Universidade Federal de Minas Gerais

Escola de Engenharia

Programa de Pós Graduação em Engenharia Elétrica

Matheus Bitarães de Novaes

Composição Musical Algorítmica Utilizando Redes Geradoras Adversárias e Algoritmos Genéticos

Belo Horizonte

2023

Matheus Bitarães de Novaes

Composição Musical Algorítmica Utilizando Redes Geradoras Adversárias e Algoritmos Genéticos

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Mestre em Engenharia Elétrica.

Orientador: Frederico Gadelha Guimarães

Coorientador: Frederico Gualberto Ferreira Coelho

Belo Horizonte

2023

Novaes, Matheus Bitarães de.
N935c Composição musical algorítmica utilizando redes geradoras adversárias e algoritmos genéticos[recurso eletrônico] / Matheus Bitarães de Novaes. - 2023.
1 recurso online (105 f. : il., color.) : pdf.

Orientador: Frederico Gadelha Guimarães.
Coorientador: Frederico Gualberto Ferreira Coelho.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Anexos: f. 87-105.

Bibliografia: f. 79-85.
Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia elétrica - Teses. 2. Inteligência computacional - Teses. 3. Redes neurais (Computação) - Teses. 4. Algoritmos genéticos - Teses. 5. Música - Teses. I. Guimarães, Frederico Gadelha. II. Coelho, Frederico Gualberto Ferreira. III. Universidade Federal de Minas Gerais. Escola de Engenharia. IV. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pela Bibliotecária Roseli Alves de Oliveira CRB/6 2121
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FOLHA DE APROVAÇÃO

"COMPOSIÇÃO MUSICAL ALGORÍTMICA UTILIZANDO REDES GERADORAS ADVERSÁRIAS E ALGORITMOS GENÉTICOS"

MATHEUS BITARÃES DE NOVAES

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica. Aprovada em 28 de março de 2023. Por:

Prof. Dr. Frederico Gadelha Guimarães
DEE (UFMG) - Orientador

Prof. Dr. Frederico Gualberto Ferreira Coelho
DELT (UFMG)

Prof. Dr. Cristiano Leite de Castro
DEE (UFMG)

Prof. Dr. Renato Tinós
DCM (USP)

Documento assinado eletronicamente por **Frederico Gadelha Guimaraes**, **Professor do Magistério Superior**, em 28/03/2023, às 23:11, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Renato Tinós**, **Usuário Externo**, em 29/03/2023, às 11:51, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Frederico Gualberto Ferreira Coelho**, **Professor do Magistério Superior**, em 29/03/2023, às 21:13, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Cristiano Leite de Castro**, **Professor do Magistério Superior**, em 03/04/2023, às 09:50, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2173586** e o código CRC **6F5744CF**.

Dedico este trabalho aos meus pais, Roberto Mazuchi e Maria Mazzarello, que sempre fizeram questão de me educarem com muito amor e dedicação.

Agradecimentos

Gostaria de agradecer a todos que me ajudaram direta ou indiretamente neste trabalho. Agradeço a minha esposa, Ana Luiza Detomi, pelo companheirismo e paciência durante este percurso. Agradeço a meus pais, por todo carinho e dedicação durante toda a minha vida. Agradeço ao meu orientador Frederico Gadelha e a meu co-orientador Frederico Coelho pela disponibilidade, atenção e aconselhamentos durante todo este trabalho. Agradeço à UFMG e ao Programa de Pós Graduação em Engenharia Elétrica pelos ensinamentos e por promover o ensino público de qualidade.

Resumo

O uso de Inteligência Computacional para geração de peças musicais está presente na literatura desde os momentos iniciais deste campo de pesquisa. Desde então, a arte algorítmica vem acompanhando os avanços tecnológicos na área e, por ser um problema que pode ser abordado sob várias óticas, diversas abordagens estão presentes na literatura com o intuito da emulação do processo artístico por computadores. Dentre os campos explorados para geração de peças musicais, os Algoritmos Genéticos e as Redes Neurais possuem significativa presença e, conforme as Redes Geradoras Adversárias (GANs) ganharam popularidade, aplicações para geração de arte começaram a emergir. Este trabalho propõe uma arquitetura composta por um algoritmo genético cuja população inicial é alimentada por redes geradoras adversárias (GANs) especializadas em gerar melodias para determinadas funções harmônicas. A função de *fitness* do algoritmo genético é um somatório ponderado de métodos heurísticos de avaliação de qualidade, onde os pesos de cada função são atribuídos pelo usuário, antes da requisição da melodia. Uma estratégia de aumento de dados para o treinamento da GAN foi proposta e validada experimentalmente. Outro experimento realizado foi uma comparação entre a qualidade das melodias geradas pela arquitetura proposta, uma GAN e uma rede LSTM onde obteve-se evidências estatisticamente significativas de que a arquitetura proposta possui melhor qualidade, de acordo com as métricas escolhidas para o trabalho. Experimentou-se também o efeito da utilização da resposta do Discriminador da GAN integrado à função de *fitness* do algoritmo genético e obteve-se indícios estatisticamente significativos de que esta abordagem possui melhor qualidade, em comparação com a utilização da função de *fitness* sem a avaliação do Discriminador.

Palavras-chave: Inteligência Computacional, Redes Geradoras Adversárias, GANs, Redes Neurais, Algoritmos Genéticos, Música, Música Algorítmica.

Abstract

The application of Computation Intelligence for musical pieces generation is present in literature since the early moments of this research field. Since then, algorithmic art has been following the technological advances in the field and, since it is a subject that can be approached by many sides, there are a diverse set of approaches in literature to emulation of the artistic process by computers. Among the research field explored for computational musical pieces generation, Genetic Algorithms and Neural Networks have significant presence and, as GANs have become more widely used, there has been an increase in the use of them for creating art. This work proposes an architecture composed of a genetic algorithm whose initial population is fed by generative adversarial networks (GANs) specialized in generating melodies for certain harmonic functions. The fitness function of the genetic algorithm is a weighted sum of heuristic methods for evaluating quality, where the weights of each function are assigned by the user, before requesting the melody. A data augmentation statregy for the GAN training data was proposed and experimentally validated. Another experiment performed was a comparison between the quality of the melodies generated by the proposed architecture, a GAN and an LSTM network. The effects of utilizing the Discriminator's evaluation on the fitness function of the generic algorithm were also experimented in a third experiment. The statistical comparison give evidences that this approach enhances melody quality in comparisson with using the fitness function without Discriminator's evaluation.

Keywords: Computational Intelligence, Generative Adversarial Networs, GANs, Neural Networks, Generic Algorithms, Music, Algorithmic Music

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

MINDS	Machine Intelligence and Data Science
UFMG	Universidade Federal de Minas Gerais
MIDI	Musical Instrument Digital Interface
SVM	Support Vector Machine
GAN	Generative Adversarial Network
RNA	Redes Neurais Artificiais
ANN	Artificial Neural Networks
LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
RL	Reinforcement Learning
SPR	Scaled Piano Roll
FID	Frechèt Inception Distance
IS	Inception Distance
KLD	Kullback-Leibler Divergence
OA	Ovarlapped Area
REST	Representational State Transfer

Sumário

Capítulo 1

Introdução

1.1 Introdução

A busca pela emulação da arte de forma computacional é presente na história desde que o conceito da computação esteve presente. Em 1842, Ada Lovelace, ao publicar seu protótipo de uma máquina capaz de realizar expressões aritméticas e analíticas, já especulava sobre a possibilidade de utilização de sua máquina para "composições musicais elaboradas e científicas" (Menabrea e Lovelace, 1842). Os primeiros registros de composições musicais geradas por computador datam de 1957, com a *Illiac Suite* (Hiller Jr e Isaacson, 1957) e de 1961 com o IBM 704, com a composição *Daisy Bell* (Sheryl, 2010), tido como primeiro registro gravado de uma voz sintética cantando uma música.

Conforme a tecnologia evolui e novas abordagens e ferramentas surgem, as aplicações no campo da arte são usadas como vitrine para as habilidades e possibilidades inerentes destes avanços. E com Aprendizado de Máquina e Algoritmos Genéticos não foi diferente (Loughran e O'Neill, 2020). O *EvoMUSART*, *International Conference on Computational Intelligence in Music, Sounds, Art and Design*, teve seu início em 2003, como parte da conferência EvoStar e se tornou uma conferência independente desde 2012 (Evostar, 2022). O aprofundamento e interesse pelo tema é crescente desde os anos 1990 e tem gerado capítulos de livros dedicados ao tema e livros inteiros como *Evolutionary Computer Music* (Miranda e Al Biles, 2007) e *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music* (Romero et al., 2008).

As *Generative Adversarial Networks* (GAN), ou Redes Geradoras Adversárias, foram propostas inicialmente em Goodfellow et al. (2014) e ganharam significativa repercussão, principalmente na área de geração de imagens. Conforme a popularidade desta rede foi aumentando, publicações de aplicações para geração de melodias começaram a emergir, como em Lee et al. (2017), Guimaraes et al. (2017) e Yang et al. (2017)

Este trabalho propõe uma arquitetura que se utiliza de redes GAN e Computação

Evolucionária para geração de melodias musicais, visando simular o processo de improvisação musical. O sistema, bem como as fundações teóricas serão expostos nos capítulos a seguir, juntamente com três experimentos que visam avaliar a qualidade das melodias geradas em comparação com outros modelos.

1.2 Justificativa

Música representa um papel considerável na sociedade. É um meio de entretenimento, comunicação e aclimatação. Existem indícios que o uso de música ambiente na indústria pode melhorar a produtividade, como apontado em Fox e Embrey (1972). Também existem evidências que o contato com a música erudita pode reduzir os níveis de ansiedade e *stress* em profissionais de serviço de emergência, como pode ser visto em Gatti (2005). O processo de composição é complexo e composto por uma série de regras a serem seguidas. Os estudos no campo de composição algorítmica podem viabilizar a criação automática de composições para, por exemplo, a geração de trilhas sonoras de forma automática para produtores de conteúdo, trilhas musicais adequadas ao sentimento do usuário, sem que estas pessoas precisem ter conhecimentos musicais. Pode-se considerar o uso também no contexto de jogos eletrônicos, onde deseja-se gerar uma trilha sonora ininterrupta, que vai se adaptar ao momento emocional do jogo. Outro uso pode ser em criar composições automáticas para auxiliar alunos e alunas iniciantes em música, sem a necessidade de criação de um banco de músicas conhecidas.

Esta pesquisa visa ampliar o conhecimento sobre composição algorítmica pela utilização da combinação de duas abordagens para a geração de melodias musicais. Através da pesquisa em Redes Neurais (especialmente Redes Geradoras Adversárias) e Algoritmos Genéticos, procura-se ampliar o entendimento sobre o processo artístico de improviso musical e como consequência, tentar adequar o processo de geração algorítmica a estes conceitos, desenvolvendo uma nova arquitetura.

O estudo sobre emulação da arte por computadores promove tanto uma proximidade dos processos computacionais com os processos humanos, quanto aumenta conhecimento sobre o alicerce mecânico onde a arte se baseia. A música é composta de regras que criam um padrão a ser executado e, a partir disso, a arte é elaborada. Aprofundar-se nos estudos de composição algorítmica é buscar tanto a transcrição dos conceitos matemáticos musicais em linguagem de máquina quanto emular as decisões humanas artísticas.

1.3 Objetivos

Este trabalho tem como objetivo propor um sistema gerador de melodias musicais com o intuito de simular o processo de improvisação musical. O sistema permite a escolha de parâmetros pelo usuário através de uma interface gráfica e envolve uma arquitetura geradora de melodias composta por uma rede geradora adversária (GAN - *Generative Adversarial Network*) combinada com um algoritmo evolucionário.

Tem-se então como objetivo a proposição e experimentação desta arquitetura e de elementos que a compõem, como uma nova metodologia de aumento de dados a ser utilizada no treinamento da GAN. O modelo proposto é comparado com um modelo composto apenas pela GAN e um outro modelo composto de uma LSTM, para que sua qualidade seja avaliada em relação a outras abordagens.

1.4 Organização do Texto

Este trabalho está organizado da seguinte maneira: Neste capítulo há uma introdução do tema, contendo a contextualização, justificativa do problema e os objetivos do trabalho. O Capítulo 2 conta com a fundamentação teórica utilizada como base para o desenvolvimento do trabalho na parte de teoria musical, representação MIDI, algoritmos genéticos e redes neurais artificiais. Há também a revisão de trabalhos anteriores no campo de Composição Algorítmica. A arquitetura utilizada na Rede Geradora Adversária (GAN) do trabalho é detalhada no Capítulo 3, com a explicação da representação dos dados, as estratégias de *data augmentation* e as topologias do Gerador e Discriminador da arquitetura da GAN. O Capítulo 4 descreve o sistema AI Hero. A arquitetura do sistema é apresentada, bem como a interface grafica e os módulos AIHeroGAN, módulo que contém as GANs do sistema, e AIHeroEVO, módulo que contém o algoritmo evolucionário. No Capítulo 5 três experimentos são apresentados. O primeiro avalia se a metodologia de aumento de dados proposta produz alguma melhoria na qualidade das GANs, após o processo de treinamento. O segundo experimento avalia a arquitetura proposta nos capítulos anteriores (GAN com Algoritmo Evolucionário), comparando-a uma rede LSTM e uma GAN. O terceiro experimento avalia os efeitos da utilização da rede discriminadora da GAN como participante da função de *fitness* do algoritmo genético. Para todos os experimentos, testes estatísticos foram realizados e as melodias geradas foram comparadas de acordo com determinadas métricas de qualidade. Tanto os testes estatísticos quanto estas métricas são introduzidas neste capítulo. A conclusão, citando as limitações do modelo e trabalhos futuros encontra-se no capítulo 6.

Capítulo 2

Composição Musical Algorítmica

2.1 Fundamentação Teórica

Esta Seção tem o objetivo de apresentar os conceitos utilizados durante o trabalho. Serão introduzidos conceitos de teoria musical pertinentes ao problema e conceitos pertencentes ao campo de inteligência computacional.

2.1.1 Teoria Musical

A música é a arte de combinar sons no tempo de forma com que a intenção do autor manifeste um sentimento no ouvinte. Na música ocidental tradicional, as composições seguem um conjunto de regras e conceitos estabelecidos. Pode-se dividir a música nas seguintes partes (Med, 1996):

- **Melodia:** Conjunto de sons organizados em ordem sucessiva, gerando uma dinâmica horizontal.
- **Harmonia:** Conjunto de sons organizados em ordem simultânea, gerando uma organização vertical de notas, ao se olhar uma partitura
- **Contraponto:** É o conjunto de melodias dispostas em ordem simultânea, gerando uma concepção vertical e horizontal.
- **Ritmo:** É a ordem e proporção em que estão dispostos os sons que constituem melodia e harmonia

A música se manifesta através de sons, que são vibrações de corpos elásticos transmitidas ao ar e propagadas como ondas sonoras em todas as direções. Estas vibrações podem ser tanto regulares, de frequência virtualmente constante (como notas de piano ou violino), ou irregulares, de frequência irregular (como som de automóveis, explosões).

Tanto sons regulares (piano, violino, etc) quanto irregulares (instrumentos de percussão) são utilizados na música (Med, 1996). Suas principais características são:

- **Altura:** Característica determinada pela frequência das vibrações. Quanto maior a frequência, mais "alto" (agudo) é o som.
- **Duração:** É o tempo de emissão das notas.
- **Intensidade:** É a amplitude das ondas sonoras, característica também chamada de "velocidade" em representações digitais.
- **Timbre:** É uma característica do agente emissor do som. Varia de acordo com cada instrumento ou voz. É uma composição de ondas sonoras que envelopam a nota principal.

2.1.1.1 Divisões temporais

Em música existem sons de diversas durações. Tanto os sons, quanto os momentos de silêncio devem ser representados em forma de notação musical. Para que esta representação seja possível, existem diferentes tipos de notas, cada uma com um valor de duração relativa. Estas notas fazem parte de uma pauta, que possui divisões, ou compassos.

Compasso é a divisão da uma peça musical, ou parte dela, em séries regulares de tempos. Pode ser entendido como o agente métrico do ritmo, como apresentado por Med (1996). Em uma partitura, as separações dos compassos são representadas por linhas verticais, chamadas de barra de compasso ou travessão, conforme pode ser visto na Figura 1. Ele é definido no início de uma partitura e durará até o final da música ou até que um novo compasso seja definido na partitura.

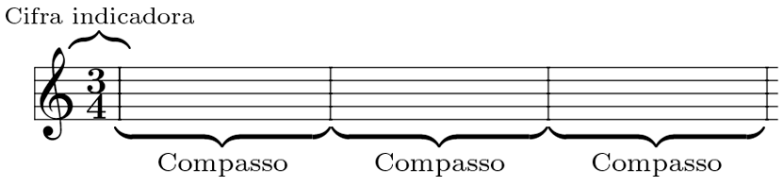

Figura 1 – Representação de compassos em uma partitura

O compasso é uma unidade regular de tempo definida conforme a formula $\frac{qt}{ql}$, onde ql é a **qualidade**, ou tipo da nota representada e qt é a **quantidade** de notas deste tipo que podem ser inseridas em sequência dentro do compasso. Por exemplo, o compasso $\frac{3}{4}$ indica uma divisão temporal onde cabem 3 notas de qualidade 4. A qualidade de uma nota indica a sua duração, conforme a Figura 2

Cada tipo de nota possui um nome específico. A nota mais duradoura é a **Semibreve**, que ocupa a duração de um compasso inteiro. A **Mínima** possui metade da duração

Nome	Imagem	Duração
Semibreve	𝅝	1
Mínima	𝅗𝅥	$\frac{1}{2}$
Semínima	♩	$\frac{1}{4}$
Colcheia	♪	$\frac{1}{8}$
Semicolcheia	𝅘𝅥𝅯	$\frac{1}{16}$
Fusa	𝅘𝅥𝅰	$\frac{1}{32}$
Semifusa	𝅘𝅥𝅱	$\frac{1}{64}$
Quartifusa	𝅘𝅥𝅲	$\frac{1}{128}$

Figura 2 – Nome, imagem e duração das notas musicais

da Semibreve. A **Semínima** possui metade da duração da **Mínima** e seguindo este mesmo padrão de divisão, tem-se a **Colcheia**, **Semicolcheia**, **Fusa**, **Semifusa** e **Quartifusa**. As pausas, momentos em que não se toca nenhuma nota, também são representadas por figuras e possuem diferentes durações, conforme Figura 3

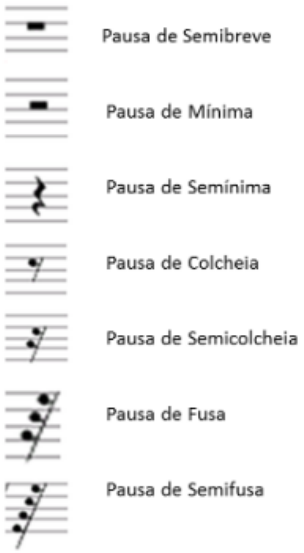

Figura 3 – Representação de pausas, isto é, momentos em que nenhuma nota é tocada

A representação de uma organização temporal das notas é indispensável para se comunicar, de forma escrita, uma sequência musical. Através destas definições é possível expressar as dinâmicas temporais desejadas por quem compõe as melodias. Outra parte importante da estruturação de uma composição em uma partitura é a representação dos intervalos musicais.

2.1.1.2 Intervalos

Como apontado no início da Seção, notas musicais referem-se a vibrações em determinadas frequências. Os intervalos, de forma geral, são as diferenças entre as frequências

dos sons. Na tentativa de criar uma fundamentação teórica matemática para os intervalos entre notas, o **Sistema Natural** foi criado. É um sistema que busca definir com precisão a frequência de cada nota musical. Porém, por este sistema nota-se que os intervalos entre as notas não são iguais. Visando abandonar a exatidão da afinação absoluta do sistema, o **Sistema Temperado** foi proposto. É um sistema que iguala os intervalos entre todas as notas, de forma a facilitar as projeções harmônicas (Med, 1996). Desta forma, define-se a **Escala temperada**, que é a divisão de uma oitava em doze partes iguais.

Na música ocidental (no sistema temperado), o menor intervalo entre notas é chamado de **semitom** ou **meio tom**. Uma oitava, isto é, um intervalo entre uma nota e a nota cuja frequência é o dobro ou a metade da nota original, contém 12 semitons. O **tom** é a soma de dois semitons.

2.1.1.3 Escalas e Tom

A partir da definição dos intervalos, é possível se definir uma escala, que é definida por uma sucessão ascendente e descendente de notas diferentes consecutivas, compreendidas dentro do limite de uma oitava. A denominação tem origem do latim "*scala*", que significa gama ou escada (Med, 1996). A escala definida por uma sequência de 12 semitons é chamada de escala cromática. A escala maior é definida pela seguinte sequência de intervalos: tom-tom-semitom-tom-tom-tom-semitom. A escala maior na tonalidade de Dó ficaria na seguinte sequência: dó-re-mi-fá-sol-lá-si-dó. Estas escalas e outros exemplos podem ser vistos na Figura 4

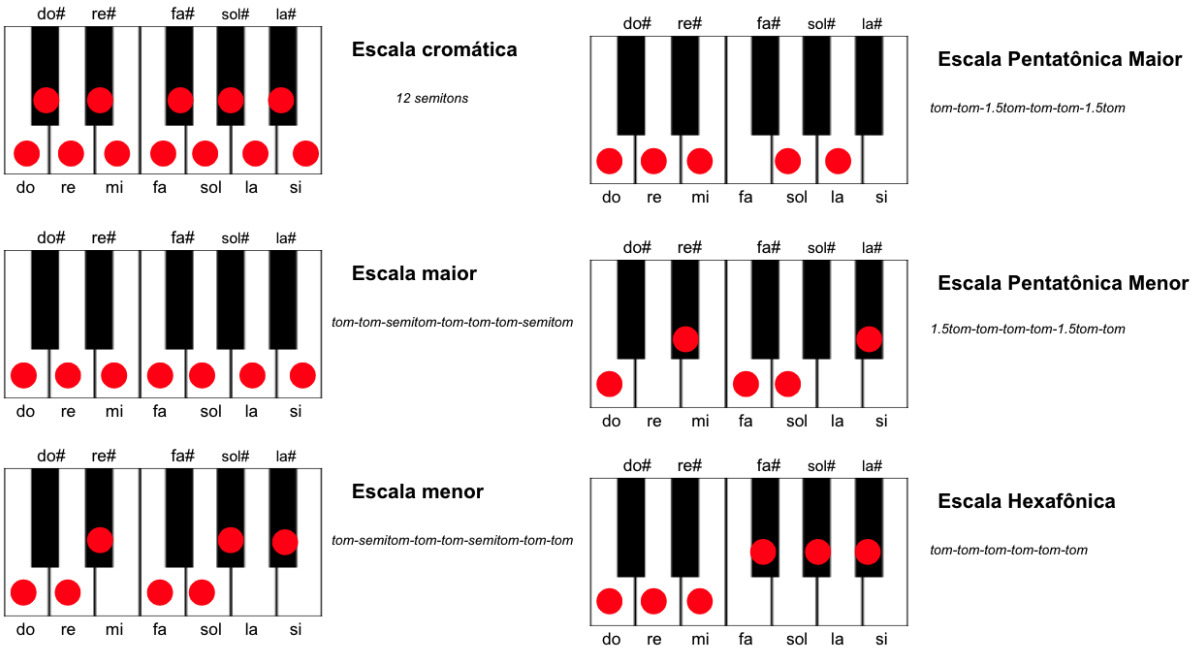

Figura 4 – Exemplos de escalas

A nota inicial da escala é chamada de **tom** e define-se tonalidade como a interde-

pendência em que os diferentes graus da escala se encontram em relação à nota tônica (primeira nota da escala), que pode ser interpretada como o centro das dinâmicas (Med, 1996). Então todo o movimento dentro da escala é feito em face do centro tonal.

2.1.1.4 Melodia e Harmonia

Pode-se definir que a música é a execução de sons no tempo, espaçados em determinados intervalos, geralmente periódicos. Na música ocidental estes sons, ou conjunto de notas, podem ser divididos em duas principais partes: melodia e harmonia.

Entende-se por melodia como uma sequência de notas espaçadas ao longo do tempo, geralmente pertencentes a uma determinada escala. As melodias são usadas na musica ocidental para definição de um tema principal, um fio condutor que é explorado durante toda a peça musical, conforme apresentado em Cavini (2012).

A harmonia consiste de uma combinação de notas tocadas ao mesmo tempo, gerando sons complexos que visam enriquecer a melodia e podem ter a intenção de conduzir quem ouve para uma sensação esperada. Dentro de uma escala, um conjunto de notas define um certo acorde.

2.1.1.5 Harmonia Funcional

Harmonia funcional é a análise das funções e sensações que cada acorde carrega dentro de uma certa tonalidade. Este conceito foi inicialmente proposto por Riemann (1899) e futuramente refinado por outros autores, como Max Regger e Hermann Grabner.

Este conceito surge a partir das teorias sobre função tonal e tonalidade, herdada de teóricos do século XVIII e XIX (por exemplo de Gottfried Weber e Georg Joseph Vogler). Estas teorias visam reduzir os acordes a sua posição fundamental, atribuindo graus de relacionamento destes acordes com a tônica. A teoria funcional, proposta posteriormente por Hugo Riemann, visa reduzir as funções tonais de determinada tonalidade a apenas três principais.

O Conceito de função no contexto de harmonia musical pode ser entendido como a propriedade de um certo acorde dada uma tonalidade. Esta propriedade é definida de acordo com a posição deste acorde em um campo harmônico. O campo harmônico descreve a relação dos acordes com o centro tonal e a relação dos acordes com a tônica é chamada de tonalidade. Existem três possíveis tonalidades e para cada tonalidade pode-se atribuir algum tipo de intenção, dinâmica ou sentimento, como explicado em Koellreutter (1986):

- **Tônica**: Repouso, resolução, relaxamento, estabilidade.
- **Dominante**: Movimento, instabilidade, tensão, aproximação.
- **Subdominante**: Movimento, tensão, afastamento.

No contexto desta dissertação, baseou-se no conceito de harmonia tonal para criar redes geradoras especialistas em cada tonalidade. Considerando que uma composição é composta de uma sequência de acordes e que, cada acorde se encaixa em uma das 3 tonalidades, considera-se que, ao se dividir a geração melódica em 3 redes especialistas, as melodias geradas por cada rede ficarão mais coesas com o contexto daquela tonalidade dentro da composição.

2.1.2 Representação digital da música e o formato (MIDI)

O padrão MIDI (*Musical Instrument Digital Interface*) é um protocolo criado em 1982 por empresas fabricantes de sintetizadores, para que fosse possível a comunicação entre instrumentos de diferentes marcas. O sinal MIDI é um sinal digital que transmite em tempo real informações de como um sinal de áudio deve ser excitado, bem como outros dados de estado do instrumento, de acordo com as entradas tocadas em um controlador. As mensagens MIDI são informações de 8 bits transmitidas em série a 31,25 kbit/s. Esta taxa foi escolhida porque é a divisão exata de 1 MHz, a velocidade com que muitos dos primeiros microprocessadores operam (Harley, 2004). O primeiro bit de cada dado identifica se a informação é um byte de estado ou de um byte de dados, e é seguido por sete bits de informação (Huber, 2012). Um bit de partida e um bit de parada são adicionados a cada byte para fins de enquadramento, de modo que um byte MIDI requer dez bits para a transmissão (Harley, 2004). A descrição das principais mensagens MIDI pode ser vista na Figura 5.

Uma mensagem MIDI pode conter 16 canais, ou seja, pode informar mensagens para 16 diferentes "instrumentos" ou 16 posições do mesmo instrumento. Há cinco tipos de mensagens: de Voz do Canal (Channel Voice), de Modo do Canal (Channel Mode), Comum do Sistema (System Common), em Tempo Real do Sistema (System Real-Time), e Exclusiva do Sistema (System Exclusive) (Huber, 2012).

Voice Message	Status Byte	Data Byte1	Data Byte2
Note off	8x	Key number	Note Off velocity
Note on	9x	Key number	Note on velocity
Polyphonic Key Pressure	Ax	Key number	Amount of pressure
Control Change	Bx	Controller number	Controller value
Program Change	Cx	Program number	None
Channel Pressure	Dx	Pressure value	None
Pitch Bend	Ex	MSB	LSB

Figura 5 – Descrição das principais mensagens MIDI

Os sinais MIDI podem ser utilizados de diversas maneiras. A mais popular é a utilização em um teclado, onde o próprio teclado é um controlador (emissor de sinais MIDI) e gerador de som (receptor de sinais MIDI), conforme Figura 6. Outro uso popular

é a utilização de um teclado controlador, que é responsável apenas pela emissão de sinais MIDI, combinado com um (ou mais) módulo(s) gerador(es) de som, conforme Figura 7.

Figura 6 – Utilização de comunicação em MIDI internamente em um teclado, que envia o sinal analógico de áudio para um amplificador.

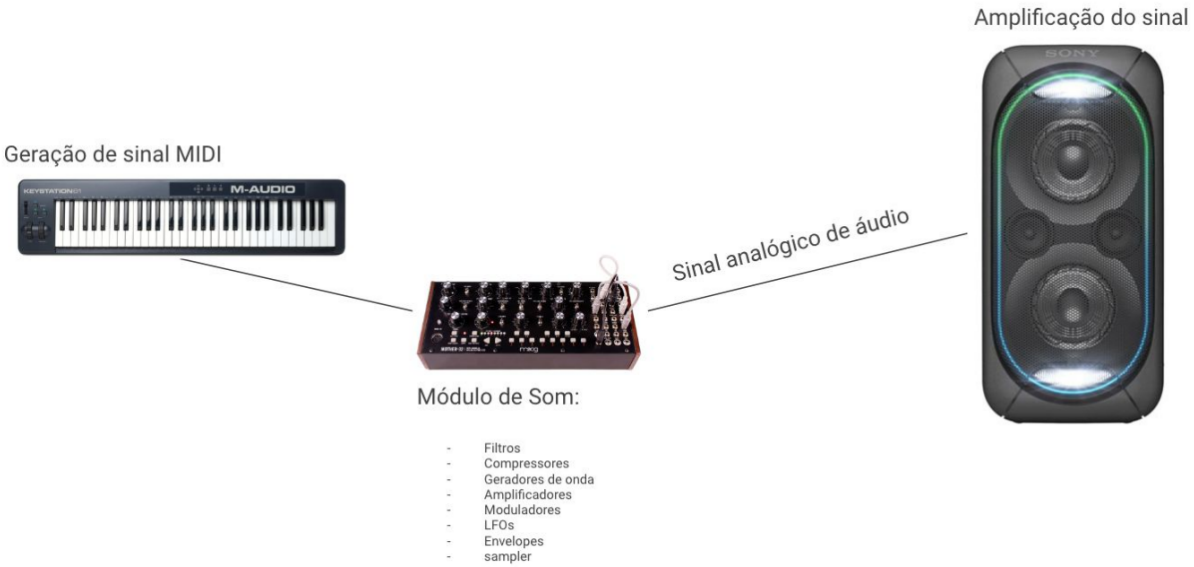

Figura 7 – Utilização de comunicação em MIDI entre um teclado e um módulo de som, que é responsável por gerar a onda sonora analógica e enviar ao amplificador.

2.1.3 Algoritmos Genéticos

A área de Computação Evolucionária estuda modelos de otimização que utilizam heurísticas baseadas em processos biológicos evolutivos encontrados na natureza. Dentro desta área, encontram-se os Algoritmos Genéticos, inicialmente introduzidos por John Holland, e publicado em seu livro *Adaptation in Natural and Artificial Systems* (Holland, 1975). Trata-se de uma forma não determinística de busca de soluções ótimas exatas ou aproximadas para problemas de otimização onde são utilizadas técnicas e conceitos inspirados na evolução biológica das espécies, como herança, mutação, cruzamento e seleção.

Esta classe de algoritmos é utilizada em otimizações de problemas complexos em campos variados, como biologia, engenharia, ciência da computação, ciência social e no campo das artes (Kumar et al., 2010).

Algoritmos genéticos baseiam-se em componentes principais como, inicialização, seleção, cruzamento e parada. Tais componentes possuem um princípio fixo, mas podem ser implementados utilizando diferentes abordagens, dependendo do problema. A seguir estes componentes serão aprofundados.

2.1.3.1 Inicialização

Para a execução do processo de evolução, é necessário ter um conjunto inicial de soluções candidatas. O processo de geração destas soluções iniciais é chamado de inicialização. Esta geração inicial pode ser aleatória ou guiada por algum tipo de conhecimento prévio do problema o que torna possível reduzir o espaço de busca do problema.

2.1.3.2 Seleção

Durante cada iteração do algoritmo, um certo número de indivíduos deve ser selecionado com base em seu grau de adaptação ao problema em questão. Este valor é definido pela função de *fitness*, cujo objetivo é avaliar cada indivíduo e atribuir um valor de qualidade, ou adaptabilidade daquela solução. Em problemas de busca de mínimo, quanto menor o valor da função de *fitness* para aquele indivíduo, melhor é aquela solução e, portanto, mais adaptado está o indivíduo ao problema.

Existem diferentes abordagens para se realizar a seleção dos indivíduos. Pode-se citar, por exemplo, o método de seleção por roleta, ou *Roulette Wheel Selection* (RWS (Bäck et al., 1997). Este método atribui uma probabilidade i de certo indivíduo ser selecionado. Esta probabilidade é proporcional ao seu *fitness* $f(i)$, como pode ser visto na equação 2.1.

$$p(i) = \frac{f(i)}{\sum_{j=1}^{n} f(j)} \tag{2.1}$$

onde n é o tamanho da população. Esta abordagem de seleção pode causar a convergência prematura para um mínimo local, caso haja a presença de indivíduos dominantes que sempre tenham altas probabilidades de seleção (Jebari e Madiafi, 2013).

Visando diminuir o risco de convergência prematura, a técnica de amostragem estocástica, ou *Stochastic Universal Sampling* (SUS) é proposta. Enquanto a seleção por roleta dá-se por uma amostragem aleatória feita de forma repetida, a abordagem SUS usa um único valor aleatório e seleciona os indivíduos de forma igualmente espaçada. É como se a roleta girasse apenas uma vez e alguns pontos igualmente espaçados fossem selecionados (Blickle e Thiele, 1996).

Outra abordagem que visa evitar a convergência prematura identificada na RWS é o método de seleção de ordenação linear, ou *Linear Rank Selection* (LRS) (Goldberg e Deb, 1991). É um método baseado em uma ordenação onde a posição n é do melhor indivíduo e a posição 1 é a do pior, de acordo com a função de *fitness*. Baseado nesta ordenação, a probabilidade de seleção de cada indivíduo é dada pela equação 2.2

$$p(i) = \frac{rank(i)}{n(n-1)} \tag{2.2}$$

Outro método de seleção, que é uma modificação do LRS é o método de ordenação exponencial, ou *Exponential Rank Selection* (ERS) (Bäck et al., 1997). Segue os mesmos princípios do LRS, porém possui uma distribuição de probabilidade exponencial entre as soluções, dada pela expressão 2.3.

$$p(i) = \exp\left(\frac{-rank(i)}{c}\right) \tag{2.3}$$

onde

$$c = \frac{2n(n-1)}{6(n-1)+n} \tag{2.4}$$

Além destes métodos, pode-se citar também o método de seleção por torneio ou *Tournament Selection* (TOS) (Hingee e Hutter, 2008). Este método é uma varação da seleção baseada em *ranking* em que um número k de indivíduos é selecionado aleatoriamente. Dentre estes indivíduos, o de maior *fitness* é selecionado. Este processo se repete até que todos os indivíduos da próxima população sejam escolhidos. Quanto maior o grupo, maior a probabilidade de selecionar indivíduos com *fitness* acima da média (Andrew, 2004).

Outra seleção possível é a *Truncation Selection* (ERS), ou seleção por truncamento (Bäck et al., 1997). É uma técnica que ordena todos os indivíduos de acordo com seus valores de *fitness* e seleciona uma porção p de indivíduos mais adaptados para passarem pela fase de reprodução.

Cada estratégia de seleção possui vantagens e desvantagens. Deve-se escolher o método que mais se encaixa no problema em questão, uma vez que algumas abordagens podem favorecer convergências precoces em mínimos locais e outras podem favorecer uma exploração global excessiva, interferindo no tempo de convergência.

2.1.3.3 Reprodução

Para que o processo de evolução biológica seja replicado, deve-se implementar uma maneira de se cruzar indivíduos adaptados, para que sejam gerados novos indivíduos que compartilhem características de indivíduos que sobreviveram ao processo seletivo. A

replicação deste conceito envolve implementar estratégias de combinação de soluções para a geração de uma nova solução, com o objetivo de se conseguir um indivíduo que possua uma adaptabilidade ainda melhor ao problema em questão.

Os métodos de cruzamento visam combinar soluções para gerar novos indivíduos. Eles podem acontecer em todos indivíduos selecionados ou pode-se optar por realizar o cruzamento condicionado a uma certa probabilidade. Neste caso, aconteceria o cruzamento com probabilidade p e nos outros casos, os indivíduos filhos seriam a réplica dos pais. Dentre os métodos de cruzamento, um dos mais amplamente utilizados é o método de cruzamento de um único ponto, ou *Single Point Crossover* (Kora e Yadlapalli, 2017). É um cruzamento onde, considerando uma modelagem de um vetor de valores, se escolhe uma posição aleatória do vetor e cria-se um filho com a parte inicial de um pai e a final do outro pai e o outro filho com a configuração contrária, conforme Figura 8.

Figura 8 – *Crossover* com 1 ponto de corte

Seguindo este mesmo princípio, tem-se também a estratégia do *crossover* com n pontos de corte, conforme Figura 9. O aumento de pontos de corte pode reduzir a performance do algoritmo, conforme apontado em Kora e Yadlapalli (2017), então deve-se analisar com cautela o número escolhido de pontos de corte.

Outra abordagem digna de menção é o *Crossover* uniforme. Define-se um vetor binário de valores aleatórios onde 1 significa selecionar o gene do pai 1 e 0 significa selecionar o gene do do pai 2. As soluções filhas são então geradas de acordo com esta máscara, conforme ilustrado na Figura 10. É um tipo de cruzamento que permite uma mistura de genes mais aleatória do que a presente nas estratégias de cruzamento anteriormente apresentadas.

Há também a possibilidade de se realizar um cruzamento de 3 pais. Esta abordagem é elaborada para modelagens binárias. Neste cruzamento, a solução filha recebe o valor

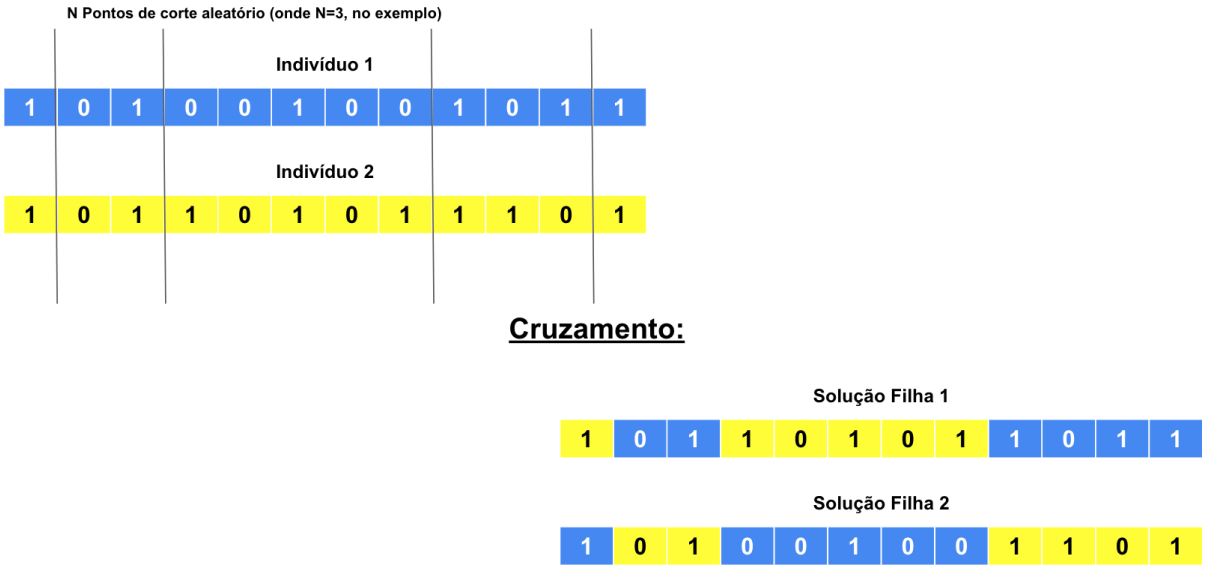

Figura 9 – *Crossover* com n pontos de corte

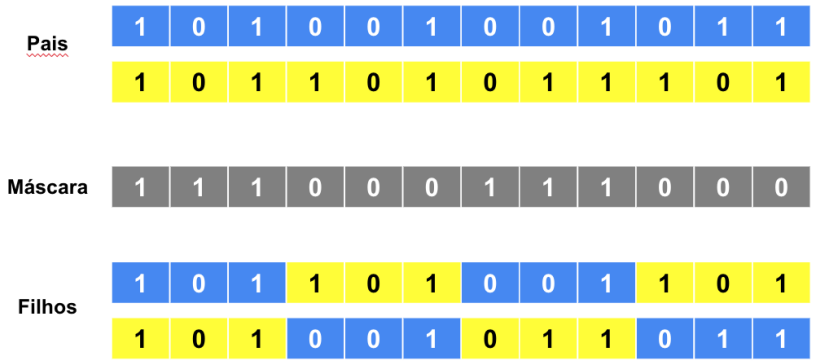

Figura 10 – *Crossover* uniforme: Define-se um vetor binário de valores aleatórios onde 1 significa selecionar o gene do pai 1 e 0 significa selecionar o gene do do pai 2. As soluções filhas são então geradas de acordo com esta máscara

de maior incidência entre os pais, para cada coluna, como pode ser visto na Figura 11. A abordagem parte do princípio que o gene a ser passado para o filho é o que possuir a maioria entre os pais selecionados.

Para modelagens não binárias, existe a possibilidade da realização do *crossover* aritmético, que seria a combinação linear entre os genes das soluções pais, com um fator de aleatoriedade, para gerar duas novas soluções filhas, conforme Figura 12.

Caso tenha-se uma modelagem por permutação, uma abordagem popular é o Cruzamento Parcialmente Mapeado, ou *Partially Mapped Crossover* (PMX) (Kora e Yadlapalli, 2017). Esta estratégia foi inicialmente proposta em Goldberg et al. (1985) para solução do Problema do Caixeiro Viajante. Um intervalo é selecionado aleatoriamente e os filhos são criados conforme os pais, porém com este intervalo trocado. Após isso,

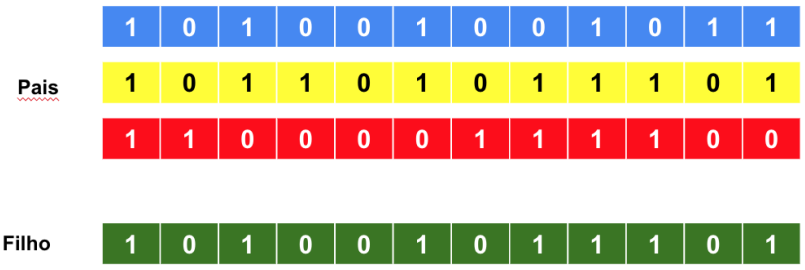

Figura 11 – *Crossover* de três pais: A solução filha recebe o valor de maior incidência entre os pais, para cada coluna

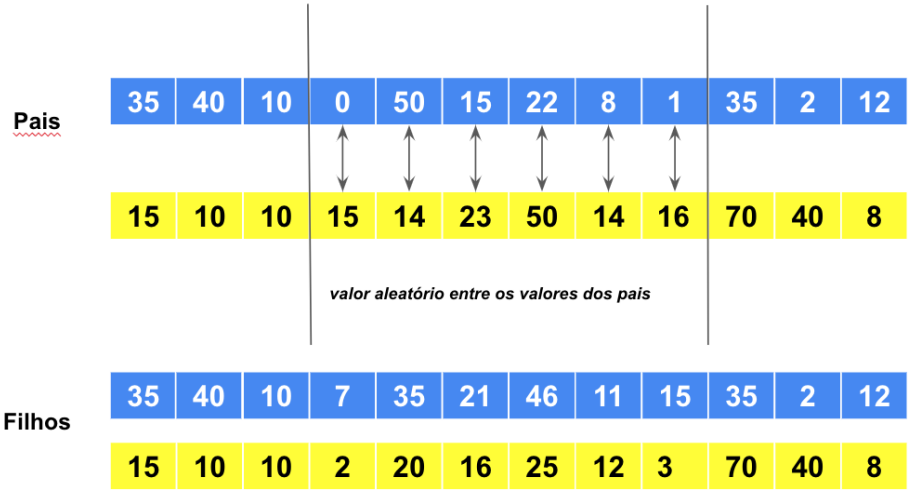

Figura 12 – *Crossover* Aritmético: Combinação linear entre os genes das soluções pais, com um fator de aleatoriedade, para gerar duas novas soluções filhas

esta relação de troca de valores do intervalo é propagada para os demais genes, como demonstrado na Figura 13.

Outra alternativa para modelagens que utilizam permutações é o *Crossover* Ordenado, ou *Order Crossover*(OX), proposto por Davis et al. (1985). Após um intervalo ser aleatoriamente selecionado, realiza-se a geração dos filhos conforme os pais, porém com a troca de genes no intervalo selecionado. Após isto, realiza-se a exclusão dos valores novos que entraram no intervalo e insere-se, em ordem, os novos valores, conforme Figura 14.

Ainda dentro de modelagens com codificação de permutação, há também o *Crossover* Cíclico, ou *Cycle Crossover* (CX). Nesta abordagem, definem-se ciclos e os filhos são gerados a partir do cruzamento destes ciclos. Para gerar um ciclo, deve-se iniciar pelo menor valor do primeiro pai (S_a) x_1 e procurar este valor x_1 na posição i correspondente no segundo pai. Ao encontrar, deve-se procurar no primeiro pai (S_b) qual o valor x_2 presente na mesma posição i. Depois, deve-se procurar em S_a qual a posição j encontra-se o valor x_2, e assim em diante, conforme demonstrado na Figura 15.

Através dos exemplos de cruzamento mostrados anteriormente, nota-se que cada

Figura 13 – *Crossover* Parcialmente Mapeado, ou *Partially Mapped Crossover*(PMX): Um intervalo é selecionado aleatoriamente e os filhos são criados conforme os pais, porem com este intervalo trocado. Após isto, esta relação de troca de valores do intervalo é propagada para os demais genes.

estratégia influenciará na velocidade de convergência, na abrangência da exploração do espaço amostral e na complexidade das operações matemáticas. É necessário ressaltar também a importância de se escolher uma estratégia de cruzamento adequada à modelagem escolhida para o problema. Em casos onde tem-se uma modelagem de dados particular, é necessário realizar adequações às modelagens acima mencionadas ou criar uma nova abordagem, porém sempre atentando-se em manter os conceitos de troca de genes entre soluções pais.

2.1.3.4 Mutação

Além do cruzamento, os processos biológicos encontrados na natureza também apresentam mutações, que são mudanças no material genético de um organismo causadas por diversos fatores, como erros na replicação do DNA, vírus, mutagênicos químicos, ou radiação (Maki, 2002). A mutação genética é uma característica indispensável para o processo evolutivo e tem um papel significativo também nos algoritmos genéticos. Em Algoritmos Genéticos a mutação é um passo onde alterações aleatórias são inseridas nas soluções geradas após a fase de cruzamento. Estas alterações aleatórias acontecem com uma probabilidade geralmente baixa, para que não se adicione uma alta taxa de aleatoriedade no processo de otimização e assim não se consiga caminhar no sentido da solução ótima.

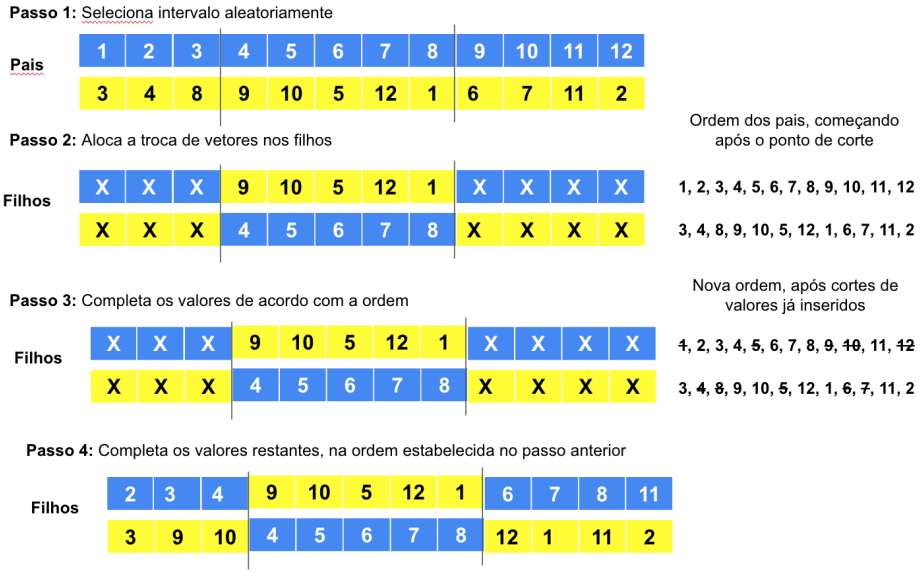

Figura 14 – *Crossover* Ordenado, ou *Order Crossover*(OX)

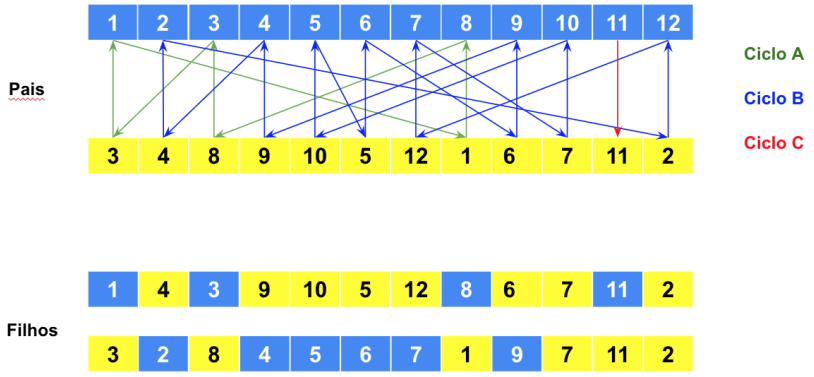

Figura 15 – *Crossover* Ciclico, ou *Cycle Crossover*(CX)

Assim como no processo de cruzamento, a metodologia de mutação é dependente da modelagem e codificação escolhida para o problema. Em uma codificação binária, por exemplo, pode-se aplicar a mudança aleatória de um bit de 0 para 1 ou 1 para zero. Em outros tipos de codificação dos dados, deve-se buscar adicionar um componente aleatório de forma que o processo de otimização não fique comprometido.

2.1.3.5 Critérios de Parada

Algoritmos Genéticos são processos que executam de forma iterativa, onde cada iteração é considerada uma geração, em alusão ao processo biológico de onde vem a inspiração. Em cada geração a população passa pelo processo de seleção, cruzamento e mutação, até que novos indivíduos sejam gerados e se tornem a população da próxima iteração. Este ciclo é repetido até que algum critério de parada seja alcançado.

Um possível critério de parada é a definição de um número máximo de gerações

que, quando atingido, tem-se o final da execução e o indivíduo com melhor *fitness* daquela geração é considerado como a solução encontrada para o problema.

Outra alternativa é estipular um número máximo de gerações em que uma solução é hegemônica. Ou seja, se uma certa solução apresentar o melhor valor de *fitness* e não for superada durante n gerações, o algoritmo deve parar e esta é a solução encontrada.

Em alguns tipos de problema e em algumas modelagens, é possível saber o valor máximo de *fitness* que pode ser alcançado. Neste caso, também é possível definir um critério de parada relacionado com a proximidade da melhor solução da execução com o ótimo conhecido. Por exemplo, pode-se entender como satisfatória uma solução que possui 90% do valor do melhor *fitness*. Desta forma, encerra-se a execução e retorna-se esta solução.

Os critérios de parada podem ser utilizados tanto separadamente quando combinados. Cada problema exigirá um tipo de requisito e a escolha do critério de parada está diretamente ligada com a qualidade da resposta do algoritmo.

2.1.4 Redes Neurais Artificiais

Redes Neurais Artificiais (RNA) ou *Artificial Neural Networks* (ANN) são sistemas de processamento de sinais compostos por um alto número de elementos que realizam operações simples de processamento e que cooperam para realizar processamentos paralelamente distribuídos com o objetivo de resolver determinada tarefa computacional. As Redes Neurais se inspiram na maneira que o cérebro humano processa informação (Macukow, 2016).

As primeiras iniciativas em construir modelos baseados no funcionamento do neurônio humano podem ser verificadas em McCulloch e Pitts (1943), onde os autores propõem redes baseadas em elementos simples, de resposta binária e limites fixos, que geravam funções lógicas. Em 1958, o Perceptron foi proposto por Rosenblatt, em Rosenblatt (1958). O Perceptron é um modelo matemático que visa emular o funcionamento biológico de um neurônio. É um classificador binário que mapeia um vetor de entrada em uma saída binária. Este mapeamento acontece através de um produto escalar entre os valores da entrada e os pesos do neurônio, que são valores estabelecidos durante o processo de treinamento da rede, o que significa que esta rede é capaz de aprender com dados. Para além do trabalho que introduziu o Perceptron, Rosenblatt também escreveu o livro *Principles of Neurodynamics*, um dos primeiros em neuro-computação (Rosenblatt, 1961).

Outro modelo significativo proposto após o Perceptron é o modelo ADALINE (*ADAptive LINear Element*) é proposto em Widrow e Hoff (1960) por dois engenheiros eletricistas Widrow and Hoff. O ADALINE, assim como o Perceptron também é uma rede de uma única camada, porém para regressão. Ambos podem aprender com dados de

treinamento. A diferença entre eles é que o ADALINE realiza a otimização dos pesos pela regra de mínimos quadrados.

Estratégias utilizando mais camadas de neurônios começaram a aparecer e surgiu a necessidade de um algoritmo de treinamento em que fosse possível atualizar os pesos de todas as camadas da rede. Com este intuito o algoritmo de *backpropagation*, proposto inicialmente por Werbos em 1974, disponível em Werbos (1994), e popularizado após o lançamento do livro *Learning Internal Representation by Error Propagation*, por Rumelhart et al. (1985). O algoritmo de *Backpropagation* é um tipo de gradiente descendente utilizado em redes neurais com o objetivo de minimização do erro (Macukow, 2016).

Este tipo de estratégia de otimização habilitou o desenvolvimento de redes com mais camadas, como por exemplo a rede *Multilayer Perceptron*, que é uma composição de várias camadas de neurônios *Perceptron*.

Existe uma grande variedade de possibilidades de arquiteturas e topologias de redes neurais. Cada uma serve a um propósito e se adéqua a certas características do problema. Levando em consideração o contexto deste trabalho, que é a utilização de redes neurais para a construção de linhas melódicas, algumas abordagens apresentam características que privilegiam o problema. Neste trabalho utilizou-se das Redes Neurais Convolucionais, ou *Convolutional Neural Networks* (CNNs). Estas redes possuem camadas internas onde operações de convolução são realizadas com os dados que trafegam por elas de uma maneira com que a dimensionalidade seja reduzida e os principais atributos relevantes à tarefa sejam ressaltados e privilegiados durante o processo de treinamento. É uma rede que obteve considerável sucesso em realizar tarefas de processamento de imagens e reconhecimento de padrões (O'Shea e Nash, 2015).

Outra possibilidade são redes que consideram os atributos temporais das sequências, as Redes Neurais Recorrentes, ou *Recurrent Neural Networks* (RNNs). Estas redes recebem como entrada uma sequência inicial de dados e visam prever o próximo valor. Dentre este tipo de arquitetura, a *Long short-term memory* (LSTM), proposta inicialmente em Hochreiter e Schmidhuber (1997), se destaca por ser uma rede que visa considerar sinais de memória de longo e curto-prazo para a previsão do próximo valor da sequência.

Para além do tipo da rede, deve-se escolher uma topologia e um processo de treinamento que sejam adequados ao problema. Uma dessas possibilidades são as Redes Geradoras Adversárias, ou *Generative Adversarial Networks*.

2.1.4.1 Redes Geradoras Adversárias (GANs)

A GAN foi inicialmente proposta por Goodfellow et al. (2014) e é uma técnica de aprendizado não supervisionado baseado em dois tipos de redes: O Gerador e o Discriminador. A relação entre estas redes pode ser vista na Figura 16

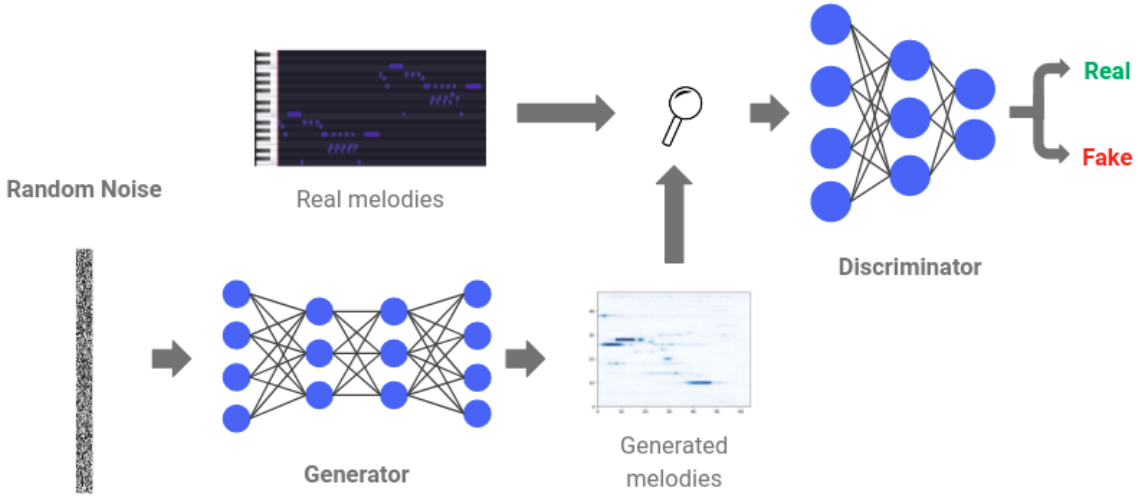

Figura 16 – Relação entre as redes geradora e discriminadora em uma GAN. O Gerador recebe um ruído aleatório como entrada e, a partir desta entrada, gera uma saída, que é avaliada pelo Discriminador. O Discriminador tem o objetivo de diferenciar as amostras criadas pelo Gerador de amostras.

O Gerador é um modelo que define implicitamente a função de densidade de probabilidade do *dataset* real $p_{model}(x)$. O modelo não necessariamente consegue avaliar toda a função de densidade de probabilidade p_{model}, mas ele consegue extrair amostras desta distribuição. A função geradora pode ser definida como $G(z, \theta_G)$, onde z é um vetor contendo valores aleatórios (chamados de semente, ou *seed*) e θ_G é um conjunto de parâmetros de aprendizado. O papel principal do Gerador é fazer a função $G(z)$ transformar o ruído aleatório de entrada em amostras que sejam indistinguíveis dos dados reais de treinamento.

O Discriminador é um modelo que recebe amostras x e retorna um valor $D(\theta_D)$, que estima se o valor inserido vem de uma distribuição real ou se é um dado falso, criado pelo Gerador. O Discriminador responde entre 0 e 1, onde 0 significa que o Discriminador avaliou a amostra como falsa e 1 como real (Goodfellow et al., 2020).

A relação entre Gerador e Discriminador é um jogo, onde cada jogador busca minimizar sua própria função de perda, dada por $J_G(\theta_G, \theta_D)$ para o Gerador e $J_D(\theta_G, \theta_D)$ para o Discriminador (Goodfellow et al., 2020), onde θ_G são os parâmetros utilizados no treinamento da rede geradora e θ_D são os parâmetros utilizados na rede discriminadora. J_G é o valor do cálculo da entropia cruzada (*cross entropy*) entre a avaliação do Discriminador e o resultado esperado pelo Gerador, que é 1. Portanto, a perda do Gerador é baixa quando ele consegue fazer com que o Discriminador avalie as amostras geradas como 1. O valor de perda no Discriminador J_D é a entropia cruzada entre a amostra real e 1 somado com a entropia cruzada entre uma amostra gerada pelo Gerador e 0.

A arquitetura inicial proposta por Goodfellow et al. (2014) foi pensada para a

geração de imagens. Portanto, é composta por redes que não consideram sequências de dados (que é o caso de melodias musicais). O trabalho de Yu et al. (2017) propõe uma GAN que gera sequências, chamada SeqGAN. Esta arquitetura se difere do conceito inicial da GAN por utilizar Redes Neurais Recorrentes em seu Gerador. Esta diferença muda a forma de treinamento da rede, uma vez que o Discriminador avalia apenas sequências completas, e não partes dela. Para resolver isto, é proposto um gerador treinado com *Reinforcement Learning* (RL), onde utiliza-se o *Policy Gradient* como abordagem para atualização dos pesos. O discriminador segue avaliando a sequência inteira e este sinal é repassado para os estados intermediários utilizando uma busca *Monte Carlo* (Yu et al., 2017).

2.2 Trabalhos Anteriores em Composição Algorítmica

Na área de composição algorítmica, é possível observar abordagens diversas para a resolução do problema. Em Papadopoulos e Wiggins (1999), as metodologias são divididas em Modelos Matemáticos, Modelos Baseados em Conhecimento, Modelos Gramaticais, Métodos Evolucionários, Sistemas de Aprendizado. E mesmo considerando estas divisões, podem existir modelos híbridos, que se aproveitam de dois ou mais metodologias para realizar a composição musical.

Uma das metodologias utilizadas para composição algorítmica é a geração de composições baseadas em modelos matemáticos. Este tipo de abordagem consiste em alimentar o sistema com um conjunto de regras pré-estabelecidas sobre o processo de composição musical. Dentro desta abordagem, nota-se a presença de soluções que utilizam a modelagem de melodias por Cadeias de Markov, como em Ames (1989), onde a nota a ser tocada no tempo $t+1$ é definida por uma probabilidade, de acordo com o estado no tempo t. Este tipo de modelagem, bem como outros processos estocásticos apresentam rápida resposta e estão presentes em soluções comerciais a partir dos anos 1980 (Zicarelli, 1987). Nota-se maior presença de processos estocásticos em composição musical entre os anos 1980 e 2000. Apesar de possuírem a vantagem de obterem rápidas respostas, é necessário que alguém insira as probabilidades para a geração da sequência temporal. Tal abordagem não incorpora muitos desvios da norma musical e, portanto, não captura níveis mais elevados de abstração musical (Papadopoulos e Wiggins, 1999). Existem também modelos baseados em Cadeias de Markov primeiro para dedução de ritmo e depois para definição das tonalidades das notas, como o *Cybernetic Composer* (Ames e Domino, 1992).

Outro exemplo de composição musical baseada em modelos matemáticos pode ser encontrado em Kosona (2010), onde uma metodologia de composição musical é proposta a partir do modelo da Catástrofe Cúspide, um dos modelos englobados pela Teoria da Catástrofe e foi inspirado pelo fenômeno da refração. Modelos matemáticos também podem

ser utilizados para representação de melodias em outros espaços e, desta forma, auxiliar no processo de composição musical ou processamento de informações por computadores. Em Chew (2000), um modelo chamado *Spiral Array* é proposto. Este modelo geométrico tem o objetivo de representar as relações entre as notas e foi utilizado para três tarefas: Determinação do tom da composição, identificação de modulações (mudanças de tom no meio da música) e encontrar a raiz do acorde (nota que dá o nome ao acorde). Em Souza et al. (2019), a aplicação *Graph Composer* é proposta como uma forma alternativa de definição de melodias, onde as notas são organizadas no espaço em formato de grafo. Desta forma, compositores podem rearranjar as melodias alterando as topologia do grafo.

Outra possível abordagem para a composição algorítmica é através de sistemas baseados em conhecimento, ou *Knowledge-based systems*. Estes sistemas, em forma geral, possuem um componente responsável por conter regras pré-definidas de comportamento e um componente decisor, que utiliza estes conhecimentos conforme o contexto, como acontece em Pachet (1991). Este trabalho propõe um *framework* para construir sistemas que simulam acompanhamentos de jazz. Os acompanhamentos são gerados de acordo com a tonalidade, acorde e a melodia que está sendo tocada no momento. Com estas informações, o sistema decisor consulta e retorna o acompanhamento mais apropriado a ser gerado. Em Ebcioğlu (1988), um sistema chamado *CHORAL* é proposto e tem como objetivo criar harmonizações de corais no estilo de *J. S. Bach*. Este sistema contém uma série de regras pré-fixadas e foi implementado em uma linguagem nova proposta, chamada BSL *(Backtracing Specification Language)*.

Uma das abordagens mais exploradas na literatura é a utilização de algoritmos evolucionários para a geração de composições musicais. As primeiras iniciativas de uso de computação evolucionária para composições musicais datam do início dos anos 1990 (Loughran e O'Neill, 2020), como pode-se observar no projeto de Horner e Goldberg em Horner e Goldberg (1991) e no projeto *GenJam* que tinha a intenção de gerar um algoritmo evolucionário para a composição de solos de *jazz* (Biles et al., 1994). Na arquitetura proposta, a avaliação de *fitness* era feita em tempo real, por humanos. Esta abordagem deixava o processo de otimização lento. Como tentativa de solucionar esta parte, o autor trocou esta avaliação de *fitness* humana por um modelo neural baseado em análises estatísticas (Biles et al., 1996) e o resultado foi tido como não satisfatório. Como uma terceira abordagem, o autor removeu a função de *fitness*, o que gerava uma evolução sem pressão seletiva, basicamente cruzando frases musicais já conhecidas (Biles, 2001), o que fez os autores questionarem se esta seria uma abordagem de evolução genética ou não. Este projeto criado por Biles evoluiu algumas vezes durante as ultimas décadas e se tornou um sistema interativo de improvisação utilizado em performances de *jazz* (Biles, 2013).

Pode-se notar na literatura a utilização de computação evolucionária para a geração não somente de melodias, mas também de padrões rítmicos, como em Horowitz

(1994), onde um sistema de geração rítmica utilizando computação evolucionária foi proposto. A função de *fitness* era uma combinação de funções objetivo ponderada por pesos e alterações de preferências inseridas pelos usuários. O autor notou que, apesar do algoritmo genético ser relativamente simples de ser implementado, a correta modelagem da função de *fitness* era um desafio.

Há também trabalhos que buscam gerar o processo melódico juntamente com o harmônico. Em Freitas (2011) os autores propõem geradores melódico e harmônico utilizando algoritmos genéticos, onde o gerador melódico possui população inicial composta de melodias que possuem um *fitness* mínimo aceitável e o gerador harmônico utiliza uma função de *fitness* multiobjetivo composta por heurísticas específicas para avaliar a qualidade de linhas harmônicas, como presença de tríades, presença da tônica, presença de notas pertencentes aos graus corretos, como explorado em Freitas e Guimaraes (2011). Posteriormente, em Freitas et al. (2012), os autores aprofundam a exploração de novas funções de avaliação de qualidade musical, através da extração e análise de características de melodias de bossa nova.

É possível verificar recentes aplicações de computação evolucionária em composição musical. Como por exemplo em Komatsu et al. (2010), que é um sistema de composição automática que utiliza programação genética em dois modelos independentes, um para a harmonia e outro para a melodia. A população inicial do gerador de melodia é criada através de um Modelo Oculto de Markov (*Hidden Markov Model*) extraído do gerador de progressão de acordes. Então conforme o gerador responsável pela harmonia evolui, o gerador de melodias também obtém melhores resultados. O trabalho posteriormente evoluiu a geração de acordes, adicionando notas de tensão na parte da geração de progressão de acordes e focou em gerar composições de *jazz blues*, conforme Kunimatsu et al. (2015). Ting et al. (2015) propõe um sistema de composição utilizando Programação Genética que recebe composições reais e procura gerar variações destas composições de acordo com certas regras estabelecidas. Quatro funções de *fitness* são propostas, todas baseadas em heurísticas. Em Loughran et al. (2015) um sistema de Evolução Gramática, ou *Grammar Evolution* (GE) é apresentado. Este sistema tem o objetivo de criar composições tonais. A função de *fitness* avalia a proximidade da tonalidade das soluções candidatas com a tonalidade desejada. Scirea et al. (2016) apresenta um sistema que consiste em um gerador de acompanhamentos (acordes) baseado em padrões e um gerador de melodias que combina *FI-2POP* (Kimbrough et al., 2004) e otimização multi-objetivo. O sistema foi bem avaliado por participantes humanos, em comparação com versões intencionalmente distorcidas e aleatórias geradas pelos autores. Posteriormente em Scirea et al. (2018), o sistema foi utilizado para criar música em tempo real capaz de expressar diferentes aclimatações em um jogo de dama.

Arquiteturas híbridas de Redes Neurais e Algoritmos Genéticos, como é o caso

desta dissertação, também podem ser encontradas na literatura. Como em (Farzaneh e Toroghi, 2020), onde um Algoritmo Genético é responsável por gerar melodias e a sua função de *fitness* é uma Rede Neural (LSTM).

Para além do Algoritmo Evolucionário, esta dissertação também faz o uso de GANs para geração de melodia. É possível também encontrar na literatura a utilização deste tipo de rede em composição algorítmica, como em Yang et al. (2017), onde é proposto um sistema que utiliza GANs convolucionais, chamado *MIDINet*. Existem também aplicações onde se utilizam GANs sequenciais, como por exemplo em Lee et al. (2017), onde é proposta a aplicação de uma *sequence generative adversarial network* (SeqGAN), citada na Seção 2.1.4.1 para criar sequências musicais polifônicas. Em Guimaraes et al. (2017) o sistema ORGAN, ou *Objective-Reinforced Generative Adversarial Networks* é proposto. Também inspirado na SegGAN, este trabalho utiliza GAN com *reinforcement learning*(RL) para gerar peças musicais.

Hoje em dia é possível encontrar empresas que fornecem soluções de geração automática de composições, como por exemplo a Soundraw, que é um sistema responsável por gerar melodias de acordo com entradas de um usuário, como gênero, instrumentos, humor, duração, entre outros (Soundraw, 2022). Outro exemplo é o sistema AIVA, uma solução de criação de composições personalizadas, utilizando inteligência artificial, com o intuito de facilitar o processo de licenciamento e uso de composições por produtores de conteúdo (SARL, 2022). Existe também a Boomy, uma plataforma que permite a criação de composições através de parâmetros escolhidos em uma interface gráfica e a disponibilização destas músicas em diversas plataformas, habilitando pessoas que não possuem conhecimento musical a criar composições e monetizá-las nas plataformas de distribuição (Corporation, 2020). Existem também produtos como o *Riffusion* (Forsgren e Martiros, 2022), que geram peças musicais de acordo com um texto do usuário, com especificações da música pretendida. Este produto realiza uma sintonia fina no modelo *Stable Diffustion* (Rombach et al., 2021), modelo popular de geração de imagens, para focar em geração de espectrogramas musicais.

Capítulo 3

Rede Geradora Adversária em Composição Musical Algorítmica

3.1 Introdução

Este capítulo contém a descrição da representação dos dados, modelagem e topologias propostas para as Redes Neurais, em específico Redes Geradoras Adversárias, no contexto de geração de melodias musicais.

Neste capítulo a representação *Scaled Piano Roll* (SPR) será apresentada, juntamente com a metodologia de *data augmentation* proposta. As topologias das redes Geradora e Discriminadora da GAN serão descritas, bem como a métrica de avaliação de qualidade utilizada para avaliar a GAN.

3.2 Representação dos dados

A escolha de uma representação de dados adequada tem um papel importante em composição musical algorítmica, uma vez que cada estratégia de representação impõe diferentes vantagens, limitações e complexidades em todo o processo de treinamento e otimização.

A representação escolhida para este projeto é a *Scaled Piano Roll (SPR)*, criada como uma variação da representação comumente chamada de *Piano Roll*, ou Rolo de Piano. *Piano Roll* é uma representação onde cada nota é distribuída ao longo do eixo y, enquanto o tempo é representado pelo eixo x. O *Scaled Piano Roll*, como pode ser visto na Figura 17, pode ser entendido como as notas do *Piano Roll*, transladadas no eixo y de acordo com o centro tonal (descrito em 2.1.1.3) em que a dada melodia foi gravada. Até onde é de conhecimento do autor, não foi encontrado outro trabalho que propôs este tipo de representação.

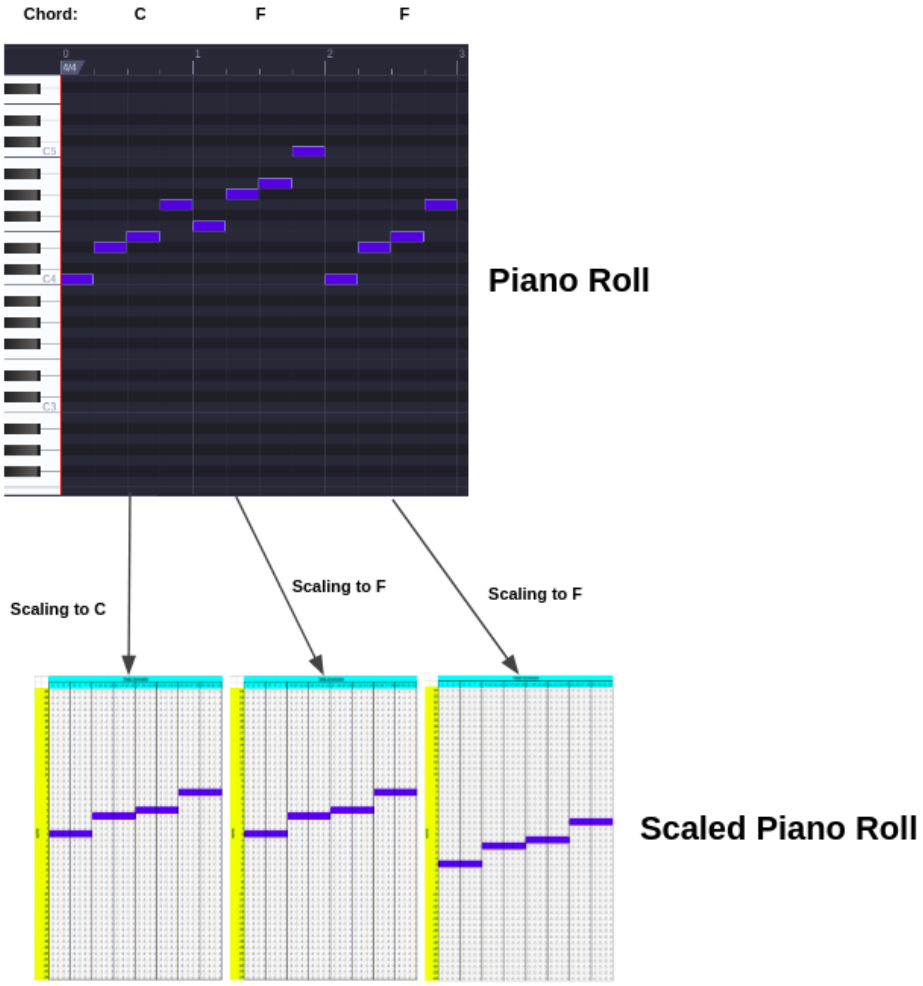

Figura 17 – Representação *Scaled Piano Roll* (SPR) em comparação com representação *Piano Roll*

Esta forma de representação dos dados melhora a generalização em comparação com a representação *Piano Roll*, pois todo o processo de treinamento e geração de melodias acontece sem a influência do acorde que está sendo tocado, uma vez que todos os dados de treinamento são transladados para o mesmo centro tonal (2.1.1.3). Então pode-se entender que o sistema gera as melodias sempre em um centro tonal fixo e padronizado. As melodias geradas são transpostas para o acorde e tonalidade durante o contexto de execução.

As dinâmicas e intenções das melodias musicais são compostas indiferentemente de acordes e tonalidades. A relação temporal entre as notas é o que conta para a qualidade ou veracidade de uma melodia. Portanto, atributos como tonalidade e acordes não precisam ser apresentados durante o treinamento do modelo.

3.2.1 Data Augmentation

Data Augmentation ou Aumento de Dados, é utilizado para expandir o conjunto de dados de treinamento através da criação de novos dados a partir da modificação

das amostras originais, de acordo com determinadas estratégias. As técnicas de *Data Augmentation* exercem um importante papel em contextos onde dados de treinamento são escassos ou onde perturbações nas amostras melhoram a generalização (como, por exemplo, em imagens). O contexto deste trabalho é um caso onde os dados de treinamento não estão amplamente disponíveis, pois necessitam ser dados MIDI, contendo apenas melodias e corretamente atribuídos a uma função harmônica. Portanto, as estratégias de *Data Augmentation* tornam-se de significativa importância nesse contexto, e estão descritas a seguir.

3.2.1.1 Mudança de duração

Muda a duração de uma nota arbitrária da melodia. Pode aumentar ou diminuir o tempo da nota, sempre deixando o resultado dentro de uma faixa pré-estabelecida de duração, como pode ser visto na Figura 18. Esta estratégia pode ser executada múltiplas vezes no *dataset* para gerar diferentes combinações.

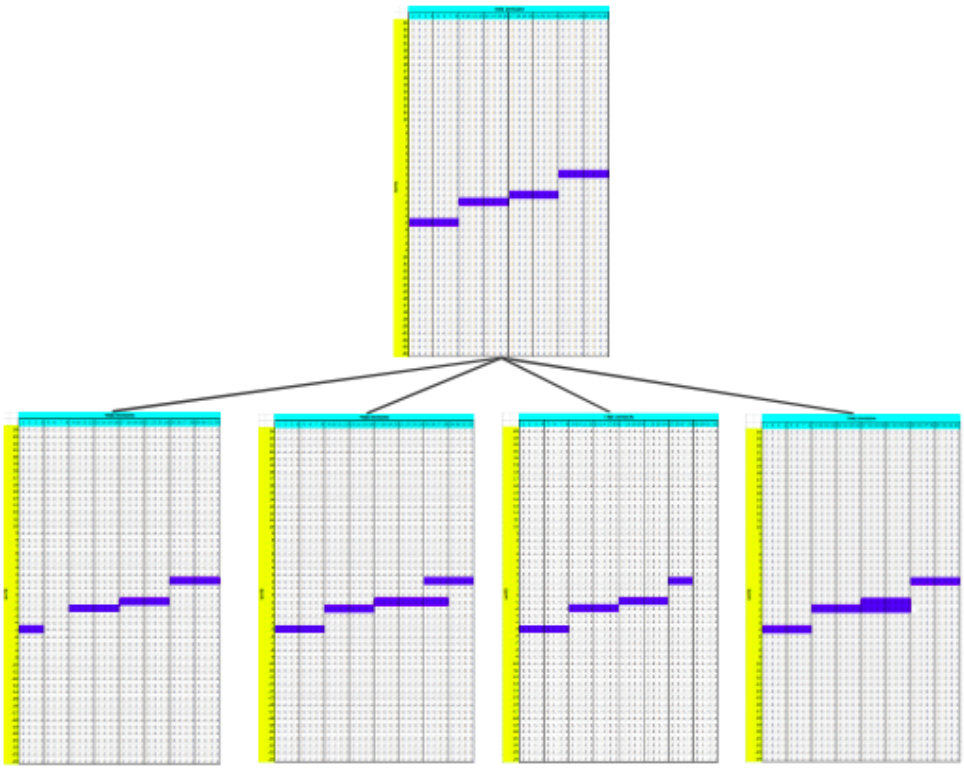

Figura 18 – Estratégia de perturbação do tempo da nota: Modifica a duração de notas aleatórias.

3.2.1.2 Mudança de oitava

Transpõe a melodia uma oitava para cima ou para baixo, conforme Figura 19. Se esta estratégia for executada mais de uma vez sobre o *dataset*, ela pode gerar melodias duplicadas. Portanto, não é recomendado que esta abordagem seja executada mais de uma vez.

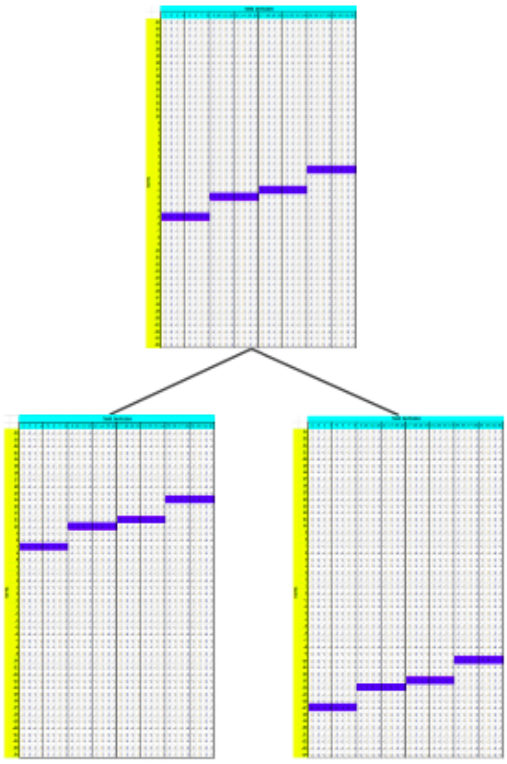

Figura 19 – Mudança de Oitava: Transpõe a melodia uma oitava para cima ou para baixo.

3.2.1.3 Adição do quinto grau

Adiciona uma nota 5 semitons acima ou abaixo de uma nota escolhida aleatoriamente. Não é recomendado executar esta estratégia mais do que 4 vezes pois melodias com múltiplas notas sendo tocadas em conjunto podem aparecer e tal característica pode soar distante de dados reais.

3.2.2 O Procedimento de *Data Augmentation*

O procedimento foi elaborado para permitir aumentar o *dataset* múltiplas vezes com diferentes estratégias. Cada estratégia i possui um fator de aumento k_i, que se refere a quantas vezes aquela estratégia será aplicada ao *dataset*. A equação 3.1 descreve a influência do processo de aumento dos dados em um *dataset* de tamanho n, com i estratégias de aumento.

$$n_{aug}(i) = n_{aug}(i-1)(k_i + 1) \tag{3.1}$$

onde $n_{aug}(i)$ é o tamanho total dos dados depois de se aplicar a estratégia de aumento i, que possui fator de aumento k_i. Um *dataset* sem nenhuma estratégia de aumento de dados é conforme **??**:

$$n_{aug}(0) = n \tag{3.2}$$

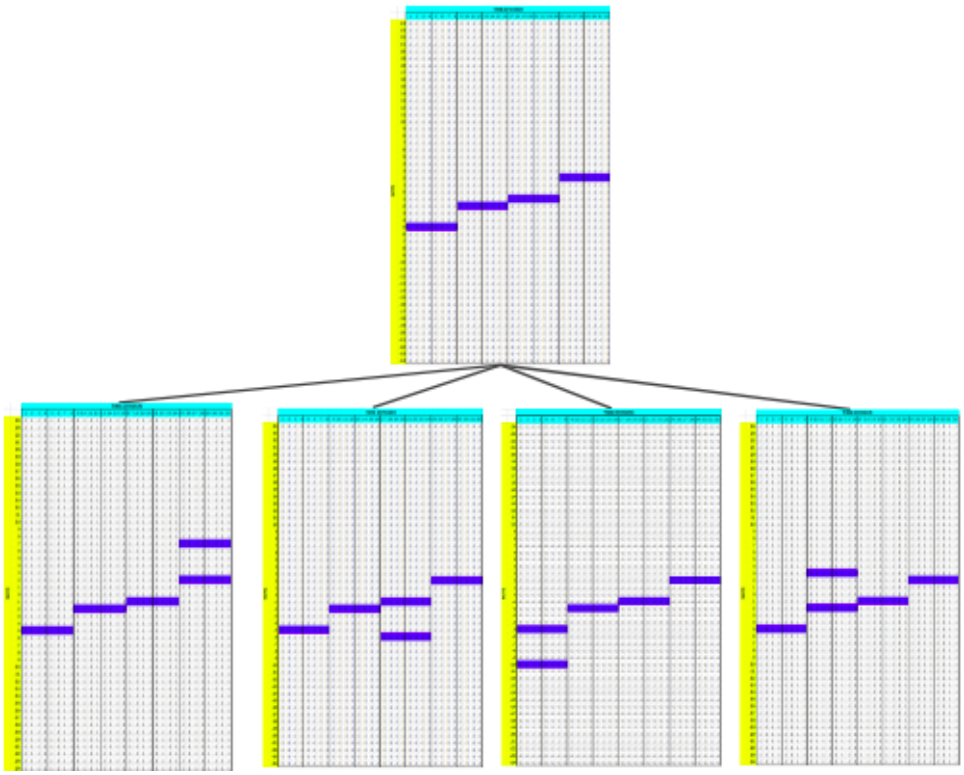

Figura 20 – Adição do quinto grau: Adiciona notas 5 semitons acima ou abaixo de uma nota aleatória.

Então, por exemplo, suponhamos 10 amostras de treinamento e as constantes abaixo:

- 1 - Mudança de duração: $k_1 = 6$
- 2 - Mudança de oitava: $k_2 = 1$
- 3 - Adição do quinto grau: $k_3 = 3$

Teremos o seguinte:

$$
\begin{aligned}
n_{aug}(0) &= n = 10 \\
n_{aug}(1) &= 10(6+1) = 70 \\
n_{aug}(2) &= 70(1+1) = 140 \\
n_{aug}(3) &= 140(3+1) = 560
\end{aligned}
$$

Portanto, após o uso das três estratégias de aumento de dados, o tamanho do *dataset* irá de 10 amostras para 560 amostras.

O experimento que valida a efetividade da utilização destas estratégias para aumento do número de amostras é descrito na Seção 5.3.

3.3 Redes Geradoras Adversárias (GAN)

Como descrito na Seção 2.1.4.1, a GAN é composta de duas redes, chamadas de Gerador e Discriminador, cujos objetivos de treinamento são opostos. O Discriminador é treinado para diferenciar amostras reais de amostras geradas pelo Gerador e o gerador é treinado para fazer com que o Discriminador avalie suas amostras como amostras reais. No contexto deste trabalho, as topologias do Gerador e Discriminador são conforme descrito abaixo e foram escolhidas após experimentações, considerando o tempo de treinamento e a capacidade de atingir bons resultados preliminares.

3.3.1 Gerador

O Gerador deve ser uma rede capaz de gerar melodias no formato *Scaled Piano Roll*, descrito anteriormente. E, para além disso, deve ser capaz de gerar um conjunto melodias em tempo hábil, para que sejam utilizadas como população no Algoritmo Genético. Um conjunto de possibilidades foi analisado.

Foi considerada a possibilidade da utilização de uma GAN sequencial (explicada com mais detalhes em 2.1.4.1), mas percebeu-se que o processo de geração de melodias tomava mais tempo, sendo proibitivo no contexto de geração de um grande número de melodias (200, por exemplo) para fazerem parte da população de um algoritmo genético, uma vez que o gerador era composto de uma Rede Neural Recorrente. Redes Neurais Recorrentes, como LSTMs, precisam que a amostra anterior seja gerada para que se gere a amostra atual. Portanto, a geração se torna mais lenta em comparação com uma rede que gera todas as amostras de uma só vez.

Portanto, a possibilidade de usar a GAN sequencial para geração de melodias no contexto deste trabalho foi descartada, priorizando a utilização da GAN conforme foi definida nos primeiros artigos propostos (descritos em 2.1.4.1), que eram focados em geração de imagens.

A topologia do Gerador é de acordo com a Figura 21, inicialmente inspirada na topologia apresentada em Tensorflow (2022). A rede recebe um vetor de ruído aleatório na entrada, com dimensão pré-determinada *noise_ dim* (que neste trabalho é um vetor de 100 valores entre 0 e 1), passa por uma camada densa (onde cada neurônio recebe como entrada todos os neurônios da camada anterior) e depois por normalização em *batch*, que possui o objetivo de estabilizar o processo de aprendizagem através da normalização da saída da camada densa. Após isto, aplica-se uma camada de ativação *leaky ReLu*. Depois destas camadas, aplica-se uma sequencia de convoluções, normalizações e ativações *leaky ReLu* até que a saída esteja na dimensão das melodias, que é (48 x 64).

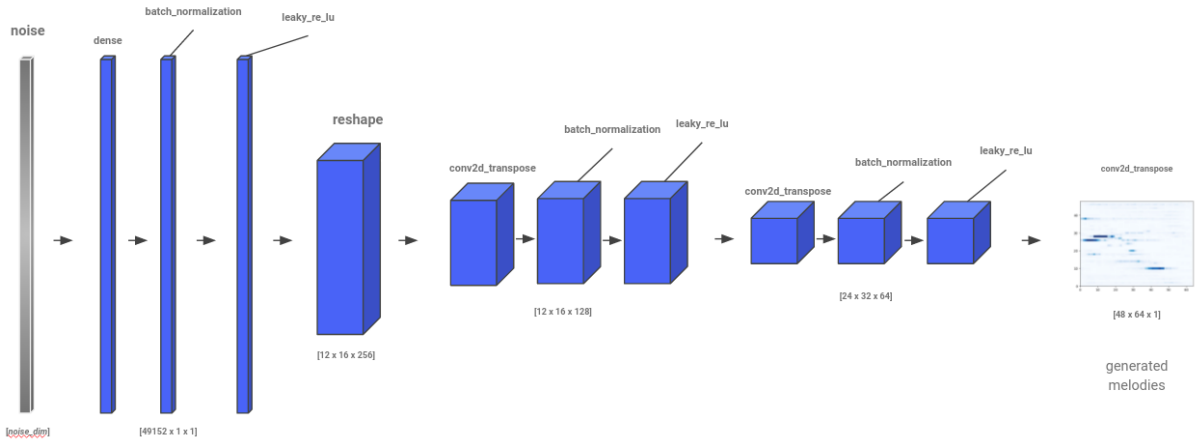

Figura 21 – Topologia do Gerador: recebe um ruído aleatório (*seed*) de dimensão *noise_ dim*, passa por uma camada densa, uma normalização em *batch* e uma camada de ativação *leaky ReLu*. Após isso, há duas camadas de convolução, normalização e ativação *leaky ReLu* para que se tenha uma saída de acordo com a dimensão pretendida.

3.3.2 Discriminador

O Discriminador deve ser uma rede capaz de receber melodias e classificá-las, indicando se correspondem a melodias reais ou geradas pelo Gerador.

A arquitetura do Discriminador é conforme a Figura 22. É uma topologia que recebe uma melodia com dimensão (48 x 64), aplica um conjunto de convoluções e mudanças de dimensão e, ao final, responde com um valor real entre 0 e 1, indicando se a melodia de entrada é real ou gerada artificialmente (1 para real e 0 para gerada artificialmente).

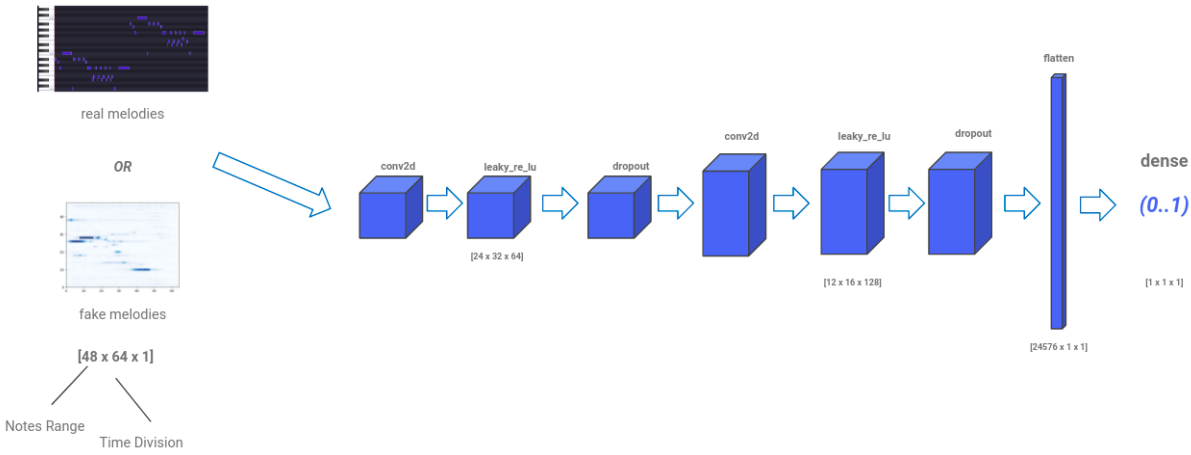

Figura 22 – Topologia do Discriminador: Recebe uma melodia com dimensão (48 x 64), aplica duas convoluções e mudanças de dimensão e tem como saída um valor entre 0 e 1, indicando se o dado de entrada é real (1) ou gerado artificialmente (0).

Capítulo 4

AI Hero: Um sistema gerador de melodias

4.1 Introdução

AI Hero é o nome do sistema proposto neste trabalho. Este sistema é acessado pelo usuário através de uma interface *web*, onde a melodia gerada também é apresentada e executada. O sistema gerador de melodias conta com um algoritmo genético em que sua população inicial é gerada por GANs treinadas especificamente para as funções harmônicas pretendidas. Neste capítulo será descrita a arquitetura deste sistema, bem como seus componentes principais e as especificações tanto das GANs quanto do Algoritmo Genético.

4.2 A Arquitetura do Sistema

A arquitetura geral do sistema proposto é conforme Figura 23. O usuário realiza requisições ao módulo **AIHeroEvo**, um algoritmo evolucionário, que otimizará as melodias para obterem melhor valor de *fitness* possível. A função de *fitness* é definida como a média ponderada de funções que descrevem características musicais, como "taxa de variedade de notas em um compasso", "quantidade de notas sendo tocadas em um compasso". O peso de cada uma dessas funções é atribuído conforme a intenção do usuário de gerar melodias com mais variedades de notas e mais ou menos "caóticas". Algoritmos evolucionários necessitam de uma população de soluções para iniciar as gerações de otimização. Esta população de soluções é fornecida pelo módulo **AIHeroGAN**, uma Rede Geradora Adversária (GAN), treinada para gerar melodias de acordo com a função harmônica solicitada. Os pesos dos treinamentos são armazenados em *checkpoints*, para que a rede possa ser reinicializada e não se perca o progresso do que já foi treinado. A rede é treinada inicialmente com melodias geradas manualmente. Novas melodias podem entrar nos dados de treinamento durante a execução, pois o usuário terá a possibilidade de aprovar a qualidade da melodia

que, caso seja aprovada, poderá ser integrada aos dados de treinamento para incrementar a qualidade do modelo.

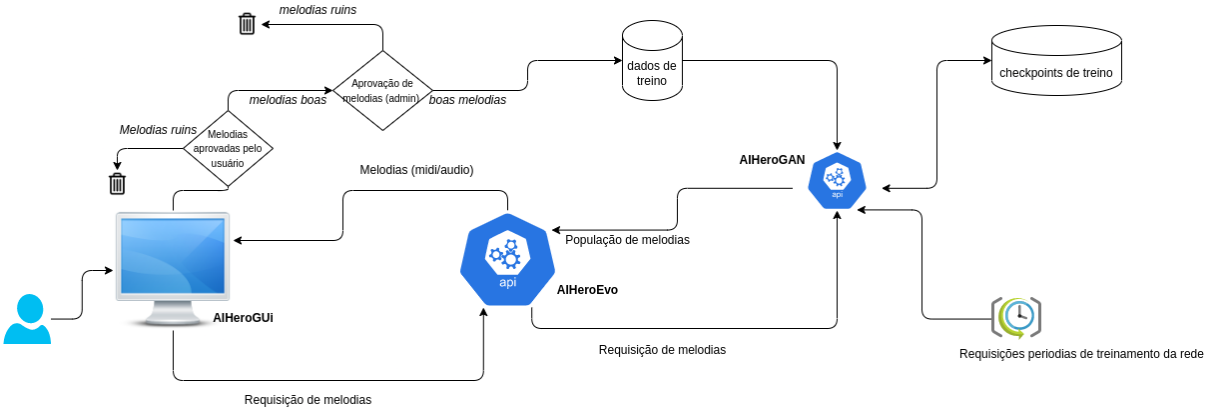

Figura 23 – Arquitetura geral do sistema. O usuário faz requisições ao módulo **AIHeroEvo** (Algoritmo evolucionário) passando as características pretendidas para a melodia. O **AIHeroEvo** utiliza um algoritmo evolucionário, de população inicial alimentada pela rede do módulo **AIHeroGAN**, para geração da melodia.

O usuário realiza uma requisição com a sequência de compassos desejados, o tom da melodia, e os acordes que serão impostos em cada compasso. Estes dados são inseridos através da interface gráfica **AIHeroGUI**, que pode ser vista na Figura 30. Tendo estas informações, o algoritmo irá gerar cada compasso separadamente.

Para a geração de um compasso, o módulo **AIHeroEvo** solicita ao módulo **AIHeroGAN** a geração de indivíduos para compor sua população inicial. Tendo a informação do tom e do acorde do compasso em questão, é possível saber qual é a função harmônica que deve ser empregada (como explicado em 2.1.1.5) e então requisita-se à GAN especialista nesta função harmônica para que gere as melodias. O módulo **AIHeroEvo** executa o processo evolutivo e retorna a melodia. O sistema então concatena os compassos gerados, agrega uma base musical de *blues* e transmite o resultado em formado MIDI para o usuário. A Interface gráfica é capaz de abrir este arquivo MIDI e tocar a melodia gerada no próprio navegador.

Este processo de comunicação entre a interface gráfica e o módulo gerador de melodias acontece conforme Figura 24. O *front-end* se comunica com o *back-end* através de requisições REST (Guilherme, 2020). O *front-end* realiza uma requisição do tipo POST para o *back-end*, com as especificações da melodia a ser gerada. O *front-end* recebe então o id daquela melodia como resposta. Esta requisição é armazenada em uma fila de processamento. Nesta fila, todas as requisições são processadas em ordem. Optou-se pela implementação de filas para habilitar requisições simultâneas por vários usuários. Como explicado anteriormente, o módulo **AIHeroEvo** gera um compasso por vez. Porém, esta ação é paralelizada em *threads*, para diminuição do tempo de resposta. Para que o *front-end*

receba a melodia gerada, é realizada uma requisição do tipo GET para o *back-end*. Caso a melodia ainda não esteja pronta, o *front-end* receberá uma resposta de código HTTP 404 (Mozilla, 2023). Caso a melodia esteja pronta, o arquivo MIDI é enviado.

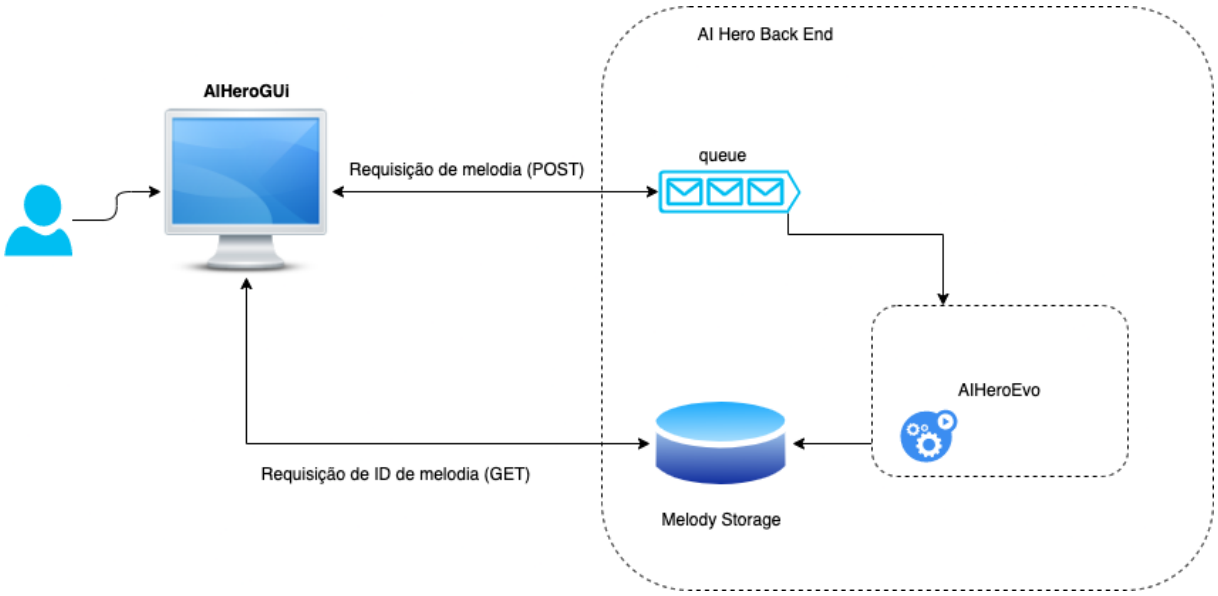

Figura 24 – Diagrama de requisição de melodias. O *front-end* se comunica com o *back-end* através de requisições do tipo REST. O processamento acontece por meio de uma fila de requisições, para que requisições de múltiplos usuários seja permitido

A seguir, a modelagem da arquitetura dos módulos internos do sistema será detalhada.

4.3 AIHeroGAN: As Redes Geradoras Adversárias por Função Harmônica

Esta seção descreve a arquitetura do módulo **AIHeroGAN**, que é conforme Figura 26 e utiliza um grupo de GANs especializadas por funções harmônicas. Cada uma das GANs especializadas possui a mesma topologia e arquitetura descrita no Capítulo 3.

O conceito de função harmônica é utilizado no sistema através da definição e treinamento de GANs especializadas em gerar melodias para cada tipo de função harmônica. Cada GAN é treinada com dados pertencentes a uma função harmônica específica.

Ao receber as especificações da melodia, é possível identificar a função harmônica analisando o tom principal e o acorde definido para um determinado compasso, conforme explicado em 2.1.1.5. Com esta informação, solicita-se a melodia à GAN correspondente àquela função harmônica. A figura 25 exemplifica a utilização das GANs especialistas na geração de compassos em uma melodia de *blues*.

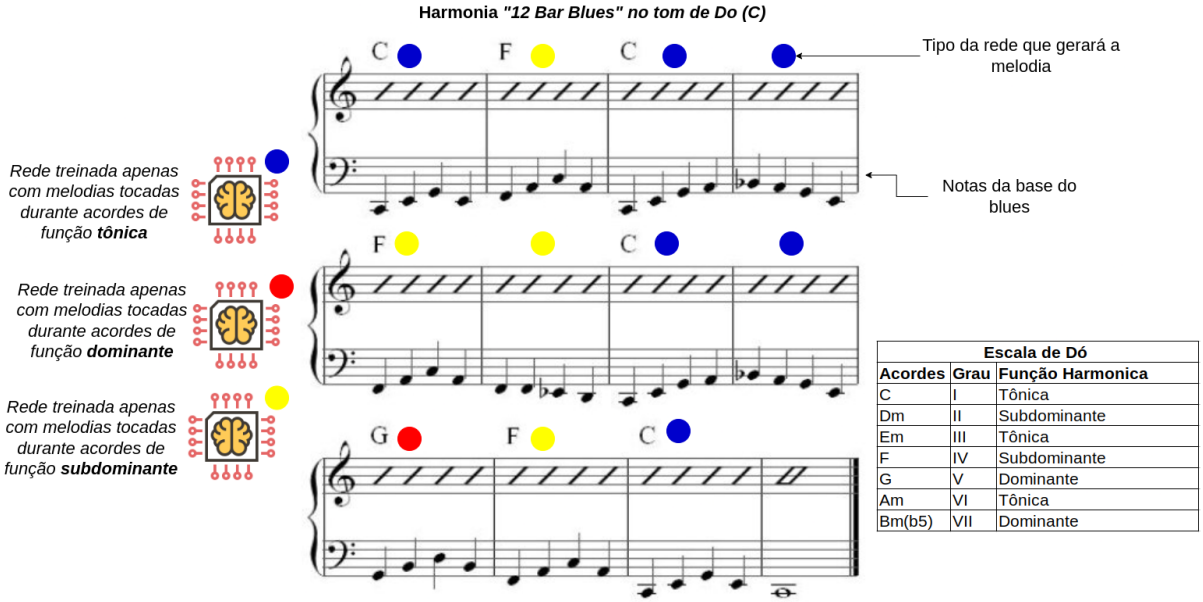

Figura 25 – Exemplo do uso do conceito de harmonia funcional aplicado ao contexto deste trabalho. Três redes geradoras de melodia são definidas. Uma para cada função harmônica: Tônica, Subdominante e Dominante. Ao se definir uma harmonia a ser executada, cada rede será responsável por gerar melodias apenas para as partes em que sua função harmônica é utilizada

O objetivo da implementação de GANs específicas é, uma vez que a música pode ser entendida como um processo de variações de dinâmicas, emular dinâmicas características de cada função harmônica, visando aumentar a coesão geral da melodia final. No contexto deste trabalho, onde são definidos geradores de compassos, este tipo de separação torna-se importante para que exista algum tipo de contexto em cada compasso a ser gerado.

4.4 AIHeroEVO: O Algoritmo Genético

Esta seção descreve as especificações consideradas e selecionadas para a implementação do algoritmo genético utilizado no trabalho.

4.4.1 Representação dos cromossomos

A representação das melodias é baseada na modelagem explicitada na Seção 3.2. Esta representação de toda a população apresenta-se como uma matriz de dimensão (N_{pop} x T x N_{midi}), onde N_{pop} é o tamanho da população, T é o número de divisões de tempo dentro de um compasso e N_{midi} é o número de notas *MIDI* a serem representadas. Cada cromossomo é representado conforme Figura 27

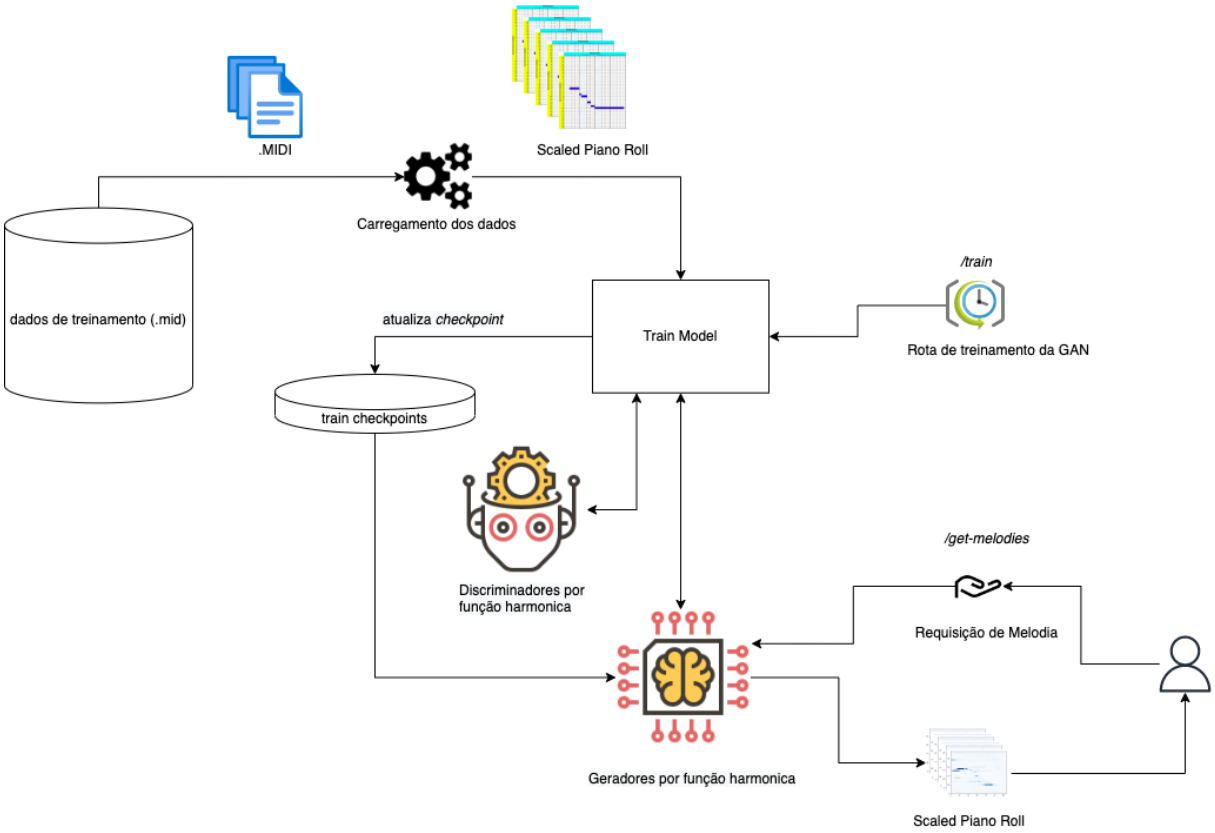

Figura 26 – Módulo **AIHeroGAN**: possui um grupo de GANs que foram treinadas de acordo com funções harmônicas específicas. Quando uma melodia é requisitada, tem-se no corpo da requisição qual é o tom da melodia e quais os acordes serão tocados em cada compasso. Com esta informação, é possível inferir as funções harmônicas e requisitar as GANs corretamente.

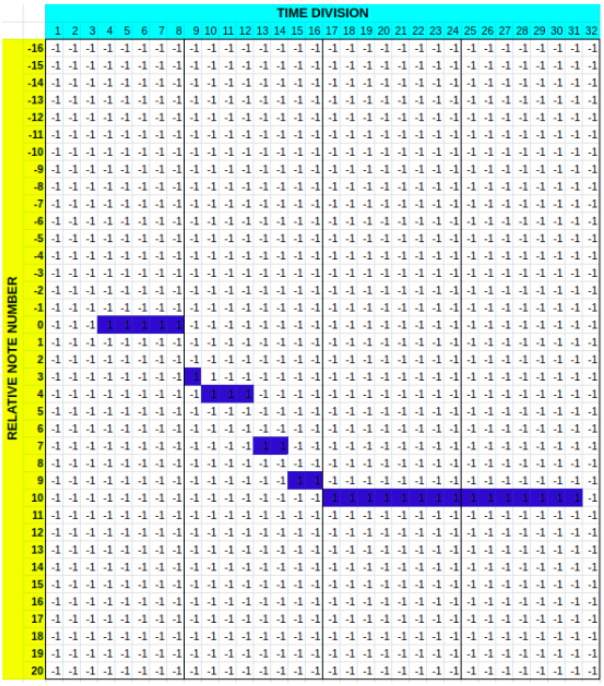

Figura 27 – Representação *Scaled Piano Roll*: Cada indivíduo é representado por uma matriz de dimensão (T x N_{midi}). As posições com valores iguais a -1 representam intervalos e as posições com valores iguais a 1 representam notas sendo tocadas.

4.4.2 Inicialização e Estrutura da População

Existem algumas possíveis formas de se inicializar a população de melodias. A primeira delas é pela geração de soluções aleatórias. Embora seja uma estratégia de rápida execução e que, em casos gerais, explora mais o espaço amostral, propiciando mais vantagem ao se encontrar o ótimo global, corre-se o risco de gerar melodias pouco naturais, o que dificulta o processo de evolução. Outra alternativa é iniciar a população de forma semi-aleatória, utilizando-se de processos geradores inspirados em melodias reais, como cadeias de Markov, ou redes neurais geradoras . Uma terceira opção é a inicialização da população utilizando um banco de melodias reais. Desta forma, garante-se que todas as melodias iniciais possuem um nível mínimo de qualidade e o algoritmo genético tratará de evoluir esta população no sentido definido pela função de *fitness*. Este tipo de abordagem de inicialização tende a ser o que gera melhores resultados, mas permite a interpretação de que o algoritmo genético não está gerando melodias, apenas modificando material já existente.

Tendo estas opções em vista, a estratégia de inicialização escolhida para este trabalho é a de se utilizar uma rede GAN (*Generative Adversarial Networks*), treinada com melodias reais, para inicialização da população. Desta forma, tem-se um processo gerador inspirado em melodias reais, porém não completamente "real" e humano.

4.4.3 Seleção

A seleção dá-se por meio de torneio. Neste tipo de seleção, um grupo de indivíduos é escolhido aleatoriamente e o melhor do grupo é selecionado. Quanto maior o grupo de indivíduos, maior a probabilidade de se selecionar indivíduos com *fitness* acima da média, como descrito em Andrew (2004). A escolha do número de indivíduos dá-se pelo parâmetro p_{tour}, que representa o percentual da população a ser escolhido para o torneio. Figura 28.

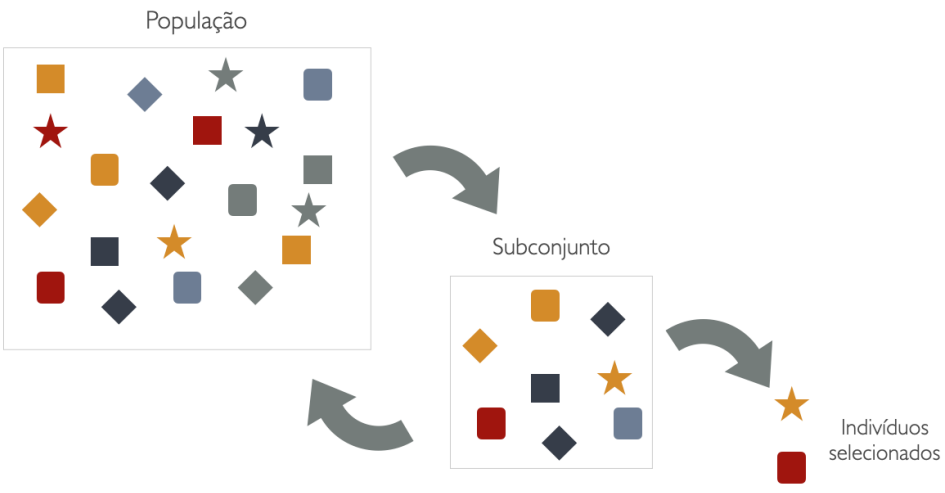

Figura 28 – Seleção por torneio

4.4.4 Cruzamento

O cruzamento ocorre após a fase de seleção e é responsável por gerar as soluções que farão parte da próxima geração de otimização. O cruzamento de duas soluções candidatas ocorre com probabilidade p_{cross}. Caso não ocorra o cruzamento, as duas soluções selecionadas são passadas para o próximo passo sem que haja modificações.

A estratégia de cruzamento foi implementada com base no *crossover* de um ponto Andrew (2004), porém com a particularidade de que os pontos de corte podem ser apenas entre divisões de tempo previamente definidas, a fim de manter-se a coesão musical na melodia, conforme demonstrado na Figura 29.

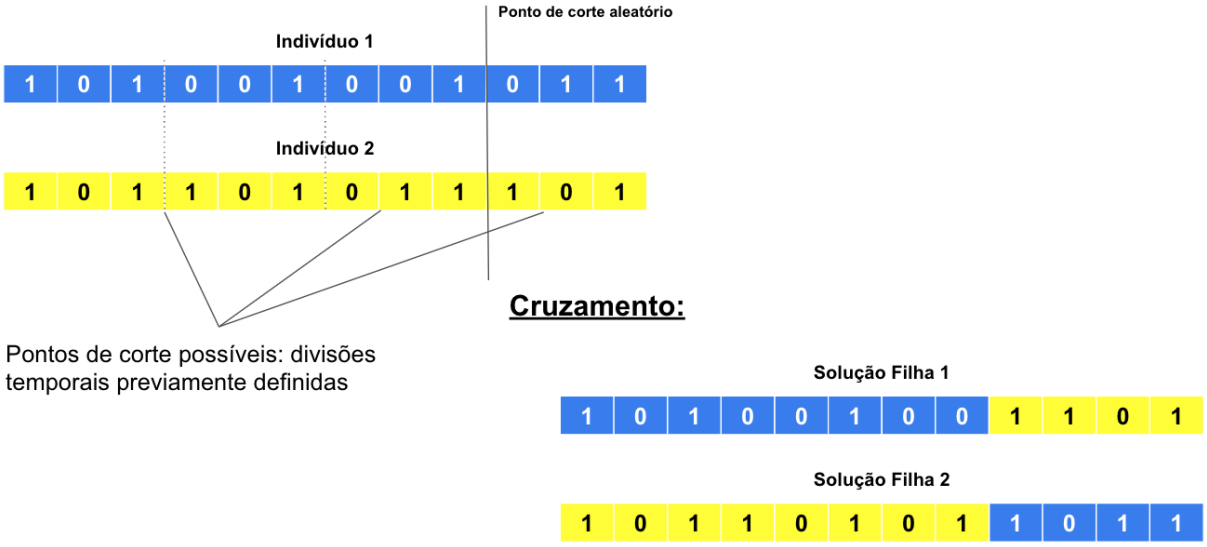

Figura 29 – Cruzamento por *crossover* de um ponto com pontos de corte definidos

4.4.5 Mutação

Após a fase de cruzamento, as soluções passam pela fase de mutação. No contexto deste trabalho, a mutação acontece com probabilidade p_{mut} e trata-se da substituição de uma solução por outra melodia gerada pela GAN, que gerou a população inicial do problema.

4.4.6 Avaliação dos indivíduos

Todo processo de seleção é guiado pela ordenação das soluções de acordo com um método de avaliação de qualidade. Quando trata-se de composições musicais geradas por computador, existem diferentes abordagens. Foram consideradas três opções para avaliação da função de *fitness*, como descrito em Freitas (2011):

- **Sem Aptidão**: Trata-se de um algoritmo genético sem qualquer avaliação de *fitness*

e portanto, nenhuma pressão seletiva. Para isso, é necessário garantir que todos os indivíduos inicializados na população sejam melodias que já possuem um "*fitness* mínimo".

- **Interativo**: É um modelo onde cada melodia é avaliada por um agente humano. Esta abordagem implica em alto tempo para a evolução do algoritmo e certa incerteza nos valores apontados por humanos, pois esta métrica de qualidade é subjetiva e varia entre indivíduos.
- **Automático**: Avaliação por meio de heurísticas para determinar se a melodia está de acordo com regras conhecidas da música. Esta abordagem implica na limitação da avaliação da qualidade musical a um conjunto de funções matemáticas estipuladas com base em conceitos teóricos de musica.

A opção escolhida foi a de calcular o *fitness* automaticamente através de uma serie de heurísticas. Foram definidas então 6 funções objetivo para a avaliação de uma melodia, conforme pode ser descrito na Tabela 1. Todas elas possuem valores que variam de -1 a 1.

Tabela 1 – Funções de *fitness*

Função
Notas coincidentes com o acorde
Notas coincidentes com a batida
Densidade de notas
Taxa de variedade de notas
Notas únicas
Notas fora da escala

Notas coincidentes com o acorde indica a taxa de notas da melodia que coincidem com a tríade do acorde em que aquela melodia está sendo tocada. Estas notas tendem a causar uma geração de conforto ao serem tocadas, pois não divergem do que é esperado dentro daquele acorde. A avaliação acontece conforme equação 4.1

$$f_1(x) = \frac{N_c(x)}{N_t(x)} \tag{4.1}$$

onde $N_c(x)$ é o número de notas da melodia que coincidem com as notas do acorde a ser tocado no compasso em questão e $N_t(x)$ é o número total de notas tocadas.

Notas coincidentes com a batida indica as notas que coincidem com a batida, ou seja, notas que tem seu início nas divisões do compasso (no caso deste trabalho, o compasso possui 4 divisões), conforme equação 4.2

$$f_2(x) = \frac{T_b(x)}{T_t(x)} \tag{4.2}$$

onde T_b é o número de notas da melodia que coincidem com as batidas do compasso e T_t é o número total de notas tocadas no tempo.

Densidade de notas indica a taxa de quantidade de notas no tempo e é dada pela equação 4.3

$$f_3(x) = \begin{cases} 0, \text{se } \frac{T_{on}(x)}{T} \leq d_{min} \\ 1, \text{se } \frac{T_{on}(x)}{T} \geq d_{max} \\ \frac{\frac{T_{on}(x)}{T(x)} - d_{min}}{d_{max} - d_{min}}, \text{caso contrário} \end{cases} \tag{4.3}$$

onde $T_{on}(x)$ é a quantidade de divisões de tempo que possuem notas sendo tocadas, d_{min} e d_{max} são os limites inferiores e superiores de densidade de notas, que neste trabalho são 0.2 e 0.8 e T é a divisão do tempo, que neste trabalho é 64.

taxa de variedade de notas indica a quantidade de diferentes alturas de notas em um compasso e é dada pela equação 4.4

$$f_4(x) = \begin{cases} 0, \text{se } \frac{N_{on}(x)}{N} \leq d_{min} \\ 1, \text{se } \frac{N_{on}(x)}{N} \geq d_{max} \\ \frac{\frac{N_{on}(x)}{N} - d_{min}}{d_{max} - d_{min}}, \text{caso contrário} \end{cases} \tag{4.4}$$

onde $N_{on}(x)$ é a quantidade de diferentes notas tocadas no compasso, d_{min} e d_{max} são os limites inferiores e superiores de variedade de notas, que neste trabalho são 0.05 e 0.15 e N é o numero de possíveis notas, que neste trabalho é 48.

Notas únicas indica a taxa de notas que são tocadas sem outra nota ser tocada simultaneamente conforme equação 4.5

$$f_5(x) = \begin{cases} 0, \text{se } \frac{T_{single}(x)}{T_{on}(x)} \leq d_{min} \\ 1, \text{se } \frac{T_{single}(x)}{T_{on}(x)} \geq d_{max} \\ \frac{\frac{T_{single}(x)}{T_{on}(x)} - d_{min}}{d_{max} - d_{min}}, \text{caso contrário} \end{cases} \tag{4.5}$$

onde $T_{single}(x)$ é a quantidade de notas únicas tocadas no compasso, d_{min} e d_{max} são os limites inferiores e superiores de variedade de notas, que neste trabalho são 0.3 e 1 e $T_{on}(x)$ é a quantidade de divisões de tempo que possuem notas sendo tocadas.

Notas fora da escala indica a taxa de notas que não possuem a escala desejada para a melodia e é dada pela equação 4.6

$$f_6(x) = \frac{N_{out}(x)}{N_t(x)} \tag{4.6}$$

onde $N_{out}(x)$ é o número de notas da melodia que não coincidem com as notas da escala a ser tocada no compasso em questão e $N_t(x)$ é o número total de notas tocadas.

A função objetivo é portanto conforme a equação 4.7

$$F_{fitness}(x) = \sum_{1}^{N} f_n(x) w_n \tag{4.7}$$

onde N é a quantidade de funções objetivo, que neste trabalho são 6, f_n é o valor da avaliação da função objetivo de índice n para uma certa melodia e w_n é o peso atribuído pelo usuário àquela função.

Outra opção considerada para este trabalho foi de se usar a resposta do Discriminador da GAN, descrito na Seção 3.3, como mais uma influência na função de *fitness* do algoritmo genético. Desta forma, a função de *fitness* $F_{fitness\_alt}$ ficaria conforme a equação 4.8

$$F_{fitness\_alt}(x) = 0.8 * F_{fitness}(x) + 0.2 * D(x) \tag{4.8}$$

onde $D(x)$ é o resultado da avaliação do Discriminador da GAN para a melodia em questão e $F_{fitness}$ é a função de *fitness* previamente definida.

Esta opção foi experimentada e comparada e os resultados estão na Seção 5.5

4.4.7 Substituição dos pais

Ao fim de uma geração, toda a população anterior é descartada e substituída pelos indivíduos selecionados para os passos de cruzamento e mutação. Após isto, uma outra geração é executada, a menos que seja atingido o critério de parada.

4.4.8 Critério de parada

Após experimentações, optou-se por utilizar dois diferentes critérios de parada:

- **Parada por limite máximo de gerações** N_{ger}: É quando se atinge o número máximo estabelecido de evolução das soluções.

- **Parada por proximidade do *fitness* máximo**: Como as funções de *fitness* variam de 0 a 1 e os pesos w_n são informados *a priori*, é possível calcular o *fitness* máximo que uma função pode atingir pela fórmula $F_{maxfit} = \sum_{1}^{N} w_n$. Termina-se a execução quando uma solução possui valor de *fitness* $F_{fitness} \geq F_{maxfit} * p_{parada}$, onde p_{parada} é um hiperparâmetro para determinar o percentual de proximidade de uma solução com o *fitness* máximo.

4.5 AIHeroGUI: A Interface Gráfica

A interface gráfica do sistema é o meio do usuário gerar melodias específicas e pode ser vista na Figura 30. Através da interface, é possível selecionar melodias geradas por dados reais (dados utilizados no processo de treinamento), melodias geradas somente pela GAN e melodias geradas pela arquitetura GAN e algoritmo evolucionário. Ao selecionar a terceira opção, a seleção de pesos é habilitada ao usuário, que pode selecionar a taxa de "variedade de notas" e a taxa de "caos" pretendida. Estes dois valores são linearmente correlacionados com os pesos das funções de *fitness* no algoritmo genético. Optou-se por reduzir a quantidade de hiper-parâmetros para caos e variedade de notas pelo usuário para simplificar o processo de escolha e melhorar a experiência de uso.

O usuário também pode definir a sequência de acordes pretendida, dentro de 12 compassos. Na implementação atual, não está habilitada a possibilidade de se escolher mais de 12 compassos e nem de trocar a base musical para outra que não seja *blues*.

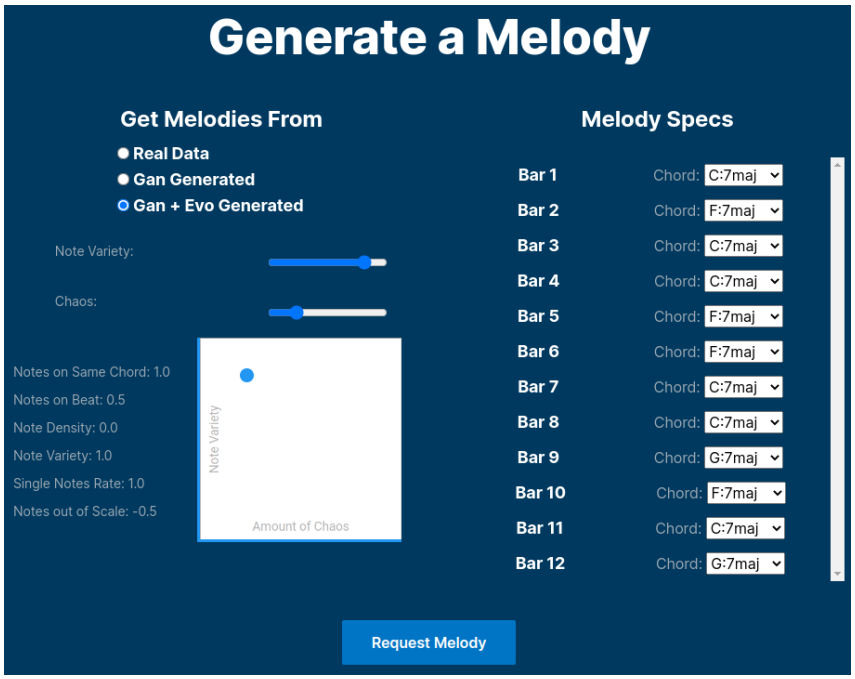

Figura 30 – Interface gráfica da aplicação AIHero. O usuário pode requisitar melodias reais, que são melodias formadas apenas por dados de treinamento, melodias geradas somente pelas GANs e melodias geradas pela arquitetura GAN com o algoritmo evolucionário. Ao selecionar a terceira opção, a escolha dos pesos é habilitada, possibilitando que o usuário tune a função de *fitness*. O usuário deve também especificar a sequência de acordes que será empregada em cada compasso.

Capítulo 5

Experimentos Computacionais

Este capítulo contém os três experimentos computacionais realizados neste trabalho. O primeiro possui o objetivo de verificar se a metodologia de aumento de dados proposta apresenta indícios de melhoria na qualidade das melodias geradas após o processo de treinamento da GAN. O segundo experimento avalia a arquitetura proposta neste trabalho, comparando-a com uma arquitetura formada somente por uma GAN e uma arquitetura composta por uma LSTM. O terceiro experimento avalia os efeitos da inclusão do discriminador na função de *fitness* utilizada pelo algoritmo genético.

5.1 Métricas de Avaliação

5.1.1 Definição

Para se realizar um teste comparativo entre dois modelos, é necessário selecionar uma ou mais métricas que avaliarão a qualidade das amostras geradas. No caso de avaliação da qualidade de melodias musicais, esta tarefa se torna complexa, uma vez que não existe uma forma definida de se avaliar uma peça musical. Os principais caminhos possíveis são o de requisitar a avaliação de humanos e o de realizar inferências com base em características extraídas de uma melodia musical. A primeira opção, para que se tenha um resultado estatisticamente significativo, requer um grupo heterogêneo de pessoas em considerável quantidade. Além da necessidade de formar este conjunto controlado de pessoas, necessita-se de uma maneira de coleta das avaliações que, no contexto deste trabalho, requer o desenvolvimento de um sistema de execução de melodias e aquisição das avaliações, que aumentaria a complexidade da implementação do trabalho. Esta opção será considerada para trabalhos futuros.

A opção de utilizar-se de métricas quantitativas baseadas em características da melodia se torna a opção viável, mesmo que ao adotar esta estratégia, tenhamos comparações de características da melodia e não da qualidade da melodia como um todo.

O trabalho de Yang e Lerch (2020) propõe as métricas baseadas em características musicais, que serão utilizadas nos experimentos 2 e 3:

- ***Total used pitch*** (*total_used_pitch*): O número de diferentes alturas em uma amostra. A saída é um valor escalar.
- ***Pitch range*** (*pitch_range*): É a subtração da nota de tonalidade mais alta pela de tonalidade mais baixa. A saída é um valor escalar para cada amostra.
- ***Total pitch class histogram*** (*total_pitch_class_histogram*): É a distribuição de alturas tonais em uma oitava, considerando as ocorrências de notas na escala cromática. Serve para indicar a concentração de ocorrências de algumas notas em comparação com as demais independente da oitava.
- ***Pitch class transition matrix*** (*pitch_class_transition_matrix*): Esta métrica, representada por uma matriz de dimensão 12x12, captura eventos de transição de tonalidade.
- ***Average inter-onset-interval*** (*avg_IOI*): Descreve o intervalo médio intra-set, ou seja, é o valor médio, em segundos, de tempo entre notas.
- ***Total used note*** (*total_used_note*): Descreve o número total de notas utilizadas em uma amostra.
- ***Note length histogram*** (*note_length_hist*): Esta métrica é o histograma das durações de notas.
- ***Note length transition matrix*** (*note_length_transition_matrix*): Está métrica visa descrever mudanças em dinâmicas rítmicas.

Estas métricas visam reduzir características musicais em valores e distribuições de dados. Desta forma, é possível compará-los e analisar as distâncias entre eles para ter-se um indicador de proximidade entre dados gerados e dados reais. As medidas de distância sugeridas em Yang e Lerch (2020) são a *Kullback-Leibler Divergence* (KLD) e a *Overlapped Area* (OA).

Kullback-Leibler Divergence (KLD) é uma medida de distância que indica o quão diferente é uma certa distribuição de probabilidade de uma outra distribuição (Kullback e Leibler, 1951), enquanto a *Overlapped Area* (OA) indica o percentual da área da distribuição de densidade de probabilidade de um conjunto de dados que está sobreposta à distribuição de densidade de probabilidade do outro conjunto de dados. Enquanto o KLD não possui um valor limite e é assimétrico, o OA é de 0 a 1 e simétrico.

A comparação de cada característica portanto se dará pela avaliação das distâncias computadas em relação aos dados de treinamento. Considera-se que, quanto mais próximo dos dados de treinamento uma melodia está, maior é sua qualidade.

Outra métrica considerada para a avaliação das melodias geradas é a *Frechèt Inception Distance* (FID), indicador de qualidade de GANs amplamente utilizada, como em Karras et al. (2019) e Karras et al. (2020). É uma métrica de avaliação quantitativa que avalia a similaridade entre amostras reais do problema e amostras geradas (Heusel et al., 2017). O valor é o resultado da distância entre a distribuição de probabilidade dos dados reais $p_w(.)$ e a distribuição dos dados gerados $p(.)$. Esta métrica é baseada no *Inception Score* (IS), (Szegedy et al., 2016), métrica que usa uma rede já treinada em uma tarefa de classificação de imagens para realizar a avaliação da qualidade.

Diferente do IS o FID compara a média e a covariância de uma camada profunda desta rede pré-treinada. Esta camada é encodada como gaussianas multidimensionais e a distância entre elas é mensurada utilizando a distância de Fréchet (Fréchet, 1957). A *Fréchet Inception Distance* (FID) $d(.,.)$ entre a distribuição gaussiana de média e covariância (m, C) obtida de $p(.)$ e a distribuição gaussiana (m_w, C_w) obtida de $p_w(.)$ é dada pela equação 5.1:

$$d^2((m, C), (m_w, C_w)) = ||m - m_w||_2^2 + tr(C + C_w - 2(CC_w)^{1/2}) \tag{5.1}$$

onde m e m_w são as médias das distribuições e C e C_w são as covariâncias.

Esta métrica possui bastante evidência de uso no contexto de imagens e não no contexto musical. Mas, pela ausência de métricas quantitativas dedicadas para avaliação de GANs no contexto musical e pelo fato de que na arquitetura proposta as melodias são geradas de forma similar a imagens, julga-se que esta métrica pode fornecer indícios significativos de qualidade, principalmente no contexto de comparação entre arquiteturas.

5.1.2 Aquisição das métricas

A aquisição da *Fréchet Inception Distance* (FID) dá-se conforme demonstrado no Algoritmo 1. Define-se inicialmente a quantidade de coletas necessárias num_fids e o percentual $sample\_perc$ de melodias que serão amostradas para o calculo da métrica. Então inicia-se o *loop* e a aquisição destes dados.

Para a aquisição das demais métricas, o fluxo é conforme o Algoritmo 2. Realiza-se a mesma da quantidade de coletas necessárias $num\_metrics$ e o percentual $sample\_perc$ de melodias que serão amostradas para o cálculo da métrica. E então, durante o *loop* realiza-se a aquisição tanto do KLD quanto do OA.

Algorithm 1 Aquisição do FID

```
sample_perc ← 0.20                                  ▷ % de melodias em cada coleta
num_fids ← 100                                      ▷ número de coletas
fids ← []
for 1 to num_fids do
    model_sample ← generated_data.sample(sample_perc)   ▷ pega 20% de amostras aleatoriamente
    fids.append(calculate_fid(model_sample))
```

Algorithm 2 Aquisição das demais métricas

```
sample_perc ← 0.10                                  ▷ % de melodias em cada coleta
num_metrics ← 100                                   ▷ número de coletas
kld_metrics ← []                                    ▷ KLD
oa_metrics ← []                                     ▷ OA
for i to num_fids do
    model_sample ← generated_data.sample(sample_perc)
    training_sample ← training_data.sample(sample_perc)
    kld_metrics.append(calculate_kld(model_sample, training_sample))
    oa_metrics.append(calculate_oa(model_sample, training_sample))
```

5.2 Testes Estatísticos

Nos experimentos apresentados neste capítulo, é realizada a comparação de amostras de múltiplos grupos distintos. Para realizar-se comparações de resultados de forma que se possa afirmar, com determinada confiabilidade, que há diferença estatística entre os grupos, deve-se aplicar um teste estatístico. A escolha do teste estatístico mais adequado varia de acordo com as características das amostras coletadas, tais como, o formato da sua distribuição, e se há homocedasticidade (homogeneidade de variância) entre os grupos. No contexto deste trabalho o testes utilizados são os testes de Friedman e de Nemenyi.

O teste de Friedman (Friedman, 1937) é um teste não-paramétrico que avalia repetidamente a variância dos rankeamentos entre as amostras. O teste de Friedman indica que ao menos um dos grupos possui uma distribuição que difere dos outros grupos. Para identificar a relação entre as distribuições, o teste de Nemenyi (Nemenyi, 1963) é aplicado. É um teste *post-hoc* que realiza comparações pareadas com o objetivo de descrever as relações entre todos os grupos do experimento.

5.3 Avaliação de metodologia de Aumento de Dados para treinamento da GAN

Este experimento tem o objetivo de verificar se a utilização da estratégia de *data augmentation* proposta na Sessão 3.2.1 melhora as melodias geradas. Para isto, é feita uma comparação entre os dados originais, dados aumentados e dados compostos de réplicas dos originais. A métrica de comparação é o FID, descrito na Sessão 5.1.

5.3.1 Modelagem do experimento

A representação dos dados é a *scaled piano roll*, descrita na Sessão 3.2 e a estratégia de *data augmentation* utilizada está descrita na Sessão 3.2.1.

Para os três sistemas do teste, as topologias das redes geradora e discriminadora são conforme descritas na Sessão 3.3. Esta mesma GAN é treinada com os 3 diferentes *datasets*: o Original, que é o conjunto de dados que possui apenas amostras reais únicas, o Aumentado, que é o *dataset* que passa pelo processo de *data augmentation*, e o Replicado, que possui o mesmo tamanho do Aumentado, porém composto apenas de cópias do *dataset* original. É importante a introdução do *dataset* Replicado no teste para avaliar-se o efeito da repetição dos dados em comparação com o aumento estratégico. A seguinte estratégia foi aplicada:

- 1 - Mudança de duração: $k_1 = 6$
- 2 - Adição do quinto grau: $k_2 = 3$
- 3 - Mudança de oitava: $k_3 = 1$

Portanto, utilizando-se as 55 amostras dos dados de treinamento, obtém-se 3080 amostras após o processo de aumento, de acordo com o processo descrito em 3.2.2.

As especificações para o treinamento da GAN foram as abaixo:

- Épocas: 60
- Tamanho do *batch*: 50
- Tamanho máximo de amostras utilizadas para a medição de qualidade: 100

Cada época representa uma iteração do treinamento, onde todas as amostras, divididas em *batches*, foram utilizadas para otimizar as perdas tando to Gerador quando do Discriminador. Houve a necessidade de definir um tamanho máximo de amostras utilizadas para a medição de qualidade pois o cálculo do FID poderia consumir toda a memória disponível no computador de teste.

O processo de treinamento foi executado 55 vezes para todos os 3 conjuntos de dados. Os resultados estão apresentados a seguir.

5.3.2 Resultados e Discussão

O *boxplot* da distribuição de todas as avaliações de FID computadas após cada processo de treinamento pode ser vista na Figura 31. As distribuições indicam que os dados Aumentados obtiveram melhores valores de FID, em média. Mas para identificar uma diferença estatisticamente significativa, é necessário realizar o teste de Friedman, seguido pelo teste de Nemenyi (ambos descritos no capítulo 5.2)

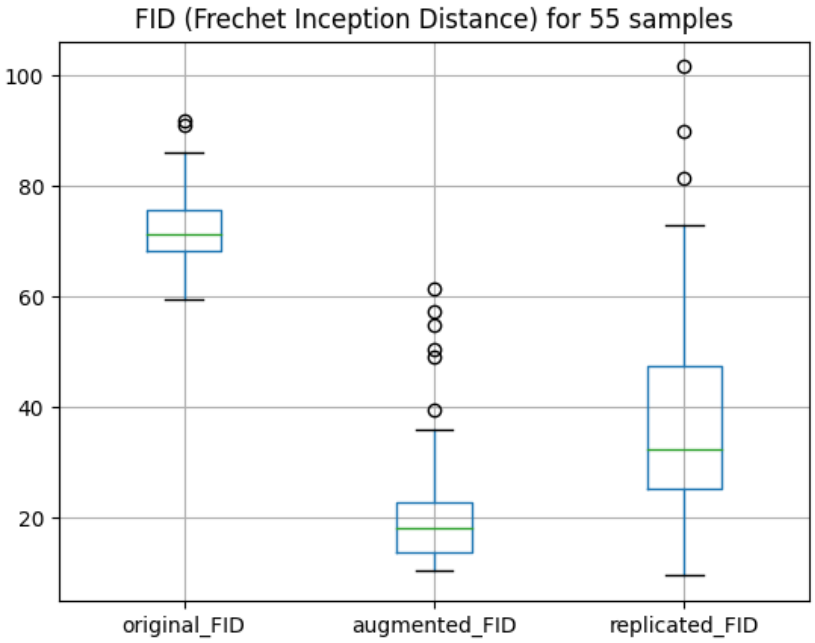

Figura 31 – Distribuição das avaliações de FID para 55 treinamentos utilizando cada tipo de *dataset*

A Tabela 2 apresenta a média e desvio padrão dos FIDs para cada treinamento e a Tabela 3 apresenta os p-valores encontrados após execução do teste de Nemenyi.

Tabela 2 – Resultados de treinamento

Dataset	**Tamanho do Dataset**	**Fréchet Inception distance (FID)**
Original	55	72.4334±6.44
Aumentado	3080	**21.7541±12.26**
Replicado	3080	39.5702±20.15

Analisando os p-valores apresentados na Tabela 3, é possível afirmar com 99% de confiança que as três distribuições diferem entre si. O *dataset* Aumentado possui o melhor

Tabela 3 – P-valores do teste de Nemenyi (FID)

datasets	*Original*	*Aumentado*	*Replicado*
Original	1	0.001	0.001
Aumentado	0.001	1	0.001
Replicado	0.001	0.001	1

valor médio de FID, seguido pelo Replicado e, por último, pelo Original. Pelos resultados apresentados, pode-se notar a influência positiva da estratégia de *data augmentation* proposta, que supera os efeitos de simplesmente inflar o conjunto de dados por replicação.

5.4 Comparação entre Arquitetura GAN Evolucionária, GAN e LSTM

Neste experimento, a arquitetura proposta, composta de um algoritmo evolucionário alimentado por uma GAN (atribuída com a sigla EVO), é comparada com um modelo composto apenas das GANs e um modelo composto de uma LSTM. Através deste experimento deseja-se perceber qual o efeito da adição do algoritmo evolucionário para melhoria da qualidade de melodias geradas pela GAN e como este resultado se compara com uma rede que considera o efeito temporal.

5.4.1 Modelagem do experimento

A representação dos dados é a *scaled piano roll*, descrita na Sessão 3.2.

Para a GAN, as topologias das redes geradora e discriminadora são conforme descritas na Sessão 3.3. O conjunto de dados utilizado passa pela estratégia de aumento de dados proposta na Sessão 3.2.1 e validada no experimento da Sessão 5.3. Os parâmetros de aumento de dados são os abaixo:

- 1 - Mudança de duração: $k_1 = 6$
- 2 - Adição do quinto grau: $k_2 = 2$
- 3 - Mudança de oitava: $k_3 = 1$

Os dados foram replicados 5 vezes antes de passarem pela estratégia de *data augmentation*. Isso foi feito para que se possibilite um aumento na diversidade de pertubações causadas pelo aumento de dados.

As especificações para o treinamento da GAN foram as abaixo:

- Épocas: 410
- Tamanho do *batch*: 50

Escolheu-se um número alto de épocas para garantir que as perdas do gerador e discriminador estivessem em equilíbrio.

Para a LSTM, decidiu-se pela seguinte topologia: Uma camada LSTM de 300 unidades, seguida por uma camada de *dropout* e uma camada densa, com cada saída representando uma possível nota.

Os parâmetros de aumento de dados foram os abaixo (mesmos da GAN):

- 1 - Mudança de duração: $k_1 = 6$
- 2 - Adição do quinto grau: $k_2 = 2$
- 3 - Mudança de oitava: $k_3 = 1$

Diferente da GAN, não houve replicação prévia de dados. Com a replicação, o treinamento da LSTM exigia mais memória do que havia disponível.

As especificações para o treinamento da LSTM foram as abaixo:

- Épocas: 160
- Tamanho do *batch*: 50

Este número de épocas foi escolhido pois percebeu-se que após 160 épocas não havia melhoria substancial na perda da rede.

A arquitetura EVO utiliza as mesmas GANs descritas acima. A arquitetura do algoritmo evolucionário é descrita na Sessão 4.4 e os hiperparâmetros foram os abaixo:

- T : 64 (número de divisões de tempo)
- N_{midi} : 48 (número de alturas do SPR)
- N_{pop} : 200 (tamanho da população)
- p_{tour} : 30% (percentual de torneio)
- p_{cross} : 50% (percentual de cruzamento)
- p_{mut} : 5% (probabilidade de mutação)

Tabela 4 – Pesos das funções de *fitness*

Função	**Peso**
Notas coincidentes com o acorde	0.5
Notas coincidentes com a batida	0.3
Densidade de notas	0.3
Taxa de variedade de notas	0.4
Notas únicas	0.2
Notas fora da escala	-0.3

Os pesos selecionados para as funções de *fitness* estão disponíveis na Tabela 4.

Para cada arquitetura, foram geradas 40 melodias de 12 compassos, ou seja, 480 compassos. As métricas foram colhidas conforme descrito na Sessão 5.1. Cada métrica foi colhida 100 vezes e os resultados estão descritos a seguir.

5.4.2 Resultados e Discussão

Ao analisar-se os resultados comparativos do FID, é possível notar que a arquitetura EVO apresenta menor média dentre os grupos e os p-valores do teste de Nemenyi, disponíveis na Figura 32, indicam que a diferença é estatisticamente significativa (considerando 99% de confiança).

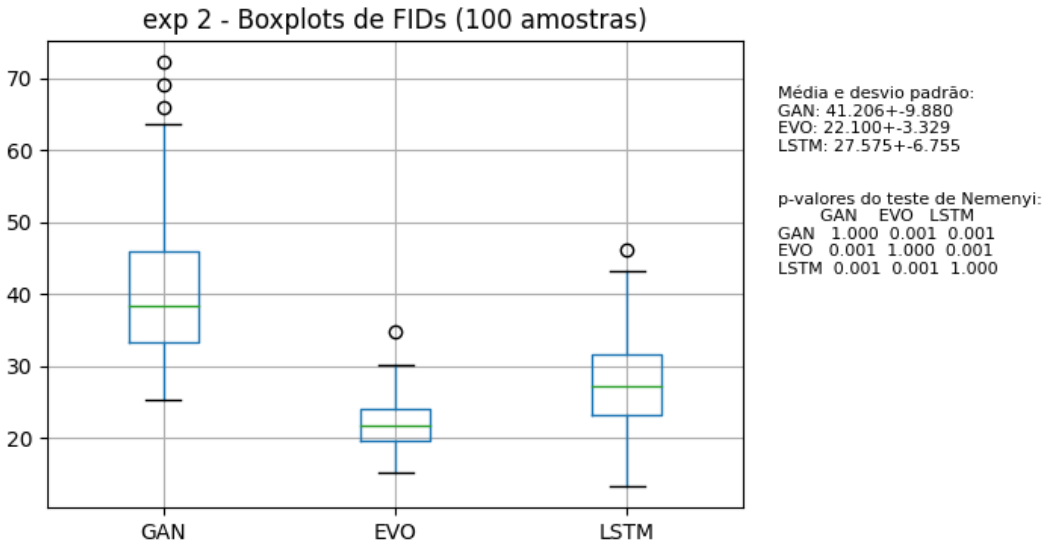

Figura 32 – Os Boxplots dos FIDs, junto com o resultado do teste de Nemenyi, indicam que a arquitetura EVO apresenta menor valor de FID e que este resultado é estatisticamente significativo

Portanto, para uma confiabilidade de 99%, pode-se afirmar que a arquitetura EVO apresentou melhor qualidade ao analisar a métrica FID em comparação com a GAN

e a LSTM. Nota-se também que a LSTM possui melhores resultados de FID que a GAN e esta também é uma diferença estatisticamente significativa.

Tabela 5 – Resultados das avaliações de FID

Modelo	**Fréchet Inception distance (FID)**
GAN	41.206±9.880
EVO	**22.100**±3.329
LSTM	27.575±6.755

O resultado das demais métricas está disponível na Tabela 6. Células em negrito indicam as colunas que obtiveram melhores valores de qualidade com diferença estatisticamente significativa, considerando confiabilidade de 95%.

Tabela 6 – Resultados das demais métricas

	KLD			OA		
Métrica	GAN	EVO	LSTM	GAN	EVO	LSTM
total_used_pitch	**0.11**±0.08	**0.11**±0.06	0.31±0.13	**0.78**±0.05	0.74±0.05	0.65±0.08
pitch_range	**0.09**±0.06	**0.08**±0.06	**0.09**±0.07	0.56±0.06	0.64±0.06	**0.72**±0.06
avg_IOI	0.14±0.11	0.11±0.06	**0.07**±0.06	**0.80**±0.05	**0.77**±0.05	0.63±0.06
total_used_note	0.07±0.06	**0.04**±0.04	0.17±0.08	**0.79**±0.06	**0.81**±0.05	0.47±0.05
total_pitch_class_histogram	**0.38**±0.19	**0.34**±0.12	0.96±0.26	0.63±0.08	**0.74**±0.03	0.46±0.05
note_length_hist	0.16±0.11	**0.07**±0.06	0.28±0.14	**0.80**±0.08	**0.79**±0.06	0.72±0.06
note_length_transition_matrix	**0.29**±0.24	0.41±0.33	0.27±0.05	0.62±0.09	**0.80**±0.05	0.76±0.52
pitch_class_transition_matrix	0.43±0.25	**0.08**±0.07	0.31±0.19	0.56±0.09	**0.87**±0.06	0.51±0.08

Para os valores de KLD, percebe-se que a arquitetura EVO só apresenta qualidade inferior em duas métricas. *avg_IOI* e *note_lenght_transition_matrix*. Uma destas tem resultado melhor para a LSTM e a outra tem resultado melhor para a GAN.

Para os valores de OA, percebe-se que a arquitetura EVO só apresenta qualidade inferior em duas métricas. *pitch_range* e *total_used_pitch*. Uma destas tem resultado melhor para a LSTM e a outra tem resultado melhor para a GAN.

No geral, pode-se afirmar que a arquitetura EVO supera ou empata com as demais e, considerando todas as métricas, é a arquitetura que apresenta maior proximidade com os dados de treinamento e, portanto, maior qualidade. As comparações detalhadas estão disponíveis no Apêndice A.

Considerando os resultados das comparações de FID e das demais métricas, é possível afirmar com 95% de confiança que a arquitetura EVO supera os demais modelos.

5.5 Discriminador da GAN como função de *fitness* do algoritmo evolucionário

Neste experimento, deseja-se avaliar a mesma arquitetura EVO do experimento anterior, comparando-a com uma arquitetura composta do mesmo algoritmo evolucionário alimentado por uma GAN, porém com uma modificação na função de *fitness* que leva em consideração a resposta da rede discriminadora da GAN. Esta arquitetura será chamada de EVO_ALT. Considerando que o resultado da função de *fitness* da arquitetura EVO é f_{evo}, a nova função de *fitness* é descrita pela equação **??**.

$$f_{evoalt}(x) = 0.8 * f_{evo}(x) + 0.2 * D(x) \tag{5.2}$$

onde $D(x)$ é o resultado da avaliação do Discriminador da GAN para a melodia em questão.

Este experimento tem como objetivo identificar se as avaliações do Discriminador podem ajudar no processo evolutivo.

5.5.1 Modelagem do experimento

As especificações da modelagem de dados, *data augmentation* e hiperparâmetros da arquitetura EVO são idênticas às do experimento disponível na Sessão 5.4.

Para as duas arquiteturas, foram geradas 40 melodias de 12 compassos, ou seja, 480 compassos. As métricas foram colhidas conforme descrito na Sessão 5.1. Cada métrica foi colhida 100 vezes e os resultados estão descritos a seguir.

5.5.2 Resultados e Discussão

Ao analisar-se os resultados comparativos do FID, é possível notar que a arquitetura EVO_ALT apresenta menor média dentre os grupos e os p-valores do teste de Nemenyi, disponíveis na Figura 33, indicam que a diferença é estatisticamente significativa (considerando 95% de confiança).

Portanto, para uma confiabilidade de 95%, pode-se afirmar que a arquitetura EVO_ALT apresentou melhor qualidade ao analisar a métrica FID em comparação com EVO.

O resultado das demais métricas está disponível na Tabela 8. Células em negrito indicam as colunas que obtiveram melhores valores de qualidade com diferença estatisticamente significativa, considerando confiabilidade de 95%.

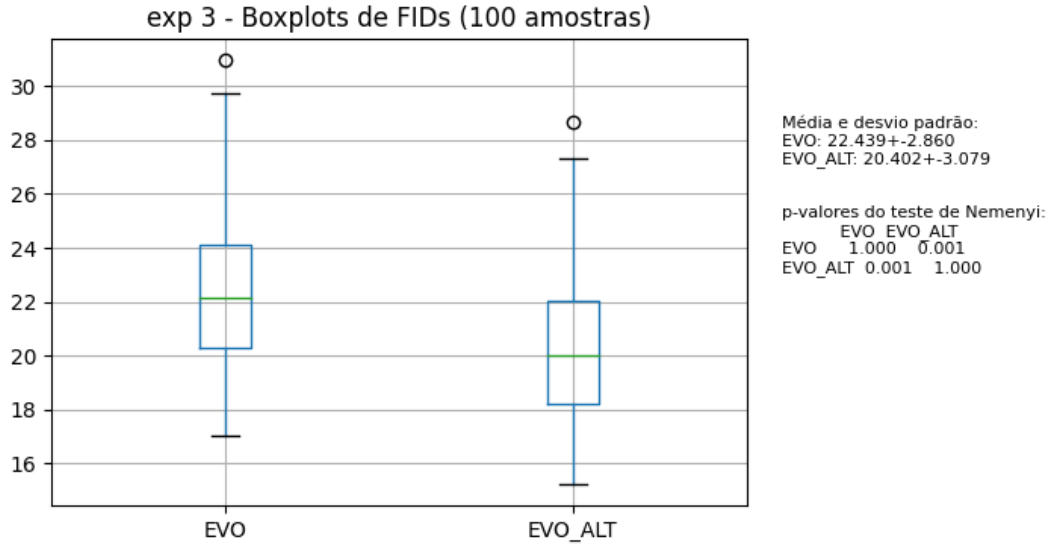

Figura 33 – Os Boxplots dos FIDs, junto com o resultado do teste de Nemenyi, indicam que a arquitetura EVO_ALT apresenta menor valor de FID e que este resultado é estatisticamente significativo

Tabela 7 – Resultados - FID

Modelo	**Fréchet Inception distance (FID)**
EVO	22.439±2.860
EVO_ALT	20.402±3.079

Tabela 8 – Resultados experimento 3

	KLD		OA	
Métrica	EVO	EVO_ALT	EVO	EVO_ALT
total_used_pitch	0.113±0.062	0.101±0.060	0.736±0.053	**0.747**±0.050
pitch_range	0.082±0.055	0.077±0.060	0.640±0.056	**0.652**±0.054
avg_IOI	0.110±0.055	0.113±0.069	0.771±0.045	0.767±0.048
total_used_note	0.044±0.038	0.035±0.032	0.810±0.053	0.803±0.051
total_pitch_class_histogram	0.339±0.122	0.330±0.147	0.736±0.031	**0.752**±0.033
note_length_hist	0.072±0.059	0.078±0.051	0.794±0.063	**0.823**±0.054
note_length_transition_matrix	0.410±0.325	0.401±0.326	0.801±0.048	0.810±0.046
pitch_class_transition_matrix	0.078±0.068	0.070±0.047	0.871±0.055	0.869±0.053

Para os valores de KLD, percebe-se que, pelo teste de Nemenyi, não foi possível identificar diferença estatisticamente significativa para nenhuma das métricas. Já para os valores de OA, foi possível identificar diferenças estatisticamente significativas em 4 das 8 métricas. As comparações detalhadas estão disponíveis no Apêndice B.

Considerando os resultados das comparações de FID e das demais métricas, é

possível afirmar com 95% de confiança que a arquitetura EVO_ALT apresenta resultados de qualidade superior à arquitetura EVO.

Capítulo 6

Conclusão

O problema de geração de música por computadores possui uma ampla diversidade de abordagens e através deste trabalho foi possível explorar mais uma. O estudo sobre emulação artística por computadores demonstra tanto as possibilidades de técnicas de algoritmo quanto nos aproxima dos processos humanos de criação artística. E neste trabalho procurou-se explorar estas duas vertentes, tanto entendendo a teoria musical utilizada por humanos durante a composição de uma peça, quanto adequando técnicas computacionais para este fim.

Este trabalho teve como objetivo propor um sistema gerador de melodias musicais visando simular o processo de improvisação musical. Pode-se notar durante o trabalho a exposição da abordagem proposta, explicitando a arquitetura e as estratégias adotadas para o cumprimento da tarefa de se emular o processo de improvisação. A arquitetura proposta contou com o desenvolvimento de um sistema que possui um algoritmo genético e Redes Geradoras Adversárias especialistas por cada função harmônica.

De acordo com os experimentos executados, pode-se afirmar que a estratégia de aumento de dados proposta aumentou a qualidade das GANs após o processo de treinamento. Foi também possível observar que a arquitetura proposta, composta por um algoritmo genético e GANs, gerou melodias de qualidade superiores a melodias geradas somente por GANs ou por uma LSTM e também foi possível verificar a influência positiva do Discriminador da GANs quando utilizado na função de *fitness* do algoritmo genético.

É preciso notar que a arquitetura proposta possui limitações. O sistema é responsável por gerar apenas a linha melódica da música, e não a base. É possível, em um trabalho futuro, desenvolver um algoritmo focado em gerar bases musicais para que a geração da música seja feita por completo.

Outra limitação é que o treinamento para os experimentos realizados nesta dissertação foi realizado com 55 compassos gravados pelo autor, considerando apenas frases musicais do ritmo *blues*. Este número, por ser pequeno, interfere diretamente na qualidade

das melodias geradas, mesmo utilizando-se da estratégia de aumento de dados proposta e a geração para outros estilos musicais não é possível. Um possível trabalho futuro é o de realizar a extração de melodias, acordes e tons, em arquivos MIDI, para que se possa realizar o treinamento usando arquivos de bases de dados públicas, com músicas completas, o que aumentaria o repertório de treinamento das GANs.

As métricas quantitativas utilizadas para avaliar as melodias musicais geradas geram evidências comparativas entre os modelos, porém, a avaliação com humanos tende a ser o teste mais confiável. Portanto, um possível trabalho futuro é a realização de um experimento com um grupo de pessoas, para medir a qualidade das melodias geradas em comparação com outros modelos disponíveis. Este experimento pode ser feito através da interface web já desenvolvida.

Deve-se também notar que, desde que a arquitetura para modelos de processamento de linguagem natural *Transformer* foi introduzida em Vaswani et al. (2017), soluções em processamento de linguagem natural ganharam tração e produtos como *ChatGPT* (OpenAI, 2022) foram lançados. Modelos que exploram a utilização dos *Transformers* para a geração de música também foram propostos, sendo o mais recente o MusicLM (Agostinelli et al., 2023), desenvolvido pela *Google*. Tendo isto em vista, um possível trabalho futuro é a exploração de modelos de linguagem aplicados para geração de melodias musicais, adicionando as estratégias de aumento de dados propostas neste trabalho.

Para mais informações, o projeto AIHero está disponível no repositórios abaixo e no site aihero.bitaraes.com.br

- github.com/matheusbitaraes/AIHero
- github.com/matheusbitaraes/AIHeroFront.

Referências

A. Agostinelli, T. I. Denk, Z. Borsos, J. Engel, M. Verzetti, A. Caillon, Q. Huang, A. Jansen, A. Roberts, M. Tagliasacchi, et al. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*, 2023.

C. Ames. The markov process as a compositional model: A survey and tutorial. *Leonardo*, 22(2):175–187, 1989.

C. Ames e M. Domino. Cybernetic composer: An overview. understanding music with ai: Perspectives on music cognition, 1992.

A. M. Andrew. Introduction to evolutionary computing, by ae eiben and je smith (natural computing series), springer, berlin, 2003, hardback, xv+ 299 pp., isbn 3-540-40184-9 (£ 30.00); book review; book review, 2004.

T. Bäck, D. B. Fogel, e Z. Michalewicz. Handbook of evolutionary computation. *Release*, 97(1):B1, 1997.

J. Biles. Straight-ahead jazz with genjam: A quick demonstration. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 9, 2013.

J. Biles, P. Anderson, e L. Loggi. Neural network fitness functions for a musical iga. 1996.

J. Biles et al. Genjam: A genetic algorithm for generating jazz solos. In *ICMC*, volume 94, pages 131–137, 1994.

J. A. Biles. Autonomous genjam: eliminating the fitness bottleneck by eliminating fitness. In *Proceedings of the GECCO-2001 workshop on non-routine design with evolutionary systems*, volume 7. Morgan Kaufmann San Francisco, CA,, USA, 2001.

T. Blickle e L. Thiele. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, 4(4):361–394, 1996.

M. P. Cavini. História da música ocidental. *São Carlos: EdUFSCar*, 2, 2012.

E. Chew. *Towards a mathematical model of tonality*. PhD thesis, Massachusetts Institute of Technology, 2000.

B. Corporation. Boomy make instant music with artificial intelligence, 2020. URL https://boomy.com/.

L. Davis et al. Applying adaptive algorithms to epistatic domains. In *IJCAI*, volume 85, pages 162–164, 1985.

K. Ebcioğlu. An expert system for harmonizing four-part chorales. *Computer Music Journal*, 12(3):43–51, 1988.

Evostar. Evomusart 2022 the 11th international conference on artificial intelligence in music, sound, art and design, 2022. URL https://www.evostar.org/2022/evomusart/.

M. Farzaneh e R. M. Toroghi. Gga-mg: Generative genetic algorithm for music generation. *arXiv preprint arXiv:2004.04687*, 2020.

S. Forsgren e H. Martiros. Riffusion - Stable diffusion for real-time music generation, 2022. URL https://riffusion.com/about.

J. Fox e E. Embrey. Music—an aid to productivity. *Applied ergonomics*, 3(4):202–205, 1972.

M. Fréchet. Sur la distance de deux lois de probabilité. *Comptes Rendus Hebdomadaires des Seances de L Academie des Sciences*, 244(6):689–692, 1957.

A. Freitas e F. Guimaraes. Melody harmonization in evolutionary music using multiobjective genetic algorithms. In *Proceedings of the Sound and Music Computing Conference*, 2011.

A. Freitas, F. G. Guimarães, e R. Barbosa. Automatic evaluation methods in evolutionary music: an example with bossa melodies. In *Parallel Problem Solving from Nature-PPSN XII: 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part II 12*, pages 458–467. Springer, 2012.

A. R. R. d. Freitas. Música evolutiva: uma abordagem computacional para composição algorítmica. Master's thesis, Programa de Pós-Graduação em Ciência da Computação, UFOP - Universidade Federal de Ouro Preto, 2011.

M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200):675–701, 1937.

M. F. Z. Gatti. *A música como intervenção redutora da ansiedade do profissional de serviço de emergência: utopia ou realidade?* PhD thesis, Universidade de São Paulo, 2005.

D. E. Goldberg e K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of genetic algorithms*, volume 1, pages 69–93. Elsevier, 1991.

D. E. Goldberg, R. Lingle, et al. Alleles, loci, and the traveling salesman problem. In *Proceedings of an international conference on genetic algorithms and their applications*, volume 154, pages 154–159. Lawrence Erlbaum Hillsdale, NJ, 1985.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, e Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144, 2020.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, e Y. Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

L. Guilherme. Rest: Conceito e fundamentos, 2020. URL https://www.alura.com.br/artigos/rest-conceito-e-fundamentos.

G. L. Guimaraes, B. Sanchez-Lengeling, C. Outeiral, P. L. C. Farias, e A. Aspuru-Guzik. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843*, 2017.

J. Harley. Electronic and computer music, revised and expanded edition. *Computer Music Journal*, 28(4):93–95, 2004.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, e S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

L. A. Hiller Jr e L. M. Isaacson. Musical composition with a high speed digital computer. In *Audio Engineering Society Convention 9*. Audio Engineering Society, 1957.

K. Hingee e M. Hutter. Equivalence of probabilistic tournament and polynomial ranking selection. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 564–571. IEEE, 2008.

S. Hochreiter e J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

J. H. Holland. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. 1975.

A. Horner e D. E. Goldberg. *Genetic algorithms and computer-assisted music composition*, volume 51. Ann Arbor, MI: Michigan Publishing, University of Michigan Library, 1991.

D. Horowitz. Generating rhythms with genetic algorithms. In *AAAI*, volume 94, page 1459. Cambridge, MA, 1994.

D. M. Huber. *The MIDI manual: a practical guide to MIDI in the project studio*. Routledge, 2012.

K. Jebari e M. Madiafi. Selection methods for genetic algorithms. *International Journal of Emerging Sciences*, 3(4):333–344, 2013.

T. Karras, S. Laine, e T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, e T. Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

S. O. Kimbrough, M. Lu, e D. H. Wood. Introducing distance tracing of evolutionary dynamics in a feasible-infeasible two-population (fi-2pop) genetic algorithm for constrained optimization. 2004.

H. J. Koellreutter. *Harmonia Funcional - Introdução à Teoria das Funções Harmônicas*, volume 3. Ricordi, 1986.

K. Komatsu, T. Yamanaka, M. Takata, e K. Joe. A music composition model with genetic programming. In *PDPTA*, pages 686–692, 2010.

P. Kora e P. Yadlapalli. Crossover operators in genetic algorithms: A review. *International Journal of Computer Applications*, 162(10), 2017.

F. Kosona. Diathlassis for flute solo: a composition based on an application of the mathematical model of cusp catastrophe. In *Proceedings of 11th WSEAS International Conference on Acoustics & Music*, 2010.

S. Kullback e R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

M. Kumar, M. Husain, N. Upreti, e D. Gupta. Genetic algorithm: Review and application. *Available at SSRN 3529843*, 2010.

K. Kunimatsu, Y. Ishikawa, M. Takata, e K. Joe. A music composition model with genetic programming-a case study of chord progression and bassline. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, page 256. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2015.

S.-g. Lee, U. Hwang, S. Min, e S. Yoon. Polyphonic music generation with sequence generative adversarial networks. *arXiv preprint arXiv:1710.11418*, 2017.

R. Loughran e M. O'Neill. Evolutionary music: applying evolutionary computation to the art of creating music. *Genetic Programming and Evolvable Machines*, pages 1–31, 2020.

R. Loughran, J. McDermott, e M. O'Neill. Tonality driven piano compositions with grammatical evolution. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 2168–2175. IEEE, 2015.

B. Macukow. Neural networks–state of art, brief history, basic models and architecture. In *IFIP international conference on computer information systems and industrial management*, pages 3–14. Springer, 2016.

H. Maki. Origins of spontaneous mutations: Specificity and directionality of base-substition, frameshift, and sequence-substitution mutageneses. *Annual review of genetics*, 36:279, 2002.

W. S. McCulloch e W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

B. Med. *Teoria da Música*. Number 4. MusiMed, 1996.

L. F. Menabrea e A. Lovelace. Sketch of the analytical engine invented by charles babbage, 1842.

E. R. Miranda e J. Al Biles. *Evolutionary computer music*. Springer, 2007.

Mozilla. Códigos de status de respostas http, 2023. URL https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status.

P. B. Nemenyi. *Distribution-free multiple comparisons*. Princeton University, 1963.

OpenAI. Chatgpt: Optimizing language models for dialogue, 2022. URL https://openai.com/blog/chatgpt/.

K. O'Shea e R. Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

F. Pachet. Representing knowledge used by jazz muscians. In *Proceedings of the International Computer Music Conference*, pages 285–285. INTERNATIONAL COMPUTER MUSIC ACCOCIATION, 1991.

G. Papadopoulos e G. Wiggins. Ai methods for algorithmic composition: A survey, a critical view and future prospects. In *AISB symposium on musical creativity*, volume 124, pages 110–117. Edinburgh, UK, 1999.

H. Riemann. *Vereinfachte Harmonielehre oder die Lehre von den tonalen Funktionen der Akkorde*. Number 9197. Augener, 1899.

R. Rombach, A. Blattmann, D. Lorenz, P. Esser, e B. Ommer. High-resolution image synthesis with latent diffusion models, 2021.

J. Romero, J. J. Romero, e P. Machado. *The art of artificial evolution: A handbook on evolutionary art and music*. Springer Science & Business Media, 2008.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

F. Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

D. E. Rumelhart, G. E. Hinton, e R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

A. T. SARL. AIVA the ai composing emotional soundtrack, 2022. URL https://www.aiva.ai/.

M. Scirea, J. Togelius, P. Eklund, e S. Risi. Metacompose: A compositional evolutionary music composer. In *International Conference on Computational Intelligence in Music, Sound, Art and Design*, pages 202–217. Springer, 2016.

M. Scirea, P. Eklund, J. Togelius, e S. Risi. Evolving in-game mood-expressive music with metacompose. In *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion*, pages 1–8. 2018.

C. Sheryl. The sounds of fighting men, howlin' wolf and comedy icon among 25 named to the national recording registry, 2010. URL https://www.loc.gov/item/prn-10-116/.

Soundraw. Soundraw ai music generator, 2022. URL https://soundraw.io/.

P. Souza, V. Rolla, J. E. Ayres, e J. E. Sánchez. Graph composer: music composition from graph design. In *Anais do XVII Simpósio Brasileiro de Computação Musical*, pages 223–224. SBC, 2019.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, e Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

Tensorflow. Deep convolutional generative adversarial network, 2022. URL https://www.tensorflow.org/tutorials/generative/dcgan?hl=pt-br.

C.-K. Ting, C.-L. Wu, e C.-H. Liu. A novel automatic composition system using evolutionary algorithm and phrase imitation. *IEEE Systems Journal*, 11(3):1284–1295, 2015.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, e I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

P. J. Werbos. *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*, volume 1. John Wiley & Sons, 1994.

B. Widrow e M. E. Hoff. Adaptive switching circuits. Technical report, Stanford Univ Ca Stanford Electronics Labs, 1960.

L.-C. Yang e A. Lerch. On the evaluation of generative models in music. *Neural Computing and Applications*, 32(9):4773–4784, 2020.

L.-C. Yang, S.-Y. Chou, e Y.-H. Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation. *arXiv preprint arXiv:1703.10847*, 2017.

L. Yu, W. Zhang, J. Wang, e Y. Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

D. Zicarelli. M and jam factory. *Computer Music Journal*, 11(4):13–29, 1987.

Apêndice A

Comparações de métricas do experimento 2

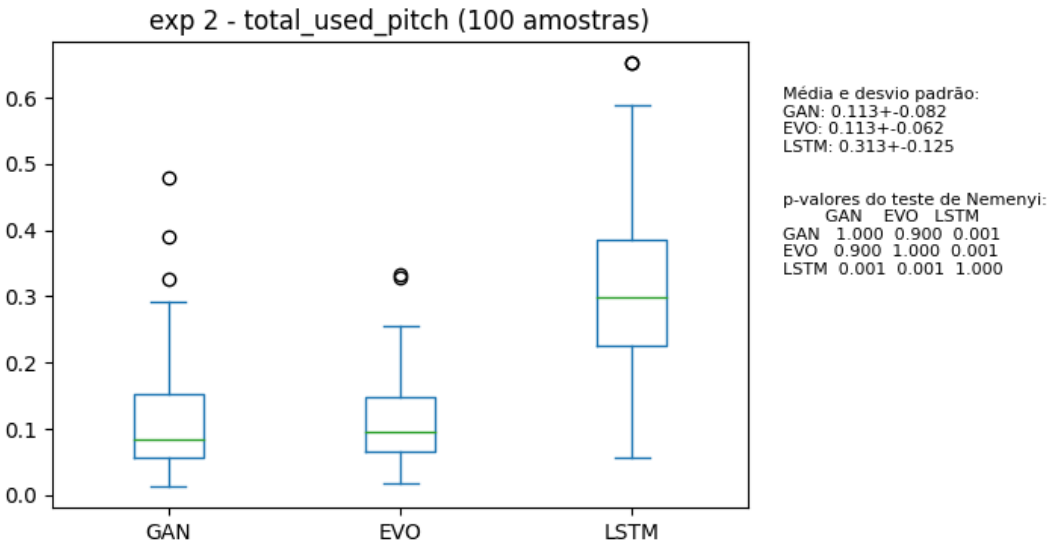

Figura 34 – exp 2 - total used pitch (100 amostras)

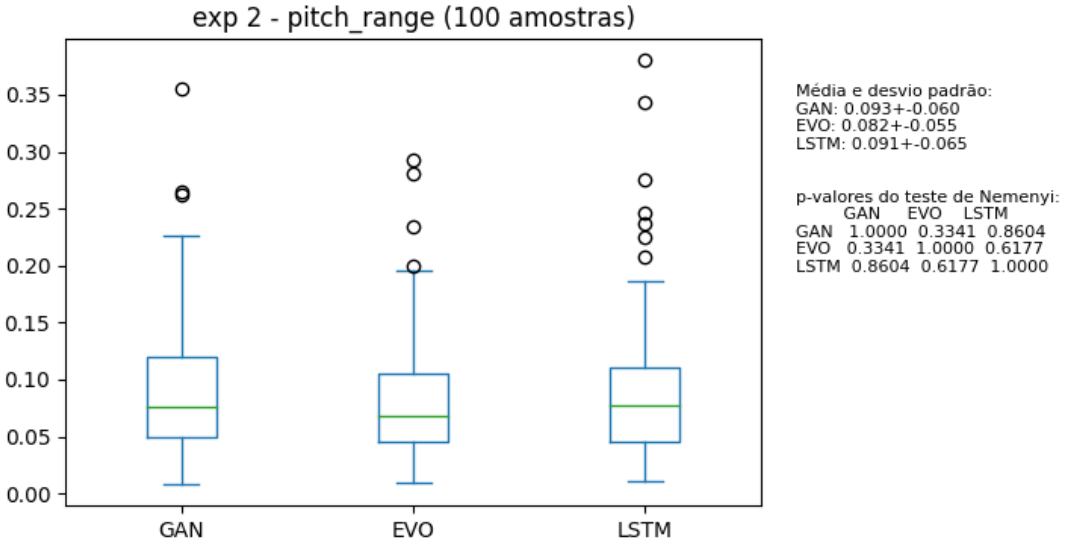

Figura 35 – exp 2 - pitch range (100 amostras)

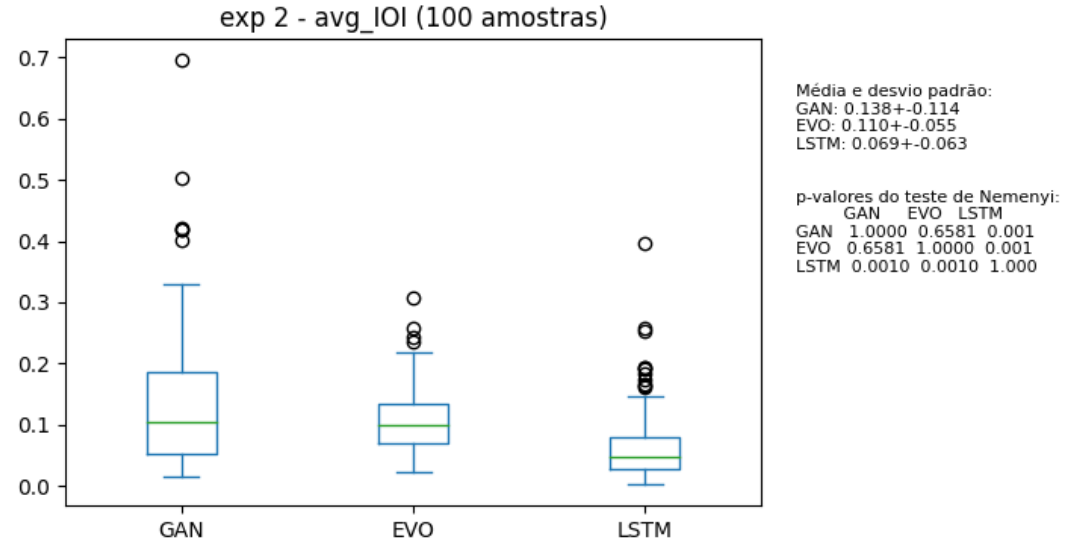

Figura 36 – exp 2 - avg IOI (100 amostras)

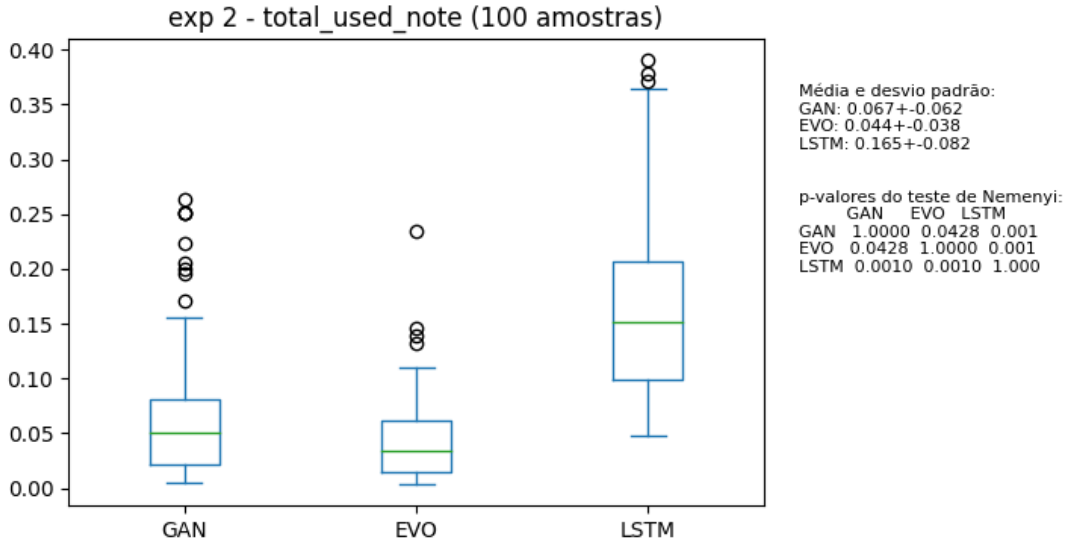

Figura 37 – exp 2 - total used note (100 amostras)

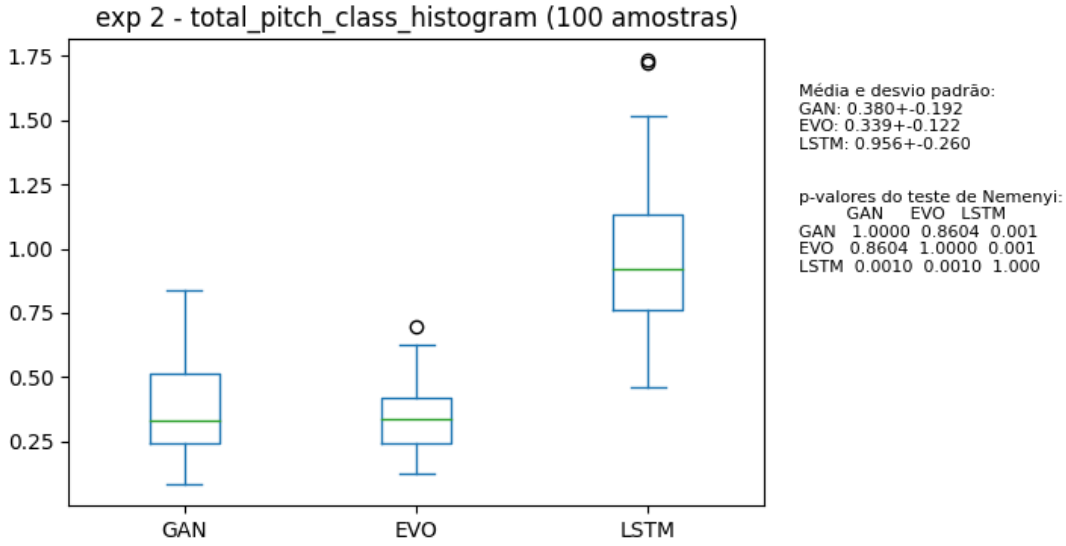

Figura 38 – exp 2 - total pitch class histogram (100 amostras)

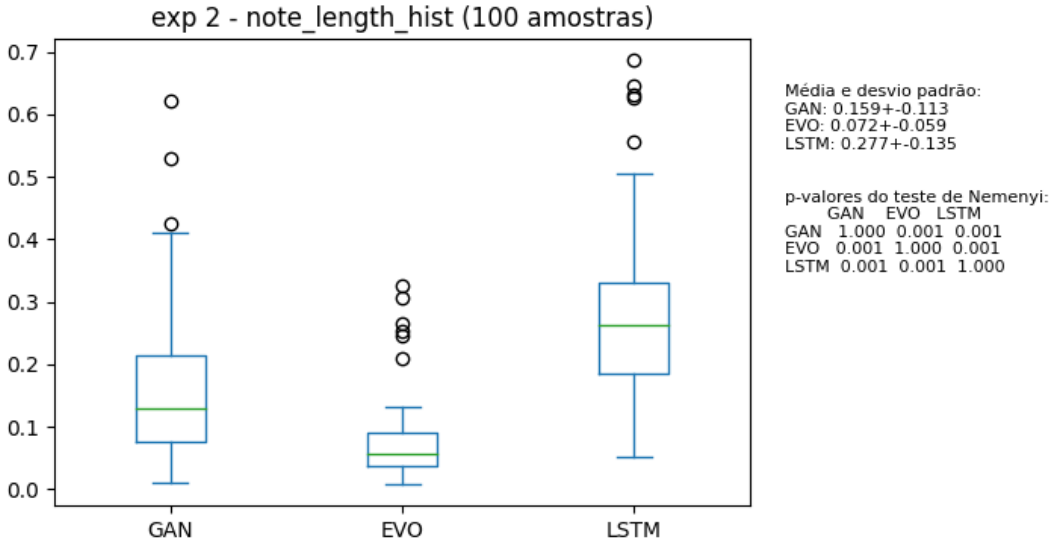

Figura 39 – exp 2 - note length hist (100 amostras)

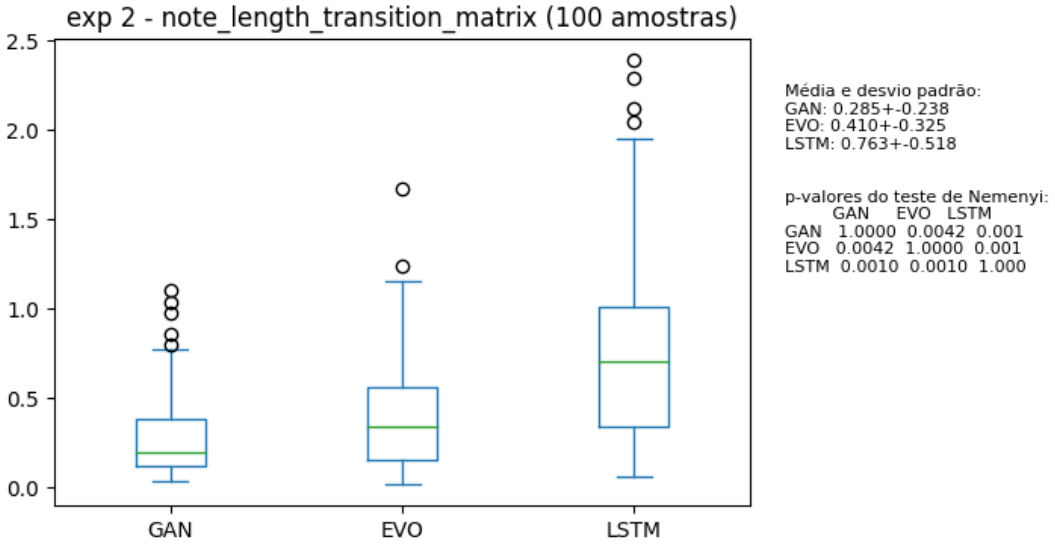

Figura 40 – exp 2 - note length transition matrix (100 amostras)

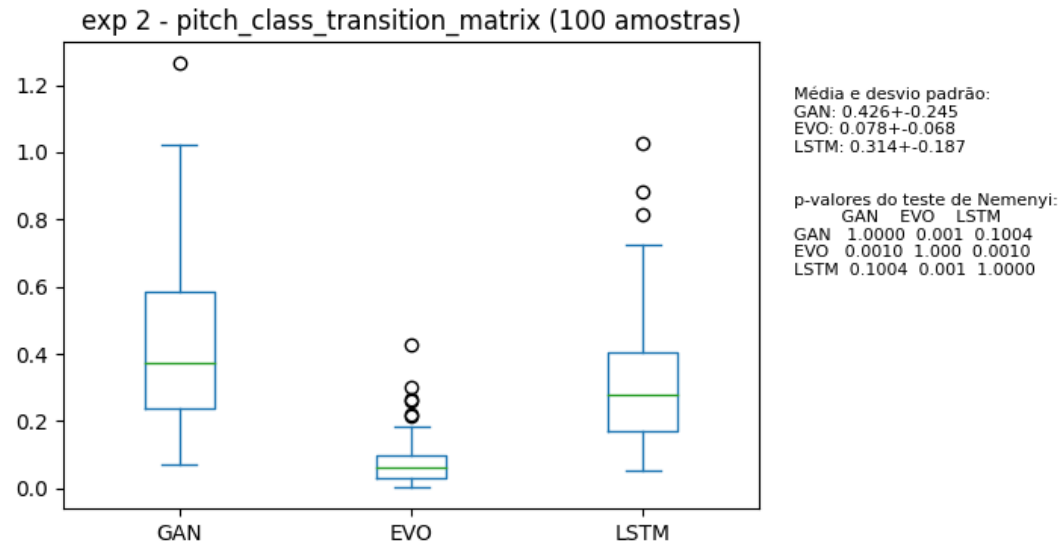

Figura 41 – exp 2 - pitch class transition matrix (100 amostras)

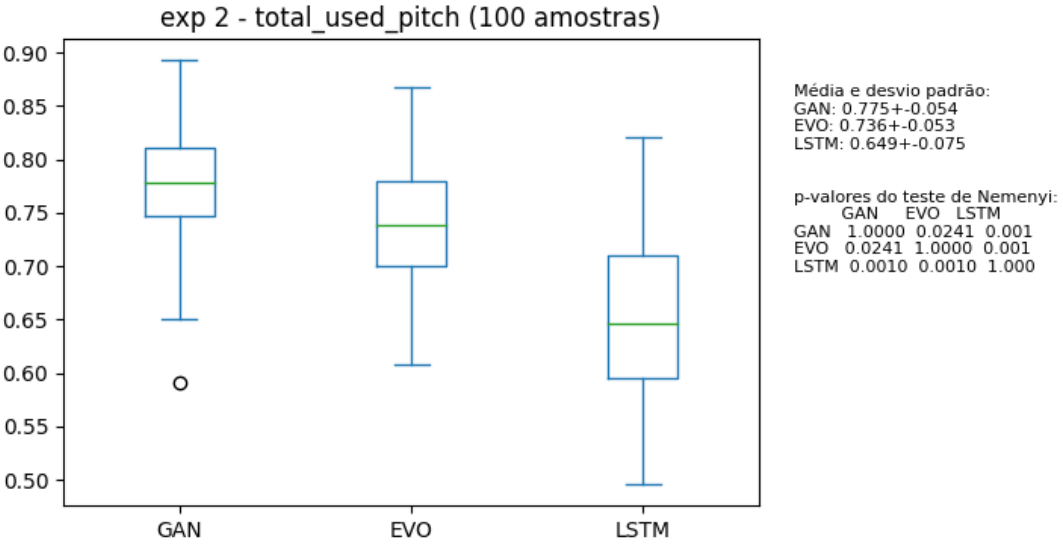

Figura 42 – exp 2 - total used pitch (100 amostras)

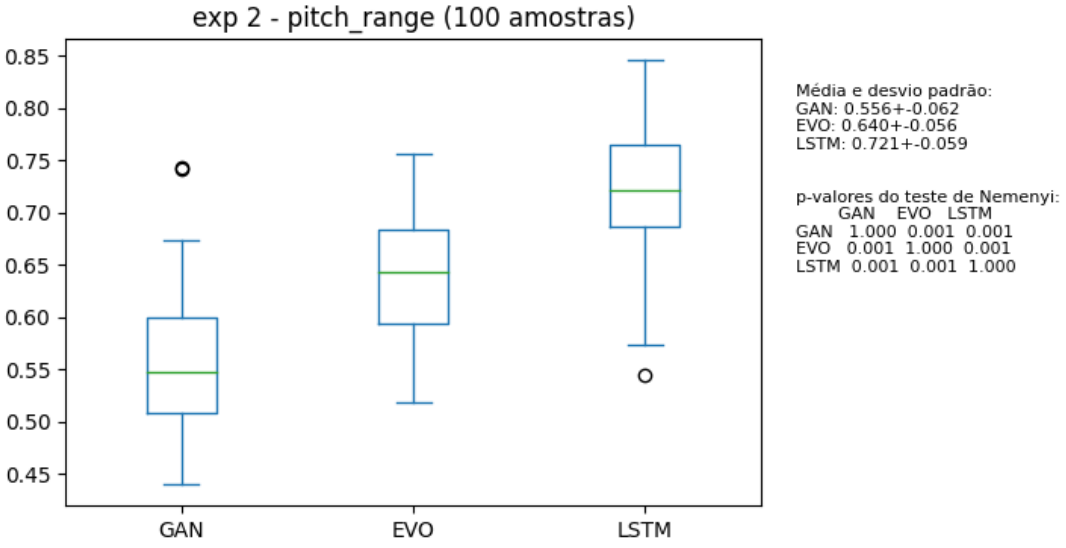

Figura 43 – exp 2 - pitch range (100 amostras)

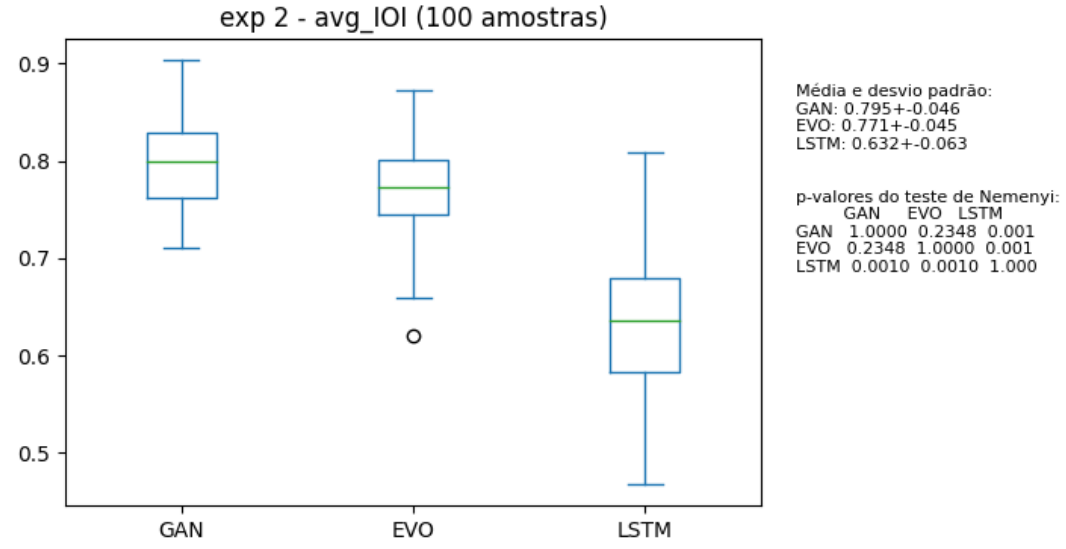

Figura 44 – exp 2 - avg IOI (100 amostras)

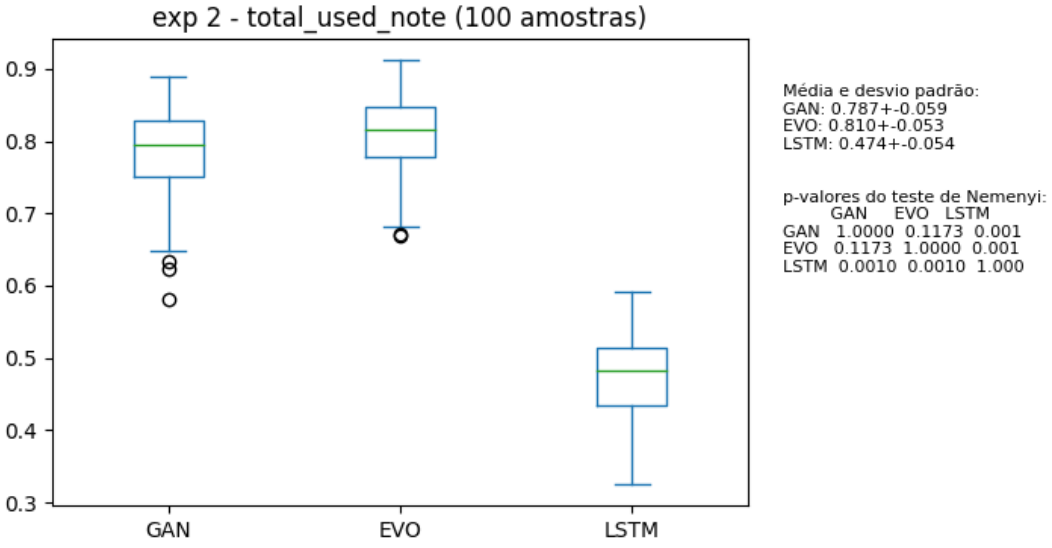

Figura 45 – exp 2 - total used note (100 amostras)

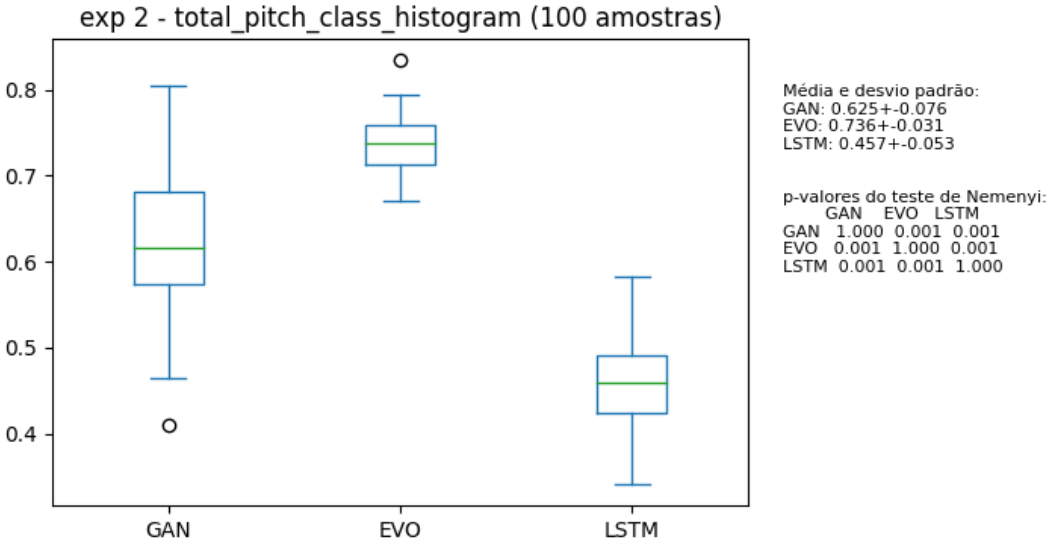

Figura 46 – exp 2 - total pitch class histogram (100 amostras)

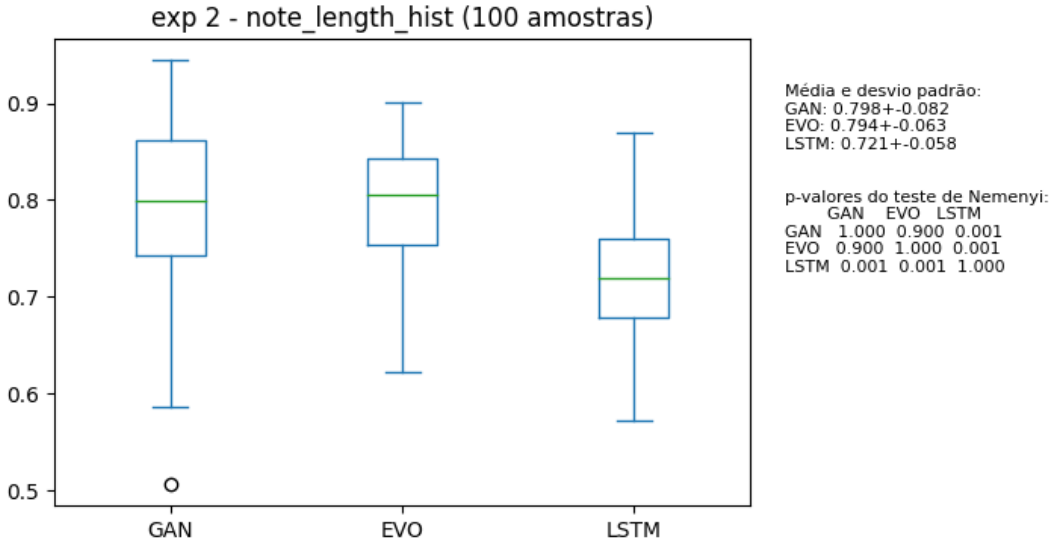

Figura 47 – exp 2 - note length hist (100 amostras)

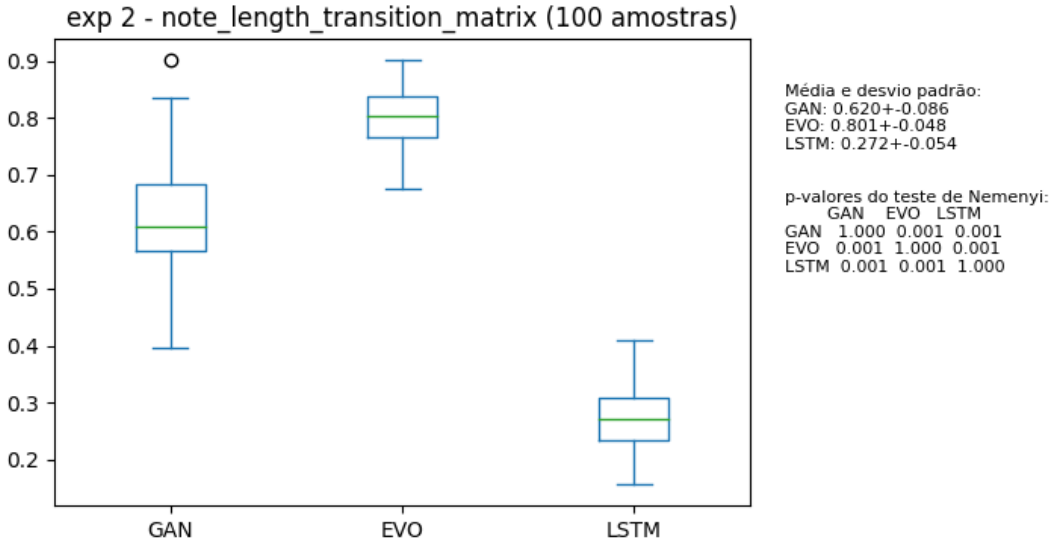

Figura 48 – exp 2 - note length transition matrix (100 amostras)

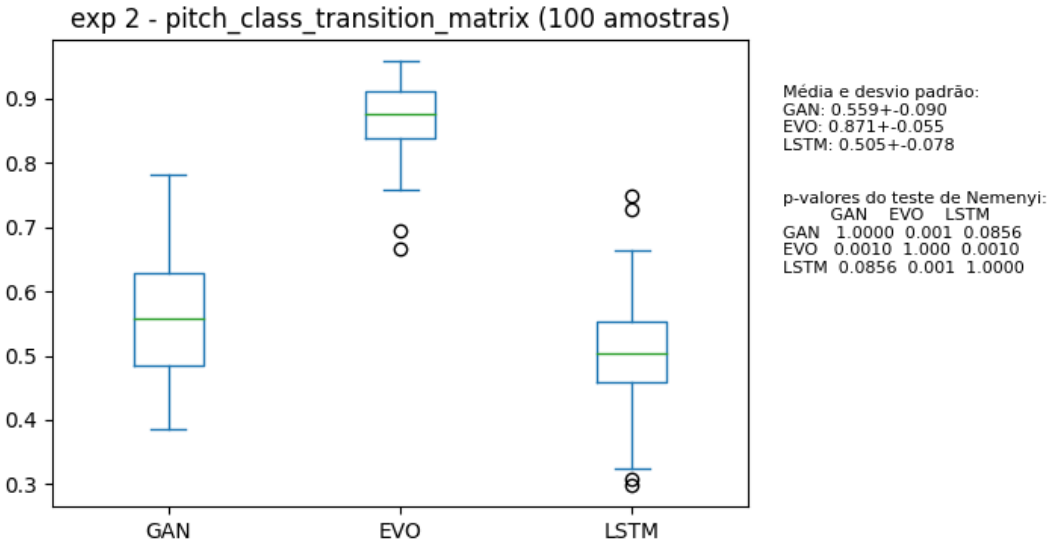

Figura 49 – exp 2 - pitch class transition matrix (100 amostras)

Apêndice B

Comparações de métricas do experimento 3

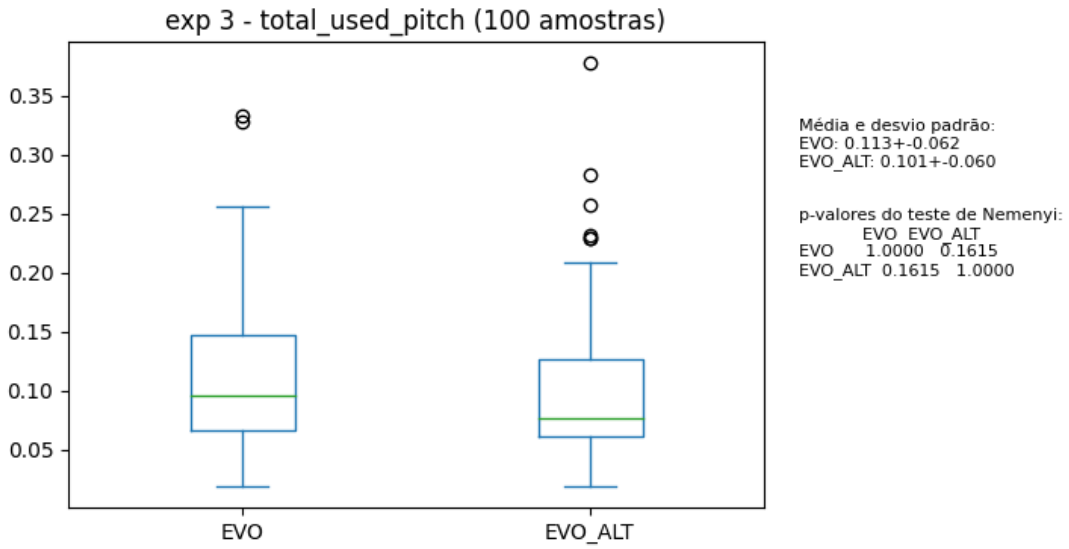

Figura 50 – exp 3 - total used pitch (100 amostras)

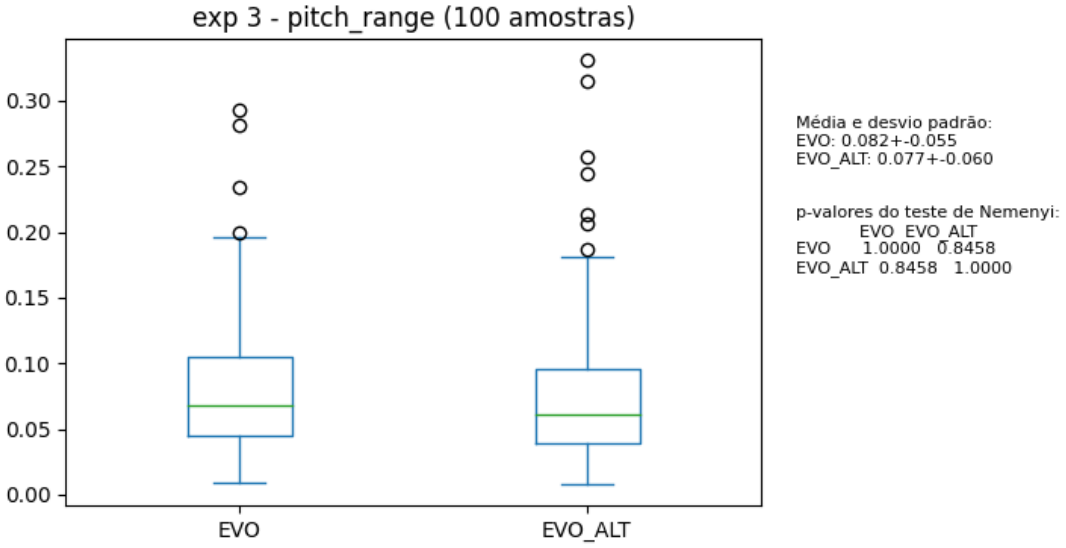

Figura 51 – exp 3 - pitch range (100 amostras)

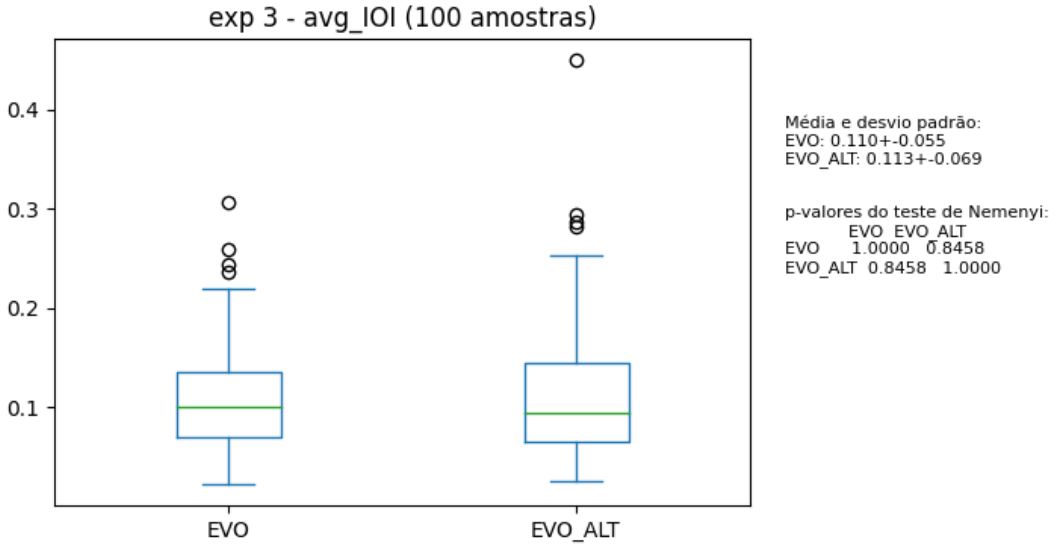

Figura 52 – exp 3 - avg IOI (100 amostras)

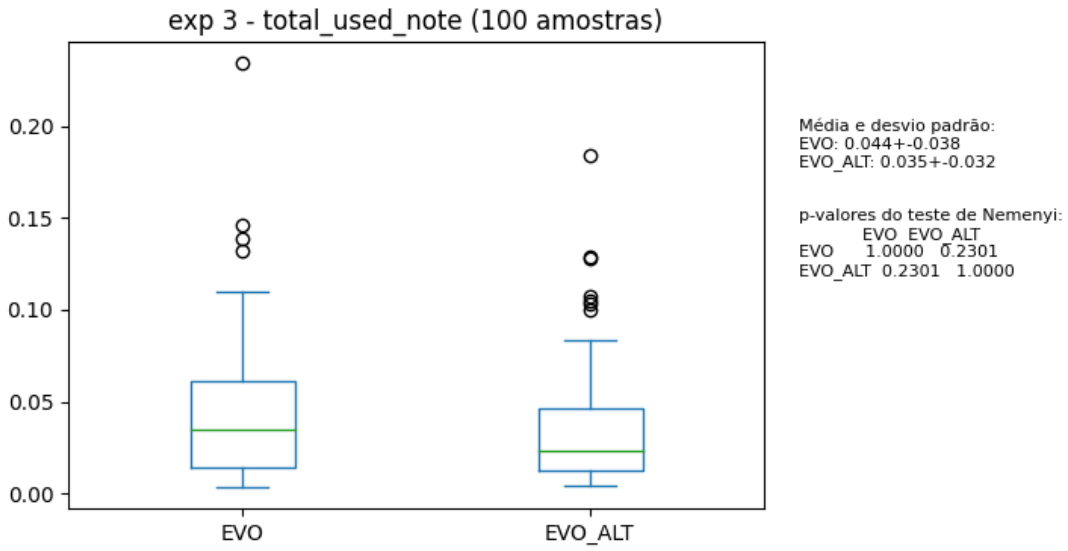

Figura 53 – exp 3 - total used note (100 amostras)

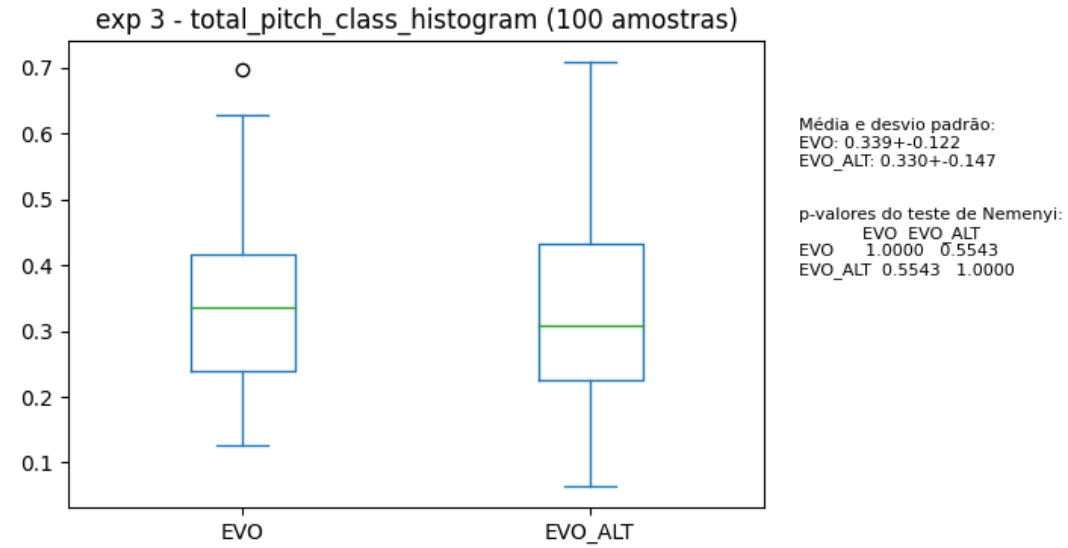

Figura 54 – exp 3 - total pitch class histogram (100 amostras)

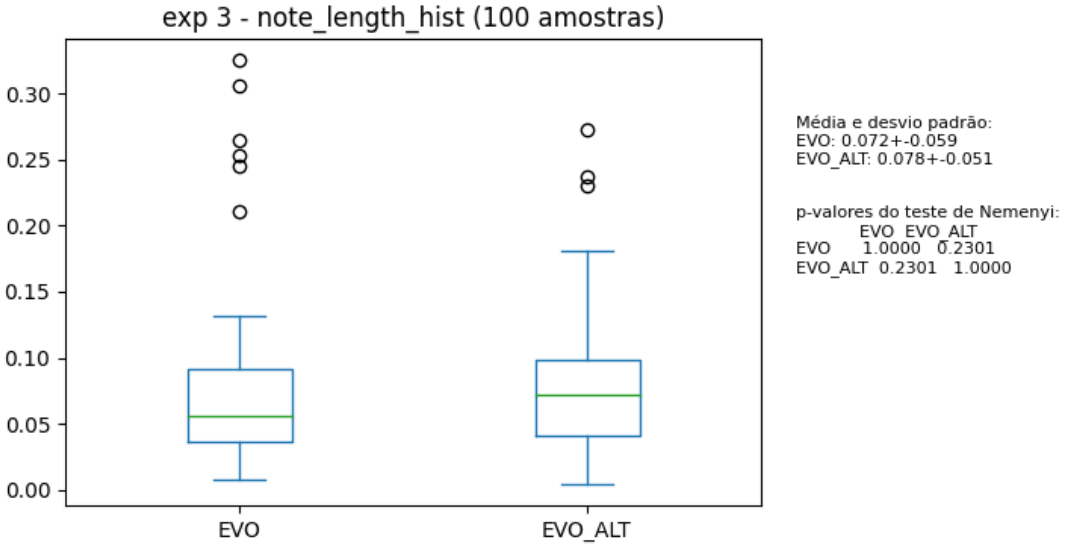

Figura 55 – exp 3 - note length hist (100 amostras)

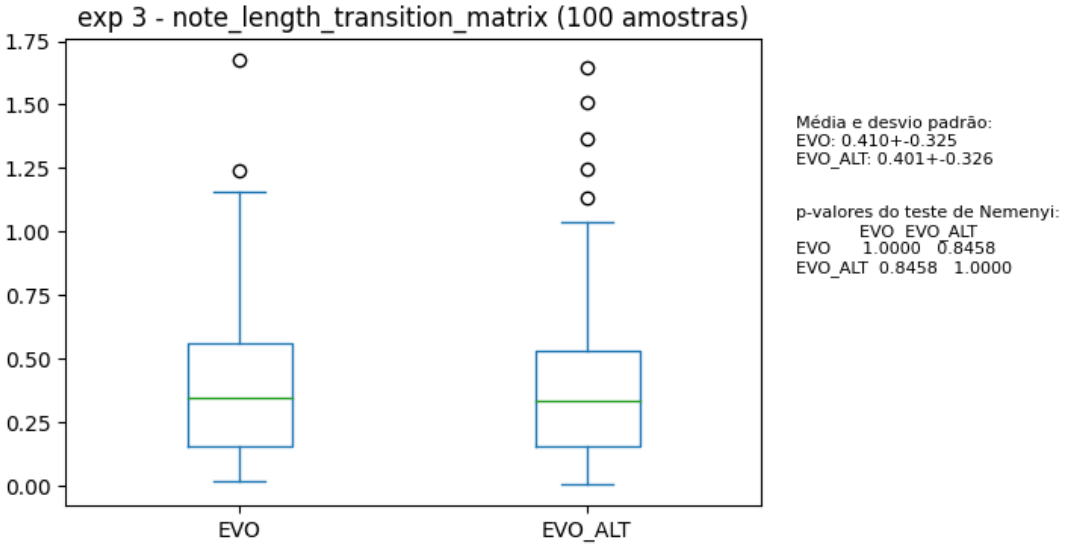

Figura 56 – exp 3 - note length transition matrix (100 amostras)

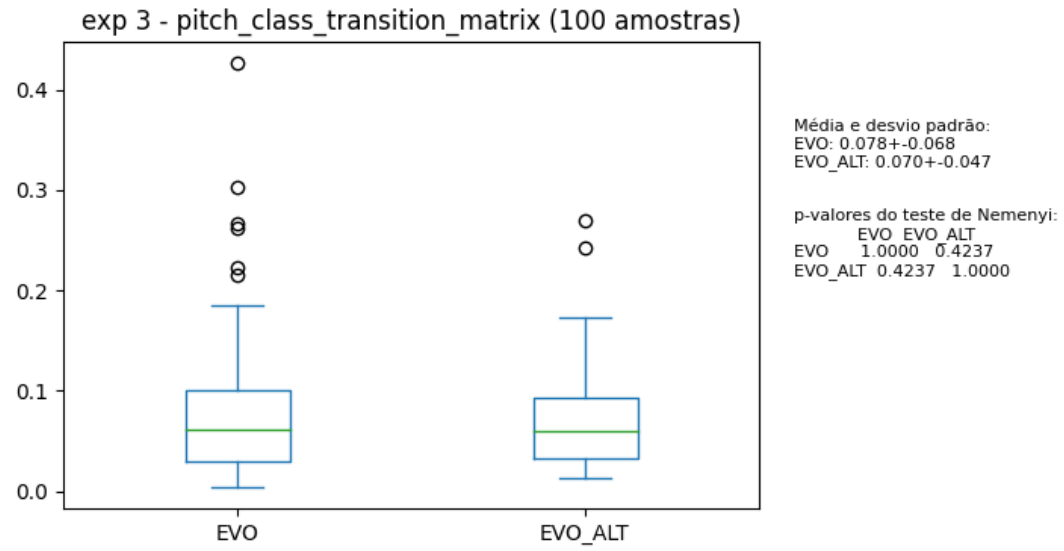

Figura 57 – exp 3 - pitch class transition matrix (100 amostras)

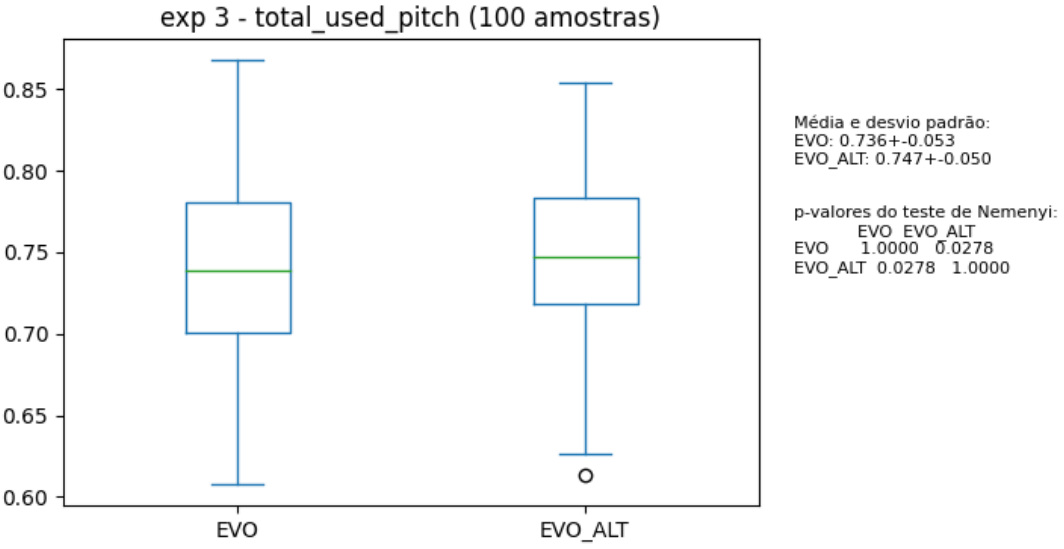

Figura 58 – exp 3 - total used pitch (100 amostras)

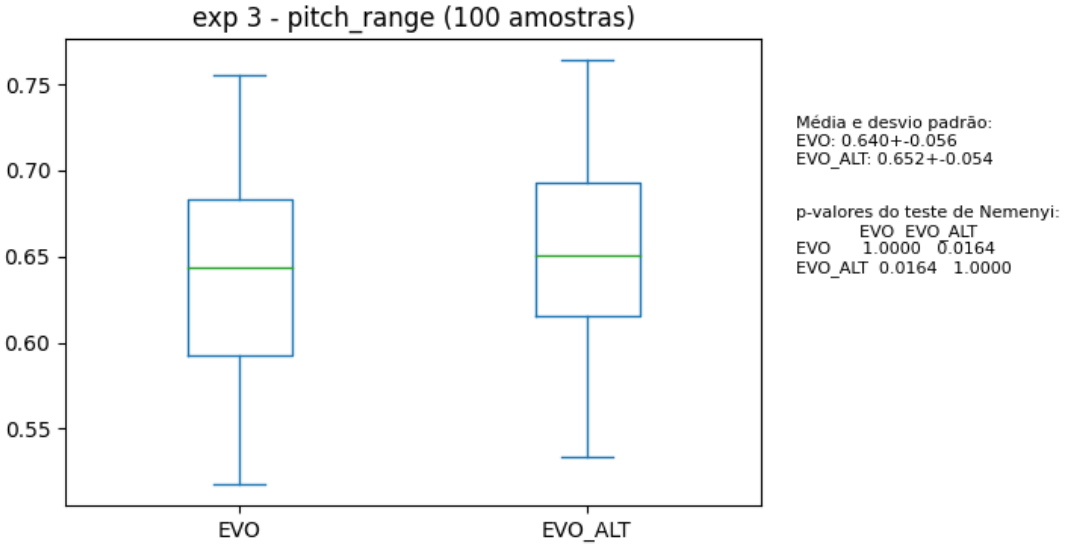

Figura 59 – exp 3 - pitch range (100 amostras)

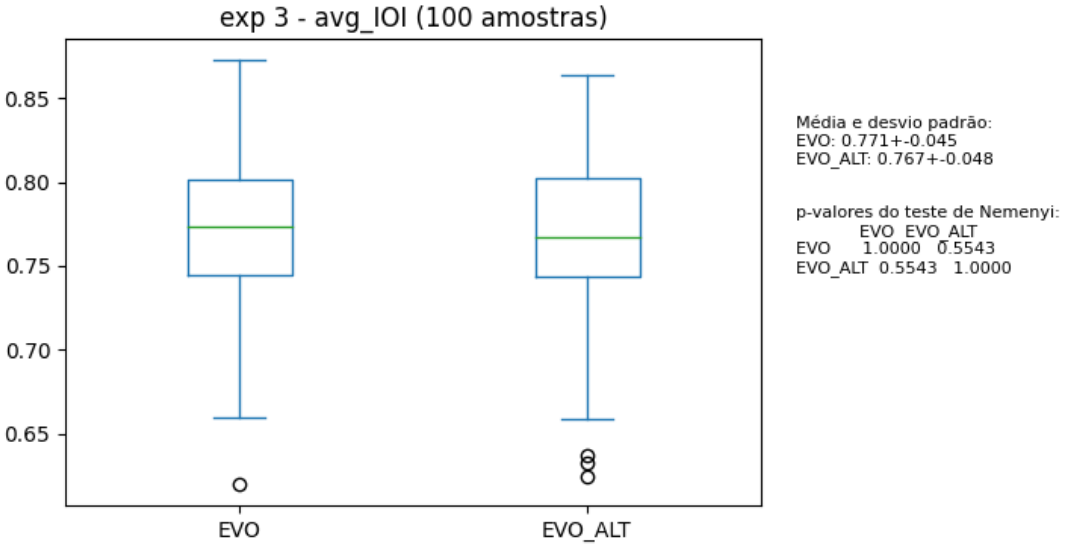

Figura 60 – exp 3 - avg IOI (100 amostras)

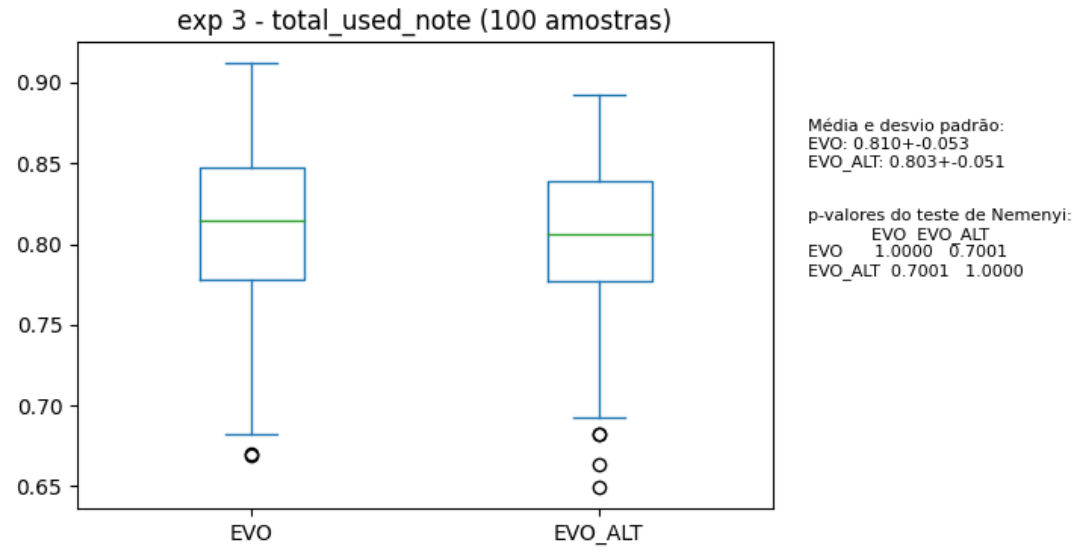

Figura 61 – exp 3 - total used note (100 amostras)

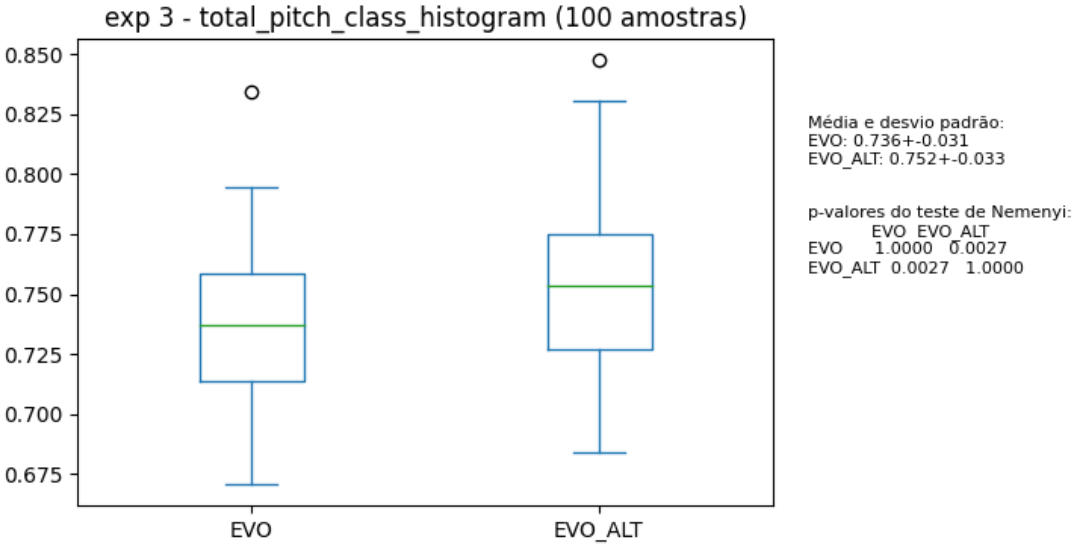

Figura 62 – exp 3 - total pitch class histogram (100 amostras)

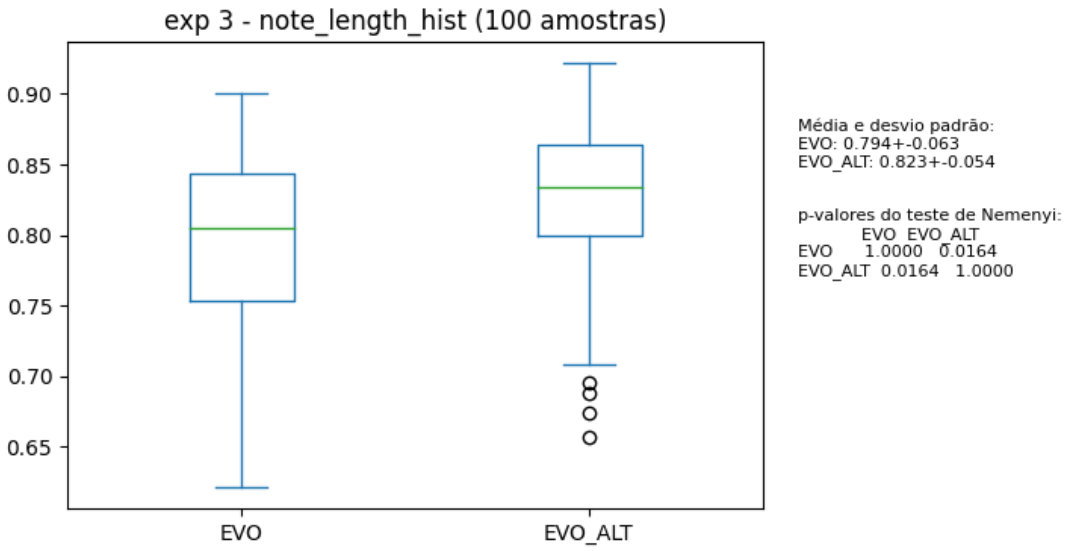

Figura 63 – exp 3 - note length hist (100 amostras)

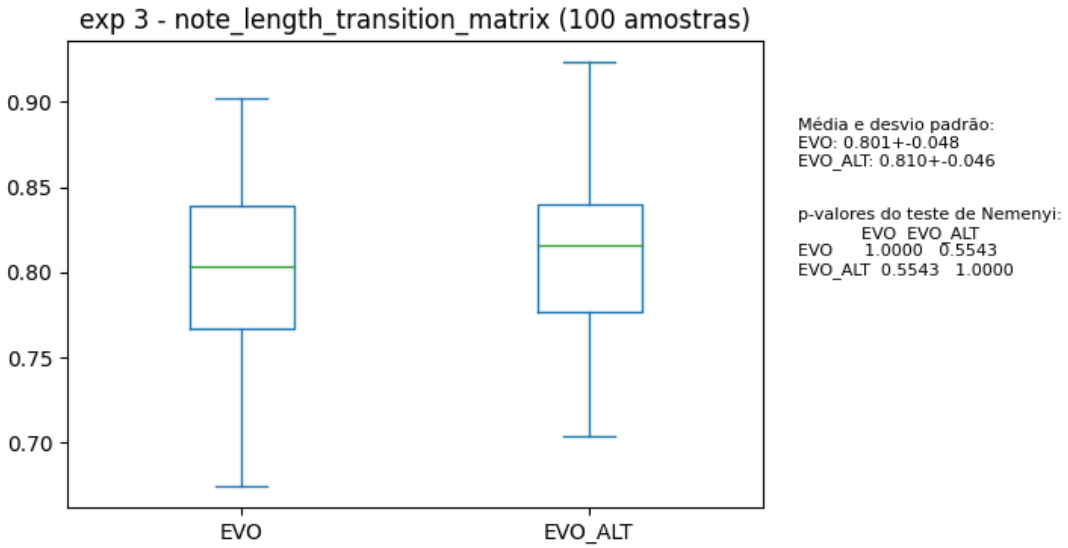

Figura 64 – exp 3 - note length transition matrix (100 amostras)

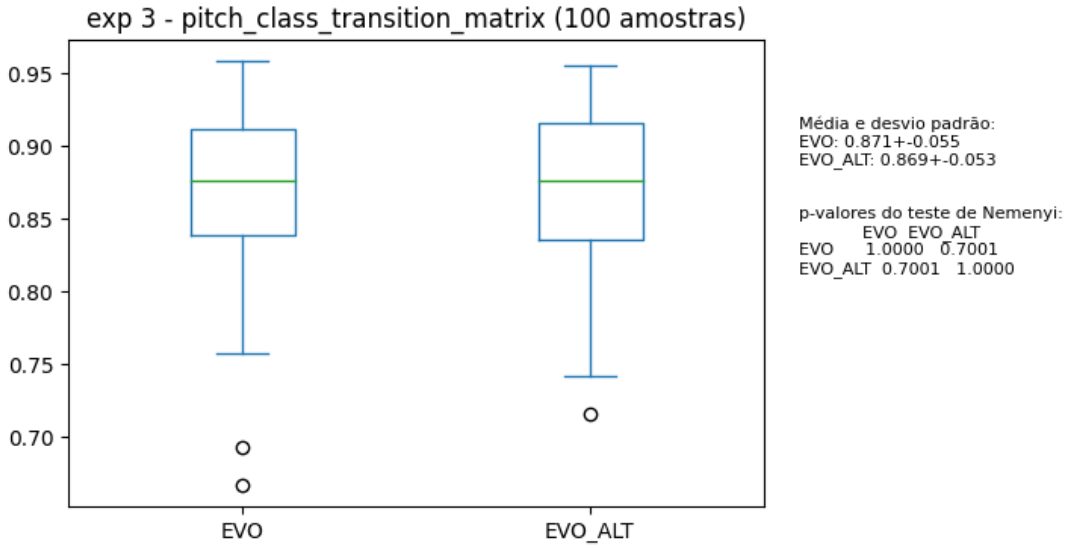

Figura 65 – exp 3 - pitch class transition matrix (100 amostras)