeto Magenta, vêm utilizando-a para esse propósito (SABÁN, 2016).

RNAs vêm sendo utilizadas desde meados dos anos 70 para análise e classificação de composições. O primeiro exemplo do uso de RNAs para composição algorítmica foi feito por Todd (1989), que consistia em uma rede recorrente de três camadas com uma saída que consistia em um sinal que representava uma nota a ser tocada na melodia. A sequência de notas na saída da rede recorrente formava uma melodia. O treinamento era feito com uma melodia específica. O resultado da série temporal era uma extrapolação dessa melodia.

Outra formas de utilização de RNAs foram vistas em (MOZER, 1991), em que um algoritmo de treinamento foi desenvolvido para capturar padrões locais e globais, e em (LEWIS, 1991), em que a rede foi treinada com sons aleatórios e músicas consideradas muito boas, adquirindo assim a capacidade de classificar os dois tipos. Para composição o método de gradiente descendente era utilizado para convergir uma sequência de sons aleatórios para uma sequência em que a rede neural treinada classificava como boa. Recentemente Coca, Romero e Zhao (2011) treinaram uma rede recorrente de Elman com músicas pré-existentes e melodias geradas em um sistema caótico pseudo-aleatório. Após treinada, a rede compunha melodias monofônicas.

Há também o uso de redes neurais para harmonizar uma melodia existente como o trabalho de Shibata (1991), em que o treinamento foi feito a partir de um *feedback* humano. E, assim como na maioria das estratégias, existem exemplos híbridos do uso de redes neurais como em (BROWNE; FOX, 2009), em que uma rede neural foi treinada com a tensão extraída do conjunto de treinamento e a rede treinada era utilizada como função de avaliação de uma implementação do *simulated annealing*.

A mistura entre redes neurais e algoritmos genéticos também é bastante utilizada na literatura. Johanson e Poli (1998) implementaram um sistema chamado GP-Music, que consistia em geração de melodias monofônicas através de um algoritmo genético em que a função de avaliação era o *feedback* de um ser humano. À medida em que os *feedbacks* humanos eram dados, uma rede neural era treinada dinamicamente para aprender o critério de avaliação. Posteriormente era possível utilizar a rede neural como função de avaliação do algoritmo. Manaris et al. (2007) extraíram características musicais derivadas da lei de Zipf de 992 peças de música clássica e treinaram uma rede neural com essas características. A partir daí a rede neural era utilizada como função de avaliação em um algoritmo de programação genética que compunha variações da décima terceira invenção de Bach em Lá menor.

2.2.5 Algoritmos Evolucionários

Algoritmos evolucionários são amplamente utilizados para realizar tarefas de composição algorítmica, seja compor uma melodia, harmonizar uma melodia existente ou criar uma melodia para uma harmonia existente. Um algoritmo evolucionário está sempre ligado à ideia de evolução de uma população de indivíduos. Vários indivíduos são gerados, os mais aptos são selecionados para que operadores de cruzamento e mutação sejam aplicados a eles, para que assim a população convirja para um ponto ótimo. No contexto da composição algorítmica, a aptidão de um indivíduo pode ser definida interativamente pelo usuário, ou pode ser modelada uma função matemática que indique a qualidade da composição. O uso de algoritmos evolucionários em composição algorítmica é de extrema importância para este trabalho, pois a abordagem multiobjetivo escolhida para tratar o problema utiliza esse paradigma. Os principais métodos evolucionários utilizados na composição algorítmica são algoritmos genéticos e programação genética.

O primeiro uso de algoritmos genéticos em composição algorítmica foi reportado em (HARTMANN, 1990) que, inspirado pelos biomorfos de Dawkin, implementou um algoritmo genético cuja função de avaliação era um *feedback* humano para compor melodias. Também com *feedback* humano como avaliação e usando programação genética, Tokui e Iba (2000) implementaram um método que compunha ritmos. Um trabalho mais recente, utilizando *feedback* interativo e regras musicais de forma híbrida, foi realizado por Fu et al. (2006). Como música é uma manifestação artística subjetiva e existem limitações e resistência em utilizar funções matemáticas para avaliação musical, implementar algoritmos interativos foi uma abordagem muito usada. Porém, a avaliação de um ser humano pode ser comprometida pelo cansaço do indivíduo que está avaliando. Existem também problemas relacionados a tempo, pois um algoritmo evolucionário precisa gerar milhares de músicas antes de entregar a solução final, mas um ser humano deve escutar uma composição inteira antes de poder avaliá-la, gerando assim um gargalo de tempo que compromete a convergência do algoritmo.

Dada a limitação de gargalo de tempo das abordagens evolutivas interativas, surge a necessidade de avaliar uma música a partir de funções matemáticas. Porém, mesmo que as teorias musicais estejam consolidadas há séculos, funções matemáticas modeladas para avaliar uma composição ainda não conseguem capturar tão bem as características musicais necessárias para compor uma música original. Horner e Goldberg (1991) implementaram um algoritmo genético em que a função de avaliação era baseada na construção temática de pontes (uma teoria musical). Biles (1991) implementou um sistema em que, a partir de uma progressão de acordes sendo tocados, improvisações de jazz eram compostas baseadas em mutações e cruzamentos que recombinavam partes dos indivíduos com uma base de dados de solos de jazz. Já a abordagem de Towsey et al. (2001) extraiu características de músicas diatônicas do oeste e modelou uma função objetivo que media

a distância dos indivíduos à média de cada característica extraída. Também foi utilizado um algoritmo genético. Um outro exemplo de extração de características foi encontrado em (ACEVEDO, 2004), porém a melodia era composta a partir de uma fuga e a função objetivo era um somatório ponderado das características extraídas dos indivíduos. Dahlstedt (2007) implementou um algoritmo de programação genética que, a partir de uma melodia tocada no piano, era gerada uma segunda voz de piano. A função de avaliação também eram somatórios de características extraídas de uma base de dados musical. Um último exemplo baseado em extração de características foi o trabalho de Wolkowicz, Heywood e Keselj (2008) em que a base de dados musical foi composta por prelúdios e fugas de Bach.

Um exemplo de composição homofônica foi encontrado em (JOHNSON; TAURITZ; WILKERSON, 2004), em que foram modeladas regras musicais para avaliar os indivíduos de um algoritmo genético. Esse trabalho compunha hinos de igreja. Uma função de avaliação alternativa foi implementada por Alfonseca, Cebrián e Ortega (2005), em que eram medidas várias distâncias de compressões normalizadas e então a função de avaliação era calculada pela diferença entre a média dessas distâncias de um indivíduo e a média de uma música alvo. Bellinger (2011) também modelou sua função de avaliação com uma música alvo, calculando probabilidades de ocorrência de uma nota.

Utilizar funções de avaliação baseadas em regras musicais em conjunto com algoritmos evolucionários é uma estratégia bastante comum. Em geral, utiliza-se o paradigma de satisfação de restrições para modelar essas regras. As regras utilizadas variam muito de acordo com o estilo que se deseja compor. Existem exemplos que utilizaram regras de Grachten (HARRIS; FITCH, 2008), regras de Kostka (DONNELLY; SHEPPARD, 2011), e outras regras cujos autores não são mencionados (ÖZCAN; ERÇAL, 2008; KHALIFA et al., 2007). Todos esses trabalhos foram destinados a compor melodias ou harmonizar uma melodia. Um conjunto de regras muito utilizado são as regras de Fux. Muitos trabalhos que a utilizam já foram citados em outras sessões. Utilizando algoritmo genético, as regras de Fux foram utilizadas em um trabalho recente por Kramann (2013). Uma característica interessante desse trabalho é a composição em tempo real. Utilizando regras de Piston e Zhang, Liu e Ting (2013) implementaram um algoritmo genético para compor melodias, porém a ponderação das regras foi feita a partir de estatísticas de downloads.

Um tipo de avaliação musical já citado e muito utilizado com algoritmos evolucionários é a avaliação baseada nas proporções da música. No trabalho de Jensen (2011) encontramos dois exemplos de composição algorítmica com função de avaliação baseada em proporções. O primeiro exemplo modelou 14 funções matemáticas baseadas em leis de Zipf e agregou-as em um somatório não ponderado na função de avaliação de um algoritmo de programação genética. O segundo método baseou-se em distribuição de similaridade, que também é relativo às proporções musicais. Em (MANARIS; HUGHES; VASSILANDONAKIS, 2011), foi implementado um sistema que usa um algoritmo genético

com uma função de avaliação que combina lei de Zipf com cadeias de Markov. A entrada do algoritmo era uma música tocada no piano e então uma segunda melodia era composta pelo método baseada na entrada. O mesmo autor implementou um outro sistema para compor harmonias utilizando também um algoritmo genético. A avaliação foi feita combinando a extração de função que segue a lei de Zipf de uma base de dados de mais de 14000 músicas clássicas (MANARIS; HUGHES; VASSILANDONAKIS, 2013).

Um trabalho mais recente que utiliza algoritmos genéticos e possui grandes diferenças das tentativas realizadas até então é o trabalho de Liu e Ting (2015) em que foi utilizado um algoritmo de mineração de padrões para identificar quais eram os padrões rítmicos mais utilizados pelo compositor asiático Jay Chou. O tipo de indivíduo desse método era uma sequência de padrões e não uma sequência de notas. Em se tratanto de algoritmos evolucionários, Liu e Ting (2016) escreveram um *survey* que apresenta os trabalhos mais recentes da área, agrupando-os em métodos de composição de melodias, acompanhamentos e formas musicais.

2.2.6 Abordagens Multiobjetivo

Apesar de eliminar o gargalo de avaliação humana, a utilização de algoritmos evolucionários para composição algorítmica requer a redução de uma melodia a um número para viabilizar a comparação entre indivíduos. Mesmo produzindo resultados musicalmente agradáveis, diferentes de uma composição aleatória, uma música é uma estrutura extremamente complexa e a quantidade de formas de avaliação existente é usualmente muito grande para ser reduzida apenas a um número. Uma alternativa para esse problema que não foi muito explorada é a abordagem multiobjetivo para tratar a composição algorítmica.

A maioria das abordagens multiobjetivo concentraram-se na tarefa de harmonizar uma melodia ao invés de compor uma melodia. O primeiro trabalho utilizando um algoritmo multiobjetivo foi encontrado em (GEIS; MIDDENDORF, 2008), em que ele combinou um algoritmo de colônia de formigas para gerar melodias e utilizou a abordagem multiobjetivo para harmonizar a melodia criada. As funções de avaliação são baseadas em regras musicais de Frank para melodia e regras de Fux para harmonia. Prisco, Zaccagnino e Zaccagnino (2010) utilizaram uma função harmônica e uma função melódica para comporem uma peça com quatro vozes a partir de uma linha de baixo existente. Carpentier e Bresson (2008) implementaram um algoritmo multiobjetivo que, dada uma melodia e uma série de restrições simbólicas definidas como entrada pelo usuário, um conjunto de soluções de orquestração era retornado. Um último exemplo de harmonização multiobjetivo foi encontrado em (FREITAS, 2011), em que as funções de avaliação eram baseadas em tríades com sétimas, e em acordes com notas dissonantes.

Em composição de melodias utilizando abordagem multiobjetivo, apenas o trabalho de Freitas (2013) foi encontrado. Nesse trabalho, foram definidas funções de avaliação

baseadas em características musicais como contorno, dissonância, intervalos entre outras para avaliar uma melodia. Contudo, devido a grande quantidade de funções, uma técnica denominada árvore de agregação foi utilizada para agrupar funções de avaliação menos conflitantes com finalidade de reduzir a dimensionalidade do problema. As funções foram reduzidas a apenas duas funções de avaliação, foi utilizado o algoritmo NSGA-II para compor melodias.

2.3 Considerações Finais

Este capítulo apresentou uma revisão bibliográfica que sumariza diversas abordagens já utilizadas para resolver o problema de composição algorítmica. A Tabela 2 sumariza os principais trabalhos apresentados neste capítulo.

Dado que existem várias formas automáticas de avaliar uma música, isto pode dar indícios de que uma abordagem multiobjetivo pode ser mais apropriada para resolver o problema. Dada essa observação e a escassez de trabalhos com essa estratégia, a abordagem multiobjetivo tornou-se um interessante paradigma para futuras pesquisas na área e será uma das principais contribuições deste trabalho em que combinamos funções de avaliações provenientes das regras de Fux e da lei de Zipf em uma abordagem multiobjetivo.

Tabela 2 – Tabela Resumo

Título	Referência	Compõe a partir:	Compõe	Tipo	Tipo de Avaliação Musical	Técnica utilizada
Evolutionary music composer integrating formal grammar	(KHALIFA et al., 2007)	Geração do zero	Harmonia e Melodia	Homofônico	Gramática e regras musicais	Algoritmo Genético
Algorithmic Composition of Jazz	(HARRIS; FITCH, 2008)	Progressão de acordes	Solo e Acompanhamento	Homofônico	Regras Grachten eNarmour	Algoritmo Genético
A Genetic Algorithm for Generating Improvised Music	(ÖZCAN; ERÇAL, 2008)	Harmonia	Melodia	Monofônico	Regras musicais do autor	Algoritmo Genético
Evolving Four-Part Harmony Using Genetic Algorithms	(DONNELLY; SHEPPARD, 2011)	Geração do zero	Melodia e Harmonia	Polifônico	Regras de Kostka	Algoritmo Genético
Darwinian Pianos: Realtime Composition Based on Competitive Evolutionary Process	(KRAMANN, 2013)	Uma frase repetida continuamente	Melodia e Harmonia	Polifônico	Regras de contraponto (Fux)	Algoritmo Genético
Evolutionary Composition Using Music Theory and Charts	(LIU; TING, 2013)	Geração do zero	Melodia e Harmonia	Polifônico	Regras Musicais De Zhang	Algoritmo Genético
Monterey Mirror: Combining Markov Models, Genetic Algorithms, and Power Laws	(MANARIS; HUGHES; VASSILANDONAKIS, 2011)	A partir de uma peça tocada por um músico	Melodia	Monofônico	Lei de Zipf	Algoritmo Genético eCadeia de Markov
Evolutionary Music Composition A Quantitative Approach	(JENSEN, 2011)	Geração do zero	Melodias	Monofônico	Lei de Zipf	Programação Genética
EvoBassComposer: a Multiobjective Genetic Algorithm for 4-voice Compositions	(PRISCO; ZACCAGNINO; ZACCAGNINO, 2010)	Linha de baixo	Harmonia	Polifônico	Função harmônica e função melódica	Algoritmo Multiobjetivo
Melody Harmonization in Evolutionary Music Using Multiobjective Genetic Algorithms	(FREITAS, 2011)	A partir de melodia	Harmonia	Polifônico	Tríades com sétima e acordes dissonantes	Algoritmo Multiobjetivo
Redução de Dimensionalidade em Problemas com Muitos Objetivos: Uma Aplicação em Composição Algorítmica	(FREITAS, 2013)	Geração do zero	Melodia	Monofônico	Funções de contorno, dissonância e intervalos	Algoritmo Multiobjetivo

3 Métricas de Avaliação Musical

Avaliação da qualidade de uma música é uma tarefa bastante subjetiva, ou seja, depende de fatores pessoais e pode variar dependendo de quem avalia a música. Devido a essa subjetividade, a avaliação musical por um computador torna-se difícil implementar, dado que um computador precisa trabalhar com números e não existe uma forma ideal de traduzir subjetividade em um número específico. Algumas abordagens foram modeladas com o objetivo de traduzir a qualidade de uma música em critérios quantitativos. Um exemplo de métrica musical é a lei de Zipf (ZIPF, 1949), que é baseada nas proporções de características de uma música.

Outro tipo de tentativa de avaliação musical é basear-se nas teorias musicais de composição existentes, como regras de Grachten e de Kostka, que já foram citadas no capítulo 2. As regras de Fux (FUX; MANN, 1971) possuem uma atenção especial na composição algorítmica por definirem numericamente eventos que não são desejáveis em uma composição, como por exemplo número de saltos por compasso. Com regras musicais em mãos, podemos quantificar eventos que não são agradáveis musicalmente e penalizar matematicamente a ocorrência desses eventos.

Neste capítulo definiremos a lei de Zipf e as regras de Fux e explicaremos detalhadamente como elas podem definir a qualidade de uma composição musical.

3.1 Lei de Zipf

George Kingsley Zipf desenvolveu uma técnica estatística para capturar as propriedades escaláveis de fenômenos humanos e naturais. Estudos apontam que a Lei de Zipf pode ser usada como critério para avaliação musical, como no trabalho de Bill Manaris e sua equipe (MANARIS et al., 2005). Esta seção explica detalhadamente como é possível utilizar a lei de Zipf para avaliação de uma música.

3.1.1 Aplicação da Lei de Zipf na Música

A lei de Zipf reflete as propriedades escaláveis de diversos fenômenos na natureza, incluindo música. A lei de Zipf assume que a frequência de ocorrência de um determinado fenômeno é inversamente proporcional à sua classificação (*rank*) da quantidade de ocorrências desses eventos, conforme a equação 1:

$$F \sim r^{-a}. \tag{1}$$

Na equação (1), F representa a frequência de um evento, r representa a classificação do evento e a é igual a 1. Quando a assume o valor 1, a equação (1) descreve o fenômeno *pink noise* (ruído rosa). O ruído rosa é o fenômeno que a lei de Zipf se baseia para explicar as proporções musicais. Outros valores de a implicam na descrição de outros fenômenos, por exemplo, *white noise*, *brown noise*, etc. Podemos inferir da equação (1) que o evento mais frequente possui taxa de ocorrência proporcional a 1, o segundo mais frequente proporcional a $\frac{1}{2}$, o terceiro mais frequente proporcional a $\frac{1}{3}$ e assim sucessivamente (basta que r assuma os valores 1, 2 e 3, respectivamente). Dessa forma, se o evento mais frequente em um fenômeno acontece 1000 vezes, espera-se que o segundo mais frequente aconteça aproximadamente 500 vezes e o terceiro aconteça 333 vezes.

Como a música é um fenômeno humano, se forem definidos fenômenos cuja ocorrência de seus eventos possa ser contada em uma música, podemos aplicar a lei de Zipf de forma a obter uma avaliação de determinada música. Jensen (2011) descreveu 14 fenômenos musicais, exclusivos de melodias, cujas ocorrências podem ser contadas para que sua distribuição seja verificada pela lei de Zipf. Os fenômenos são:

- **Nota:** Valor inteiro MIDI de uma nota musical $(n \in [0{,}127])$;
- **Tom Cromático:** Resto da divisão do inteiro MIDI por 12 $(tc \in [0{,}11])$;
- **Duração:** Duração da nota em batimentos $(d \in \mathbb{R}^+)$;
- **Nota e Duração:** Par ordenado de nota e duração $(n{,}d)$;
- **Tom Cromático e Duração:** Par ordenado de tom cromático e duração $(tc{,}d)$;
- **Distância de Nota:** Intervalo de tempo entre duas notas repetidas $(tn \in \mathbb{R}^+)$;
- **Distância de Tom Cromático** Intervalo de tempo entre dois tons cromáticos repetidos $(ttc \in \mathbb{R}^+)$;
- **Intervalo Melódico:** Intervalo musical nas melodias $(im_i = n_i - n_{i-1})$;
- **Bigrama Melódico:** Par de intervalos melódicos adjacentes (im_i, im_{i+1});
- **Trigrama Melódico:** Trinca de intervalos melódicos adjacentes $(im_i, im_{i+1}, im_{i+2})$;
- **Ritmo:** Duração da nota mais pausas subsequentes $(r \in \mathbb{R}^+)$;
- **Intervalo Rítmico** Razão entre um ritmo e seu antecessor $(ir = \frac{r_i}{r_{i-1}})$;
- **Bigrama Rítmico:** Par de intervalos melódicos adjacentes (ir_i, ir_{i+1});
- **Trigrama Rítmico:** Trinca de intervalos melódicos adjacentes $(ir_i, ir_{i+1}, ir_{i+2})$.

Existem outros fenômenos musicais que podem ser definidos caso a necessidade seja calcular a distribuição de ocorrências em harmonias, porém não serão abordados neste trabalho. Além disso, a forma de cálculo da lei de Zipf é a mesma independente do evento.

3.1.2 Cálculo da Lei de Zipf

Como em um fenômeno, a ocorrência de um evento é inversamente proporcional à sua classificação, com $a = 1$, se gerarmos um gráfico do logaritmo das ocorrências de um evento pelo logaritmo da classificação de um evento, esperamos que os pontos desse

gráfico se aproximem de uma reta com coeficiente angular igual a -1. Portanto, quanto mais próximo esse gráfico estiver dessa reta, melhor será a avaliação da música para esse evento.

Para verificar se um fenômeno está de acordo com a lei de Zipf, devemos primeiro escolher um fenômeno específico, por exemplo intervalos melódicos. Por definição, um intervalo melódico é a diferença entre o valor MIDI inteiro de uma nota e valor MIDI da nota que a antecede. Após analisar todas as notas de uma música hipotética, suponha que foram encontrados os seguinte intervalos melódicos: $(1, 2, 3, 1, -5, 0, 0, 1, 3, 3, 0, 0, 0, -2)$.

Analisando esses intervalos, podemos ver que o invervalo 0 ocorreu 5 vezes, o intervalo 1 ocorreu 3 vezes, o intervalo 2 ocorreu 1 vez, o intervalo 3 ocorreu 3 vezes, o invervalo -2 ocorreu 1 vez e o intervalo -5 ocorreu 1 uma vez, formando a Tabela 3.

Tabela 3 – Ocorrências de Intervalos Melódicos

Intervalo Melódico	-5	-1	0	1	2	3
Ocorrências	1	1	5	3	1	3

Como a lei de Zipf prevê que a ocorrência de um evento deve ser inversamente proporcional à sua classificação quanto ao número de ocorrências, devemos ordenar as ocorrências para prosseguir com o cálculo. Dessa forma, obtemos os valores da Tabela 4.

Tabela 4 – Ocorrências de Intervalos Melódicos Ordenadas

Intervalo Melódico	0	1	3	2	-5	-1
Ocorrências	5	3	3	1	1	1
Classificação	1	2	3	4	5	6

Após ordenar as ocorrências dos intervalos melódicos e definir as classificações de cada evento, é necessários aplicar a função logaritmo natural tanto nas ocorrências, quanto nas classificações dos eventos. Após a aplicação dessa função, obtemos a Tabela 5. Feito isto, basta realizar um ajuste linear para obter a reta que melhor se adequa aos logaritmos das ocorrências da música analisada.

Tabela 5 – Logaritmo de Ocorrências e Classificações

log(Ocorrências)	1.6094	1.0986	1.0986	0	0	0
log(Classificação)	0	0.6931	1.0986	1.3863	1.6094	1.7918

A Figura 2 mostra o gráfico dos logaritmos das classificações e das ocorrências junto com seu ajuste linear, ou seja, a melhor reta que passa pelos pontos em questão. Para nossa reta, obtemos um coeficiente angular igual a $-1{,}005$ que é próximo de -1, ou seja, nossa música hipotética possui uma boa distribuição de ocorrências de intervalos melódicos.

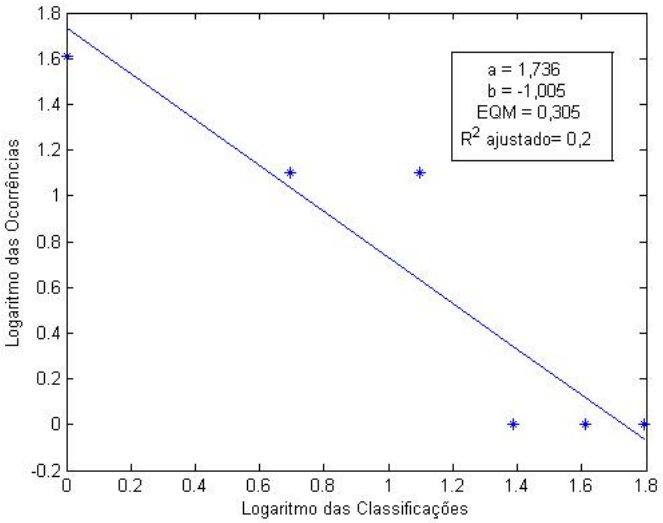


Figura 2 – Gráfico do Logaritmo das Ocorrências pelo Logaritmo das Classificações

Essa forma de cálculo da lei de Zipf é definida em (MANARIS et al., 2005). Para qualquer fenômeno que definirmos, a forma de cálculo da lei de Zipf é a mesma. Contar as ocorrências dos diferentes eventos daquele fenômeno e calcular o coeficiente angular do logaritmo das ocorrências pelo logaritmo da classificação. O resultado dessa análise é sempre um coeficiente angular e uma reta ajustada e cabe ao pesquisador decidir como utilizar esses dados na avaliação musical.

3.2 Fux

Em 1725, Johann Fux elaborou uma série de regras de composição denominadas regras de contraponto através de espécies. O objetivo inicial dessas regras era servir como ferramenta pedagógica para estudantes aprenderem como compor músicas polifônicas com duas vozes. As regras de contraponto eram organizadas através de uma hierarquia de níveis onde o primeiro nível era mais restritivo que o segundo, ou seja, tinha uma menor liberdade de escolha de duração e tons das notas compostas. O terceiro nível possuia maior liberdade de escolha que o segundo nível e assim sucessivamente até o quinto nível. A cada nível acima, maior complexidade musical era empregada nas composições. As regras de contraponto eram ensinadas a alunos em sequência determinada pelos níveis e, apesar de existirem a mais de dois séculos, são fundamentais até hoje nos ensinamentos musicais (NORDEN, 1969).

Uma característica intrínseca às regras de Fux é que elas definem critérios horizontais e verticais para composição de uma música. Critérios horizontais definem o relacionamento entre as notas tocadas em sequência e os critérios verticais estão relacionados a notas tocadas em vozes diferentes. Em geral, a melodia de contraponto composta a partir

das regras de Fux é sempre composta em relação a uma primeira melodia denominada *cantus firmus* (CF), que é uma sequência de notas horizontalmente harmônicas, todas elas com duração de uma semibreve (figura musical que representa 4 tempos). A composição do CF também é definida nas regras de Fux.

As melodias de contraponto compostas pelo contraponto de primeira espécie consistem em melodias cuja duração das notas são apenas 4 tempos (assim como no *cantus firmus*). Um exemplo de regra de contraponto de primeira espécie é: "No máximo dois saltos consecutivos. Um salto é um intervalo com mais de dois semitons". No contraponto de segunda espécie é definido um conjunto de regras diferente do contraponto de primeira espécie, e durações representadas por mínimas (figura musical que representa 2 tempos) são permitidas. No contraponto de terceira espécie figuras de apenas um tempo musical são permitidas, no de quarta espécie notas ligadas são permitidas e por fim, no contraponto de quinta espécie são permitidas notas ligadas, semibreves, mínimas e semínimas (figura que representa um tempo musical).

Dada as diversas formalizações existentes das regras de Fux, este trabalho baseia-se nas regras encontradas em (SALZER; SCHACHTER, 1969), pois possui uma definição quantitativa, o que torna fácil a transformação das regras em funções de avaliação. Como este trabalho visa compor melodias monofônicas, apenas os critérios horizontais do contraponto de quinta espécie serão apresentados.

3.2.1 Regras de Contraponto de Quinta Espécie

O contraponto de quinta espécie define dezenove critérios horizontais para orientar um músico no processo de composição (HERREMANS, 2014). Devido a abstração musical feita neste trabalho, que será detalhada no capítulo 4, apenas dezessete desses dezenove critérios serão abordados neste trabalho. Algumas das regras citadas são definidas em função do tamanho unitário de música denominado L. Assim como em (HERREMANS, 2014), este trabalho define L como um conjunto de 4 compassos. Existem também regras que são definidas a partir do tom da música, por exemplo a regra 1. Portanto, para utilizar as regras de Fux é necessário definir esse tom, que é denominado chave. As dezessete regras de contraponto utilizadas neste trabalho são:

1. Oitavas devem acontecer em um passo;
2. Deve haver apenas um clímax, que só pode ser repetido após um tom de vizinhança;
3. O clímax deve cair em um tempo forte, exceto se for repetido após um tom de vizinhança;
4. Apenas intervalos consonantes são permitidos;
5. Conjuntura: movimentos graduais devem ser predominantes. Dois a quatro saltos por L são ótimos;
6. Cada salto longo deve ser seguido por um movimento gradual;

7. Mudar de direção após cada salto longo;
8. O clímax deve ser melodicamente consonante com a tônica;
9. Não mais do que dois saltos consecutivos;
10. Não mais do que dois saltos longos por L;
11. Não haver muitos movimentos graduais na mesma direção;
12. A direção deve mudar no mínimo três vezes por L;
13. A nota final deve ser tônica;
14. A penúltima nota deve ser o tom conducente e preferencialmente na mesma oitava que a nota final se encontra;
15. O início e o fim de um movimento deve ser consonante;
16. Deve haver um balanceamento entre movimentos ascendentes e descendentes. Movimentos maiores que nove semitons devem ser evitados;
17. O maior intervalo permitido entre duas notas consecutivas é a oitava perfeita.

A fim de transformar as regras de contraponto em funções matemáticas devemos definir o conceito de alguns termos citados nas regras que, devido ao teor musical, costumam não ser conhecidos por leigos (por exemplo: clímax, movimento gradual e intervalos) e que o entendimento deles é necessário para compreender a metodologia. Esses termos estão definidos em (HERREMANS, 2014) e estão descritos abaixo:

- **Intervalo:** Distância em quantidade de semitons entre duas notas;
- **Oitava:** Nota que dista doze semitons da nota chave;
- **Passo ou Movimento Gradual:** Intervalo de um ou dois semitons;
- **Clímax:** Nota cujo tom é o mais alto na música;
- **Tom de vizinhança:** Nota posicionada entre duas notas iguais e que é aproximada e deixada por um movimento gradual;
- **Tempo forte:** Primeiro tempo (ou primeira nota) de um compasso;
- **Intervalos Consonantes:** Intervalos de 0, 1, 2, 3, 4, 5, 7, 8, 9, 12 semitons (ou mais oitavas - múltiplos de 12 semitons);
- **Salto:** Intervalo de mais de dois semitons;
- **Salto Longo:** Intervalo de mais de 4 semitons;
- **Tônica:** Nota definida como chave da melodia;
- **Tom conducente:** Tom que representa a sétima nota da escala (seja maior ou menor);
- **Movimento (Ascendente ou Descendente):** Sequência de notas cujos intervalos sejam sempre um passo e sempre na mesma direção;
- **Intervalo de Movimento:** Intervalo entre primeira e última nota do movimento.

3.2.2 Funções de Regras de Fux

De posse das regras de contraponto de quinta espécie, é necessário elaborar uma forma de, a partir delas, avaliar matematicamente uma música. A forma proposta por

Herremans (2014) consiste em atribuir, para cada uma das dezessete regras, um *subscore* numérico cujo valor varia entre 0 e 1, de forma que cada vez que uma melodia não satisfaça uma regra, é adicionada uma penalização no subscore correspondente.

As fórmulas de cálculo dos subscores elaboradas por Herremans (2014) são apresentadas nas equações de (2) a (18). Em todas as funções, o símbolo $\sum$ (somatório) é utilizado para indicar a contagem da quantidade de ocorrências de um evento, seja ele violação da regra ou não.

O numerador da equação (2) consiste em verificar se as oitavas contidas na melodia não são precedidas por passo $(nppp)$ e se não são deixadas por passo $(ndpp)$. Cada vez que uma violação acontecer é acrescentado 1 no numerador. O denominador é igual o total de oitavas na melodia. A equação (3) conta a quantidade de clímax na melodia, subtrai uma unidade e divide pela quantidade de notas. A equação (4) conta a quantidade de clímax em tempos não fortes e divide pela quantidade de clímax. Esse procedimento de contar a quantidade de violações de uma regra é encontrado e calculado de forma análoga nas equações (5), (7), (8), (10), (16), (17) e (18).

As equações (6), (11), (12), (13) definem intervalos em que as violações de saltos, saltos longos, movimentos e mudança de direção são aceitáveis. Dentro desse intervalo, as funções assumem o valor 0, fora dele o procedimento padrão de contabilizar o número de violações é aplicado. As equações (9) e (14) são funções binárias que assumem o valor 0 caso o clímax e a nota final sejam tônicas respectivamente. E por fim, a equação (15) assume o valor 0 caso a penúltima nota seja um tom conducente e esteja na mesma oitava que a nota final, 0.5 caso seja apenas um tom conducente e assume o valor 1 caso não seja um tom conducente.

$$subscore_1(s) = \frac{\sum 8ths\ nppp + \sum 8ths\ ndpp}{2\ \mathrm{X}\ \sum 8ths}; \tag{2}$$

$$subscore_2(s) = \frac{\sum \text{Clímax} - 1}{\text{tamanho}}; \tag{3}$$

$$subscore_3(s) = \frac{\sum \text{Clímax em tempos não fortes (exceto em tom de vizinhança)}}{\text{Clímax (exceto em tom de vizinhança)}}; \tag{4}$$

$$subscore_3(s) = 1 - \frac{\sum \text{Intervalos Consonantes}}{\text{Intervalos}}; \tag{5}$$

$$subscore_5(s) = \begin{cases} 0 & \text{se } 2L \leq \sum Saltos \leq 4L \\ \frac{\sum \text{Saltos}-(4L)}{\text{Tamanho}-1-(4L)} & \text{se } 4L < \sum Saltos \\ \frac{\sum \text{Saltos}}{\text{Tamanho}--(4L)} & \text{se } 2L > \sum Saltos \end{cases}; \tag{6}$$

$$subscore_6(s) = \frac{\sum \text{Saltos Longos nspp}}{\text{Saltos Longos}}; \tag{7}$$

$$subscore_7(s) = \frac{\sum \text{Saltos Longos nsp}\Delta\text{d}}{\text{Saltos Longos}}; \tag{8}$$

$$subscore_8(s) = \begin{cases} 0 & \text{se Clímax é consonante com tônica} \\ 1 & \text{se Clímax não é consonante com tônica} \end{cases}; \tag{9}$$

$$subscore_9(s) = 1 - \frac{\sum 3 \text{ Saltos Consecutivos}}{\text{Tamanho} - 3}; \tag{10}$$

$$subscore_{10}(s) = \begin{cases} 0 & \sum \text{Saltos Longos} \leq 2L \\ \frac{\sum \text{Saltos Longos} - (2L)}{\text{Tamanho} - 1 - 2L} & \sum \text{Saltos Longos} > 2L \end{cases}; \tag{11}$$

$$subscore_{11}(s) = \begin{cases} 0 & \text{se maior sequência} \leq 5 \\ \frac{\text{Tamanho da maior Sequência de Passos}}{\text{Tamanho}} & \text{se maior sequência} > 5 \end{cases}; \tag{12}$$

$$subscore_{12}(s) = \begin{cases} 0 & \text{se} \sum \text{Mudança de Direção} \geq 3L \\ 1 - \frac{\sum \text{Mudança de Direção}}{3L} & \text{se} \sum \text{Mudança de Direção} < 3L \end{cases}; \tag{13}$$

$$subscore_{13}(s) = \begin{cases} 0 & \text{se nota final é tônica} \\ 1 & \text{se nota final não é tônica} \end{cases}; \tag{14}$$

$$subscore_{14}(s) = \begin{cases} 0 & \text{se penúltima nota é tom conducente} \\ 0.5 & \text{se penúltima nota é tom conducente em oitava diferente da nota final} \\ 1 & \text{se penúltima nota não é tom conducente} \end{cases}; \tag{15}$$

$$subscore_{15}(s) = \frac{\sum \text{Intervalo de Movimento Dissonante}}{\sum \text{Intervalo de Movimento}}; \tag{16}$$

$$subscore_{16}(s) = \frac{\sum \text{Intervalo de Movimento} > 9}{\sum \text{Intervalo de Movimento}}; \tag{17}$$

$$subscore_{17}(s) = \frac{\sum \text{Intervalo} > 12 \text{ semitons}}{\sum \text{Intervalo}}. \tag{18}$$

Após cálculo de todos os subscores é realizado um somatório ponderado de acordo com a equação (19). O valor dessa equação indicará a qualidade de uma melodia em

relação às regras de Fux, sendo que quanto mais próximo de zero for esse valor, maior será a qualidade da melodia.

$$f(s) = \sum_{i=1}^{17} a_i.subscore_i(s) \tag{19}$$

3.3 Considerações Finais

Neste capítulo foram apresentadas as métricas de avaliação musical que serão utilizadas neste trabalho como critério de avaliação de melodias. Considerando a escolha de uma métrica baseada em proporções musicais (lei de Zipf) e uma baseada em teoria musical (regras de Fux), obtemos duas visões diferentes de se enxergar uma melodia, o que torna bastante conveniente a implementação de um algoritmo multiobjetivo. No capítulo 4, serão detalhados os conceitos de um algoritmo genético e do algoritmo muiltiobjetivo NSGA-II e como eles foram utilizados neste trabalho para realizar a composição algorítmica.

4 Algoritmo Genético Aplicado a Composição Algorítmica

Conforme visto no capítulo 2, algoritmos genéticos são amplamente utilizados em composição algorítmica. Definida uma métrica de avaliação musical, podemos utilizar uma heurística computacional para compor melodias de acordo com a métrica definida. Neste capítulo, destalharemos os conceitos de um algoritmo genético e do algoritmo NSGA-II e mostraremos como eles podem ser usados em conjunto com as métricas definidas no capítulo 3 para compor melodias monofônicas.

4.1 Algoritmos Genéticos

Inspirados na teoria da evolução de Darwin, algoritmos genéticos fazem parte de um conjunto de técnicas heurísticas de otimização, cuja finalidade é encontrar soluções de um problema matemático que sejam próximas da solução ótima. Apesar da idéia dos algoritmos genéticos existirem desde os anos 40, sua implementação computacional surgiu apenas no final da década de 60 em (HOLLAND, 1975).

Assim como na teoria de Darwin, algoritmos genéticos imitam o processo de evolução de uma população baseados na seleção natural de indivíduos mais aptos, em que esses, assim como na natureza, se reproduzem mais vezes, gerando mais filhos que herdam as características fenotípicas dos pais. A Figura 3 mostra um fluxograma de um algoritmo genético, em que definimos um indivíduo que representa uma solução do problema, geramos uma população aleatória desses indivíduos, avaliamos a população, aplicamos os operadores de seleção, cruzamento e mutação e definimos a nova população. Esse processo é repetido até que um critério de parada seja atingido, em geral o número de gerações escolhido. Os algoritmos genéticos constituem uma classe de algoritmos que são caracterizados pela existência da execução desses procedimentos, porém existem diversas maneiras diferentes de executá-los.

4.1.1 Definição do Indivíduo

O primeiro passo para implementação de um algoritmo genético é a definição do indivíduo. A modelagem desse indivíduo varia de acordo com o problema que queremos otimizar. Se estivermos utilizando um algoritmo genético para maximizar uma função matemática de duas variáveis, o indivíduo pode ser modelado por um par ordenado de números reais (x_1,x_2), porém pode haver outras formas de modelar esse mesmo indivíduo.

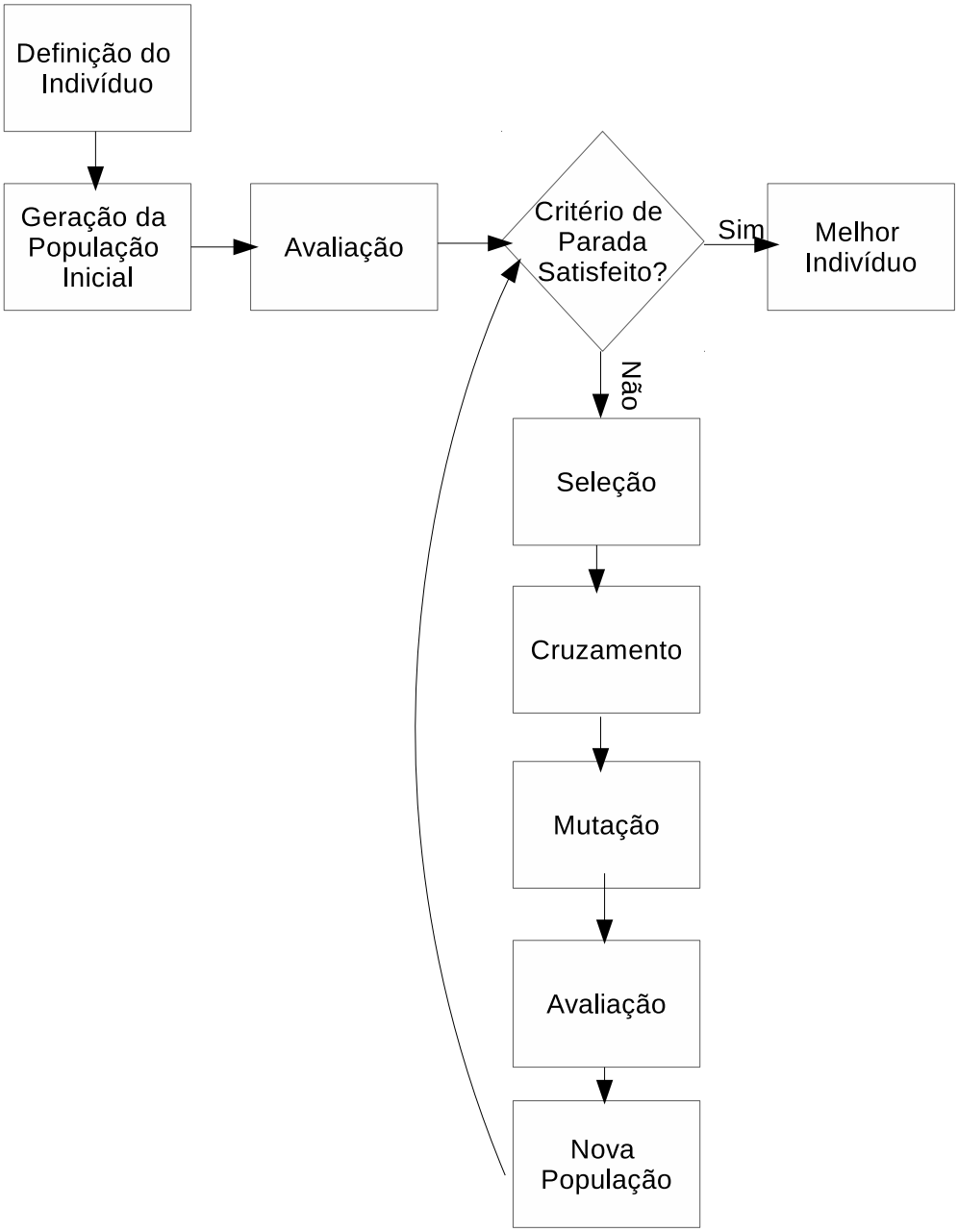


Figura 3 – Fluxograma de um algoritmo genético.

4.1.2 Função de Avaliação

A função de avaliação ou aptidão define a qualidade de um indivíduo na população. Ela é parte intrínseca de qualquer algoritmo genético e a partir dela podemos avaliar se

um indivíduo é bom ou ruim. Assim como na definição do indivíduo, definir uma função de avaliação depende fortemente do problema a ser otimizado por um algoritmo genético.

Caso estejamos tratando do exemplo de maximização de uma função de duas variáveis, a função de avaliação do problema pode ser definida como a própria função matemática. Por exemplo: $f(x_1,x_2) = -x_1x_2$, sendo que x_1 e $x_2 \in [-1,1]$. Sabemos que os indivíduos que maximizam essa função são os pares ordenados $(1, -1)$ e $(-1,1)$. Portanto, através dos operadores de seleção, cruzamento e mutação, um algoritmo genético para resolver a maximização dessa função, convergiria para um indíviduo próximo desses indivíduos.

4.1.3 Operadores

Assim como a seleção natural utiliza-se da reprodução (cruzamento) e mutação para evoluir indivíduos de uma população, os algoritmos genéticos utilizam-se de operadores para explorar o espaço de busca do problema e evoluir uma população. Os operadores básicos dos algoritmos genéticos são operadores de seleção, cruzamento e mutação.

4.1.3.1 Seleção

A ideia do operador de seleção é, a partir da função de avaliação e de uma política de seleção, privilegiar os indivíduos mais aptos de uma população. Essa política de seleção pode ser implementada de diversas formas, por exemplo a roleta. Na roleta somamos os valores de aptidão de todos os indivíduos da população e calculamos a proporção que cada indivíduo contribui para esse somatório. Essa proporção é a chance do indivíduo ser selecionado.

Outro operador de seleção muito utilizado é o torneio. Nesse método, k indivíduos são selecionados aleatoriamente de uma população e aquele que for o mais apto (tiver melhor valor na função de avaliação) entre os k indivíduos é selecionado. Um torneio com $k = 2$ é um torneio binário. Operadores de seleção são utilizados para selecionar indivíduos para realizar o cruzamento e podem também ser utilizados para selecionar indivíduos que comporão a nova população.

4.1.3.2 Cruzamento

Na seleção natural, o processo de cruzamento combina dois indivíduos de uma população para gerar um ou mais indivíduos, sendo que esses possuem características dos pais. Nos algoritmos genéticos não é diferente. Utilizamos operadores de seleção para selecionar indivíduos mais aptos e, de acordo com uma probabilidade definida, efetuamos o cruzamento entre esses indivíduos.

A definição de um operador de cruzamento depende fortemente do problema a ser otimizado. Assim, como a definição do indivíduo e da função de avaliação, existem diversas formas possíveis de modelar um operador de cruzamento. No caso do exemplo da otimização de uma função de duas variáveis, um possível operador de cruzamento seria calcular a média das variáveis independentes de dois indivíduos e gerar um novo indivíduo cujo par ordenado é composto por essas médias.

4.1.3.3 Mutação

É comum em algoritmos genéticos estagnarmos em ótimos locais e obtermos uma solução que não necessariamente é a melhor e pode nem estar próxima da ótima. Operadores de mutação tem o objetivo de evitar esse problema, propiciando a diversidade da população.

Assim como na seleção natural, mutações não ocorrem frequentemente, portanto operadores de mutação tendem a ter uma taxa baixa. No caso de otimização de funções matemáticas poderíamos definir um operador de mutação como a inversão do sinal de uma das variáveis independentes. Ou seja, mutando o indivíduo (x_1,x_2) podemos obter os indivíduos $(-x_1,x_2)$ ou $(x_1, -x_2)$. Operadores de mutação também dependem fortemente do problema a ser resolvido pelo algoritmo genético.

4.1.4 Definindo a Nova População

Como já foi dito, algoritmos genéticos baseiam-se na evolução de uma determinada população através das gerações. Após aplicarmos operadores de cruzamento e mutação, é necessário definir quais indivíduos comporão a próxima geração. Em um algoritmo genético a quantidade de indivíduos de uma população é sempre constante, ou seja, a geração $i + 1$ possui o mesmo número de indivíduos da geração i.

Não existe forma correta para definir a próxima geração de indivíduos, porém sua definição deve ser coerente com as características de um algoritmo genético, ou seja, a próxima geração usualmente é mais apta que a anterior. Um método comum é fazer com que a nova população seja composta por filhos gerados nos cruzamentos de acordo com a taxa de cruzamento e o restante seja selecionado através do operador de Seleção. Outro exemplo é mesclar a população de filhos com a população anterior e selecionar os indivíduos através do operador de seleção.

Existe também o conceito de elitismo, em que o(s) melhor(es) indivíduos de uma geração são sempre escolhidos para compor a próxima geração. A definição da próxima geração também depende do problema a ser resolvido. Após a execução de todas as gerações, o melhor indivíduo é selecionado como solução do problema.

4.2 Aplicação de Algoritmos Genéticos em Composição Algorítmica

A utilização de um algoritmo genético como heurística para solução de um problema, requer que seja definido como será codificado o indivíduo, como serão efetuados os procedimentos de cruzamento e mutação e ainda outros parâmetros mais simples como tamanho da população e número de gerações. Nesta seção descreveremos o algoritmo genético implementado para realizar a tarefa de composição de melodias monofônicas.

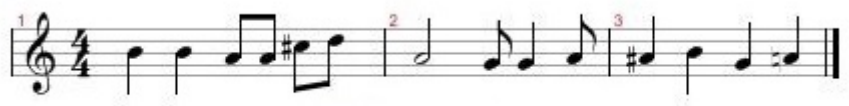
4.2.1 Codificação do Indivíduo

Neste trabalho, consideramos uma melodia monofônica como uma melodia de uma única trilha ou instrumento e que não possui sons simultâneos. Dada essa abstração, uma forma intuitiva de enxergar uma melodia monofônica é por um vetor de notas musicais. A representação de uma nota musical é feita a partir de uma estrutura de dados semelhante a utilizada por Herremans (2014). Porém, temos apenas três atributos, nota (inteiro), duração (real) e compasso (inteiro). A descrição de todos pode ser vista na Tabela 6.

Tabela 6 – Estrutura de Dados de uma nota Musical

Atributo	Tipo	Descrição
nota	inteiro	Valor inteiro MIDI do tom da nota
duração	real	Duração em termos de número de tempos
compasso	inteiro	Qual compasso da melodia a nota se encontra

Além de conter o vetor de notas descrito, o tipo de indivíduo definido neste trabalho possui algumas restrições musicais. As notas de um indivíduo estão divididas em compassos de forma que o começo de um compasso sempre coincide com o começo de uma nota, seja ela uma nota tocada ou uma pausa, conforme Figura 4, em que podemos ver uma partitura de uma melodia e abaixo uma tabela com sua codificação. Nessa tabela, a primeira linha corresponde a enésima nota da melodia, a segunda linha corresponde ao seu valor inteiro MIDI (formato de arquivo explicado no capítulo 1), a terceira à sua duração e a quarta o compasso em que se encontra. Durante a geração de um indivíduo aleatório, notas com tom e duração aleatórias são geradas conforme faixa de valores que será explicitada no capítulo 5. Caso a duração exceda o valor estabelecido no compasso (definido como 4 tempo neste trabalho), ela é descartada e uma nova nota é gerada aleatoriamente.



Nota	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Valor	59	59	57	57	61	62	57	55	55	57	58	59	55	57
Duração	1.0	1.0	0.5	0.5	0.5	0.5	2.0	0.5	1.0	0.5	1.0	1.0	1.0	1.0
Compasso	1	1	1	1	1	1	2	2	2	2	3	3	3	3

Figura 4 – Tipo de indivíduo implementado.

4.2.2 Função de Avaliação Musical

No capítulo 3 apresentamos a lei de Zipf, as regras de Fux e detalhamos como elas podem ser usadas quantitativamente para avaliar a qualidade de uma melodia. Porém, a utilização de algoritmos genéticos necessita da definição de uma função de avaliação que avalie tanto indivíduos (melodias) ruins quanto indivíduos bons. Essa função de avaliação deve ser reduzida a um número para que seja possível a comparação entre indivíduos para orientar a sua seleção durante a execução. Neste trabalho, o problema de composição algorítmica foi definido como um problema de maximização, ou seja, durante a comparação de indivíduos, o indivíduo que obtiver maior valor na função de avaliação será considerado mais apto. Nesta seção definiremos as funções de avaliação que foram utilizadas no nosso método de composição algorítmica.

4.2.2.1 Funções de Avaliação baseadas na Lei de Zipf

No capítulo 3 apresentamos os conceitos da lei de Zipf, como ela pode ser observada na música e como identificar se uma determinada música é considerada boa através da distribuição dos seus fenômenos musicais. Portanto, sabemos que os coeficientes angulares dos fenômenos musicais devem se aproximar do valor -1.

Porém, como possuimos diversos fenômenos musicais que podem ser definidos, torna-se difícil que as regressões lineares realizadas para todos os fenômenos apresentem o coeficiente angular desejado simultaneamente. Além disso, mesmo que os coeficientes angulares das regressões sejam próximos de -1, nada garante que cada um dos logaritmos das ocorrências dos eventos sejam bem representados pela regressão encontrada. Outra questão que precisa ser definida é como será calculado o valor de aptidão de um indivíduo de uma população de um algoritmo genético a partir desse coeficiente angular.

Para responder essas perguntas apresentaremos nesta seção duas funções matemáticas que são utilizadas como funções de avaliação musical de um indivíduo. Uma delas foi proposta por Jensen (2011), que consiste em uma função gaussiana do coeficiente angular. A outra, que está sendo proposta pela primeira vez neste trabalho, é baseada no

menor somatório dos erros quadráticos médios das regressões lineares.

4.2.2.1.1 Função Gaussiana

Sabemos que o coeficiente angular extraído de uma melodia agradável, conforme a lei de Zipf, deve apresentar um valor próximo de -1. Porém, para evoluir uma música através de um algoritmo evolucionário, precisamos avaliar melodias boas e melodias ruins, ou seja, precisamos definir um valor numérico que seja bom quando o coeficiente é próximo de -1 e que piore à medida que o coeficiente angular se distância de -1.

Uma função matemática com essa característica é a função gaussiana da equação (20), proposta por Jensen (2011). Essa função possui a variável independente x que indica o coeficiente angular da regressão extraído da melodia. O parâmetro T representa o coeficiente angular alvo (no nosso caso, -1) e o parâmetro λ indica a tolerância da gaussiana, ou seja, o quão longe de -1 admitimos que o coeficiente angular seja. A Figura 5 mostra o gráfico da função gaussiana para λ assumindo os valores $0.5, 1.0, 1.5$ e 2.0. Podemos ver que quanto menor o valor de λ, menor será a tolerância a um coeficiente angular diferente de -1.

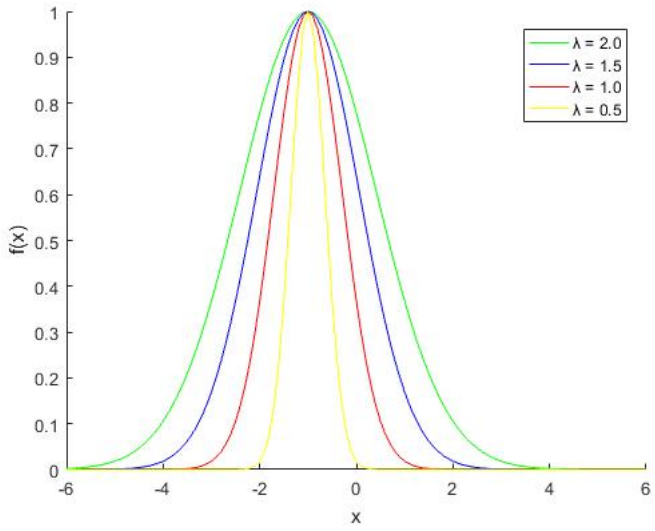


Figura 5 – Gráfico Função Gaussiana

No trabalho de Jensen (2011) o valor de λ escolhido foi igual a 0.5. Para cada um dos fenômenos definidos no capítulo 3 temos uma variável dependente f_m que denota o valor numérico da gaussiana. Após substituir os parâmetros definidos, obtemos a função da equação (21). Podemos inferir dessa equação, que cada função gaussiana assume valores no intervalo $[0,1]$, em que 1 é considerado bom e 0 ruim.

$$f_m(x) = e^{-(\frac{T-x}{\lambda})^2} \tag{20}$$

$$f_m(x) = e^{-(\frac{-1-x}{0.5})^2} \tag{21}$$

Após definir uma função para avaliar a qualidade do coeficiente angular para cada fenômeno musical, precisamos elaborar uma forma de agrupar todos os valores encontrados em apenas um valor. A forma escolhida para agregar esses valores foi a de um somatório simples de todos os valores da função gaussiana, conforme equação 22, em que N é o número de fenômenos musicais, M representa a melodia avaliada, x_i o i-ésimo fenômeno musical, f_m a função gaussiana e F representa a função de avaliação que é utilizada na comparação de melodias. Como no capítulo 3 foram definidos quatorze fenômenos musicais para cálculo da lei de Zipf e o problema foi definido como um problema de maximização, a função F assume valores no intervalo $[0,14]$, em que 14 é o melhor valor de avaliação possível. Todas as abstrações para modelagem da função gaussiana foram propostas por Jensen (2011).

$$F(M) = \sum_{i=1}^{N} f_m(x_i) \tag{22}$$

4.2.2.1.2 Função Erro Médio Quadrático

Apenas obter um coeficiente angular -1 em uma regressão linear não é garantia que a melodia avaliada siga a lei de Zipf. Uma regressão linear é feita ajustando uma reta $bx + a$ de forma que os coeficientes a e b escolhidos minimizem o erro quadrático médio, ou seja, minimizem o somatório da equação (23). Se esse erro for alto, significa que os pontos não estão bem alocados ao longo da reta, portanto a regressão linear não é boa e a melodia não obedece a lei de Zipf. A Figura 6 apresenta uma comparação entre duas regressões lineares cujo coeficiente angular é próximo de -1. Na regressão da esquerda vemos que os pontos no plano cartesiano não estão bem posicionados em cima da reta de regressão, gerando um erro quadrático médio alto. Já a regressão da esquerda mostra uma distribuição apropriada dos eventos de um fenômeno musical.

$$min \sum (Y_i - bX_i - a)^2 \tag{23}$$

Partindo do pressuposto que o erro quadrático médio é um fator tão importante quanto o coeficiente angular, foi proposta neste trabalho uma função de avaliação que não leva em consideração o quão próximo o coeficiente angular está de -1, mas sim o quão bem as ocorrências são representadas pela regressão linear. Elaboramos a função matemática da equação (24) que, para cada fenômeno, calculamos o erro quadrático médio dos eventos em relação a regressão linear. A variável dependente fze_m representa o valor

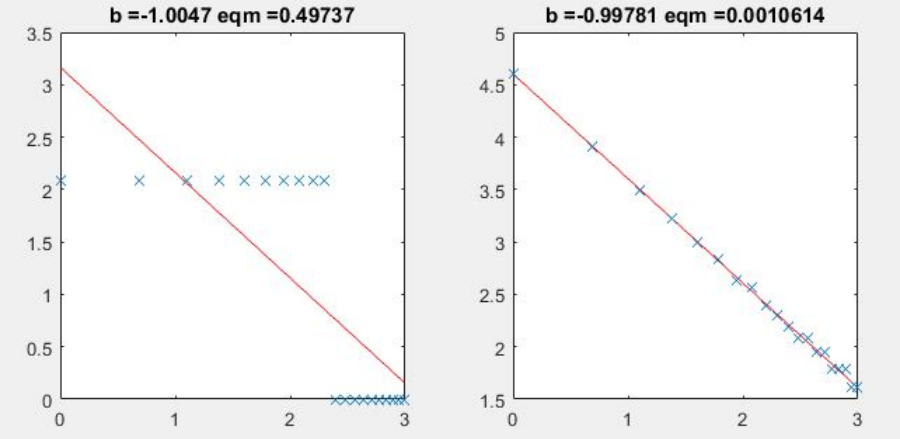


Figura 6 – Diferença do erro quadrático médio para duas regressões com coeficientes angulares próximos de -1

da função para cada fenômeno m, o numerador é o somatório dos erros da regressão e o denominador q representa a quantidade de eventos do respectivo fenômeno.

$$fze_m(M) = \frac{\sum(Y_i - bX_i - a)^2}{q} \tag{24}$$

Enfim, de forma análoga à função gaussiana, definimos a função de avaliação baseada no erro quadrático médio pela equação (25), como o somatório dos erros quadráticos médios de cada fenômeno multiplicado por -1, em que FZE representa o valor numérico da avaliação musical, M é a melodia a ser avaliada, $fze_m(M)$ é o erro quadrático médio de cada fenômeno. Podemos inferir então, que a função de avaliação pode assumir qualquer valor no intervalo $(-\infty,0]$. Como estamos tratando de um problema de maximização e quanto mais próximo de 0 melhor o valor da função, o somatório dos erros foi multiplicado por -1, propiciando coerência no problema, em que erros com magnitude maior, possuem menor valor de avaliação.

$$FZE(M) = -\sum_{i=1}^{N} fze_m(M) \tag{25}$$

Obtemos, assim, o modelo de otimização da equação (26), em que $fze_m(M)$ é o erro quadrático médio de cada fenômeno da lei de Zipf, n é uma nota qualquer na melodia, n_{ij} é a j-ésima nota do i-ésimo compasso, $d()$ é uma função cuja imagem é a duração de uma nota musical e C é o conjunto de compassos da melodia. O conjunto de restrições do modelo, definem que o começo de um compasso sempre coincidirá com o começo de uma nota.

$$\begin{gathered} max(-\textstyle\sum_{i=1}^{N} fze_m(M)) \\ s.a.: \textstyle\sum d(n_{ij}) = 4, \forall i \in C \end{gathered} \tag{26}$$

4.2.2.2 Função de Avaliação baseada nas Regras de Fux

Diferente da lei de Zipf, a própria definição da regra de Fux em (HERREMANS, 2014) já fornece uma fórmula matemática fechada, pronta para ser utilizada como função de avaliação. A única ressalva a se fazer é que nessa definição, a composição algorítmica foi definida como um problema de minimização, e neste trabalho trata-se de um problema de maximização. Portanto, adaptamos a equação (19) para que quanto maior o valor da função, maior seja a qualidade da música. Obtemos assim a função de avaliação da equação (27), em que $Fux(M)$ é o valor da função de avaliação da melodia M e $subscore_i(M)$ é o valor de cada uma das dezessete regras de Fux. O valor da função de avaliação está presente no invervalo $[-17,0]$, em que assume o valor -17 caso todas as regras apresentem o valor 1, caracterizando uma melodia ruim, e assume o valor 0 caracterizando uma boa melodia.

$$Fux(M) = -\sum_{i=1}^{17} subscore_i(M) \tag{27}$$

O modelo de otimização utilizado para função de avaliação baseada nas regras de Fux é apresentado na equação (28), em que $subscore_i(M)$ é o valor de cada uma das dezessete regras de Fux para melodia M e o conjunto de restrições é o mesmo do modelo da equação 26.

$$\begin{array}{c} max(-\sum_{i=1}^{17} subscore_i(M)) \\ s.a.: \sum d(n_{ij}) = 4, \forall i \in C \end{array} \tag{28}$$

4.2.3 Operadores de Cruzamento

O operador de cruzamento foi nomeado de *crossOverBar* (Cruzamento de Compasso). Conforme Figura 7, nesse operador é selecionada uma posição aleatória no vetor de notas de dois indivíduos selecionados pelo operador de seleção, concatenam-se as notas antes da posição aleatória do primeiro indivíduo com as notas depois da posição aleatória do segundo indivíduo gerando assim um novo indivíduo. Após isto, as notas depois da posição aleatória do primeiro indivíduo são concatenadas com as notas antes da posição sorteada do segundo indivíduo, gerando assim o segundo filho. É importante ressaltar que apenas as notas em finais de compasso são pontos de cruzamento aceitáveis, portanto a posição aleatória não é sempre a mesma nos dois vetores de notas, ou seja, caso seja sorteada a posição 45 que está no final do compasso 5, na segunda melodia a nota selecionada será a posição cuja nota é a última nota do compasso 5, e assim as notas antes da quadragésima quinta nota do primeiro vetor serão concatenadas com o subvetor do segundo indivíduo contendo da nota posterior ao compasso 5 até a última nota da melodia, garantindo assim que as novas melodias criadas tenham sempre a mesma quantidade de compassos. É importante ressaltar que sempre que aplicamos esse operador de cruza-

mento, os indivíduos gerados sempre obedecerão à restrição de compasso estabelecida nos modelos matemáticos das equações (26) e (28). Além disso, os filhos provenientes do cruzamento herdarão parte dos vetores de notas dos pais e, por consequência, suas características boas.

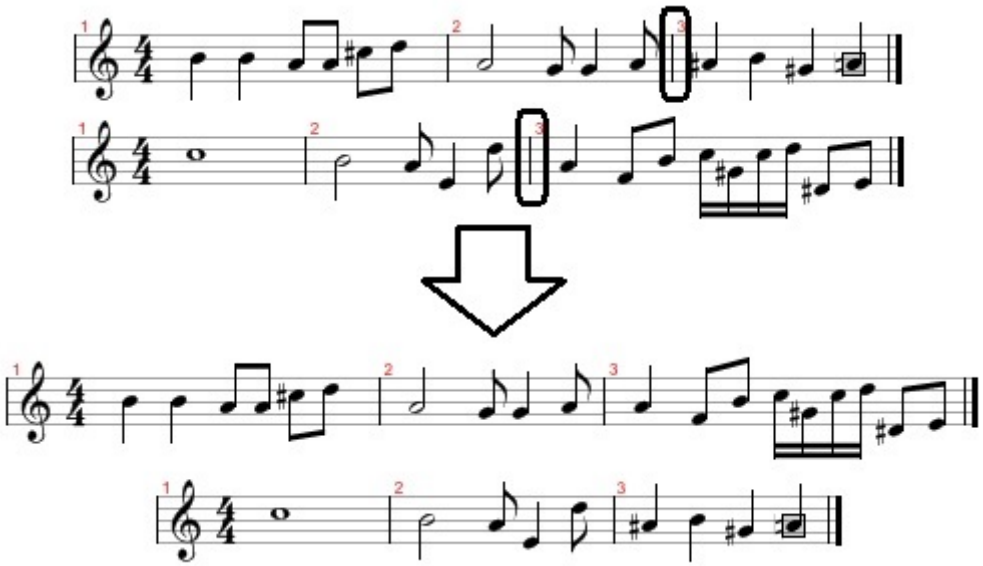


Figura 7 – Execução do operador *crossOverBar*.

4.2.4 Operadores de Mutação

Os operadores de mutação definidos neste trabalho tem o objetivo de melhorar sutilmente o valor da função de avaliação. Quando tratamos de melhorar a função de avaliação, estamos pensando em mutações mais tendenciosas, que, apesar de conter aleatoriedade no processo, estaremos alterando partes da melodias que são indesajáveis perante a função de avaliação. Os operadores de mutação são:

- ***mutateRhythmTrigram:*** Modifica um trigrama rítmico;
- ***mutateMelodicTrigram:*** Modifica um trigrama melódico;
- ***mutateMelodicAndRhythmTrigram:*** Combinação das duas mutações acima;
- ***mutateCopyingPartMusic:*** Substitui uma parte da melodia por uma outra parte aleatória da própria melodia.

O operador *mutateRhythmTrigram* é baseado no fenômeno musical Trigrama Rítmico citado no capítulo 3. Dado que um trigrama rítmico é uma tripla ordenada de três intervalos rítmicos, um trigrama rítmico é formado por uma sequência de 4 notas musicais. A ideia central desse operador é selecionar um trigrama rítmico na melodia de acordo com um critério específico e modificar os tempos das duas notas centrais da sequência de quatro notas, modificando, assim, todos os intervalos rítmicos do trigrama. O exemplo da Figura 8 mostra uma execução do operador em que as durações das notas Sol e Sol Sustenido do

trigrama circulado, com duração de uma colcheia, são alteradas para uma semínima e uma mínima respectivamente.

O critério escolhido para escolha do trigrama rítmico foi verificar se a reta obtida na regressão linear possui coeficiente angular de magnitude maior ou menor do que o coeficiente angular alvo. Se o coeficiente tiver maior magnitude, significa que a reta da regressão linear precisa ter uma inclinação menor, portanto escolhemos um trigrama rítmico da metade esquerda da reta de regressão. Se o coeficiente tiver magnitude menor, a reta precisará ter inclinação maior, portanto tomaremos um trigrama da metade direita da reta de regressão. No exemplo da Figura 9, que apresenta uma regressão linear do fenômeno trigrama rítmico, obtemos um coeficiente angular de valor $-0{,}72$. Como o valor esperado do coeficiente é -1, devemos aumentar sua magnitude, ou seja, o operador de mutação seleciona um trigrama rítmico da metade esquerda da reta para ser modificado (trigramas que contribuem para os eventos cujos números de ocorrências são 1 ou 2).

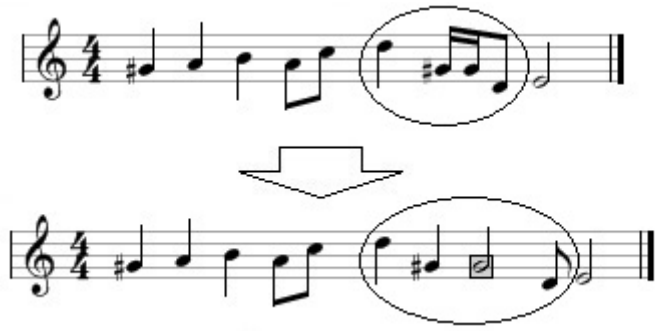


Figura 8 – Modificação do trigrama rítmico pelo operador mutateRhythmTrigram

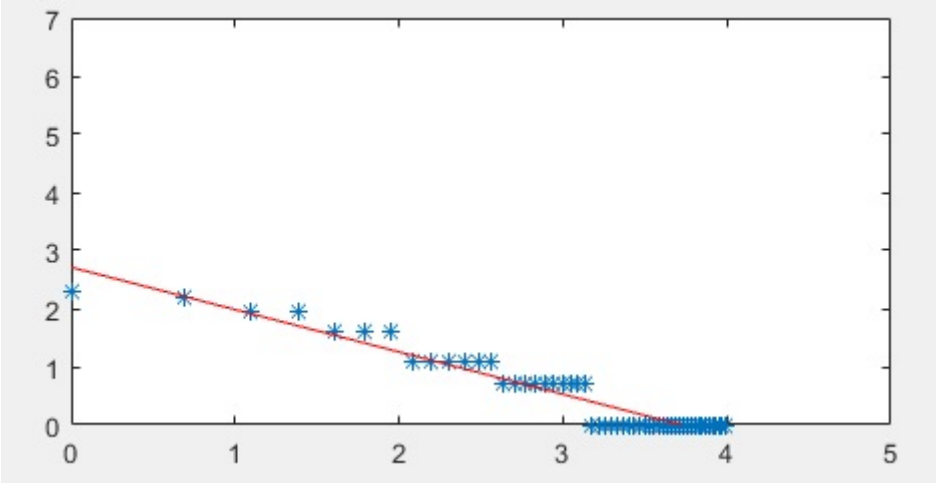


Figura 9 – Regressão linear ($y = bx + a$) do fenômeno trigrama rítmico de uma música composta neste trabalho: $b = -0{,}72$

O procedimento do operador *mutateMelodicTrigram* é análogo ao mutateRhythmTrigram, porém, baseado no fenômeno trigrama melódico. A forma de selecionar o trigrama a

ser modificado é a mesma, porém o atributo musical que é modificado é o tom da nota e não sua duração. Novamente as duas notas do interior do trigrama que serão modificadas pelo operador, conforme Figura 10 em que as notas Sol e Sol Sustenido do trigrama circulado são modificadas para Fá e Mi respectivamente. O operador *mutateMelodicAndRhythmTrigram* nada mais é do que executar o operador *mutateRhythmTrigram* e após isto executar o operador *mutateMelodicTrigram*. Obtendo assim notas com duração e tom diferentes.

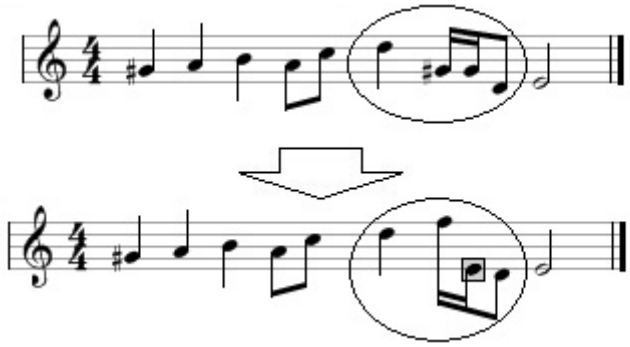


Figura 10 – Modificação do trigrama melódico pelo operador mutateMelodicTrigram

Partindo do pressuposto que a lei de Zipf é uma lei que se baseia na repetição de eventos, podemos inferir que em uma melodia, a repetição desses eventos contribua para uma boa avaliação pela lei de Zipf. Pensando nesse fato, implementamos um operador de mutação denimonado *mutateCopyingPartMusic*. Esse operador consiste em selecionar um subconjunto de notas consecutivas da melodia, cujo tamanho do subconjunto varia aleatoriamente entre 5% e 10% do tamanho da melodia e selecionar um segundo subconjunto de mesmo tamanho que o primero, porém, que não contenha nenhuma das notas do primeiro subconjunto. O segundo subconjunto então é substituído por uma cópia do primeiro subconjunto.

Ao compor melodias que respeitam o compasso da música, podemos encontrar problemas ao alterar a duração de uma nota, pois o compasso que aquela nota se encontra poderá exceder os quatro tempos permitidos ou poderá faltar tempos para completar o compasso. Para contornar esse problema, implementamos uma política em que sempre que um compasso tiver seu tempo excedido, deveremos excluir notas aleatórias que não sejam as notas que acabaram de sofrer alteração até que ele fique com os quatro tempos esperados. Caso o compasso precise de notas adicionais, deveremos adicionar notas da mesma forma que a geração de indivíduos respeitando o compasso, ou seja, nunca adicionar uma nota que faça com que o compasso exceda o tempo permitido.

4.2.5 Definição da Próxima Geração

A definição da próxima geração implementada utiliza o conceito de elitismo, mantendo sempre o indivíduo melhor avaliado da população anterior. Para os demais indivíduos,

criamos uma nova população que engloba tanto os indivíduos da geração anterior quanto os filhos gerados nos cruzamentos. O operador de seleção torneio binário é empregado para selecionar dentre toda a população os indivíduos que irão compor a próxima geração. O operador de mutação é aplicado apenas nos indivíduos que já estavam na geração anterior.

4.3 Abordagem Multiobjetivo

Diferente da otimização mono-objetivo, a otimização multiobjetivo consiste em encontrar uma solução que represente um bom valor numérico não para uma função, mas para duas ou mais funções de avaliação. Problemas multiobjetivos são complexos de serem resolvidos, pois pode haver funções de avaliação conflitantes em que sempre que melhoramos o valor de uma, pioramos o valor de outra. Para resolver esses problemas utilizam-se métodos de otimização multiobjetivo.

4.3.1 Otimização Multiobjetivo

Cunha, Takahashi e Antunes (2013) definem a otimização multiobjetivo como o problema de otimização vetorial (POV) da equação (29), em que $f(.) : \mathbb{R}^n \rightarrow \mathbb{R}^m$ representa o vetor funções de avaliação do problema, $g_i(x)$ e $h_j(x)$ as restrições de desigualdades e igualdades e m e n são inteiros que definem as dimensões do problema. Os vetores $x \in \mathbb{R}^n$ constituem o espaço de parâmetros, ou seja as possíveis soluções para o problema e compõem o espaço de parâmetros χ. Os vetores $f(x) \in \mathbb{R}^m$ os valores de avaliação que x pode assumir. O objetivo do problema é encontrar um conjunto de vetores $\chi^* \in \chi$ que sejam boas soluções para o problema modelado. Esse conjunto de vetores de soluções é denominado **conjunto pareto-ótimo**.

$$POV \begin{cases} \chi^* = x^* \in \mathbb{R}^n \\ x^* = argmin_x f(x) \\ \text{sujeito a : } \begin{cases} g_i(x) \leq 0, \forall i = 1, ..., r \\ h_j(x) = 0, \forall j = 1, ..., p \end{cases} \end{cases} \tag{29}$$

Dado que em um problema de otimização multiobjetivo precisamos otimizar duas ou mais funções matemáticas simultaneamente, uma solução x do problema que possui um valor de função de avaliaçao $f_1(x)$ melhor do que o valor de função de avaliação $f_1(y)$, mas possui $f_2(x)$ pior do que $f_2(y)$ não pode ser considerada uma solução melhor do que a solução y. Para comparar duas soluções em um problema multiobjetivo precisamos definir o conceito de **dominância**. Dado um problema de minimização, uma solução $x_1 \in \chi$ domina uma solução $x_2 \in \chi$ ($x_1 \preceq x_2$) se $f_i(x_1) \leq f_i(x_2), \forall i$ e $\exists i : f_i(x_1) < f_i(x_2)$. Dada esta definição, uma solução x^* para um problema multiobjetivo só pode ser considerada uma boa solução se ela não for dominada por nenhuma outra solução factível do problema.

Chamamos esta solução de **solução pareto-ótima**. A Figura 11 mostra um conjunto pareto-ótimo de um problema hipotético, em que as soluções A, B e E não são dominadas por nenhuma outra solução. Nessa figura, os eixos x e y definem valores de um problema arbitrário.

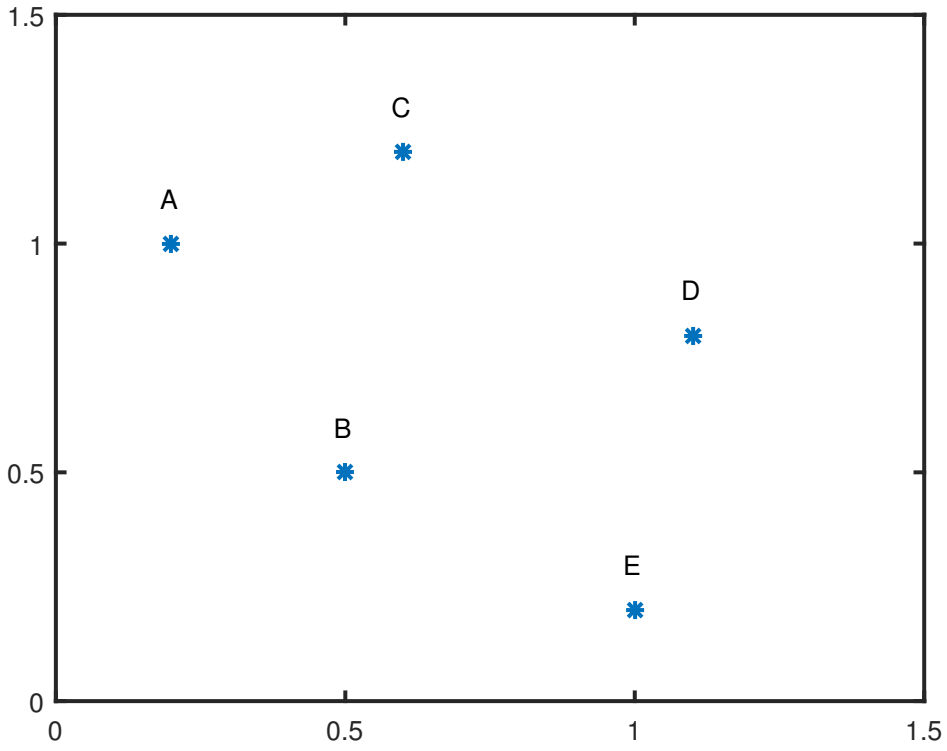


Figura 11 – Exemplo de conjunto pareto-ótimo de um problema de minimização, formado pelos pontos A, B e E.

4.3.2 Método NSGA-II

Para tratar o problema de conflito entre funções de avaliação existem vários métodos na literatura. Uma estratégia comum é a utilização de métodos escalares que são responsáveis por gerar soluções eficientes, transformando problemas multiobjetivos em mono-objetivos. Dentre os métodos mais comuns estão soma ponderada (P_λ), ϵ-restrito (P_ϵ), *goal attainment* (P_χ) e programação alvo (P_μ). Porém existem também estratégias que não reduzem o problema a um problema mono-objetivo. Um método comum é o *nondominated sorting genetic algorithm* (NSGA-II).

O algoritmo NSGA-II (DEB et al., 2002) é um algoritmo genético que utiliza um método de ordenação rápida não dominada (*fast nondominated sorting*) para ordenar os indivíduos da população em diferentes níveis de dominância, utiliza o conceito de preservação da diversidade para classiciar indivíduos no mesmo nível de dominância e usa

o conceito de elitismo para definir a próxima geração. A estrutura do NSGA-II pode ser vista no Algoritmo 1.

Algoritmo 1: Algoritmo NSGA-II. Extraído de (DEB et al., 2002) e adaptado para este trabalho.

início
  $t \leftarrow 0$ // Geração atual
  $P_t \leftarrow populacaoInicial()$ //Gera população inicial
  $P_t \leftarrow avaliacaoSolucoes(P_t)$ $Q_t \leftarrow 0$ //População de filhos
  repita
    $R_t = P_t \cup Q_t$ //R_t é a população de pais e filhos
    $F = fastNonDominatedSort(R_t)$
    $P_{t+1} \leftarrow \varnothing$
    $i = 1$
    repita
      $crowdingDistanceAssignment(F_i)$
      $P_{t+1} = P_{t+1} \cup F_i$
      $i = i + 1$
    até $|P_{t+1}| + |F_i| \leqslant tamanhoPopulacao$;
    $ordenar(F \preceq n)$ //Ordena a fronteira F_i
    $P_{t+1} = P_{t+1} \cup F_i[1 : (tamanhoPopulacao) - |P_{t+1}|]$ //Adiciona os indivíduos restantes para completar a população
    $Q_{t+1} = operadoresGeneticos(P_{t+1})$
    $Q_{t+1} = avaliacaoSolucoes(Q_{t+1})$
    $t = t + 1$
  até *Critério de parada não for satisfeito*;
fim

O primeiro passo para execução do NSGA-II é gerar uma população inicial (aleatória ou com algum critério) e avaliá-la de acordo com as funções de avaliação definidas. Após isso utilizamos os operadores de seleção, cruzamento e mutação para gerar uma população de filhos com o mesmo número de indivíduos e mesclamos estas duas populações gerando uma grande população. O NSGA-II executa o procedimento *fast nondominated sorting* para ordenar toda a população mesclada em níveis de dominância. Os indivíduos no primeiro nível de dominância não são dominados por nenhum outro indivíduo da população, constituindo o conjunto pareto-ótimo daquela iteração do algoritmo. O segundo nível de dominância é formado por indivíduos que são dominados por apenas um indivíduo e assim sucessivamente.

Após a ordenação por nível de dominância o NSGA-II calcula o valor estimado de densidade (medida utilizada para preservação da diversidade) de cada indivíduo do primeiro nível de dominância utilizando o método *crowding distance assignment* e inclui os indivíduos na próxima geração. Esse procedimento é repetido para os demais níveis de dominância até que o tamanho da nova população seja menor ou igual ao tamanho definido como parâmetro para população. Caso todos os indivíduos contido no último nível de dominância

verificado não possam ser inseridos na nova população (devido a extrapolação do tamanho definido), deve-se ordenar os indivíduos no nível de dominância pelo valor de densidade e inserir os indivíduos de maior valor de densidade até que a população atinja seu valor máximo. Esta população constituirá a próxima geração do NSGA-II.

O procedimento *fast nondominated sorting*, cuja estrutura é encontrada no Algoritmo 2, consiste primeiramente em selecionar um indivíduo p na população, verificar em toda a população se o indivíduo p domina um indivíduo q. Se q for dominado, inserimos q em um vetor de indivíduos dominados de p. Caso contrário verificamos se q domina p. Se isto for verdade incrementamos o contador de nível de dominância n_p de p em uma unidade. Após isto, se n_p for 0, inserimos p no primeiro nível de dominância da população. Repetimos esse procedimento para todos indivíduos da população até formar todo primeiro nível de dominância.

Para formar os próximos níveis de dominância, o *fast nondominated sorting* seleciona cada indivíduo q contido no vetor de indivíduos dominados de cada indivíduo do primeiro nível, reduz em uma unidade seu contador de nível de dominância n_q e caso n_q se torne 0, este indivíduo é adicionado no próximo nível de dominância. Repetimos esse procedimento até que o nível de dominância F_i seja um conjunto vazio.

A medida de densidade, também chamada de *crowding distance*, é calculada pelo procedimento do Algoritmo 3. Para cada função de avaliação, ordenamos os indivíduos em um nível de dominância pelo valor de função de avaliação em questão. Os indivíduos nos extremos do nível de dominância recebem o valor infinito. Os valores no interior do nível de dominância recebem o valor calculado na linha 10.

Para comparar dois indivíduos no NSGA-II, utilizamos o *crowded-comparison operator* que consiste em verificar primeiramente se o nível de domiância do indivíduo i é menor que o do indivíduo j. Se for menor, i é melhor que j. Se for igual, verificamos o valor de densidade. Se o valor de densidade de i for maior, i é melhor que j. Caso contrário j é melhor que i. Se o valor do nível de dominância de j for maior do que o de i, j também será melhor do que i.

4.3.3 Aplicação do NSGA-II na Composição Algorítmica

Um ponto importante na utilização do NSGA-II é que as funções de avaliação utilizadas devem ser ambas de maximização ou ambas de minimização. Em nossa implementação, utilizamos as funções das equações (24) e (27) como funções de avaliação de um problema de maximização. Além disso, para que o procedimento *crowding distance assignment* seja realizado corretamente, temos que definir o domínio de valores das funções de avaliação como sempre positivos ou sempre negativos. Como nossas funções de avaliação pertencem aos intervalos $(-\infty, 0]$ e $[-17{,}0]$ respectivamente, essas definições não são necessárias

Algoritmo 2: Procedimento *fast nondominated sorting*. Extraído de (DEB et al., 2002) e adaptado para este trabalho.

início
  para *cada* $p \in P$ **faça**
    $S_p \leftarrow \varnothing$ //Conjunto de soluções que p domina
    $n_p \leftarrow 0$ //Contador de soluções que dominam p
    $F_1 \leftarrow \varnothing$ //Primeira fronteira não dominada
    para *cada* $q \in P$ **faça**
      se *(*$p \preceq q$ *)* **então**
        $S_p = S_p \cup q$//Se p domina q, então adiciona q no conjunto S_p
      fim
      senão
        se *(*$q \preceq p$ *)* **então**
          $n_p = n_p + 1$//Se q domina p, então incrementa o contador n_p
        fim
      fim
    fim
    se *(*$n_p = 0$ *)* **então**
      $P_{rank} = 1$ //Se o contador n_p é zero, então é uma solução não-dominada
      $F_1 = F_1 \cup p$ //Pertence à primeira fronteira
    fim
  fim
  $i \leftarrow 1$ // Contador de fronteiras, inicia na primeira fronteira (soluções não-dominadas encontradas anteriormente)
  repita
    $Q \leftarrow \varnothing$ //Conjunto de soluções pertencentes a fronteira $i + 1$
    para *cada* $p \in F_i$ **faça**
      para *cada* $q \in S_p$ **faça**
        $n_q = n_q - 1$ //Para cada solução q dominada por p, decrementa-se o contador
        se $n_q = 0$ **então**
          $q_{rank} = i + 1$ //Se n_p é zero, então não existe outra solução que domina q
          $Q = Q \cup q$ //Pertence à fronteira $i + 1$
        fim
      fim
    fim
    $i = i + 1$
    $F_i = Q$ //Fronteira i recebe as soluções encontradas do conjunto Q
  até $F_i \neq \varnothing$;
fim

Algoritmo 3: Procedimento *crowding distance assignment*. Extraído de (DEB et al., 2002) e adaptado para este trabalho.

início
  $l = |I|$
  para *cada* i **faça**
    $I[i]_{distancia} = 0$ //Inicializa todos os atributos como zero
  fim
  para *cada função de avaliação* m **faça**
    $I = Ordena(I,m)$// Ordena usando o valor de cada função de avaliação
    $I[1]_{distancia} = I[l]_{distancia} = \infty$ //Valor infinito para soluções limite
    para $i = 2$ *até* $l - 1$) **faça**
      $I[i]_{distancia} = I[i]_{distancia} + (I[i+1].m - i[i-1].m)/(f_m^{max} - f_m^{min})$
      //Cálculo do crowding distance
    fim
  fim
fim

em nossa aplicação.

Os operadores genéticos de mutação e cruzamento utilizados são os mesmos citados na seção 4.2. Apenas o operador de seleção deve ser modificado. O operador de seleção implementado deve, ao invés de selecionar o indivíduo com maior valor de função de avaliação, selecionar o indivíduo considerado melhor pelo operador *crowded-comparison operator*.

Definimos como nosso modelo de otimização multiobjetivo o modelo apresentado na equação (30). As funções de avaliação a serem maximizadas são exatamente iguais às funções das equações (26) e (28) e o conjunto de restrições de compasso também é o mesmo. Dada essas definições, o algoritmo NSGA-II está pronto para ser aplicado ao nosso método de composição algorítmica.

$$\begin{gathered} max(-\sum_{i=1}^{17} subscore_i(M), -\sum_{i=1}^{N} fze_m(x_i)) \\ s.a. : \sum d(n_{ij}) = 4, \forall i \in C \end{gathered} \tag{30}$$

4.4 Considerações Finais

Neste capítulo, apresentamos toda fundamentação teórica referente a algoritmos genéticos mono-objetivos e multiobjetivos, necessária para o entendimento dos métodos de composição algorítmica elaborados neste trabalho. Explicamos também as particularidades inerentes aos nossos métodos, tais como operadores genéticos, funções de avaliação e outros parâmetros necessários para aplicação de algoritmos genéticos. No capítulo 5 apresentaremos todos os experimentos executados com os diferentes parâmetros e operadores implementados. Apresentaremos também análises estatísticas que evidenciam

a qualidade musical dos métodos propostos neste trabalho.

5 Análise e Discussão dos Resultados

Nos capítulos 3 e 4 apresentamos toda fundamentação teórica relacionada a algoritmos genéticos, métricas de avaliação musical e como foram utilizadas neste trabalho as abordagens mono-objetivo e multiobjetivo para composição de melodias. Neste capítulo mostraremos os resultados obtidos com as análises gráficas e estatísticas dos métodos propostos. Apresentaremos também uma outra contribuição deste trabalho, que é uma metodologia de avaliação baseada nas respostas de questionários por pessoas que escutaram as melodias compostas pelos métodos de composição propostos neste trabalho. Todas as melodias compostas neste trabalho podem ser escutadas através da página do SoundCloud do autor (LOPES; SANTOS; MARTINS, 2016).

5.1 Codificação e Simplificação Musical

A primeira questão a ser tratada quando trabalhamos com composição algorítmica é definir como a melodia será interpretada pelo computador. Portanto, tipo de arquivo, estruturas de dados e bibliotecas são as primeiras questões a serem definidas. Dado que teremos que realizar várias operações matemáticas em uma melodia, a escolha da estrutura de dados deve ser feita de forma a facilitar o tratamento e as operações musicais contidas neste trabalho. O mesmo vale para o tipo de arquivo e biblioteca. O tipo de arquivo escolhido deve conter informações que podem ser traduzidas facilmente através de uma biblioteca para estrutura de dados escolhida.

5.1.1 Formato de Arquivo e Biblioteca

Já foi mencionado no capítulo 1 a existência do formato MIDI. Em um arquivo MIDI estão presentes instruções contendo informações de qual nota deve ser tocada e qual sua duração. Em um arquivo MIDI, uma nota musical é representada por um número inteiro. Essa representação facilita o cálculo de diversos passos contidos nas métricas de avaliação. Além disso o formato MIDI é amplamente difundido, sendo compatível com a maioria dos software reprodutores de mídia musical e ainda é o formato mais utilizado na literatura.

Dado que este trabalho utiliza as métricas de avaliação contidas em (HERREMANS, 2014), utilizamos a mesma biblioteca de codificação e decodificação em MIDI. A biblioteca utilizada foi a JMusic para Java, desenvolvida por Sorensen e Brown (1998). Através dela conseguimos obter a informação da nota tocada como um número inteiro e da duração tocada como um número de ponto flutuante.

5.1.2 Simplificações Musicais

Tratando-se de composição algorítmica e devido a grande variedade de combinações de notas e durações diferentes é comum encontrarmos na literatura restrições para os valores que o tom da nota e a duração assumem, se ligaduras são permitidas ou não, entre outras.

Neste trabalho realizamos dois tipos de composições, composições livres e composições que utilizam apenas notas da escala de Dó maior. Restringimos os valores de tons a três oitavas, ou seja, 36 tons diferentes para composições livres, com o atributo nota apresentado na Tabela 6 podendo assumir valores MIDI entre 36 e 72, e 21 tons diferentes para composições em Dó maior, com o atributo nota podendo assumir apenas os valores MIDI entre 36 e 72 que representam notas na escala de Dó maior. Notas ligadas não serão permitidas e as durações permitidas são de $4, 2, 1, 0.5$ e 0.25 tempos. As melodias compostas possuem um tamanho total de 30 compassos.

5.2 Resultados Numéricos

Antes de realizar qualquer experimento, decidimos verificar como as funções de avaliação avaliam melodias geradas aleatoriamente. Para isso, geramos 10 mil melodias aleatórias, avaliamos todas elas pela função Fux e pela função ZipfEQM e plotamos os pares ordenados encontrados no gráfico da Figura 12, em que o eixo x representa a avaliação dos indivíduos pela função Fux e o eixo y pela função ZipfEQM. Essa visualização é importante pois nos dá uma uma noção da disposição das melodias aleatórias dentro do espaço de busca, ou seja, uma visão do ponto inicial da otimização.

Podemos ver na Figura 12 que, apesar do espaço de busca consistir no intervalo $[-17{,}0]$ para função Fux e $(-\infty, 0]$ para a função ZipfEQM, composições aleatórias encontram-se num espaço de busca menor, entre os intervalos $[-11{,}92; -7{,}81]$ para função Fux e $[-7{,}31; -3{,}47]$ para função ZipfEQM, ou seja, uma composição aleatória consegue capturar algumas das características de uma melodia considerada boa pelas funções de avaliação. A partir daí, basta evoluir essa população inicial aleatória para obter melodias melhores.

Para realizar essa tarefa implementamos 4 métodos de composição algorítmica. Afim de verificar a performance dos métodos implementados, nós conduzimos uma série de experimentos gráficos e estatísticos. Os métodos implementados neste trabalho são:

1. Algoritmo genético mono-objetivo com função de avaliação Fux;
2. Algoritmo genético mono-objetivo com função de avaliação ZipfEQM;
3. Algoritmo NSGA-II com funções de avaliação Fux e ZipfEQM;
4. Algoritmo NSGA-II com funções de avaliação Fux e ZipfEQM e com uma população

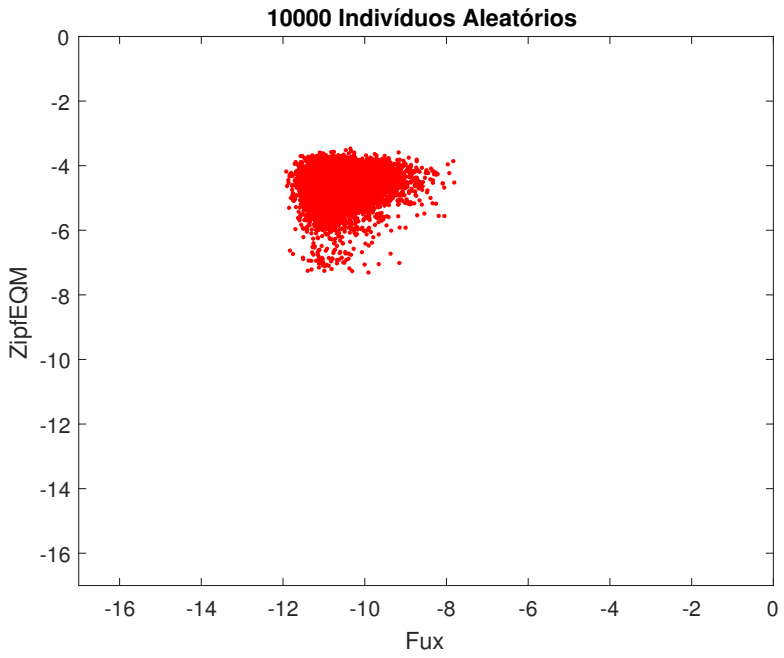


Figura 12 – Avaliações de melodias aleatórias.

inicial contendo 66 indivíduos. Desses 66, 33 são os melhores indivíduos de 33 execuções distintas do método 1 e 33 do método 2.

5.2.1 Experimentos com Métodos Mono-objetivos

O primeiro experimento com algoritmos mono-objetivos conduzido foi verificar a curva de convergência média de 33 execuções diferentes dos métodos 1 e 2 para diferentes taxas de cruzamento e mutação. Esse experimento é importante para verificar se os operadores de cruzamento e mutação implementados neste trabalho são bons o suficiente para encontrar bons valores para as funções de avaliação propostas.

O gráfico esquerdo da Figura 13 mostra a curva de convergência média para cada combinação de parâmetros utilizada na composição de melodias pelo método 1 e mostra como o valor de avaliação da função ZipfEQM evolui de acordo com a otimização orientada pela função de avaliação Fux. O gráfico direito mostra o análogo para otimização orientada pela função ZipfEQM. As taxas de cruzamento utilizadas foram $0{,}9$ e $0{,}7$ e as taxas de mutação foram $0{,}3$ e $0{,}1$. O número de gerações escolhido foi 1000.

A primeira observação feita a partir da Figura 13 é que, o valor médio encontrado na geração 500, de ambos os métodos, é muito próximo do valor encontrado na geração 1000, sendo assim, o número de gerações escolhido é suficiente para que os métodos 1 e 2 convirjam. Além disso, ambos valores médios finais encontrados na geração 1000 estão muito próximos de 0 (valor ótimo para ambas funções de avaliação). Isso quer dizer que, somos capazes de otimizar as funções de avaliação definidas para este trabalho.

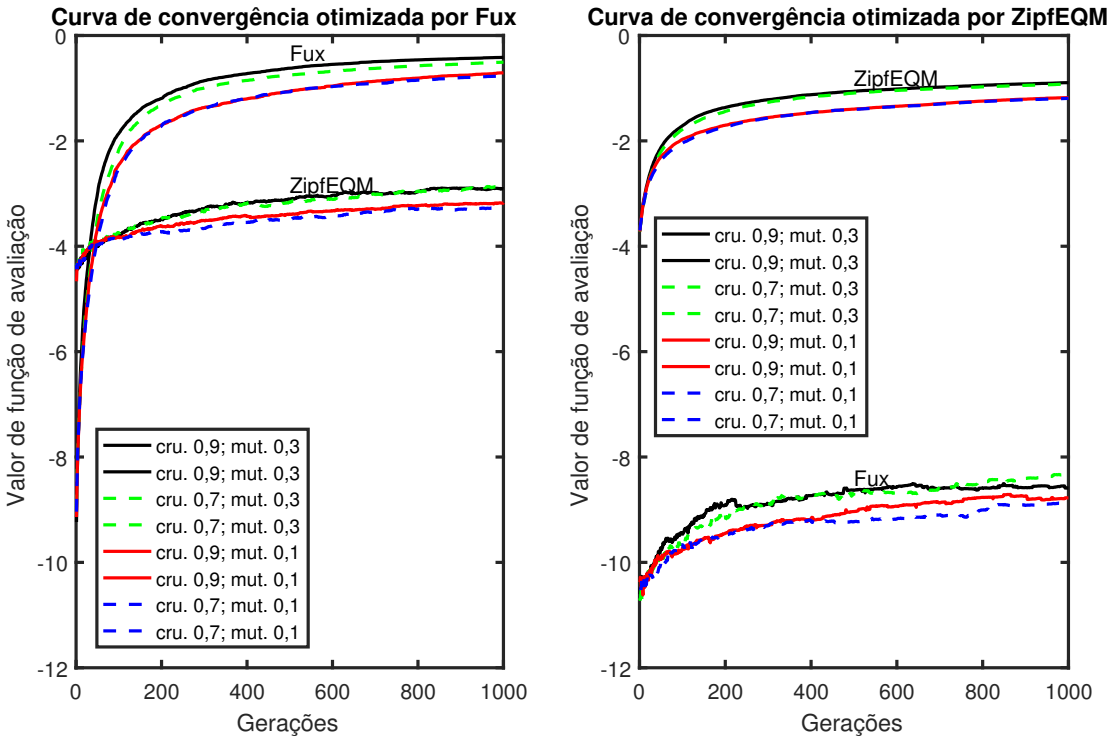


Figura 13 – Curva de convergência de 33 execuções dos métodos mono-objetivos.

Outro ponto importante é que os valores médios de avaliação de uma população evoluída pelo método 1 avaliados pela função ZipfEQM, não apresentam bons resultados para função baseada na lei de Zipf. O mesmo acontece com os valores de uma população evoluída pelo método 2 quando é avaliada pela função Fux. Isso quer dizer que otimizar a função de avaliação Fux, não otimiza simultaneamente a função de avaliação ZipfEQM. Ou seja, implementar um algoritmo multiobjetivo que otimiza ambas funções simultaneamente pode ser uma boa abordagem para tratar este problema de composição algorítmica.

A respeito das taxas de cruzamento e mutação, podemos ver que a taxa de mutação $0{,}3$ obteve os dois melhores valores médios finais para ambos métodos mono-objetivos e a taxa de cruzamento $0{,}9$ obteve melhor resultado para o método 1. Para o método 2 não é possível visualizar graficamente se a taxa de cruzamento $0{,}9$ é melhor ou pior que a taxa $0{,}7$. Apesar do experimento gráfico nos mostrar os melhores valores finais, é necessária uma análise estatística dos dados para evidenciar se realmente existe uma diferença entre os parâmetros escolhidos para taxas de cruzamento e mutação.

A técnica mais apropriada para analisar a influência das taxas de cruzamento e mutação nos métodos mono-objetivos é a análise de variância com 2 fatores (ANOVA-2), que propõe um modelo linear para os dados desde que as premissas de normalidade (os valores proveêm de uma população que segue a distribuição normal), homocedasticidade (os grupos possuem igualdade de variâncias) e independência dos resíduos (não existe

correlação entre os erros de medição) sejam satisfeitas. Em todos os testes procuramos o valor crítico de 95% de confiabilidade.

Utilizando o software R realizamos o teste de Shapiro-Wilk para normalidade das populações contendo os resultados das 33 execuções dos métodos 1 e 2, encontrando o p-valor $2{,}62X10^{-11}$ e $0{,}006$, ou seja, rejeitamos a hipótese de normalidade para ambas as amostras. Para prosseguir com os testes estatísticos utilizamos uma transformação chamada *aligned rank transformation*, proposta em (WOBBROCK et al., 2011), em que o objetivo é transformar os dados em uma forma não paramétrica para que a ANOVA possa ser realizada sem a premissa de normalidade. Contornando a necessidade de normalidade, realizamos os testes de Breusch-Pagan para homocedasticidade e de Durbin-Watson para independência dos resíduos para ambos os métodos.

Para homocedastididade encontramos o p-valor $0{,}111$ para o método 1 e $0{,}039$ para o método 2. O p-valor encontrado para o método 1 é suficiente para garantir a homocedasticidade, porém o p-valor do método 2 extrapolou o valor crítico, gerando evidências de que parte da variância é explicada pelos fatores. Apesar disso, dado que a premissa de homocedasticidade necessária é uma premissa fraca para ANOVA, prosseguiremos com os testes tendo em mente a possibilidade de não homocedasticidade do método 2.

Para a independência dos resíduos encontramos o p-valor $0{,}412$ tanto para o método 1 quanto para o método 2, ou seja, a independência dos resíduos está garantida. As Tabelas 7 e 8 mostram as tabelas ANOVA para os métodos 1 e 2 respectivamente. Em ambas tabelas **GL** é o grau de liberdade dos fatores, **GLR** o grau de liberdade dos resíduos, **Valor F** é o valor da estatística de teste, **P-valor** é o próprio p-valor, **SQ** é a soma dos quadrados dos fatores, e **SQR** a soma dos quadrados dos resíduos. **Cru** é o fator taxa de cruzamento, **Mut** é o fator taxa de mutação e **Cru*Mut** é a interação entre os fatores.

Podemos ver nos dois métodos que apenas o fator taxa de mutação foi considerado influente na otimização mono-objetivo. Como temos apenas dois níveis (taxas de mutação $0{,}1$ e $0{,}3$), podemos inferir pelo gráfico da Figura 13 que a taxa de mutação $0{,}3$ alcança resultados melhores em ambos os métodos. Nada se pode afirmar a respeito da influência da taxa de cruzamento.

Tabela 7 – Tabela ANOVA do método 1

	GL	**GLR**	**Valor F**	**P-Valor**	**SQ**	**SQR**
Cru	1	128	0,15	0,69	218,94	185017
Mut	1	128	72,97	0,00	68637,12	120399
Cru*Mut	1	128	0,44	0,50	619,67	179862

Tabela 8 – Tabela ANOVA do método 2

	GL	**GLR**	**Valor F**	**P-Valor**	**SQ**	**SQR**
Cru	1	128	0,11	0,91	17,45	190455
Mut	1	128	200,82	0,00	117006,81	74576
Cru*Mut	1	128	0,08	0,77	124,12	191116

5.2.2 Experimentos com Métodos Multiobjetivos

Para analisar o comportamento dos métodos 3 e 4, que utilizam a abordagem multiobjetivo, plotamos para cada combinação de parâmetros utilizada nos métodos mono-objetivos (taxas de cruzamento $0{,}7$ e $0{,}9$ e mutação $0{,}3$ e $0{,}1$) um gráfico que mostra o conjunto pareto-ótimo combinado de 33 execuções dos métodos multiobjetivo. O conjunto pareto-ótimo combinado consiste em tomar o conjunto pareto-ótimo de cada uma das 33 execuções, inserí-los em uma mesma população e executar uma iteração do NSGA-II para obter um novo conjunto pareto-ótimo.

Os gráficos das Figuras 14, 15, 16 e 17, mostram o pareto-ótimo combinado para 33 execuções dos métodos 3 e 4. Cada ponto no gráfico representa um indivíduo otimizado diferente. Em todas as figuras, o eixo x representa o valor da função de avaliação de Fux do indivíduo e o eixo y o valor da função de avaliação ZipfEQM. Os pontos exibidos em x vermelho são indivíduos do conjunto pareto-ótimo combinado do método 4, os asteriscos azuis são os indivíduos do método 3. As cruzes verdes são os 33 melhores indivíduos otimizados no método 2 e as cruzes pretas os 33 melhores indivíduos otimizados pelo método 1.

O primeiro ponto importante que podemos observar é o impacto da taxa de mutação encontrado nas Figuras 14 e 16. Podemos ver que uma taxa de mutação mais alta faz com que o pareto-ótimo combinado do método 3 fique mais próximo do pareto-ótimo combinado do método 4. Além disso, os valores finais alcançados aparentam ser melhores do que com a taxa de mutação $0{,}1$, principalmente quando se trata dos valores da função Fux. Outra observação importante é que utilizar uma população inicial contendo os melhores indivíduos da otimização mono-objetivo faz com que o pareto-ótimo combinado do método 4 contenha uma diversidade maior de indivíduos em relação aos valores da função de avaliação Fux. A possível causa dessa diversidade é que é mais fácil aumentar o valor da função de avaliação Fux de um indivíduo já otimizado por ZipfEQM do que o contrário. Além disso, como previsto, o pareto-ótimo do método 4 domina todos os outros indivíduos dos métodos 1, 2 e 3. Por se tratar de uma otimização mono-objetivo, esperamos que um indivíduo otimizado pelos métodos 1 e 2 sejam sempre melhores em suas respectivas funções de avaliação do que indivíduos otimizados em um método multiobjetivo. Caso isso ocorra, inserir esses melhores indivíduos como população inicial para o método 4, sempre acarretará em uma dominância do método 4 em relação aos outros. Podemos concluir que numericamente o método 4 sempre gerará indivíduos melhores em relacão a ambas funções simultaneamente do que

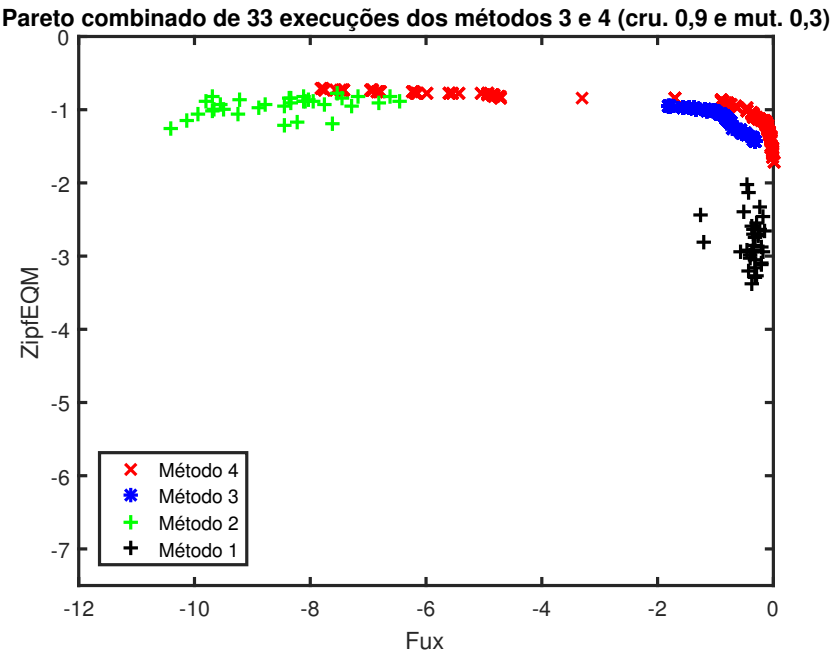


Figura 14 – Pareto combinado dos métodos 3 e 4. O método multiobjetivo 4 é representado pelos pontos em formato de cruz vermelha e o método multiobjetivo 3 é representado pelos pontos em formato de asteriscos azuis. As cruzes verdes e pretas representam os 33 melhores indivíduos dos métodos mono-objetivos 1 e 2 respectivamente.

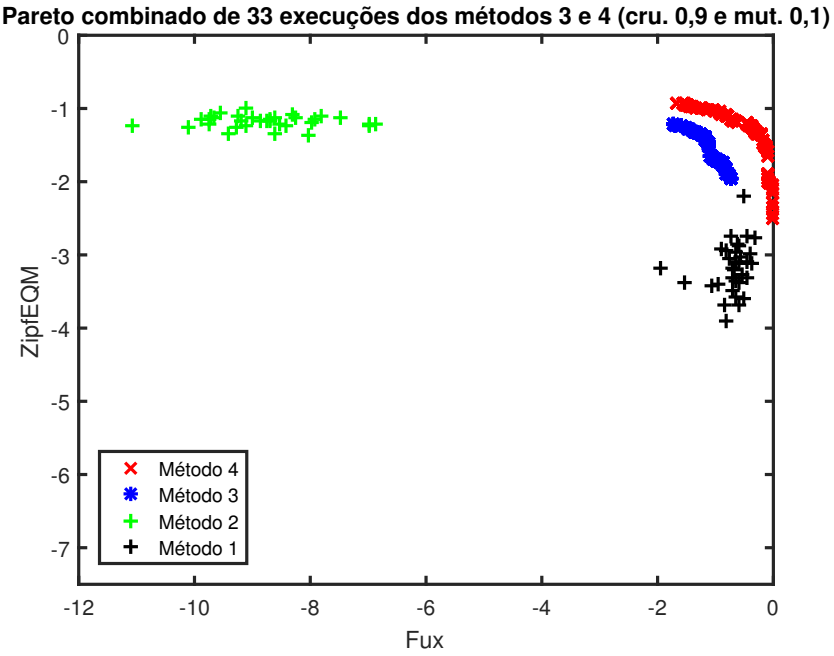


Figura 15 – Pareto combinado dos métodos 3 e 4. Legenda análoga a Figura (14).

os outros métodos.

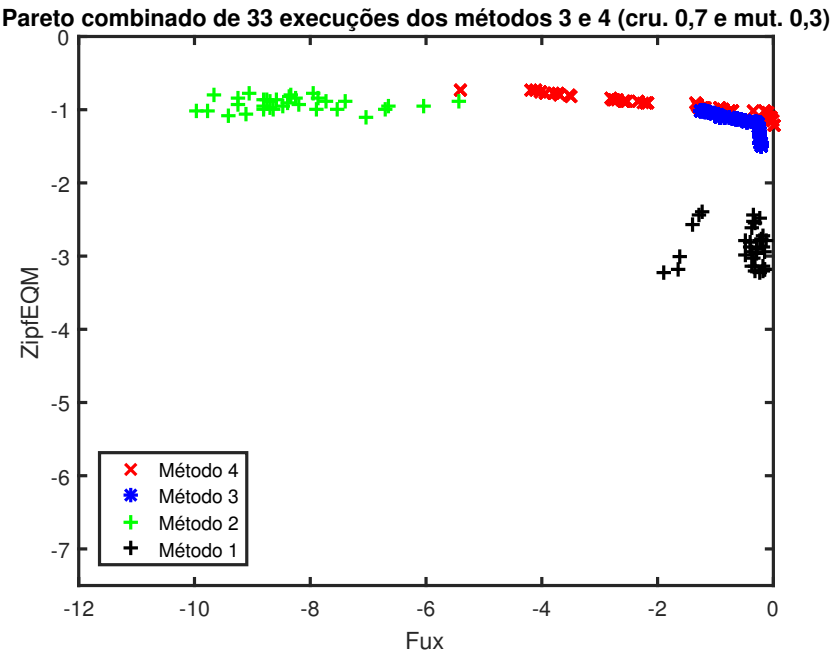


Figura 16 – Pareto combinado dos métodos 3 e 4. Legenda análoga a Figura (14).

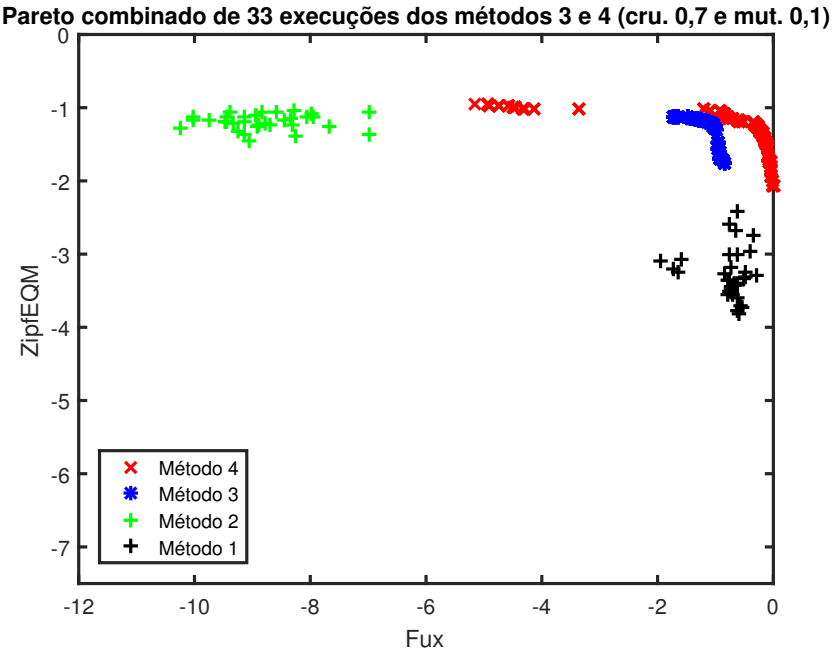


Figura 17 – Pareto combinado dos métodos 3 e 4. Legenda análoga a Figura (14).

5.2.3 Avaliação de Músicas Reais

Além dos experimentos numéricos anteriores, realizamos um experimento para verificar se as funções de avaliação utilizadas avaliam bem melodias monofônicas compostas por músicos. A Tabela 9 mostra os valores de funções de avaliação para 3 trechos de melodias de peças compostas por músicos. As melodias avaliadas foram "Ambos"de José

Aurélio Garcia (extraída do site *Classical Archieve* (SCHWOB, 1994)), "Le Carnaval des Animaux, 10. Voliere"de Saint Säens (extraída do site *Kunst Der Fuge* (KUNSTDERFUGE, 2002)) e "Sonnerie de Ste. Genevieve du Mont de Paris"de Marin Marais (extraída do site *Classical Archieve*.

Tabela 9 – Músicas compostas por músicos avaliadas por Fux e ZipfEQM

	Fux	ZipfEQM
Ambos	-8,606	-6,301
Le Carnaval des Animaux, 10. Voliere	-4,383	-5,978
Sonnerie de Ste. Genevieve du Mont de Paris	-5,460	-3,195

O valor de função de avaliação da peça "Ambos"encontra-se dentro do intervalo de valores de avaliação encontrados para melodias aleatórias mostrado na Figura 12. Já a peça "Le Carnaval des Animaux, 10. Voliere"possui valor de função de avaliação Fux fora desse intervalo e ZipfEQM dentro do intervalo. A peça "Sonnerie de Ste. Genevieve du Mont de Paris"possui os dois valores de avaliação fora do intervalo.

Dado que as peças escolhidas para serem avaliadas são peças de música clássica era esperado que todas elas estivessem fora do intervalo para função Fux. Não observamos isto apenas para peça "Ambos". Como os trechos foram escolhidos arbitrariamente e que não necessariamente essas peças foram compostas em Dó maior, os valores de função de avaliação Fux podem ser penalizados pelas regras de valores binários utilizadas (regras das equações (14) e (15)).

Já para a função ZipfEQM a explicação mais plausível é que as peças escolhidas não possuem eventos musicais que seguem bem a distribuição de Zipf. Apesar de a função ZipfEQM não ter encontrado bons valores para melodias compostas por músicos, esse resultado reforça o fato de que as funções de avaliação em questão não são harmônicas entre si, justificando uma abordagem multiobjetivo.

5.3 Resultados Qualitativos

Dado que uma melodia é um elemento subjetivo, isto é, sua qualidade depende da opinião do ouvinte, torna-se interessante realizar uma análise qualitiva das composições algorítmicas. Dada essa necessidade, elaboramos um questionário para ser respondido por pessoas que escutam as melodias compostas. A partir dessas respostas, faremos os testes estatísticos para verificar a qualidade das melodias compostas. Além disso, elaboramos uma métrica denominada coeficiente de criatividade. Esta métrica mede a qualidade de um método de composição algorítmica baseada nas respostas do questionário elaborado.

5.3.1 Elaboração do Questionário

O questionário elaborado possui uma estrutura simples, nós pedimos as pessoas que dessem uma nota de 1 a 10 para 3 melodias diferentes, em que uma nota 1 significa que a melodia escutada é desagradável e uma nota 10 denota uma melodia agradável. Cada uma das 3 melodias eram selecionadas aleatoriamente de 3 diferentes grupos.

A primeira melodia vinha de um grupo de 3 melodias que eram compostas aleatoriamente, ou seja, um aglomerado de notas sem nenhum significado.

O segundo grupo de melodias é o grupo formado por melodias compostas por métodos de composição algorítmica que serão validados pela análise do questionário.

O terceiro grupo continha trechos da peça "Ambos"de José Aurélio Garcia (extraída do site *Classical Archieve* (SCHWOB, 1994)), "Le Carnaval des Animaux, 10. Voliere"de Saint Säens (extraída do site *Kunst Der Fuge* (KUNSTDERFUGE, 2002)) e "Sonnerie de Ste. Genevieve du Mont de Paris"de Marin Marais (extraída do site *Classical Archieve* (SCHWOB, 1994)). Os trechos escolhidos possuem duração próxima das durações das melodias compostas. Esses compositores foram selecionados por não serem muito populares no país de origem do autor deste trabalho, propiciando assim um teste cego para quem responde o questionário. Para que o teste estatístico não seja enviesado, nós alteramos os instrumentos de todos os trechos para "piano".

Adicionalmente, perguntamos às pessoas que informassem qual seu nível de instrução musical entre os 6 possíveis níveis abaixo:

- Sou músico profissional;
- Faço faculdade de música;
- Sou músico amador;
- Participo ou já participei ocasionalmente de algum grupo musical;
- Já toquei um instrumento há muito tempo;
- Não sei nada sobre música.

Os dois primeiros níveis representam pessoas com um alto conhecimento musical, os dois do meio representam pessoas com um conhecimento musical moderado e os dois últimos, pessoas com baixo conhecimento musical.

Após a elaboração dos questionários, realizamos análises estatísticas de duas formas distintas. Primeiramente realizamos as estatísticas com o segundo grupo de melodias contendo melodias compostas pelos 4 métodos propostos nesse trabalho. Em seguida, escolhemos o método 4 e selecionamos melodias em Dó maior e livres, compostas com os operadores de mutação *mutateMelodicAndRhythmTrigram* e *mutateCopyingPartMusic* para verificar a diferença na percepção desse tipo de composição pelas pessoas. A escolha do método 4 foi feita por ter apresentado bons resultados nos questionários e por ter a melhor qualidade audível segundo opinião dos autores.

5.3.2 Coeficiente de Criatividade

Além da abordagem qualitativa baseada em questionários, elaboramos neste trabalho uma métrica que denota a quantidade de criatividade contida em um método de composição algorítmica. Esse método consiste em verificar a distância que as médias de avaliação das respostas do questionário das melodias compostas estão das médias de avaliação das melodias aleatórias e das médias das melodias reais. O método é calculado pela equação (31), em que M_{ca} representa a média de avaliação de um método de composição algorítmica, M_r é a média de avaliação das melodias reais, M_a é a média das melodias aleatórias e CC é o coeficiente de criatividade.

$$CC = \frac{M_{ca} - M_a}{M_r - M_a} \tag{31}$$

Um resultado esperado para um método de composição algorítmica é que quando uma pessoa avalia uma música proveniente desse método, esta deve ser avaliada com uma nota melhor que uma melodia aleatória e menor ou igual que uma melodia real. Se isso ocorrer, o valor de CC sempre estará entre 0 e 1. Quanto mais perto de 1 CC estiver, mais criativo o método é para compor melodias. Outro ponto importante a ser analisado é que como o coeficiente de criatividade é calculado em função de dados de questionários, CCs iguais não implicam em métodos iguais, assim como melodias com mesma nota não implicam em melodias iguais.

5.3.3 Estatísticas de Questionário dos 4 Métodos Iniciais

Para realizar as estatísticas dos questionários dos 4 métodos iniciais, escolhemos para representar a segunda melodia do questionário, melodias provenientes de um grupo de 12 melodias, 3 melodias compostas para cada um dos métodos propostos neste trabalho.

Para selecionar as melodias que compunham o segundo grupo, nós avaliamos por nós mesmos 5 melodias de cada método implementado com cada combinação de taxas de cruzamento e mutação descritas na seção 5.2, totalizando 20 melodias para cada método. Escolhemos para compor o segundo grupo as 3 melodias que soaram mais agradáveis aos ouvidos do autor e que eram provenientes das execuções com taxa de cruzamento igual a $0{,}7$ e mutação $0{,}3$. Esta combinação foi escolhida por obter maior média de avaliação durante o processo de avaliação dos autores.

Utilizamos os testes estatísticos de Kruskal-Wallis e Dunn contidos no software R para realizar esse experimento. Esses testes foram escolhidos porque nossa amostragem não possuia o mesmo número de réplicas para todos os grupos (46 respostas para o grupo com 3 melodias aleatórias, 46 respostas para os 4 grupos com melodias compostas e 46 respostas para o grupo com 3 melodias reais), necessitando então de técnicas não

paramétricas para realizar o teste. Nós separamos as respostas do questionário em 6 grupos:

- Grupo 1: Melodias aleatórias;
- Grupo 2: Melodias compostas pelo método 1;
- Grupo 3: Melodias compostas pelo método 4;
- Grupo 4: Melodias compostas pelo método 3;
- Grupo 5: Melodias compostas pelo método 2;
- Grupo 6: Trechos de melodias reais.

A Figura 18 mostra, para cada dos 6 grupos a quantidade de ocorrências de cada nota no questionário. Podemos ver que o grupo de melodias aleatórias possui maior concentração de notas baixas. As melodias compostas por músicos possuem maior concentração nas notas mais altas e as melodias compostas pelos 4 métodos possuem concentração de notas médias.

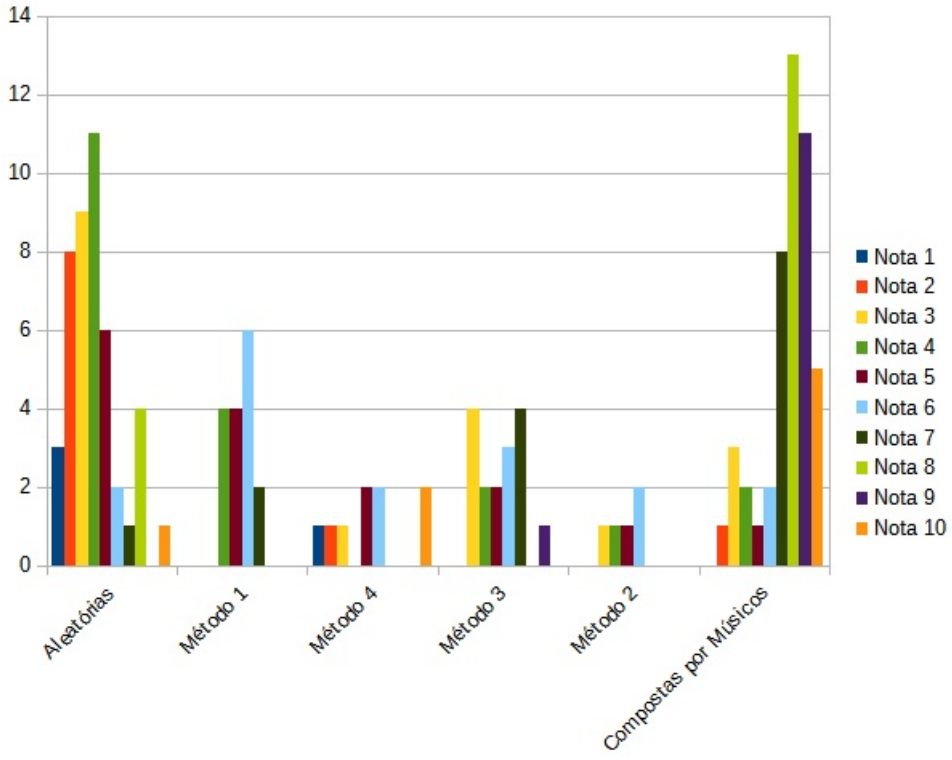


Figura 18 – Respostas dos questionários por grupos.

Após separar o quesionário em 6 grupos, nós realizaremos uma análise dos resíduos, que é uma técnica para verificar a adequabilidade de um teste estatístico, para ter certeza que esses testes podem ser aplicados. Na Figura 19 não distinguimos nenhum padrão na distribuição dos resíduos. Para conferir a independência dos resíduos, realizamos o teste de Durbin-Watson e encontramos o p-valor de $0{,}06$, ou seja, não atingimos o valor crítico de 95% que torna nossos resíduos independentes. Portanto, podemos aplicar os testes de

Kruskall-Wallis e Dunn.

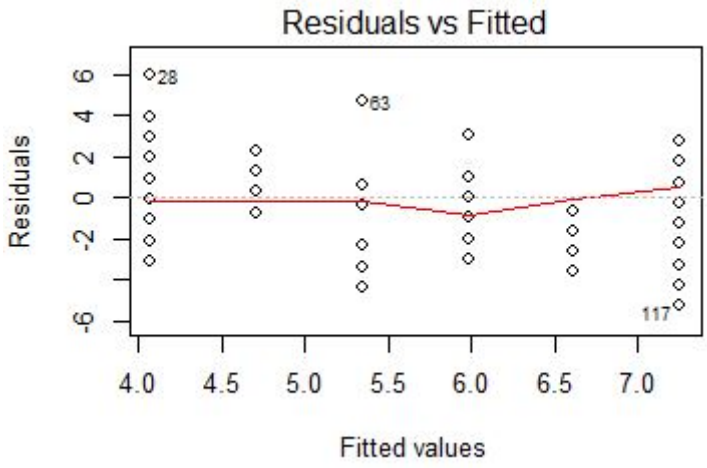


Figura 19 – Resíduos amostrais das respostas do questionário para os métodos 1, 2, 3, e 4.

Garantindo a independência dos resíduos, nós utilizamos também o software R para realizar os testes de Kruskall-Wallis e de Dunn. Os testes são exibidos na Tabela 10. Para o teste de Kruskall Wallis encontramos o p-valor 0, isto indica que existem evidências estatísticas de que existe diferenças entre os grupos estudados. O teste de Dunn nos mostra que os grupos 2, 3 e 4 vêm de uma população diferente do grupo 1. Entretanto, não existem evidências que o grupo 5 é diferente do grupo 1. Também foi detectado pelo teste que todos os grupos são estatisticamente diferentes do grupo 6 (grupo de melodias reais) e que não há evidências estatísticas de que os métodos propostos são distintos entre si.

Tabela 10 – Testes de Dunn para comparação múltipla dos grupos das respostas do questionário para os métodos 1, 2, 3 e 4.

	1	**2**	**3**	**4**	**5**
2	0,0205				
3	0,0710	0,4449			
4	0,0325	0,4355	0,4997		
5	0,2397	0,3059	0,3584	0,3463	
6	0,0000	0,0021	0,0073	0,0011	0,0102

Podemos inferir desses resultados que 3 dos métodos propostos adquiriram uma qualidade musical que os diferencia de uma melodia aleatória. Porém, nenhum deles possui qualidade musical suficiente para se equiparar a uma melodia composta por um compositor real. O método 2, otimização mono-objetivo com função de avaliação ZipfEQM, não possui qualidade musical suficiente para ser diferenciado de uma melodia aleatória.

Após realizada os testes de comparações múltiplas, calculamos os coeficientes de criatividade para os 4 métodos propostos e encontramos os respectivos valores, com 3 dígitos de significância.

- Método 1: $0{,}389$
- Método 2: $0{,}224$
- Método 3: $0{,}371$
- Método 4: $0{,}377$

A partir dos coeficientes, podemos dizer que o método 1 tem $38{,}9\%$ de criatividade comparado a um compositor humano, o método 3, $37{,}1\%$ e o método 4, $37{,}7\%$. O método 2 possui $22{,}4\%$ de criatividade, porém o teste de Dunn não detectou diferenças estatísticas entre melodias aleatórias e melodias compostas por este método.

Adicionalmente à análise dos questionários, fizemos um breve experimento a fim de identificar a influência do nível de instrução musical nas notas dadas para as melodias contidas no questionário. A Tabela 11 mostra as médias de avaliação para cada grupo de questionário (melodias aleatórias, compostas e reais) agrupadas por nível de instrução musical. Optamos por agrupar todas as notas referentes a melodias compostas porque a quantidade de avaliações por grupo é muito pequena. Isto se deve a pequena quantidade de pessoas com alto nível de instrução musical que respondeu ao questionário (apenas 1 pessoa).

Tabela 11 – Influência da instrução musical na avaliação de melodias dos métodos 1, 2, 3 e 4.

	Melodia Aleatória	Composição Algorítmica	Melodia Real
Pouca instrução	$4{,}057$	$5{,}212$	$7{,}229$
Instrução moderada	$4{,}100$	$5{,}000$	$8{,}300$
Muita Instrução	$2{,}000$	$4{,}000$	$9{,}000$

Podemos visualizar uma tendência de que quanto maior o nível de instrução musical, maior a avaliação de uma música real. Em relação a melodias aleatórias e compostas, não há diferença visual significante em relação aos níveis de instrução baixo e moderado. Quanto ao nível de instrução alto, parece haver uma queda nas médias de avaliação de melodias aleatórias e compostas, porém, como apenas 1 amostra de avaliação foi feita por esse nível, afirmar qualquer tendência pode ser um equívoco. Dada essa escassez de amostras, não pode ser realizado um teste estatístico para comparação de médias, porém, apesar dessa análise ser superficial, ela nos fornece a hipótese de que o nível de instrução musical pode impactar nas avaliações.

5.3.4 Estatísticas de Questionário com Outras Mutações e em Dó Maior

Dados os resultados obtidos com a análise do questionário da seção anterior, escolhemos um dos 4 métodos iniciais para realizar composições com outro operador de mutação e composições em Dó maior. Apesar do método 1 possuir um coeficiente de criatividade

maior, escolhemos o método 4 pois ele representa uma abordagem multiobjetivo elaborada exclusivamente neste trabalho. Para realizar esses experimentos, elaboramos novamente um questionário com a mesma estrutura que o anterior. Os grupos de melodias aleatórias e reais foram os mesmos. Porém o segundo grupo era formado por 3 melodias compostas pelo método 4, 3 pelo método 4 com o operador de mutação *mutateCopyingPartMusic*, e 6 composições de ambos em Dó maior. O método de escolha das 3 melodias de cada método foi o mesmo utilizado na seção anterior, ou seja, melhor nota na avaliação feita pelo autor. Utilizamos novamente os testes de Kruskall-Wallis e Dunn para realizar esse experimento, porém dessa vez adquirimos 62 respostas para nosso questionário, separadas pelos seguintes grupos:

- Grupo 1: Melodias aleatórias;
- Grupo 2: Melodias compostas pelo método 4;
- Grupo 3: Melodias compostas pelo método 4 com mutação mutateCopyingPartMusic em Dó maior;
- Grupo 4: Melodias compostas pelo método 4 em Dó maior;
- Grupo 5: Melodias compostas pelo método 4 com mutação mutateCopyingPartMusic;
- Grupo 6: Trechos de melodias reais.

De forma análoga a Figura 18, a Figura 20 mostra, para cada dos 6 grupos a quantidade de ocorrências de cada nota no questionário. Podemos perceber a mesma tendência do questionário anterior, em que o grupo de melodias aleatórias possui concentração de notas baixas. As melodias compostas por músicos possuem concentração nas notas mais altas e as melodias compostas pelos 4 métodos possuem concentração nas notas médias.

Verificamos a premissa de independência dos resíduos utilizando o teste de Durbin-Watson para execução do teste de Kruskall-Wallis e encontramos o p-valor $0{,}2517$, ou seja, o teste não rejeitou a hipótese de independência. O gráfico dos resíduos pode ser visto na Figura 21, em que não visualizamos nenhum padrão em sua distribuição, portanto realizamos o teste de Kruskall-Wallis, cujo p-valor encontrado foi 0, e de Dunn exibido na Tabela 12.

Tabela 12 – Teste de Dunn para comparação múltipla dos grupos das respostas dos questionários para melodias compostas em Dó maior e com diferentes operadores de mutação.

	1	**2**	**3**	**4**	**5**
2	0,0271				
3	0,0381	0,4837			
4	0,0657	0,2547	0,2794		
5	0,0096	0,3998	0,3871	0,1641	
6	0,0000	0,0033	0,0473	0,0000	0,0059

O resultado obtido com o teste de comparação múltipla foi semelhante ao obtido

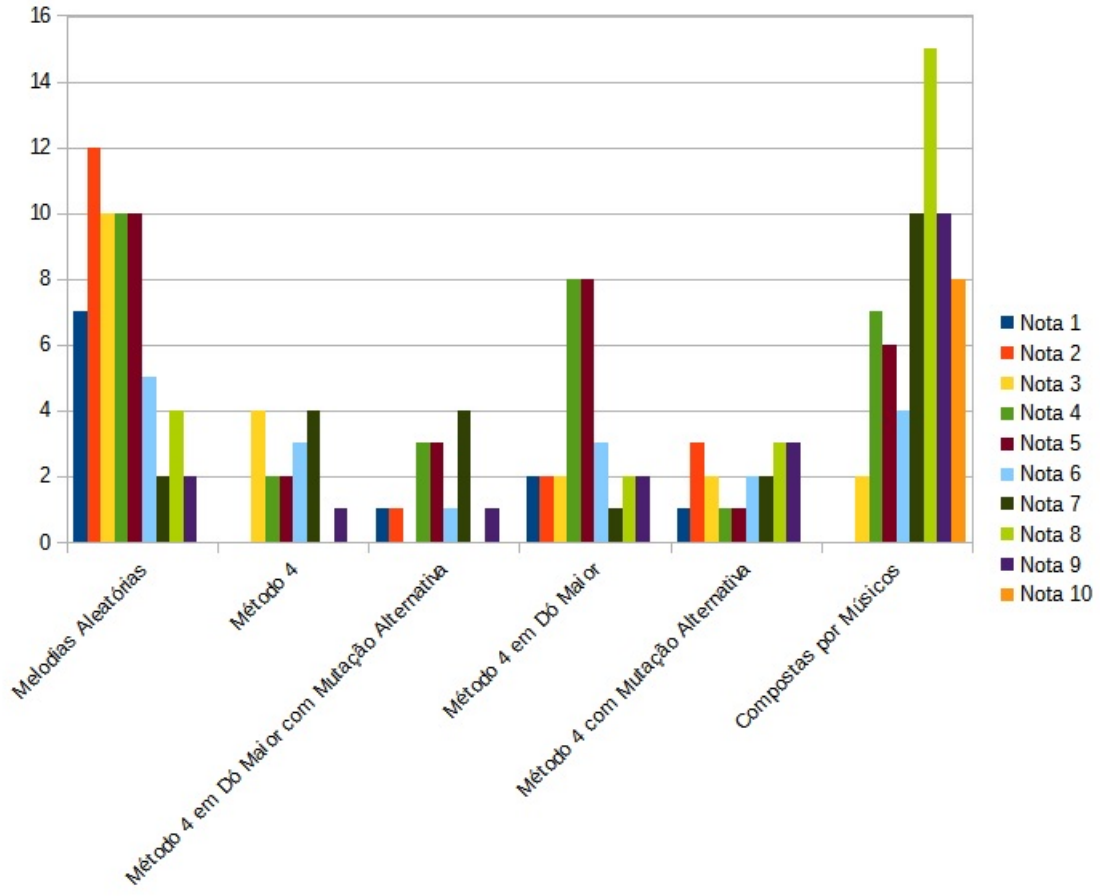


Figura 20 – Respostas dos questionários por grupos.

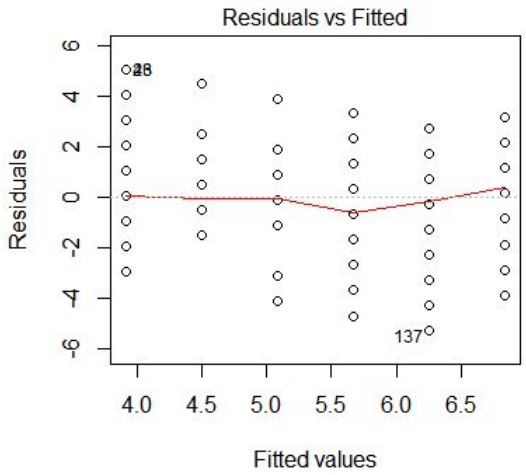


Figura 21 – Resíduos amostrais das respostas do questionário para melodias compostas em Dó maior e com diferentes operadores de mutação.

com as estatísticas do questionário da seção anterior. Todos os métodos de composição algorítmica implementados são estatísticamente diferentes de uma melodia aleatória, com exceção do método representado pelo grupo 4, melodias compostas pelo método 4 em dó maior. Mas, apesar de não ter atingido o valor crítico de 95%, seu p-valor é $0{,}0657$, muito

próximo ao valor crítico e muito menor que o p-valor $0{,}2397$ encontrado para o método 2, ou seja, podemos dizer que esse método possui qualidade musical suficiente para se distinguir de uma composição aleatória. Além disso, todos os métodos continuam estatiscamente diferentes do grupo de melodias reais.

Após realizados os testes de comparação múltiplas, calculamos os coeficientes de criatividade para cada parametrização diferente do método 4. Encontramos os respectivos valores, com 3 dígitos de significância.

- Método 4: $0{,}416$
- Método 4 com outra mutação: $0{,}386$
- Método 4 em dó maior: $0{,}249$
- Método 4 com outra mutação em dó maior: $0{,}473$

Como as médias para as melodias aleatórias e reais foram calculadas a partir das respostas do segundo questionário, obtivemos uma leve diferença no coeficiente de criatividade do método 4, que era $0{,}389$ e passou a ser $0{,}416$. Essa mudança já era prevista pois CC é calculado a partir das notas obtidas nos questionários, e estas podem variar dependendo de quem escuta. De qualquer forma, o valor de $41{,}6\%$ de criatividade permanece próximo ao valor anterior. A utilização do operador de mutação *mutateCopyingPartMusic* resultou num coeficiente de criatividade de $38{,}6\%$. Como o valor é muito próximo do coeficiente do método 4, preferimos não concluir nada a respeito da eficácia desse operador.

Ao compor melodias apenas com notas contidas na escala de dó maior, verificamos uma queda no coeficiente de criatividade do método 4, $24{,}9\%$. Esse resultado foi uma surpresa, pois restringir as possíveis notas a serem geradas para notas na escala de Dó maior deveria facilitar o processo de composição algorítmica. Apesar de concedida esta facilidade, uma melodia com apenas notas de uma determinada escala pode se tornar monótona aos ouvidos de uma pessoa. Por exemplo, uma mesma sequência de notas sendo repetidas indefinidamente pode ser desagradável aos ouvidos, mesmo estando dentro de uma escala. O mesmo não aconteceu quando compusemos melodias em dó maior utilizando o operador *mutateCopyingPartMusic*, que possui coeficiente de criatividade $47{,}3\%$. Podemos inferir que o operador de mutação em questão foi capaz de organizar melhor as notas da escala de dó maior de forma a produzir uma melodia mais agradável para as pessoas.

Também verificamos superficialmente a influência do nível de instrução musical na avaliação das melodias para as parametrizações do método 4. A Tabela 13 mostra as médias de avaliação agrupadas por nível de isntrução, assim como análise feita na subseção anterior, incluindo os critérios de agrupamento.

Nesse experimento, pessoas com nível de instrução moderado avaliaram melhor

Tabela 13 – Influência da instrução musical na avaliação de melodias compostas em Dó maior e com diferentes operadores de mutação

	Melodia Aleatória	Composição Algorítmica	Melodia Real
Pouca instrução	$3{,}926$	$5{,}219$	$7{,}185$
Instrução moderada	$4{,}136$	$4{,}769$	$7{,}500$
Muita Instrução	$-$	$-$	$-$

as melodias aleatórias do que pessoas com pouca instrução. Esse resultado foi desfavorável ao experimento análogo anterior, cujo resultado apresentara o oposto. Dado que a diferença entre as médias é muito pequena, a melhor conclusão possível é afirmar que não há influência do nível de instrução na avaliação de melodias aleatórias. O mesmo vale para melodias compostas algoritmicamente. Apesar do resultado obtido ser favorável ao experimento análogo anterior, a diferença encontrada foi muito pequena $(0{,}45)$ para afirmar qualquer influência. Porém para validar essa conclusão precisaríamos de uma amostragem maior de avaliações musicais, pois novamente nenhuma pessoa com nível de instrução alto respondeu ao questionário.

Para avaliação de melodias reais, a diferença entre as médias de pessoas com poucas instrução e instrução moderada foi $0{,}315$, um valor considerado baixo, portanto não afirmamos nada sobre a influência da instrução musical em sua avaliação.

Dado que os experimentos com instrução musical não possuem amostras com pessoas de alto nível de instrução, deixaremos essa questão em aberto para um possível trabalho futuro. Acreditamos que realizar um experimento mais detalhado com apenas dois níveis de instrução não nos fornecerá uma visão realística dessa influência.

6 Conclusão

Neste trabalho estudamos diferentes técnicas de composição algorítmica e utilizamos uma abordagem multiobjetivo que combina regras e proporções para realizar a tarefa de composição de melodias monofônicas, cuja utilização na literatura é inédita. Propusemos pela primeira vez neste trabalho uma função de avaliação baseada na lei de Zipf em que o objetivo é minimizar o erro quadrático médio de uma regressão linear. A nova função de avaliação, apesar de não ter conseguido resultados satisfatórios para compor melodias em uma abordagem mono-objetivo, se mostrou muito importante para auxiliar a otimização multiobjetivo no processo de compor melodias que satisfaçam tanto as regras de Fux como a lei de Zipf.

A função de avaliação baseada nas regras de Fux neste trabalho já foi amplamente utilizada, portanto sua utilização em uma otimização mono-objetivo não caracteriza uma nova abordagem. Entretanto, sua utilização em uma abordagem multiobjetivo, combinada com a função de avaliação baseada na lei de Zipf, pavimentou a construção de dois métodos (métodos 3 e 4 do capítulo 5) inovadores para tratar o problema em questão. Os 4 métodos implementados neste trabalho compõem melodias explorando um espaço de notas musicais de 3 oitavas e não utiliza nenhuma técnica de pós-processamento (métodos para melhorar a qualidade do som após a composição das melodias) para melhorar as composições. Dos 3 métodos inovadores criados neste trabalho, os 2 multiobjetivos possuem evidências estatísticas suficientes que os diferenciam de uma composição completamente aleatória, o que nos permite concluir que obtivemos sucesso em prover um comportamento criativo a um método de composição algorítmica.

Em adição aos métodos implementados, nós também propusemos pela primeira vez, uma metodologia qualitativa de validação de métodos de composição algorítmica baseada na análise estatística das respostas de questionários elaborados neste trabalho. Dado que a qualidade de uma melodia depende de critérios subjetivos, como a opinião de um ouvinte, essa abordagem é uma grande contribuição para comunidade de pesquisa da área. Além da metodologia baseada em questionários, criamos uma métrica que mede a quantidade de criatividade de um método de composição algorítmica denominada coeficiente de criatividade, que quando, amparado por testes estatísticos, pode medir a distância entre um método de composição algorítmica, uma composição original e uma composição aleatória. Apesar dessa metodologia possuir falhas e ser suscetível a críticas, esperamos que ela possa ser melhorada no futuro e se torne amplamente utilizada na avaliação de métodos de composição algorítmica.

6.1 Trabalhos Futuros

Apesar da composição algorítmica ser pesquisada desde os anos 50, o surgimento e aprimoramento de novas tecnologias e técnicas de inteligência artificial sempre abrem espaço para novas descobertas. Dado que o uso de abordagens multiobjetivo é escasso na área, um trabalho futuro interessante seria combinar funções de avaliação baseadas no aprendizado de máquina, como redes neurais, em conjunto com regras musicais de Fux ou outras não utilizadas ainda dessa forma.

Muitos trabalhos da área codificam um indivíduo como um vetor de notas e apenas recentemente esse paradigma está sendo quebrado. Uma codificação de um tipo de indivíduo diferente, combinado a operadores genéticos apropriados para trabalhar com uma dada codificação, pode gerar resultados que não foram possíveis alcançar com as codificações tradicionais.

Outra opção de trabalho futuro seria atribuir características de gênero musical às composições. Por exemplo, utilizar métricas de reconhecimento para verificar se a melodia pertence a um estilo musical e adicionar esse reconhecimento a função de avaliação. Técnicas de pós-processamento também podem ser utilizadas para melhorar a qualidade de uma melodia composta por um método de composição algorítmica.

6.2 Considerações Finais

Este trabalho alcançou os objetivos descritos no capítulo 1 relacionados a implementação de um método multiobjetivo para compor melodias e de sua validação baseada em respostas de questionários. A função de avaliação baseada na lei de Zipf proposta não conseguiu gerar sozinha uma melodia considerada boa pelos ouvintes, porém sua utilização foi de grande importância para que os métodos multiobjetivos implementados pudessem proporcionar características de proporções musicais às melodias compostas.

Referências

ACEVEDO, A. G. Fugue composition with counterpoint melody generation using genetic algorithms. **Proceedings of the International Conference on Computer Music Modeling and Retrieval**, pp, p. 96—-106, 2004. Citado na página 14.

ALFONSECA, M.; CEBRIÁN, M.; ORTEGA, A. . Evolving computer-generated music by means of the normalized compression distance. **In Proceedings of the WSEAS International Conference on Simulation, Modelling and Optimization**, pp, p. 343—-348, 2005. Citado na página 14.

AMES, C. Automated composition in retrospect: 1956-1986. **Leonardo**, v. 20, p. 169–185, 1987. Citado na página 6.

AMES, C.; DOMINO, M. Understanding music with ai, chap cybernetic composer: an overview. **The MIT Press**, pp, p. 186–205, 1992. Citado na página 11.

ANDERS, T.; MIRANDA, E. R. Constraint programming systems for modeling music theories and composition. **ACM Computing Surveys**, v. 43, p. 30–38, 2011. Citado na página 9.

ARIZA, C. Two pioneering projects from the early history of computer-aided algorithmic composition. **Computer Music Journal**, v. 35, p. 40–56, 2011. Citado na página 6.

BARONI, M.; JACOBONI, C. Proposal for a grammar of melody: The bach chorales. **Les Presses de l'Université de Montréal**, 1978. Citado na página 8.

BELLINGER, E. Little ludwig, an evolutionary learning machine for musical composition. **Journal of the ACM**, v. 1, 2011. Citado na página 14.

BILES, J. A. Autonomous genjam: Eliminating the fitness bottleneck by eliminating fitness. **Proceedings of the 2001 Genetic and Evolutionary Computation**, pp, p. 337—-441, 1991. Citado na página 13.

BROOKS, F. P. et al. An experiment in musical composition. **Electronic Computers, IRE Transactions on**, v. 3, p. 175–182, 1957. Citado na página 1.

BROWNE, T. M.; FOX, C. Global expectation-violation as fitness function in evolutionary composition. **Proceedings of the Conference on Applications of Evolutionary Computation**, pp, p. 538–546, 2009. Citado na página 12.

BULLEY, J.; JONES, D. Genotator: an environment for exploring the application of evolutionary techniques in computer-assisted composition. **Organised Sound**, v. 4, p. 127–133, 1999. Citado na página 8.

BULLEY, J.; JONES, D. Variable 4: A dynamical composition for weather systems. **Proceedings of the International Computer Music Conference**, 2011. Citado na página 8.

CARPENTIER, G.; BRESSON, J. Interacting with symbol, sound, and feature spaces in orchidée, a computer-aided orchestration environment. **Computer Music Journal**, v. 34, p. 10–27, 2008. Citado na página 15.

COCA, A. E.; ROMERO, R. A. F.; ZHAO, L. Generation of composed musical structures through recurrent neural networks based on chaotic inspiration. **Proceedings of the International Joint Conference on Neural Networks**, pp, p. 3220–3226, 2011. Citado na página 12.

CUNHA, A. G.; TAKAHASHI, R.; ANTUNES, C. H. **Manual de Computação Evolutiva e Metaheurística**. Belo Horizonte: [s.n.], 2013. Citado na página 40.

DAHLSTEDT, P. Autonomous evolution of complete piano pieces and performances. **Proceedings of the European Conference on Artificial Life**, 2007. Citado na página 14.

DAVISMOON, S.; ECCLES, J. Combining musical constraints with markov transition probabilities to improve the generation of creative musical structures. **Proceedings of the European Conference on the Applications of Evolutionary Computation**, 2010. Citado na página 9.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm. **IEEE Transactions on evolutionary computation**, v. 6(2), p. 182–197, 2002. Citado 5 vezes nas páginas x, 41, 42, 44 e 45.

DELGADO, M.; FAJARDO, W.; MOLINA-SOLANA, M. Inmamusys: Intelligent multiagent music system. **Expert Systems with Applications**, 2008. Citado na página 10.

DONNELLY, P.; SHEPPARD, J. Evolving four-part harmony using genetic algorithms. **Applications of Evolutionary Computation**, pp, p. 273–282, 2011. Citado 2 vezes nas páginas 14 e 17.

EIGENFELDT, A.; PASQUIER, P. Realtime generation of harmonic progressions using controlled markov selection. **Proceedings of the International Conference on Computational Creativity**, 2010. Citado na página 10.

FERNÁNDEZ, J. D.; VICO, F. Ai methods in algorithmic composition: A comprehensive survey. **Journal of Artificial Intelligence Research**, v. 48, p. 513–582, 2013. Citado 3 vezes nas páginas vii, 7 e 8.

FIELD, E. elfridge. Beyond midi: The handbook of musical codes. **Cambridge, MA: MIT Press**, 1997. Citado na página 2.

FREITAS, A. R. R. de. **Redução de Dimensionalidade em Problemas com Muitos Objetivos: Uma Aplicação em Composição Algorítmica**. 145 f. Tese (Doutorado em Engenharia Elétrica) — Programa de Pos-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, Belo Horizonte, 2013. Citado 2 vezes nas páginas 15 e 17.

FREITAS, F. G. Alan de. Melody harmonization in evolutionary music using multiobjective genetic algorithms. **In Proceedings of the Sound and Music Computing Conference**, 2011. Citado 2 vezes nas páginas 15 e 17.

FU, T. yang et al. Evolutionary interactive music composition. **Genetic and Evolutionary Computation Conference**, p. 1863–1864, 2006. Citado na página 13.

FUX, J.; MANN, A. The study of counterpoint from johann joseph fux's gradus ad parnassum - 1725. **Norton, New York**, 1971. Citado 2 vezes nas páginas 3 e 18.

GEIS, M.; MIDDENDORF, M. Creating melodies and baroque harmonies with ant colony optimization. **International Journal of Intelligent Computing and Cybernetics**, v. 1, p. 213–218, 2008. Citado na página 15.

GILLICK, J.; TANG, K.; KELLER, R. M. Learning jazz grammars. **Proceedings of the Sound and Music Computing Conference**, pp, p. 125–130, 2009. Citado na página 8.

GWEE, N. **Complexity and Heuristics in Rule-Based Algorithmic Music Composition**. Tese (Ph.D. thesis) — Louisiana State University, Louisiana, 2002. Citado na página 9.

HARRIS, R.; FITCH, J. **Algorithmic Composition of Jazz**. April 2008. 130 f. Dissertação (BSc. Computer Science with Honours) — University of Bath, 2008. Citado 2 vezes nas páginas 14 e 17.

HARTMANN, P. Natural selection of musical identities. **Proceedings of the International Computer Music Conference**, 1990. Citado na página 13.

HERREMANS, D. **Compose equivalent Compute: Computer Generation and Classification of Music through Operations Research Methods**. Tese (Ph.D. thesis) — Universiteit Antwerpen, Antwerpen, 2014. Citado 11 vezes nas páginas ix, 1, 2, 3, 4, 22, 23, 24, 31, 36 e 47.

HERREMANS, D.; SORENSEN, K. Composing first species counterpoint with a variable neighbourhood search algorithm. **Journal of Mathematics and the Arts**, pp, p. 169–189, 2012. Citado na página 9.

HERREMANS, D.; SORENSEN, K. Composing fifth species counterpoint music with a variable neighborhood search algorithm. **Expert Systems with Applications**, v. 40(16), p. 6427–6437, 2013. Citado na página 9.

HERREMANS, D.; SORENSEN, K.; CONKLIN, D. Generating structured music using quality metrics based on markov models. **Working Paper 2014019 Faculty of Applied Economics, University of Antwerp**, 2014. Citado na página 11.

HILLER, L. A.; ISAACSON, L. M. Musical composition with a high-speed digital computer. **Journal of the Audio Engineering Society**, v. 6, p. 154–160, 1958. Citado na página 6.

HOLLAND, J. H. Adaptation in natural and artificial systems [s.l.]. **Cambridge, Massachusetts**, 1975. Citado na página 27.

HORNER, A.; GOLDBERG, D. E. Genetic algorithms and computer-assisted music composition. **Proceedings of the International Conference on Genetic Algorithms,**, pp, p. 337—-441, 1991. Citado na página 13.

JENSEN, J. H. **Evolutionary Music Composition: A Quantitative Approach**. April 2011. Dissertação (Master of Science in Computer Science) — Norweigian University of Science and Technology, Noruega, 2011. Citado 6 vezes nas páginas 14, 17, 19, 32, 33 e 34.

JOHANSON, B.; POLI, R. Gp-music: an interactive genetic programming system for music generation with automated fitness raters. **Proceedings of the Annual Conference on Genetic Programming**, pp, p. 181–186, 1998. Citado na página 12.

JOHNSON, M. D.; TAURITZ, D. R.; WILKERSON, R. W. Evolutionary computation applied to melody generation. **Proceedings of the Artificial Neural Networks in Engineering**, 2004. Citado na página 14.

JONES, K. Compositional applications of stochastic processes. **Computer Music Journal**, v. 5, p. 45–61, 1981. Citado na página 10.

KHALIFA, Y. M. et al. Evolutionary music composer integrating formal grammar. **Proceedings of the Genetic and Evolutionary Computation Conference**, pp, p. 2519—-2526, 2007. Citado 3 vezes nas páginas 8, 14 e 17.

KRAMANN, G. Darwinian pianos: Realtime composition based on competitive evolutionary process. **EvoMUSART**, pp, p. 37—-46, 2013. Citado 2 vezes nas páginas 14 e 17.

KUNSTDERFUGE. **Kunst Der Fuge**. 2002. Disponível em: <http://www.kunstderfuge.com>. Citado 2 vezes nas páginas 55 e 56.

LEWIS, J. P. Music and connectionism, chap. creation by refinement and the problem of algorithmic music composition. **The MIT Press**, 1991. Citado na página 12.

LIDOV, D.; GABURA, J. A melody writing algorithm using a formal language model. **Computer Studies in the Humanities and Verbal Behavior**, v. 4, p. 138–148, 1973. Citado na página 8.

LIU, C.-H.; TING, C.-K. Evolutionary composition using music theory and charts. **IEEE Symposium on Computational Intelligence for Creativity and Affective Computing**, pp, p. 63—70, 2013. Citado 2 vezes nas páginas 14 e 17.

LIU, C.-H.; TING, C.-K. Music pattern mining for chromosome representation in evolutionary composition. **Proceedings of the IEEE Congress on Evolutionary Computation**, pp, p. 2145–2152, 2015. Citado na página 15.

LIU, C.-H.; TING, C.-K. Computational intelligence in music composition: A survey. **IEEE Transactions on Emerging Topics in Computational Intelligence**, pp, p. 63—70, 2016. Citado na página 15.

LO, M.; LUCAS, S. M. Evolving musical sequences with n-gram based trainable fitness functions. **Proceedings of the IEEE Conference on Evolutionary Computation**, pp, p. 601–608, 2006. Citado na página 11.

LOPES, H. B.; SANTOS, V. F. dos; MARTINS, F. V. C. **https://soundcloud.com/henrique-barros-lopes/tracks**. 2016. Disponível em: <https://soundcloud.com/henrique-barros-lopes/tracks>. Citado na página 47.

MANARIS, B.; HUGHES, D.; VASSILANDONAKIS, Y. Monterey mirror: Combining markov models, genetic algorithms, and power laws. **Proceedings of the IEEE Conference on Evolutionary Computation**, 2011. Citado 2 vezes nas páginas 14 e 17.

MANARIS, B.; HUGHES, D.; VASSILANDONAKIS, Y. Harmonic navigator: A gesture-driven, corpus-based approach to music analysis, composition, and performance. **Artificial Intelligence and Interactive Digital Entertainment**, 2013. Citado 2 vezes nas páginas 11 e 15.

MANARIS, B. et al. Zipf's law, music classification, and aesthetics. 2005. Citado 2 vezes nas páginas 18 e 21.

MANARIS, B. et al. A corpus-based hybrid approach to music analysis and composition. **Proceedings of the AAAI National Conference on Artificial intelligence**, pp, p. 839–845, 2007. Citado na página 12.

MARTIN, A.; JIN, C. T.; BOWN, O. Implementation of a real-time musical decisionmaker. **Proceedings of the Australasian Computer Music Conference**, pp, p. 103–108, 2012. Citado na página 11.

MASON, S.; SAFFLE, M. L-systems, melodies and musical structure. **Leonardo Music Journal**, v. 4, p. 31–38, 1994. Citado na página 8.

MCINTYRE, R. A. Bach in a box: the evolution of four part baroque harmony using the genetic algorithm. **Proceedings of the IEEE Conference on Evolutionary Computation**, pp, p. 852–857, 1994. Citado na página 9.

MCKAY, C. **Automatic Genre Classification of MIDI Recordings**. June 2004. Dissertação (Master Thesis) — McGill University, Montreal, 2004. Citado na página 2.

MOORER, J. A. Music and computer composition. **Communications of the ACM**, v. 15, p. 104–113, 1972. Citado na página 10.

MORALES, E.; MORALES, R. Learning musical rules. **Proceedings of the International Joint Conference on Artificial Inteligence**, 1993. Citado na página 9.

MOZER, M. C. Music and connectionism, chap. connectionist music composition based on melodic, stylistic, and psychophysical constraints. **The MIT Press**, p. 195–211, 1991. Citado na página 12.

NELSON, G. L. Real time transformation of musical material with fractal algorithms. **Computers and Mathematics with Applications**, v. 1, p. 109–116, 1996. Citado na página 8.

NORDEN, H. **Fundamental Counterpoint**. Boston: Crescendo Publishing Co., 1969. Citado na página 21.

NORTH, T. A technical explanation of theme and variations: A computer music work utilizing network compositional algorithms. **Ex Tempore**, v. 5, 1991. Citado na página 11.

PACHET, F. Evolving musical sequences with n-gram based trainable fitness functions. **Proceedings of the International Conference on Music and Artificial Intelligence**, pp, p. 103–108, 2002. Citado na página 11.

PADBERG, H. A. **Computer-composed canon and free-fugue**. Tese (Ph.D. thesis) — Saint Louis University, St. Louis, 1964. Citado na página 6.

PAIEMENT, J.-F.; ECK, D.; BENGIO, S. Probabilistic melodic harmonization. **19th Conference of the Canadian Society for Computational Studies of Intelligence**, pp, p. 218–229, 2006. Citado na página 11.

PEARCE, M.; MEREDITH, D.; WIGGINS, G. Motivations and methodologies for automation of the compositional process. **Musicae Scientiae**, v. 6 (2), p. 119–147, 2002. Citado na página 1.

PECK, J. M. **Explorations in algorithmic composition: Systems of composition and examination of several original works**. Dissertação (Mestrado) — State University of New York, Oswego, 2011. Citado na página 8.

PEREIRA, F. et al. A connectionist approach to algorithmic composition. **Proceedings of the Conference on Computational Models of Creative Cognition**, 1997. Citado na página 10.

PESTANA, P. Lindenmayer systems and the harmony of fractals. **Chaotic Modeling and Simulation**, v. 1, p. 91–99, 2012. Citado na página 8.

PINKERTON, R. C. Information theory and melody. **Scientific American**, v. 194, p. 77–86, 1956. Citado na página 1.

PONSFORD, D.; WIGGINS, G.; MELLISH, C. Statistical learning of harmonic movement. **Journal of New Music Research**, v. 28 (2), p. 150–177, 1999. Citado na página 11.

PRISCO, R. D.; ZACCAGNINO, G.; ZACCAGNINO, R. Evobasscomposer: a multi-objective genetic algorithm for 4-voice compositions. **In Proceedings of the Genetic and Evolutionary Computation Conference**, 2010. Citado 2 vezes nas páginas 15 e 17.

QUICK, D. **Generating music using concepts from Schenkerian analysis and chord spaces**. [S.l.], 2010. Citado na página 8.

RACZYNSKI, S.; FUKAYAMA, S.; VINCENT, E. **Melody harmonisation with interpolated probabilistic**. [S.l.], 2012. Citado na página 11.

RADER, G. M. A method for composing simple traditional music by computer. **Communications of the ACM**, v. 17, p. 631–638, 1974. Citado na página 8.

ROTHGEB, J. **Harmonizing the unfigured bass: A computational Study**. Tese (Ph.D. thesis) — Yale University, 1968. Citado na página 9.

SABÁN, A. **O Projeto Magenta de Google compôs uma canção através da inteligência artificial**. 2016. Disponível em: <http://br.blogthinkbig.com/2016/06/10/o-projeto-magenta-de-google-compos-uma-cancao-atraves-da-inteligencia-artificial/>. Citado na página 12.

SALZER, F.; SCHACHTER, C. **Counterpoint in composition: the study of voice leading**. [S.l.]: Columbia University Press, 1969. Citado na página 22.

SCHWANAUER, S. Machine models of music, chap. a learning machine for tonal composition. **The MIT Press, Cambridge**, pp, p. 511–532, 1993. Citado na página 9.

SCHWOB, P. R. **Classical Archieves**. 1994. Disponível em: <http://www.classicalarchives.com>. Citado 2 vezes nas páginas 55 e 56.

SHIBATA, N. A neural network-based method for chord/note scale association with melodies. **NEC Research and Development**, v. 32, p. 453–459, 1991. Citado na página 12.

SIMON, I.; MORRIS, D.; BASU, S. Mysong: Automatic accompaniment generation for vocal melodies. **Proceedings of Sound of Music**, pp, p. 725–734, 2008. Citado na página 11.

SORENSEN, A.; BROWN, A. **JMusic Music Composition in Java**. 1998. Disponível em: <http://explodingart.com/jmusic/>. Citado na página 47.

THOMAS, M. T. Vivace: A rule based ai system for composition. **Proceedings of the International Computer Music Conference**, pp, p. 267–274, 1985. Citado na página 9.

TIPEI, S. Mp1: a computer program for music composition. **Proceedings of the Annual Music Computation Conference**, 1975. Citado na página 10.

TODD, P. M. A connectionist approach to algorithmic composition. **Computer Music Journal**, v. 13, p. 27–43, 1989. Citado na página 12.

TOKUI, N.; IBA, H. Music composition with interactive evolutionary computation. **Proceedings of the Generative Art Conference**, 2000. Citado na página 13.

TOWSEY, M. et al. Towards melodic extension using genetic algorithms. **Educational Technology Society**, v. 4, p. 54—-56, 2001. Citado na página 13.

ULRICH, J. W. Proceedings of the international joint conference on artificial inteligence. **Communications of the ACM**, pp, p. 865–872, 1977. Citado na página 8.

VERBEURGT, K.; FAYER, M.; DINOLFO, M. A hybrid neural-markov approach for learning to compose music by example. **Proceedings of the Canadian Conference on Advances in Artificial Intelligence**, v. 28 (2), p. 480–484, 2004. Citado na página 11.

WOBBROCK, J. O. et al. The aligned rank transform for nonparametric factorial analyses using only anova procedures. **Conference on Human Factors in Computing Systems**, p. 143–146, 2011. Citado na página 51.

WOLKOWICZ, J.; HEYWOOD, M.; KESELJ, V. Evolving indirectly represented melodies. **Proceedings of the European Conference on Artificial Life**, pp, p. 603–608, 2008. Citado na página 14.

ZIPF, G. K. **Human Behavior and the Principle of Least Effort**. [S.l.: s.n.], 1949. Citado na página 18.

ÖZCAN, E.; ERÇAL, T. A genetic algorithm for generating improvised music. **Proceedings of the International Conference on Artificial Evolution**, pp, p. 266–277, 2008. Citado 2 vezes nas páginas 14 e 17.

Apêndices

APÊNDICE A – Partituras de Melodias Compostas no Trabalho

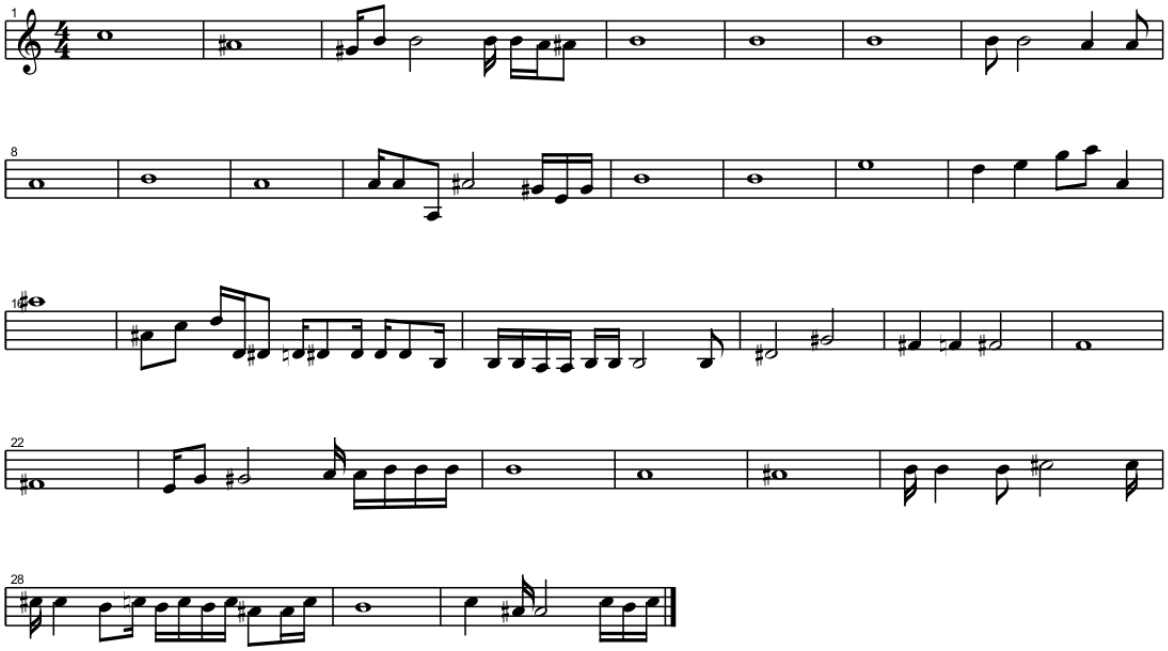


Figura 22 – Partitura da melodia composta utilizando o método 4 da capítulo 5, com taxa de cruzamento $0{,}7$ e mutação $0{,}3$.

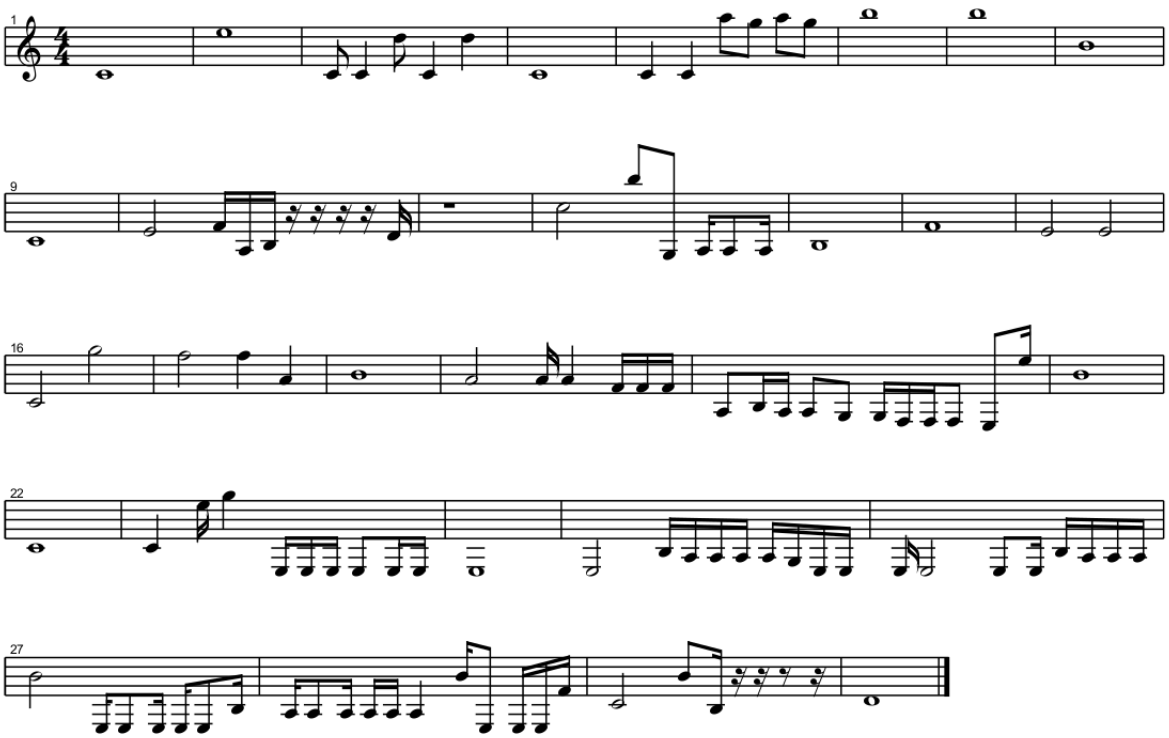


Figura 23 – Partitura da melodia composta utilizando o método 4 da capítulo 5, com taxa de cruzamento $0{,}7$ e mutação $0{,}3$ em Dó maior.

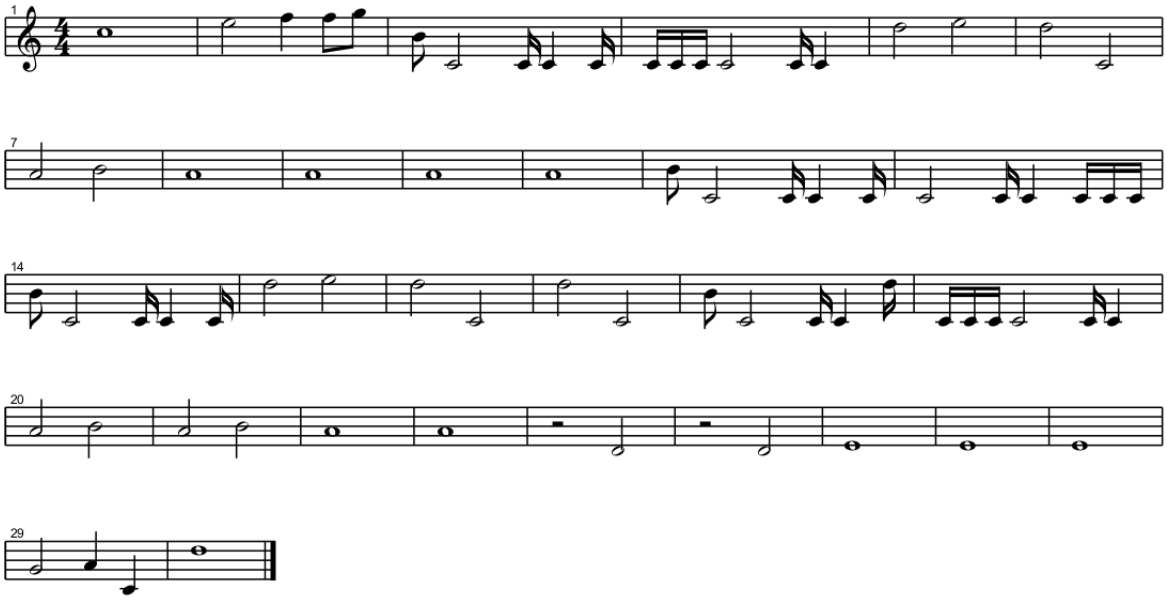


Figura 24 – Partitura da melodia composta utilizando o método 4 da capítulo 5, com taxa de cruzamento $0{,}7$ e mutação $0{,}3$, em Dó maior e com a mutação *mutateCopying-PartMusic*.