PEDRO CIPRIANO FEIJÃO

"UM ALGORITMO DE CRIAÇÃO DE IMPROVISOS COM HARMONIA DE JAZZ"

Campinas
2004

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

Pedro Cipriano Feijão

UM ALGORITMO DE CRIAÇÃO DE IMPROVISOS COM HARMONIA DE JAZZ

Tese de mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica.
Área de concentração: Eletrônica, Microeletrônica e Optoeletrônica.

Orientador: Prof. Dr. Furio Damiani

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA TESE DEFENDIDA PELO ALUNO, E ORIENTADA PELO PROF. DR. FURIO DAMIANI

Campinas
2004

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

F324a

Feijão, Pedro Cipriano, 1975-
Um algoritmo de criação de improvisos com harmonia de jazz / Pedro Cipriano Feijão. --Campinas, SP: [s.n.], 2004.

Orientador: Furio Damiani.
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Musical por computador - Composição. 2. Markov, Processos de. 3. Sistemas estocásticos. 4. Inteligência Artificial. I. Damiani, Furio, 1943-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: An algorithm for the composition of Jazz solos
Palavras-chave em Inglês: Computer composition, Markov chains, Stochastic systems, Artificial intelligence
Área de concentração: Eletrônica, Microeletrônica e Optoeletrônica
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Adolfo Maia Junior, Jônatas Manzolli
Data da defesa: 26-02-2004
Programa de Pós Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Nome do Candidato : Pedro Cipriano Feijão

Data da Defesa : 26.02.2004

Título da Tese : "Um Algoritmo de Criação de Improvisos com Harmonia de Jazz"

Prof. Dr. Furio Damiani (03.961-6):_____

Prof. Dr. Adolfo Maia Junior:_____

Prof. Dr. Jônatas Manzolli:_____

À MINHA AVÔ LOURDES.

Agradecimentos

Agradeço, Ao Prof. Furio Damiani por ter sido um excelente orientador, por toda a sua paciência e incentivo.

Aos meus pais, por incentivarem incondicionalmente todas as minhas escolhas, mesmo quando não as consideram as melhores.

À minha linda esposa Giovana, pelo amor, carinho e paciência em todos os momentos.

À CAPES, pelo apoio financeiro.

A maravilhosa disposição e harmonia do universo só pode ter tido origem segundo o plano de um Ser que tudo sabe e tudo pode. Isto fica sendo a minha última e mais elevada descoberta.

Isaac Newton

Resumo

Um levantamento detalhado dos trabalhos em composição computacional será apresentado. Um algoritmo de composição e performance em tempo real de solos de jazz foi criado, inspirado no estilo bebop. Técnicas utilizadas por músicos de bebop durante os seus improvisos foram implementadas usando-se várias regras. Um novo modelo estocástico, baseado no raciocínio de um músico durante o seu solo, escolhe a saída do algoritmo. O usuário pode avaliar a saída do algoritmo em tempo-real, e isto será utilizado em um processo de aprendizado supervisionado para modificar os parâmetros do modelo estocástico. Algumas amostras de saídas do algoritmo serão mostradas, junto com as conclusões.

Palavras-chave: Composição Algorítmica, Cadeias de Markov, Métodos Estocásticos, Treinamento Supervisionado, Música, Jazz.

Abstract

An extensive review of the most prominent works in the field of computer music is presented. A real-time algorithm for composition and performance of jazz solos in the style of bebop was created. Techniques that bebop musicians use on their solos were implemented using several rules. A new stochastic model, inspired by the thinking of a musician during his solo chooses the algorithm output. The user evaluates the algorithm output in real-time, and this input is used in a supervised learning process to change the stochastic model parameters. Samplings of the algorithm output are shown, along with concluding remarks.

Key-words: Algorithmic Composition, Markov Chains, Stochastic Methods, Supervised Training, Music, Jazz.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Um certo grau de formalismo é a base da composição musical ocidental. Regras formais de composição vêm sendo utilizadas desde o século XI, quando Guido d'Arezzo criou uma relação entre notas musicais e as vogais em textos religiosos. Regras rigorosas foram desenvolvidas para a composição de contraponto, nos períodos Clássico e Barroco. Outro exemplo é o serialismo, onde regras estritas ditam a ordem da escolha de vários elementos musicais, com algumas variações permitidas. Algoritmos de composição estão, consciente ou inconscientemente, arraigados na composição musical ocidental.

Mas um conjunto de regras sozinho não é suficiente para determinar uma composição. O autor escolhe quais regras deve seguir, em que ordem, e até quais deve quebrar, usando a sua criatividade. Uma das perguntas fundamentais a qualquer sistema, não só de composição, mas de inteligência artificial, é: Como simular a criatividade? Podemos considerar que a primeira forma de simular a criatividade com algoritmos foi com a aleatoriedade. O mais antigo exemplo conhecido do uso de eventos aleatórios em composição é a peça Musikalisches Würfelspiel (música de dados), de Mozart, onde as variações que seriam aplicadas em um certo tema inicial eram decididas pelo resultado obtido jogando dados. John Cage utilizou largamente eventos aleatórios em suas composições. A sua peça Reunion é executada ao se jogar xadrez em um tabuleiro equipado com fotorreceptores, onde cada movimento ativa um som, o que faz com que a peça seja diferente cada vez que é executada.

O computador permite que regras e métodos de simulação de criatividade sejam programados; composições podem ser geradas quase que instantaneamente, com o poder de processamento atual. Mas, para implementar um algoritmo de composição, duas perguntas fundamentais devem ser respondidas: como escolher e implementar as regras, ou seja, a base de conhecimento musical do sistema e como simular a criatividade. O conhecimento musical pode ser codificado com regras derivadas diretamente do estilo de composição ou aprendidas pelo computador através de exemplos musicais. Para simular a criatividade várias abordagens já foram tomadas, como processos aleatórios, computação evolutiva e aprendizagem de máquina. Uma outra questão aparece freqüentemente: Como simular um crítico, ou seja, como saber se o resultado, seja de um pequeno trecho, seja da peça completa, é bom ou ruim? O crítico de um sistema de composição algorítmica pode estar automaticamente implementado no seu conhecimento musical ou processo criativo, ou pode estar independente, controlando o processo de composição à medida que ocorre, seja na forma de um módulo independente do módulo de composição, seja na forma de um crítico humano.

O primeiro objetivo desta pesquisa é fazer um levantamento detalhado de vários sistemas de com-

posição algorítmica computacional, dando uma ampla visão da área e mostrando como as questões acima já foram abordadas. O segundo objetivo é implementar um algoritmo de composição, apresentando algumas novas abordagens para a resolução das questões levantadas. O algoritmo sugerido, o Bebopper, tem o seu conhecimento musical implementado por meio regras que simulam técnicas utilizadas por músicos de jazz para criar improvisos melódicos em tempo real. A criatividade é introduzida por meio de um novo modelo estocástico proposto, que teve como inspiração o raciocínio de um músico durante o seu improviso. O usuário pode participar como crítico: sua avaliação em tempo real dos improvisos geradas será utilizada em um processo de aprendizado supervisionado para ajustar os parâmetros do modelo estocástico.

1.1 Organização

No capítulo 2 é apresentado um levantamento detalhado dos trabalhos recentes em composição e performance computacional. As seções do capítulo estão categorizadas pelo tipo de técnica utilizada, com exceção da última, a Performance Expressiva, que não é uma técnica, mas sim uma área que tem despertado grande interesse, onde são utilizadas várias técnicas diferentes. O capítulo 3 está dedicado à apresentação do algoritmo criado. Na seção 3.1 é feita uma introdução e visão geral do algoritmo. A seção 3.2 explica as técnicas utilizadas para a criação do improviso melódico. Na seção 3.3 são apresentados os detalhes da criação de uma seção rítmica para o acompanhamento do improviso. Na seção 3.4 está apresentado o modelo estocástico proposto. No capítulo 4 serão apresentados alguns resultados, conclusões e perspectivas futuras.

Capítulo 2

Modelos Existentes

2.1 Processos Estocásticos

Processos estocásticos foram usados para composição computacional extensivamente no passado e continuam a ser usados atualmente, embora dificilmente como abordagem única, e muito mais como uma das partes de um sistema. Uma das principais razões de se usar processos estocásticos é a baixa complexidade, o que os torna bons candidatos para aplicações em tempo real, especialmente quando não se tinha a capacidade computacional que se tem hoje. Com o passar do tempo, outros métodos passaram a ser factíveis de serem implementados em tempo real, e os métodos baseados puramente em processos estocásticos foram sendo em parte substituídos por novas abordagens. Os dois métodos estocásticos mais utilizados em composição algorítmica são cadeias de Markov e distribuições aleatórias.

2.1.1 Cadeias de Markov

Cadeias de Markov foram formuladas em 1906 pelo matemático Andrei Markov (1856-1922). Ele usou o seu modelo para derivar tendências ortográficas da língua Russa, utilizando o texto Eugene Onegin, de Aleksandr Pushkin [4]. Cadeias de Markov são um tópico fundamental na área de processos estocásticos, e foram utilizadas para modelar vários tipos de processos aleatórios em diversas áreas. Cadeias de Markov são sistemas discretos usados para representar a tendência de um determinado estado ou evento se seguir a outro, ou a um conjunto de outros estados imediatamente anteriores. Uma cadeia de Markov pode ser representada por uma matriz composta das probabilidades de um estado se seguir a um ou mais estados anteriores. Um exemplo simples de uma cadeia de Markov com 3 estados é dado na figura 2.1. Cada vértice do grafo representa um estado, e os números sobre cada um dos arcos indicam a probabilidade de sair de um estado e chegar a outro. Por exemplo, a probabilidade de ir para o estado 2, se o estado atual é o 1, é de 0,25.

Cadeias de Markov foram provavelmente o método estocástico mais usado para a composição algorítmica, especialmente nos primeiros anos da composição computacional, onde eram necessários métodos pouco complexos. A primeira aplicação existente de composição computacional utilizou cadeias de Markov com probabilidades de transição nota a nota. Lejaren Hiller utilizou o computador ILLIAC para compor a Illiac Suite, com a ajuda de Leonard Isaacson e Robert Baker [121]. No final

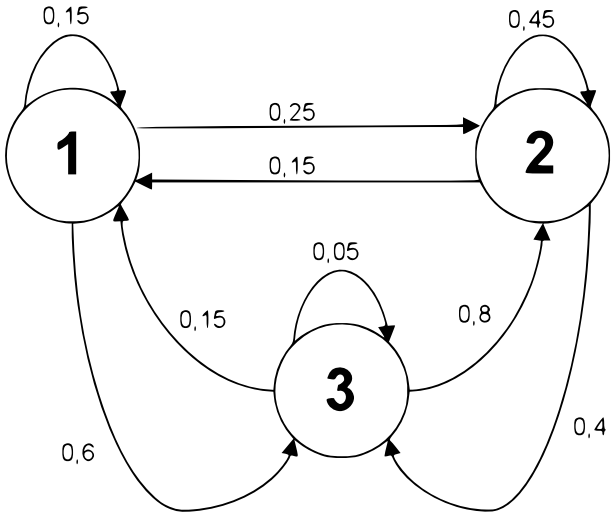

Figura 2.1: Um exemplo de uma cadeia de Markov

dos anos 50 e início dos anos 60, Hiller e Baker desenvolveram uma das primeiras linguagens para composição computacional, o MUSICOMP, que consistia em um conjunto de sub-rotinas divididas em 3 partes: a geração, onde a matéria-prima musical é gerada; a modificação, onde se aplicam variações sobre o material gerado anteriormente e a seleção, responsável por escolher qual o material gerado que deve entrar na composição.

Iannis Xenakis, um dos mais famosos pesquisadores da área, utilizou outros processos estocásticos, além de cadeias de Markov. Em seu livro *Formalized Music* [147], ele descreve eventos musicais em termos de três componentes: frequência, duração e intensidade. Esses três elementos foram combinados em forma de vetor e utilizados como estados em uma cadeia de Markov [55].

Boa parte das aplicações com cadeias de Markov utiliza matrizes de transição nota a nota. A escolha da nota atual depende de 1 ou mais notas anteriores. Zicarelli, no seu *M and Jam Factory* [149] utiliza essa abordagem, com quatro cadeias de Markov de 1ª a 4ª ordem sendo utilizadas alternadamente, de acordo com um conjunto de parâmetros definidos pelo usuário. As matrizes de transição podem ser derivadas de melodias fornecidas pelo usuário. As melodias geradas com esse tipo de abordagem, apesar de apresentarem semelhanças, principalmente estatísticas, com melodias humanas, não possuem sua objetividade nem estrutura. Seria o mesmo que, com matrizes de transição letra a letra em uma determinada língua, tentar obter frases ou até parágrafos com algum significado.

Cadeias de Markov foram utilizadas pela primeira vez para a análise musical por Franz [55]. Ele montou cadeias de Markov de primeira a terceira ordem com as probabilidades de transição entre notas tiradas de nove improvisos diferentes do saxofonista John Coltrane, executados sobre a sua composição *Giant Steps*, para depois analisá-las e tirar conclusões sobre o estilo desse músico. Franz considerou que redes de Markov são úteis como método de análise musical, servindo como um primeiro passo na busca de métodos de análise quantitativa para elementos historicamente analisados apenas qualitativamente, como estilo musical e criatividade.

Uma variação relativamente recente em cadeias de Markov são as Cadeias Ocultas de Markov, ou *Hidden Markov Chains (HMM)*. Para um tutorial sobre cadeia ocultas de Markov, veja o texto the Dugad e Desay [48].

Cadeias ocultas de Markov têm uma maior flexibilidade para capturar níveis musicais mais complexos, quando comparadas a cadeias de Markov [39]. A sua forma de utilização geralmente envolve algum tipo de treinamento, onde temos como entrada uma seqüência de eventos, e procura-se descobrir a cadeia oculta de Markov que geraria tal seqüência com maior probabilidade.

No seu excelente tutorial sobre cadeias ocultas de Markov, Dugad e Desay mostram que a maioria das aplicações de HMM se reduz à resolução de duas questões [48]:

- Dada uma HMM, como calcular a probabilidade de gerar a seqüência de entrada?
- Como ajustar os parâmetros dessa HMM de forma a maximizar essa probabilidade?

Se for possível resolver essas questões, poderemos encontrar uma HMM que reproduz seqüências estatisticamente próximas à seqüência de entrada, o que pode ser útil para capturar e gerar o estilo de um certo compositor, por exemplo. Vários métodos já foram sugeridos para resolver cada uma das questões [48]. Clement utilizou HMM's para modelar uma gramática estocástica livre de contexto (*SCFG - stochastic context-free grammars*) que busca capturar macroestruturas musicais, analisando uma seqüência de entrada com elementos musicais [39]. Os elementos musicais em questão são progressões harmônicas, mas o modelo sugerido pode ser utilizado para modelar outros elementos facilmente. Assumindo que cada elemento possa ser capturado de forma independente, o grande desafio é como combinar cada elemento de forma a descrever um estilo musical. Para o treinamento, foram utilizadas as técnicas de *forward-backward* [126] para encontrar a probabilidade de ocorrência da seqüência de entrada e o algoritmo de maximização de esperança de Baum-Welsh [111] para o ajuste dos parâmetros do modelo. Segundo os autores, embora tenham sido encontrados problemas na aplicação dos algoritmos utilizados para o aprendizado de línguas naturais, cadências harmônicas são suficientemente simples para que possam ser aprendidas. Em relação a outros aspectos musicais, cadeias de Markov podem não ser adequadas, mas algoritmos para o aprendizado de gramáticas livres de contexto podem ter sucesso.

Outro modelo que utiliza HMM é apresentado na dissertação de Allan [3], para resolver um problema clássico de composição algorítmica: a harmonização em 4 vozes de uma linha melódica dada. A tarefa de harmonização foi dividida em três partes: a) Construção de um esqueleto harmônico; b) O esqueleto de acordes, onde o esqueleto harmônico é preenchido com notas musicais; c) A ornamentação, adicionando notas no contratempo para embelezamento e melhor fluência das linhas melódicas individuais.

As duas primeiras partes são resolvidas com HMM, usando-as de forma *gerativa*: primeiro as HMM são treinadas, tendo como seqüências de entrada harmonias de J.S. Bach. A idéia básica é que em uma seqüência musical de entrada, as notas da melodia são as saídas, geradas pelos acordes da harmonia, que são definidos como os estados ocultos.

Para o ajuste dos parâmetros das HMM, foram utilizadas duas abordagens: o método de estimação do máximo a posteriori (MAP) usando o algoritmo de Viterbi [51] e amostragens da probabilidade condicional dos estados dadas as saídas.

Após o treinamento, o algoritmo executou muito bem a tarefa de harmonização de melodias de entrada, conseguindo capturar nuances do estilo de Bach. Entre as vantagens apresentadas pelo autor está a ausência de um conhecimento pré-programado: todo o conhecimento harmônico do sistema é

obtido através de exemplos. Outra grande vantagem é que o sistema é capaz de planejar a harmonização globalmente, ao invés de maximizar o valor das transições locais, uma característica de HMM's. Cadeias de Markov já conseguiram se mostrar boas ferramentas para sistemas de composição e análise musical. Porém, existem algumas desvantagens a serem levantadas: encontrar as probabilidades de transição nem sempre é tarefa simples, e devem ser analisadas muitas peças musicais para que se tenha a garantia de um sistema robusto. Além disso as cadeias de Markov têm dificuldade de capturar níveis musicais abstratos, apesar dessa dificuldade ser um pouco atenuada com o uso de HMM's. A não ser que sejam utilizadas em conjunto com outras técnicas, métodos que utilizem cadeias de Markov geralmente têm como saída composições com pouca estrutura musical.

Minsky [98] tem uma explicação para refutar o uso de processos estocásticos em música:

> *Although stochastic methods have been widely used, studies of their application to music has been concentrated on different selection methods and processes instead of considering the implications on thus-synthesized music. Especially when using a cognition-oriented viewpoint, it is difficult to apply stochastic methods to simulated mental processes. Mental processes seem not to be random in the stochastic sense, because human beings try to rationalize their choices logically, even if the rationalizations are complicated or vague.*

2.1.2 Distribuições Aleatórias

Números aleatórios são utilizados para tomada de decisões. Em música são utilizadas geralmente selecionar a próxima nota da composição [90]. Uma excelente discussão sobre a utilização de distribuições aleatórias, bem como outros métodos estocásticos em música foi apresentada em dois artigos por Charles [5, 6]. Além disso, Ames apresenta o seu método de obter equilíbrio em uma distribuição aleatória: *statistical feedback*. Segundo Ames, um dos maiores problemas de distribuições aleatórias é que precisa-se de um número relativamente grande de eventos para que a distribuição seja representada proximamente. Se o número de eventos não é grande, a distribuição não teve tempo suficiente para "falar". O seu método diminui bastante o número de eventos necessários para que uma distribuição seja bem representada.

2.2 Sistemas Caóticos não Lineares e Fractais

Um sistema caótico pode ser definido como um sistema dinâmico de comportamento complexo, determinado por equações recursivas não-lineares, com um alto grau de imprevisibilidade devido à sua sensibilidade às condições iniciais. O interesse em estudar esse tipo de sistemas está no fato de que a maioria dos sistemas encontrados no mundo real é não linear, e apresenta um comportamento caótico em alguma extensão. Entre os exemplos que já foram modelados através de sistemas caóticos estão a previsão do tempo, flutuações populacionais, mecânica de fluidos e fenômenos elétricos oscilatórios. Algumas de suas características, como a auto-similaridade e auto-correlação, além da baixa complexidade computacional para serem geradas, fizeram com que sistemas caóticos fossem utilizados para a composição computacional.

Um sistema caótico pode ser representado pela equação

$$y_{t+1} = f(y_t) \tag{2.1}$$

onde a função f é não linear. À medida que os valores de y_t vão sendo iterados, os seus valores traçam uma órbita. Essas órbitas são de grande interesse matemático, e formam a essência de sistemas caóticos. O comportamento de f pode ser bastante variado. Muitas vezes seu conjunto-imagem é ilimitado. Em outros casos a função tem um comportamento oscilatório, gerando uma órbita periódica. Em qualquer caso, o ponto ou conjunto de pontos para o qual a função f converge é chamado de *atrator caótico* ou *estranho*, e pode ser definido pelo conjunto A da equação abaixo [66]:

$$A = \bigcup_{i=\infty}^{2\infty} f^{\circ n}(y) \quad k = 0, 1, 2, \ldots \quad f : \mathbb{R}^n \to \mathbb{R}^n \tag{2.2}$$

onde está sendo usada a notação de composição de funções, $f^{\circ 2}(y) = f\left(f(y)\right)$.

Como dito acima, alguns sistemas caóticos têm atratores que são um conjunto finito. O valor da função vagueia pelo espaço sem nunca voltar a um ponto já percorrido. Após um número infinito de iterações, o atrator preenche uma determinada fração do espaço, tendo portanto uma dimensão fracional. Tais atratores são chamados de *fractais*.

Representações gráficas de fractais podem assumir belíssimas formas, o que motivou o aparecimento de vários sistemas de composição que os utilizam. Apesar de produzirem alguns trechos interessantes, as peças geradas por sistemas caóticos não tem estrutura e parecem vagar sem objetivo. Além disso, é difícil julgar a sua qualidade, pois o seu conhecimento musical não é derivado de trabalhos ou pensamento humano [119].

2.3 Sistemas Especialistas

Sistemas especialistas, no sentido que utilizaremos aqui, são sistemas que utilizam um conjunto regras e restrições simbólicas pré-programadas como base do seu conhecimento, para alcançar um objetivo desejado. O uso de sistemas especialistas parece uma escolha natural em música especialmente quando se está modelando um domínio específico e bem conhecido, ou se quer introduzir estruturas ou regras explícitas. Uma grande vantagem é que o seu raciocínio é bem definido; é fácil explicar seu funcionamento através das regras. Um problema que já se tornou clássico em música computacional é o de harmonização em quatro vozes: Dada a voz da melodia de entrada(soprano), encontrar as outras três vozes(contralto, tenor e baixo) que a harmonizem adequadamente. Como estão implícitas no problema várias restrições, como a tessitura das vozes e *voice leading*, este problema foi resolvido na maioria das vezes utilizando sistemas baseados em regras.

Ebcioglu desenvolveu sua própria linguagem (*Backtracking Specification Language*) e a utilizou para desenvolver o seu sistema *CHORAL* de harmonização no estilo de J.S.Bach [49].

Outra técnica utilizada foi CSP (*constraint satisfaction programming*). O objetivo da CSP é restringir progressivamente os valores possíveis das variáveis do modelo, até que se chegue a um valor único para cada uma.

Pachet e Roy utilizaram CSP para o problema de harmonização [117]. O conhecimento é representado utilizando uma visão orientada a objetos. A harmonização é modelada como um conjunto

de restrições. O algoritmo é dividido em dois passos: No primeiro, as restrições são aplicadas aos intervalos entre as notas de uma mesma voz, pertencentes a acordes consecutivos; no segundo, são aplicadas aos intervalos entre as vozes de um mesmo acorde. O algoritmo de busca é baseado em *backtracking*.

Henz e Lauer implementaram um sistema musical em duas partes: arranjo, com o sistema *AARON*, e o sistema de composição, o *COMPOzE*, este último utilizando CSP, através do sistema de programação *Oz* [71]. O *COMPOzE* cria composições em quatro vozes tendo como entrada o plano musical elaborado pelo *AARON*.

Phon-Amnuaisuk e Wiggins compararam um sistema baseado em regras e um algoritmo genético para a harmonização, chegando à conclusão que algoritmos genéticos não são apropriados para o uso em música [124]. Posteriormente o sistema de harmonização baseado em regras foi sofisticado com uma estrutura hierárquica, onde cada componente do modelo trabalha em um nível diferente [122].

Um outro problema onde parece lógico o uso de CSP é a composição de melodias em contraponto. Apesar da utilização do contraponto ter variado bastante durante toda a história da música ocidental, uma característica comum a todos os estilos de contraponto é um conjunto rigoroso de restrições. Vendo a composição de contrapontos por esse ângulo, Jones criou o CPA (*counterpoint assistant*), um assistente para a criação de contraponto atonal, utilizando CSP [89]. A vantagem do CPA em relação a trabalhos anteriores é que ele não está preso a nenhum estilo específico: o usuário define as restrições, podendo simular um estilo existente ou criar o seu próprio. Além disso, o CPA permite melodias cromáticas, sem nenhuma restrição diatônica. O CPA não possui nenhuma regra tradicional de contraponto implementada *a priori*. O usuário é responsável pela entrada de todas as restrições que serão utilizadas. As restrições estão divididas em duas categorias, melódicas e harmônicas, e a abordagem escolhida foi totalmente empírica: O usuário entra com um número de linhas melódicas, de onde serão tirados os intervalos e padrões rítmicos para a montagem das linhas do contraponto; essas são as restrições melódicas. Depois, devem ser adicionados os acordes permitidos, de 2 a 6 notas, definindo as restrições harmônicas. O usuário pode ainda ativar algumas restrições globais, como a proibição do uso de oitavas e uníssono, ou do cruzamento entre vozes. Pode também definir o número máximo de notas simultâneas a qualquer momento, o número de notas em solo (sem harmonia) e o número mínimo de acordes de n notas, para n de 2 a 6. Com isso, é possível ter um grande controle sobre a densidade do contraponto gerado. Depois de ter todas as entradas definidas, o CPA encontra todos os contrapontos possíveis que satisfazem as restrições. O número de resultados pode chegar a centenas. Para facilitar a visualização, os resultados são ordenados crescente ou decrescentemente de acordo com um entre vários critérios, como menor número de notas diferentes, menor número de acordes diferentes ou menor número de notas em solo, por exemplo. Segundo o autor, a arte ao usar o CPA consiste definir e refinar as restrições, progressivamente, de forma a encontrar resultados úteis, pois as primeiras tentativas geralmente devolvem um número excessivo de resultados. Com a experiência, o CPA pode se tornam um grande assistente para a composição de contrapontos, como mostram as composições apresentadas pelo autor.

Gwee examinou o processo de composição do ponto de vista da teoria da complexidade computacional e do aprendizado de máquina [69]. Para estudar essas questões, ele implementou um sistema de composição de uma classe de contraponto conhecida como *species counterpoint*. O problema de composição de contrapontos sobre melodias dadas será modelado como um problema de CSP. As regras

do sistema estão divididas em 2 partes: as regras que modelam o estilo musical (*correctness rules*), que incluem regras melódicas e harmônicas referentes a esse tipo de contraponto. Essas regras garantem que o contraponto estará correto, mas não são suficientes para que o contraponto seja criativo ou artístico. Para isso, são introduzidas as regras artísticas (*artistry rules*), que tentam incluir algum tipo de criatividade, e são baseadas principalmente em preferências musicais de ouvintes dos tempos atuais. Para a verificação das regras, foi utilizada lógica *fuzzy*, considerada mais adequada, especialmente para as regras artísticas. Em seguida, o autor faz uma interessante discussão sobre a complexidade dos algoritmos envolvidos na resolução do problema de encontrar os contrapontos que satisfazem as restrições. Essa é uma das maiores contribuições dessa pesquisa, visto que tratar da complexidade é algo raro ou quase inexistente no campo da composição computacional. Na segunda parte do trabalho, Gwee tenta fazer com que as composições do sistema sejam tão próximas a uma composição humana de forma a não ser possível distinguir entre uma e outra, como uma espécie de teste de Turing. Para isso, o primeiro passo é saber avaliar uma composição de forma mais humana possível. Ele tentou duas abordagens: O *Jeppesen Experiment*, onde é coletado um conjunto de composições de um *expert* (o músico dinamarquês Knud Jeppesen) e se determina a importância relativa que ele deu a cada uma das regras do sistema, dessa forma ajustando os seus pesos. A segunda abordagem envolve o treinamento supervisionado de uma rede neural para que ela seja capaz de avaliar uma composição de forma semelhante a um humano. Com essas duas formas de avaliação, Gwen utilizou dois métodos heurísticos de busca para encontrar composições sub-ótimas: busca *best-first* com *branch-and-bound* e um algoritmo genético.

Outra aplicação para a composição de contrapontos é o *Doctor Webern* [20]. O usuário entra com um conjunto de temas que sofrem variações musicais, como inversões, permutações e alterações rítmicas, entre outras. Os temais iniciais e as variações serão utilizados para a composição das vozes. A composição será feita em um estilo musical, determinado por um conjunto de restrições, divididas em três grupos: restrições polifônicas, harmônicas ou variacionais. As polifônicas incluem regras sobre intervalos e restrições a certos paralelismos. Restrições harmônicas excluem determinados tipos de acordes, dependente ou independentemente do contexto. As restrições variacionais determinam o tipo de variações que podem ser utilizadas, restringindo as variações não utilizáveis em um dado estilo, por exemplo. Um programa, chamado *Doctor Helmholtz*, será desenvolvido para executar a saída do *Doctor Webern*, através de várias técnicas de síntese de som.

Uma das maiores limitações levantadas quando se trata de performance musical computacional é a falta de expressividade, o resultado mecânico da execução. Por causa disso, uma área da composição computacional que tem atraído bastante atenção ultimamente é a performance expressiva, ou seja, como tornar mais expressiva a execução musical de um computador. Friberg, junto com os seus colegas Johan Sundberg e Lars Fryden, desenvolveu um conjunto de regras para converter uma partitura escrita em uma performance musical expressiva convincente [57, 58]. As regras aplicam desvios de intensidade, duração e ritmo nas notas da partitura, e foram derivadas num processo de análise-por-síntese. O conjunto de regras foi posteriormente implementado em LISP dando origem ao sistema denominado *Director Musices* [59, 24]. Mais detalhes sobre essa pesquisa serão dados na seção 2.9, performance expressiva.

Um modelo computacional baseado em um regras simbólicas tem a vantagem de ter o seu funcionamento facilmente explicado, mas também apresenta algumas desvantagens. Representar o co-

nhecimento de forma simbólica nem sempre é tarefa simples, mesmo com a ajuda de especialistas, especialmente quanto se tenta implementar as exceções às regras, algo necessário em música. Outra grande desvantagem é que o espaço musical pode ficar limitado pelas regras, diminuindo a criatividade.

2.4 Gramáticas Gerativas

Em música, certo grau de hierarquia existe naturalmente. Notas são combinadas para formar frases, que por sua vez formam seções, movimentos e peças inteiras. Um problema presente em grande parte dos algoritmos de composição é a falta de macroestrutura. Uma composição que consiste de um único nível hierárquico - uma sucessão de melodias não relacionadas, por exemplo - dificilmente despertará interesse. Uma forma de conseguir uma estrutura hierárquica é utilizar uma gramática para a composição musical, da mesma forma que vem sendo usada para linguagem natural.

As gramáticas gerativas, criadas por Noam Chomsky e apresentadas em seu livro *Syntactic Structures* [37], são mecanismos que simulam a capacidade de uma pessoa de criar frases gramaticalmente corretas em sua língua nativa. A idéia básica é que a linguagem, apesar de possuir uma estrutura extremamente complicada, apresenta alguma regularidade matemática.

Uma gramática gerativa é formada por um conjunto de regras (e palavras) capaz de formar frases na língua escolhida, de forma a nunca criar uma frase errada gramaticalmente. As regras transformam sucessivamente classes de hierarquia alta em classes de hierarquia mais baixa, até chegar em classes na hierarquia mais baixa, que serão substituídas por palavras. Esse processo é realizado por meio de *augmented transition networks*(ATN).

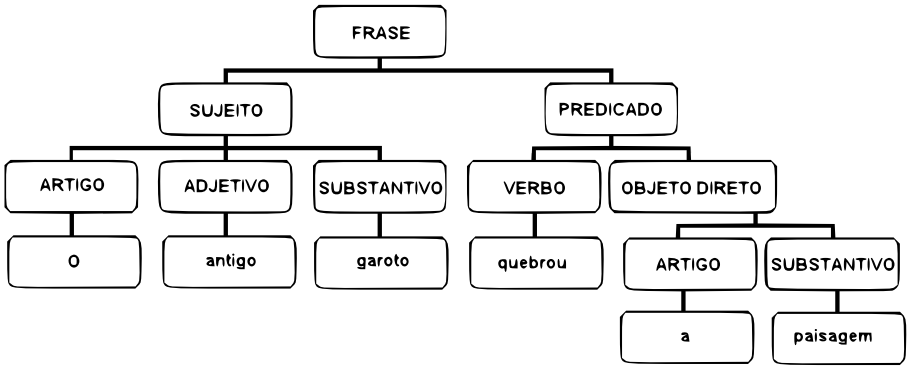

Figura 2.2: Exemplo do funcionamento de uma gramática gerativa

Na figura 2.2 podemos ver um exemplo de uma frase construída por uma gramática gerativa. Como podemos notar na frase gerada, apesar de ter sentido gramatical, as frases criadas por gramáticas gerativas podem não ter nenhum sentido semântico. Esse problema é resolvido em versões mais avançadas onde o significado das palavras é levado em consideração.

A mais conhecida aplicação de gramáticas gerativas em música é o livro *A generative theory of tonal music* [101]. Os autores tentam descrever como um ouvinte treinado percebe intuitivamente a estrutura musical de uma peça, e partindo disso apresentam um sistema de análise gramatical de música tonal.

Um uso bastante comum de gramáticas em música é a indução de estilo musical, para posterior composição de novas peças dentro do estilo capturado. Essa é exatamente a proposta de David Cope, em seu sistema EMI (*Experiments in music Intelligence*) [43, 44]. Com um algoritmo de reconhecimento de padrões aplicado em peças musicais de um autor desejado, o sistema encontra alguns padrões utilizados repetidamente, que são chamados de assinaturas do compositor, e que são incluídas em uma biblioteca. Uma ATN é responsável pelo rearranjo das assinaturas de uma biblioteca em uma nova composição, de uma forma bem estruturada. O EMI já criou composições no estilo de vários compositores eruditos, como J.S. Bach, Ludwig van Beethoven, Johannes Brahms e Béla Bártók. O EMI é um dos sistemas de composição computacional mais conhecidos, e um dos que apresenta melhores resultados. Apesar disso, é censurado fortemente, especialmente por críticos musicais que se recusam a acreditar que uma máquina pode compor músicas que emocionem. Em resposta a tais críticas, que se estendem à composição algorítmica como um todo, e não só ao EMI, Cope escreveu um artigo bastante interessante [45].

Também na área de indução de estilo musical, Buzzanca desenvolveu uma gramática para o reconhecimento de estilo musical [27], usando as regras de melodia de Baroni, Dalmonte e Jacoboni [13]. As regras dizem respeito à organização e ocorrência de certos padrões melódicos, estrutura intervalar, organização rítmica, entre outros aspectos. Uma composição de Giovanni Legrenzi foi analisada buscando a detecção de indicadores estilísticos. Outras composições, de Legrenzi e outros compositores também foram analisadas, para verificar se o programa é capaz de encontrar as diferenças entre os estilos. Em um estudo posterior, Buzzanca mostra que gramáticas, sozinhas, não são suficientes para resolver o problema de reconhecimento de estilo, e apresenta um método de reconhecimento baseado em redes neurais [28]. A rede funciona melhor para reconhecimento, e a gramática para a geração de exemplos; em conjunto, formam um sistema mais eficiente (ver seção 2.7).

Steedman desenvolveu uma gramática gerativa para criação de harmonias em progressões de blues de 12 compassos [135], e nais tarde a refinou usando gramáticas categoriais [136].

Algumas desvantagens do uso de gramáticas devem ser levadas em consideração. Gramáticas são estruturas hierárquicas, enquanto a música não é totalmente; portanto, ambiguidade pode ser necessária para melhorar o poder de representação da gramática [129]. Além disso, é um processo computacionalmente caro, dificultando o seu uso para implementações em tempo real, especialmente quando se inclui semântica.

2.5 Autômatos Celulares

Autômatos celulares são sistemas dinâmicos discretos. Podemos defini-los como uma matriz cujos elementos são denominados *células*, cada uma podendo assumir um estado dentre um certo número de estados possíveis. As células são atualizadas de forma síncrona em instantes discretos de tempo, de acordo com uma ou mais regras locais. O estado de uma célula em um instante de tempo atual depende apenas do seu estado e do estado de algumas células próximas – definidas como a sua *vizinhança* – no instante de tempo anterior. Para definir um autômato celular precisamos apenas das suas regras de atualização e a sua configuração inicial.

Autômatos celulares têm se mostrado ferramentas muito importantes em modelagem e simulação, encontrando aplicações em áreas diversas, como física, química, biologia e até sociologia. Uma carac-

terística importante de autômatos celulares é que, apesar de a sua dinâmica ser controlada apenas por regras locais, diversos comportamentos globais tendem a ocorrer ao longo do seu desenvolvimento - o que é chamado de *comportamento emergente*.

McAlpine, Miranda e Hoggar apresentaram o *CAMUS 3D* (*Cellular Automata MUSic in 3 Dimensions*), um sistema que utiliza dois autômatos celulares conhecidos - o Jogo da Vida e o *Demon Cyclic Space* - para composição musical [104]. Como é explicado no artigo, O *CAMUS 3D* é uma versão melhorada do *CAMUS*, dos mesmos autores, com a principal diferença de que os dois autômatos celulares agora têm três dimensões, ao invés de duas. Com a utilização de autômatos celulares para a composição, os autores esperam que padrões musicais e o surgimento de um estilo musical próprio do sistema apareçam espontaneamente durante o processo de composição, como consequência do comportamento emergente dos autômatos celulares.

O processo de composição começa com a iniciação do Jogo da Vida com uma configuração inicial escolhida pelo usuário e o *Demon Cyclic Space* é iniciado aleatoriamente. Após esse passo, os dois autômatos são executados simultaneamente. A cada instante de tempo, as 3 coordenadas de cada célula viva no Jogo da Vida são utilizadas para montar um acorde de 4 notas, que pode ter suas notas tocadas simultaneamente ou sequencialmente. O estado da célula correspondente no *Demon Cyclic Space* determina o instrumento que executa tal acorde. Cada coordenada da célula indica o intervalo entre as notas do acorde. A fundamental do acorde e a ordenação e duração das notas - no caso das notas serem executadas sequencialmente - são escolhidas através de processos estocásticos, cujos parâmetros podem ser especificados pelo usuário na iniciação do sistema.

As composições geradas apresentam um estilo bastante específico, centrado em eventos musicais de 4 notas. Isso é resultado direto da arquitetura do algoritmo, bem como do comportamento emergente dos autômatos. Os autores acreditam que a natureza orgânica dos autômatos celulares é responsável pelos bons resultados, considerados melhores do que de outros algoritmos de composição que utilizam técnicas semelhantes.

Uma limitação atual do sistema é que ele gera a composição uma célula por vez, usando uma ordem fixa para a varredura do Jogo da Vida. Isso significa que a música gerada é monofônica, e surge uma ordem fixa de progressão harmônica. A solução para esse problema seria o processamento paralelo das células, o que por sua vez aumentaria bastante a complexidade do sistema. Além disso há a questão do conteúdo harmônico gerado, que teria de ser avaliado de alguma forma, para evitar harmonias ruins. Os autores têm a intenção de, ao implementar o processamento paralelo, implementar também uma rede neural para classificar os acordes resultantes, treinada para responder a harmonia de forma similar à pessoa que opera o sistema.

Bilotta e Pantano desenvolveram um algoritmo genético para testar uma nova forma de avaliação do *fitness*, baseado na consonância [21]. É definido um autômato celular, cujas regras de evolução formam o genoma da população do algoritmo genético. São definidas algumas regras de musificação do autômato - a transformação de suas células em música. A peça musical resultante tem o seu *fitness* calculado de acordo com a consonância entre as suas notas, e o algoritmo genético continua a sua evolução quando toda a população tiver o seu *fitness* avaliado. A motivação dos autores é verificar as propriedades emergentes em sistemas de inteligência artificial em música, trabalhando com *fitness* baseado em consonância, seguindo a hipótese de que a evolução está presente na música e é possível detectar nessa evolução propriedades emergentes e organização de alto-nível, que de certa forma se

parecem com a evolução histórica da música ocidental, da forma que a conhecemos.

2.6 Computação Evolutiva

A computação evolutiva é um ramo da ciência da computação que propõe um paradigma alternativo ao processamento de dados convencional. Este novo paradigma, diferentemente do convencional, não exige, para resolver um problema, o conhecimento prévio de uma forma de encontrar uma solução. A computação evolutiva é baseada em mecanismos evolutivos encontrados na natureza, tais como a auto-organização e o comportamento adaptativo. Estes mecanismos foram descobertos e formalizados por Darwin em sua teoria da evolução natural, segundo a qual a vida na terra é o resultado de um processo de seleção, pelo meio ambiente, dos mais aptos e adaptados, que tem mais chances de reproduzir-se.

O termo *computação evolutiva* foi introduzido por volta de 1960 por Rechenberg e Schwefel [10]. A computação evolutiva está baseada em algumas idéias básicas que, quando implementadas, permitem simular em um computador o processo de passagem de gerações da evolução natural: a codificação das soluções, a população de soluções, a forma de avaliação dos indivíduos e os operadores genéticos.

A codificação da solução significa encontrar uma forma de representação simbólica da solução do problema tratado; uma estrutura que determine unicamente a solução do problema, como um vetor, uma matriz, um conjunto ou uma árvore, por exemplo, dependendo do problema. A codificação das soluções de um problema não é única, podendo ser modelada de várias formas diferentes, da forma que se achar mais conveniente. Essa codificação é denominada *genótipo* da solução.

A população de soluções é um conjunto de indivíduos (genótipos) representando soluções potenciais para o problema, que sofrerão o processo evolutivo. Os indivíduos mais adaptados vão continuar na população e devem se reproduzir, enquanto os menos adaptados irão morrer (sair da população). Para avaliar a adaptabilidade de um indivíduo, é necessário criar uma forma de avaliação dos indivíduos, denominada *função de* fitness, que não precisa ter nenhum conhecimento da solução do problema, bastando apenas conseguir avaliar quão adaptado (próximo da solução) é o indivíduo, ou menos do que isso - basta saber comparar dois indivíduos e indicar qual é o melhor. Com as indicações da função de *fitness*, são determinados quais os indivíduos que resistiram ao processo evolutivo. Sobre eles serão aplicados os chamados *operadores genéticos*, o equivalente computacional da reprodução e mutações genéticas. Os genótipos dos indivíduos são combinados e modificados por operações como *crossover* entre dois ou mais genótipos e *mutações*. Esses operadores são responsáveis pela criação de uma nova população, da geração seguinte, que novamente sofrerá o processo evolutivo, até que se encontre alguma condição de parada.

Resumindo, o conceito chave na computação evolutiva é o de *adaptação*: uma população inicial de soluções evolui, ao longo das gerações que são simuladas no processo, em direção a soluções mais adaptadas, isto é, com maior valor da função de *fitness*, por meio de operadores de seleção, mutação e recombinação.

Provavelmente a área mais conhecida dentro da computação evolutiva são os *algoritmos genéticos*, desenvolvidos por John Holland e o seu grupo de pesquisadores na Universidade de Michigan, e apresentados no livro *Adaptation in Natural and Artificial Systems* [73]. Outra classe importante é a

programação genética, apresentada por Koza em 1992 [96].

2.6.1 Algoritmos Genéticos

Algoritmos genéticos (GAs) são métodos de busca que têm motivação no processo de evolução natural. Para se encontrar a solução de um problema, é construída uma população de soluções em potencial. Os indivíduos são codificados simbolicamente como uma lista de atributos, denominada *genótipo*. Para cada indivíduo da população é atribuído um valor que indica o quão próximo ele está da solução. Esse valor é interpretado como a sua capacidade de adaptação ao meio, e é denominado de *fitness*. Os indivíduos com melhor *fitness* sobrevivem ao processo evolutivo e por meio de recombinações e mutações - denominados *operadores genéticos* - geram a população seguinte, que novamente sofre a pressão evolutiva, até que se interrompa o processo. Dessa forma, o *fitness* da população tende a aumentar gradativamente, até que se encontre um indivíduo que é a solução do problema, ou esteja suficientemente próximo dela.

Algoritmos genéticos apresentam diversas vantagens em relação a métodos tradicionais. Uma grande vantagem é o fato de não ser necessário ter muito conhecimento sobre o problema; basta apenas saber como avaliar o *fitness* de um indivíduo. Algoritmos genéticos se mostraram métodos de busca bastante eficientes, especialmente em problemas com espaço de solução muito grande. Além disso, a habilidade de retornar múltiplas soluções, propriedade desejada em domínios criativos, os torna bons candidatos para um método de busca para aplicações musicais [119].

Para a apresentação das aplicações, os algoritmos genéticos serão divididos de acordo com a forma de obtenção do *fitness*: formalístico (baseado em regras de teoria musical e fórmulas matemáticas), interativo (determinado pelo usuário) e neural (uma rede neural treinada para a avaliação do *fitness*).

Aplicações com *fitness* Formalístico

Horner e Goldberg estudaram o uso de algoritmos genéticos para a transfomação temática entre duas melodias, ou seja, modificar uma melodia durante um determinado período de tempo até que se transforme na segunda melodia [75]. No seu algoritmo, cada indivíduo representa uma série de variações sobre o tema inicial (troca de notas, inversão, mudanças de ritmo). Existem dois *fitness*: o primeiro mede a proximidade entre o tema inicial e o tema final gerado pelo indivíduo; o segundo verifica se a duração dos dois temas é compatível. Dado um tema inicial, um tema final e o número de notas entre os dois, o algoritmo genético chega a indivíduos que contêm variações que transformam o primeiro tema no segundo, utilizando operadores genéticos usuais, como crossover e seleção por torneio. Segundo os autores, a metodologia apresentada pode ser inclusive utilizada em outros aspectos de composição, como manipulação de timbres.

O problema de harmonização de uma linha melódica, já apresentado em outras seções, pode ser formulado como um problema de busca: encontrar, no espaço de possíveis harmonizações, aquelas que satisfazem as restrições determinadas. A utilização de algoritmos genéticos é uma boa escolha neste caso pois, como já foi dito, são eficientes quando o espaço de solução é muito grande, além da vantagem da obtenção de múltiplas soluções.

McIntyre criou um algoritmo genético para harmonização no estilo barroco [106]. O *fitness* é dividido em três etapas, em ordem de importância. Cada etapa seguinte só é calculada se foi obtido

um valor acima de 85% do máximo na etapa anterior. Desta forma se pretende atingir as propriedades mais importantes primeiro, e fazer o ajuste fino depois, acelerando a convergência. A primeira etapa é o *fitness* harmônico: como estão distribuídos os acordes e a distância entre as vozes. A segunda etapa avalia o movimento harmônico e o interesse. A última etapa avalia a suavidade do movimento das vozes. Os resultados obtidos foram considerados satisfatórios e, segundo o autor, é possível aplicar o mesmo método para outros estilos, sabendo encontrar as regras que os caracterizam melódica e harmonicamente.

Horner e Ayer apresentaram outro método de harmonização, um GA baseado em dois grupos de restrições [74]. O primeiro grupo define como cada acorde pode ter a sua distribuição de notas. O segundo define como cada voz pode mudar de um acorde para outro. O processo de harmonização ocorre em duas etapas, uma para cada grupo de restrições. Na primeira etapa monta-se um conjunto de acordes possíveis que satisfazem as restrições do primeiro grupo. Na segunda etapa ocorre o algoritmo genético, onde cada indivíduo representa uma sequência de acordes do conjunto gerado, e o seu *fitness* é o número de restrições do segundo grupo que não foram satisfeitas, ou seja, quanto mais próximo de zero, melhor o seu *fitness*.

Outro estudo em harmonização por Phon-Amnuaisuk, Tuson e Wiggins [123] deixa claro que o sucesso do uso de GA's depende da quantidade de informação codificada no sistema. Ao invés de funcionar como um GA geral, o seu algoritmo possui mais informações sobre o domínio do problema dado, que o tornam mais eficiente. As informações acerca do domínio estão codificadas na representação, nos operadores com sentido musical e no *fitness*, baseado em regras básicas de harmonização. As saídas do sistema foram corrigidas por um professor da faculdade de música da Universidade de Edinburgh, de acordo com o critério que ele utiliza para os seus alunos de harmonia do 1º ano. O melhor exemplo conseguiu 50%, garantindo a aprovação no curso. Apesar disso, a maioria das saídas ficou com notas em torno de 30%, principalmente por não apresentar uma progressão harmônica com um plano geral, pois no sistema só há informação sobre transições locais entre acordes. Os autores concluem que GA's não são ideais para a simulação do pensamento musical humano, e que algoritmos baseados em regras são mais adequados para a harmonização [124].

Papadopoulos e Wiggins usaram um algoritmo genético para evoluir melodias de jazz sobre uma progressão harmônica [118]. A construção dos operadores e da função que avalia o *fitness* foi feita de forma a imitar as ferramentas utilizadas comumente por um músico. Foram escolhidos operadores genéticos com sentido musical, como inversões, alterações rítmicas e operações de cópia de trechos do genótipo, bem como crossover de 1 e 2 pontos. A função de *fitness* é uma soma ponderada de oito funções que analisam características da melodia, penalizando características negativas, como a existência de intervalos muito grandes e notas fora do acorde, e bonificando características positivas, como notas do acorde em tempos fortes. Há ainda a possibilidade de incluir um algoritmo simples de reconhecimento de padrões na função de *fitness*, para atrair o desenvolvimento temático. De acordo com os autores, a melhor descrição do sistema é que ele reduz o espaço de busca eliminando soluções que tem maior probabilidades de serem musicalmente inadequadas.

Todd e Werner criaram um modelo de coevolução de duas subpopulações, compositores e críticos, onde as características de uma afetam a geração seguinte da outra [140]. O modelo é baseado numa população de pássaros, onde cada macho (compositor) tem uma melodia para tentar atrair uma fêmea (crítico), que avalia a melodia. O macho é representado apenas pela melodia, uma sequência de 32

notas restritas uma escala cromática em 2 oitavas. A fêmea é representada por uma matriz de transição 24x24, que contêm as probabilidades de transição de uma nota a outra durante a execução da melodia, e é utilizada para avaliar a melodia do macho. Foram usados 3 métodos diferentes de avaliação pelas fêmeas. O processo de seleção utilizado (seleção sexual) difere dos outros utilizados em composição musical (seleção natural) e mostrou-se bastante eficiente para manter a diversidade na população, tanto a diversidade sincrônica quanto anacrônica.

Um sistema evolutivo de composição em tempo real, o *Vox Populi*, foi desenvolvido por Moroni e colegas [113, 112]. Cada indivíduo da população representa um acorde de 4 vozes (baixo, tenor, contralto e soprano). Foi definido um critério de consonância, calculado em função da sobreposição entre os componentes harmônicos de duas vozes. O valor da consonância entre duas vozes varia entre 0 (não há sobreposição) e 1 (sobreposição total - as vozes são a mesma nota). O *fitness* de um indivíduo é baseado em três itens: o *fitness* melódico, que é calculado como o máximo da consonância entre as notas do acorde e um centro tonal definido; o *fitness* harmônico, calculado como máximo da consonância entre as notas do acorde; e o *fitness* de intervalo de vozes, que é um bônus para cada voz que pertence ao seu intervalo permitido. O processo evolutivo se inicia com indivíduos aleatórios, e cada nova geração é formada por indivíduos gerados por meio de operadores de *crossover* e *mutação*, com os mais aptos se reproduzindo com maior chance, além de um novo grupo de indivíduos aleatórios. Ao se iniciar a performance, é iniciado o processo evolutivo. Um parâmetro denominado controle biológico determina o número de gerações até que se selecione um indivíduo, o mais apto no momento, que será enviado para um buffer de performance. Um outro parâmetro, controle rítmico, varia a fatia de tempo em que as notas do buffer serão enviadas para a placa MIDI, controlando diretamente o ritmo da composição. Uma característica importante do *Vox Populi* é que o usuário pode alterar em tempo real os seus parâmetros através de uma interface gestual, modificando radicalmente a composição. Um bloco de desenho interativo supre uma área na qual duas curvas em 2D, uma vermelha e uma azul, podem ser desenhadas. A curva azul controla os controles biológico e rítmico, e a curva vermelha controla o centro tonal e o tamanho do intervalo permitido para as vozes. Logo, uma altera o ritmo, e a outra o *fitness* do processo evolutivo. Com a mudança desses parâmetros, a população terá de se adaptar a novas condições, fazendo com que a composição mude de característica gradualmente, por mais drástica que seja a mudança. Os autores sugerem o uso dessa interface como os gestos de um maestro. Cada gesto gera uma sonoridade diferente, e com suficiente treinamento, o usuário pode aprender a associar os gestos com suas sonoridades correspondentes, podendo direcionar a composição gerada.

Aplicações com *fitness* Interativo

Como a tarefa de avaliar a qualidade de uma composição musical é altamente subjetiva, algumas aplicações se baseiam na avaliação de um crítico humano para determinar o *fitness* dos indivíduos, ou direcionar o tipo ou o estilo da saída do sistema. Uma aplicação que utilize GA com essa abordagem é comumente denominada de *interactive genetic algorithm*(IGA).

Horowitz desenvolveu um sistema de geração de ritmos onde cada indivíduo da população é representado por uma série de notas e pausas, de 1 compasso de duração [76]. O usuário atribui um *fitness* para cada indivíduo de acordo com suas preferências ou do objetivo onde quer chegar. Usando técnicas tradicionais de reprodução, como *crossover* e *mutação*, novas populações que se aproximam

aos critérios desejados vão sendo geradas. Além disso, uma função de *fitness* determinística permite a evolução automática de populações durante um certo número de gerações, tornando o processo mais rápido. Essa função analisa parâmetros como síncope, densidade e repetição. O usuário inicia os parâmetros dessa função de *fitness* de acordo o objetivo que procura, e um segundo algoritmo genético vai evoluindo populações desses parâmetros. O uso de algoritmos genéticos em paralelo acelerou a convergência para ritmos que satisfazem as exigências do usuário.

Biles usou um algoritmo genético para criar o seu sistema de improvisação em Jazz, o *GenJam* [16]. *GenJam* é um GA que modela um aluno aprendendo a improvisar, com o seu desempenho (*fitness*) sendo medido pelo seu professor (usuário). Dada uma série de acordes como entrada, o sistema cria um improviso sobre essa progressão. O sistema possui 2 populações distintas: compassos e frases, o que lhe dá uma maior flexibilidade. Os operadores genéticos utilizados não foram operadores padrão, como *crossover* e *mutação*, mas sim operadores com algum sentido musical, como inversões, transposições e alterações rítmicas. Enquanto o GenJam executa suas frases, o usuário pode avaliar a frase em tempo real, afetando o *fitness* do indivíduo sendo executado. As populações vão sendo alteradas pelos operadores genéticos, com os indivíduos de maior *fitness* se reproduzindo mais vezes e com maior chance de sobrevivência, o que melhora a qualidade da população, e consequentemente dos improvisos gerados. O sistema de Biles conseguiu bons resultados musicais. O seu maior problema é o *gargalo de* fitness, a ineficiência causada pelo fato de o usuário ter de avaliar as frases uma por uma durante a execução, limitando o tamanho da população e tornando a evolução lenta.

Para tentar resolver esse problema, Biles tentou usar uma rede neural para a avaliação do *fitness* [19], sem muito sucesso, como veremos na próxima seção. Outra forma de contornar esse problema foi simplesmente eliminar o *fitness*, no *AutoGenJam* [18]. A população inicial é gerada a partir de frases retiradas de um livro de frases de jazz. Sem a avaliação do *fitness*, o processo seletivo também foi eliminado. Ocorrem apenas mutações com sentido musical e um crossover inteligente, que procura manter a musicalidade das frases originais. Com isso, alguma novidade é introduzida sem que se destrua a qualidade das frases originais. Esta versão do GenJam também produziu bons resultados, até melhores do que o GenJam original em alguns casos. Mas será que o GenJam ainda é um algoritmo genético, mesmo sem *fitness* e processo seletivo? Todas as descrições de GA apresentam como passo fundamental uma avaliação do *fitness*, inexistente no AutoGenJam, embora ele apresente todos os outros passos essenciais. Vendo pelo lado da seleção por melhores indivíduos em busca de um aumento da qualidade da população a cada geração, o AutoGenJam não é um GA. Mas Biles considera o AutoGenJam ainda um GA, onde a informação sobre o problema foi totalmente codificada na iniciação da população e nos operadores, sendo excluída da função de *fitness*. Outro ponto importante é que dessa forma se elimina a convergência para algumas poucas soluções de alto *fitness*, que destrói a diversidade, algo desastroso neste sistema onde não se busca uma solução única, mas sim uma população inteira de soluções. A falta da pressão seletiva sobre a população faz com que essa convergência não ocorra, garantindo a diversidade.

Uma novidade interessante introduzida no *GenJam* é o *Interactive GenJam* [17], onde além do modo tradicional de operação, o *GenJam* pode improvisar interativamente com um solista humano, num modo conhecido em música popular como "quatro e quatro" (trading fours), onde cada solista improvisa durante quatro compassos, alternadamente. O GenJam escuta os 4 compassos do solista humano através de um conversor *pitch-to-MIDI*, aplica algumas mutações musicais, e toca o resultado

nos 4 compassos seguintes, que serão um desenvolvimento temático da frase humana.

Jacob desenvolveu um sistema de composição que, ao invés de construir melodias através de notas individuais, trabalha diretamente com frases musicais, aplicando variações sobre um conjunto de motivos iniciais [85]. Trabalhando com blocos maiores é possível reduzir o espaço de busca imensamente, além de garantir uma coesão temática inerente ao sistema. O algoritmo se divide em três módulos: *composer*, que gera material musical, *ear*, que avalia esse material, e *arranger*, que constrói a composição final. O usuário fornece um conjunto inicial de motivos, que são usados pelo composer para gerar frases através de variações musicais aplicadas sobre os motivos. O usuário atribui um *fitness* a cada uma dessas frases. Esse *fitness* será usado em um algoritmo genético que evolui os parâmetros do *composer*, para que ele simule cada vez melhor a forma de composição do usuário. As frases são combinadas em blocos polifônicos, que são passados para o ear, que contêm um conjunto de transições harmônicas permitidas. As frases que contêm transições inválidas são descartadas. O módulo *ear* tem os seus parâmetros (transições harmônicas permitidas) iniciados de forma aleatória. Durante o processo de composição, o usuário atribui um *fitness* a cada transição de acordo com o seu gosto ou objetivo, e o *ear*, assim como o composer, vão evoluindo na direção desejada pelo seu usuário. Quanto um certo número de frases foi aceito, o *arranger* gera ordenações dessas frases, que também são avaliadas pelo usuário e recombinadas, em um algoritmo genético, num processo similar ao problema do caixeiro viajante para encontrar a melhor ordenação.

Em um artigo seguinte [86] defende que existem dois tipos de criatividade: a inspiração (*inspiration*) e a transpiração (*hard-work*). A primeira é pouco conhecida, portanto impossível de ser modelada atualmente. Já a segunda é essencialmente algorítmica. O objetivo então é achar as regras que modelam essa segunda forma de criatividade. Para isso, é preciso um algoritmo que simule o mais próximo possível o método de composição de seu usuário, e um mecanismo que consiga avaliar rapidamente a viabilidade de um trecho musical.

Na maioria dos casos, os IGAs conseguem bons resultados estéticos, pois a avaliação humana é bem melhor do que as outras formas encontradas até então. Apesar disso, os IGAs tem algumas desvantagens. Primeiro, a ineficiência, o *gargalo de* fitness; não é possível criar populações muito grandes e nem rodar muitas iterações, já que o usuário tem que avaliar cada um dos indivíduos da população a cada geração. Em alguns casos são usadas representações simplificadas para os indivíduos da população, para reduzir o imenso espaço de busca, comprometendo a qualidade do resultado. Além disso, este método mostra pouco sobre os processos mentais de composição, escondidos na mente do avaliador do *fitness*.

Aplicações com *fitness* Neural

O uso de redes neurais para avaliar o *fitness* permite que, ao invés de avaliar diretamente ou ter de criar regras que calculem o *fitness*, o usuário possa treinar o algoritmo em um conjunto de exemplos musicais conhecidos, e esperar que a rede neural consiga atribuir o *fitness* como o usuário o faria, em novos exemplos. Para uma introdução a redes neurais, veja a seção 2.7.1.

Gibson e Byrne desenvolveram um sistema que produz pequenas composições em quatro partes, em um domínio reduzido por uma série de restrições [65]. Uma restrição assegura que todas as composições tenham o mesmo tamanho; restrições de tonalidade previnem mudanças de tom, e restrições harmônicas permitem apenas os acordes da tônica, subdominante e dominante do tom. Uma rede

neural foi treinada de forma supervisionada para atribuir o valor bom ou ruim a ritmos de 4 tempos, avaliados pelos autores. Essa rede foi usada para guiar um algoritmo genético na busca de novos bons ritmos. Cada indivíduo da população representa um conjunto de 4 ritmos de 4 tempos, e o *fitness* do indivíduo é calculado de acordo com o *fitness* de cada ritmo que o compõe. Ao final do algoritmo genético, o melhor indivíduo encontrado tem o seu ritmo preenchido por notas, para formar uma melodia. Esse processo é executado por duas redes neurais. Uma determina o intervalo entre notas subsequentes, e a outra controla a estrutura harmônica. O sistema foi capaz de gerar melodias com um certo sucesso, mas devido ao grande número de restrições, as composições eram pouco ousadas.

Para resolver o problema da ineficiência da avaliação humana do *fitness*, Biles, Anderson e Loggi introduziram no GenJam uma rede neural que serviria de filtro para que os indivíduos com *fitness* muito baixo não fossem apresentados ao mentor humano, acelerando o processo evolutivo [19]. Nos experimentos verificou-se que foi necessário um número muito grande de vértice ocultos para obter uma boa performance, e mesmo nos casos em que a rede foi capaz de aprender o conjunto de treinamento, falhou no conjunto de teste. Várias combinações de diferentes parâmetros estatísticos foram tentadas como entrada para a rede, como o número de notas novas em um compasso, o tamanho do maior intervalo, o número de mudanças de direção, entre outros, sem sucesso. Tentou-se utilizar também histogramas de intervalos e notas como entrada, sozinhos ou em combinação com os parâmetros estatísticos anteriores, também sem sucesso. A conclusão dos autores é de que seres humanos escutam música de maneira complexa e sutil, que não é facilmente capturada por modelos estatísticos simples. A automação do *fitness* pode ser possível com o uso de técnicas de inteligência artificial com bastante informação musical codificada, como as utilizadas pela comunidade de pesquisa em cognição musical no estudo de modelos de inteligência artificial para a escuta e análise musical. Tais abordagens podem funcionar, mas são difíceis de implementar e necessitam de bastante capacidade computacional.

Uma desvantagem do uso de redes neurais para a avaliação do *fitness* é que o processo de treinamento pode consumir muito tempo. Além disso, após o término do treinamento, não há a possibilidade de ser modificada a forma como a rede neural avalia o *fitness*. Isso significa que novos indivíduos gerados podem ser avaliados com baixo *fitness*, quando de fato possuem um *fitness* alto, devido ao método de treinamento empregado e a incapacidade da rede de se atualizar para conseguir avaliar novos casos que não estavam presentes no conjunto de treinamento. Para resolver esses problemas, Burton e Vladimirova [25] utilizam uma rede neural ART (*Adaptive Resonance Theory*) [115] para a avaliação do *fitness*. Redes Neurais ART são redes auto-organizadoras, que utilizam treinamento não-supervisionado e algoritmos de *clustering* para o reconhecimento de padrões. Uma vantagem do treinamento não-supervisionado é que não é necessário um conhecimento inicial sobre a forma de classificação dos padrões - a própria rede determina a classificação. Não há limite para o número de *clusters* de padrões criados. Se um padrão analisado é suficientemente próximo de um *cluster* existente, ele é adicionado a esse *cluster*, caso contrário um novo *cluster* é criado para acomodar o padrão. A similaridade entre os padrões é controlada de acordo com um parâmetro de vigilância, um limite que indica se um padrão é similar a algum outro padrão já classificado, ou se é necessário criar um novo *cluster*.

Em um outro artigo, Burton e Vladimirova testaram a sua rede ART em um algoritmo genético para a geração de ritmos [26]. A rede neural utilizou dados de alguns arquivos MIDI existentes com padrões rítmicos. O processo evolutivo foi gerando categorias de padrões rítmicos similares. Os

autores comentam que os resultados obtidos indicaram que redes neurais ART podem ser uma boa opção de avaliação de *fitness*, com vantagens sobre outros tipos de redes neurais.

A principal característica das redes ART é que elas resolvem o dilema *estabilidade-plasticidade* [70], que pode ser resumido com a pergunta: Como pode um sistema de aprendizado ser projetado para ser adaptativo, respondendo a eventos significativos e mantendo a estabilidade em eventos irrelevantes, e com isso aprendendo novas situações, além de preservar o conhecimento já adquirido?

2.6.2 Programação Genética

A programação genética é uma extensão de algoritmos genéticos apresentada por Koza, em seu livro *Genetic Programming: On the Programming of Computers by Means of Natural Selection* [96]. O processo evolutivo é idêntico ao de algoritmos genéticos, mas a representação é diferente. Ao invés de listas de atributos, o genótipo é uma *árvore de atributos*. Em programação genética o indivíduo que deve ser evoluído é um programa de computador definido como uma árvore de funções e terminais. Cada função é um ramo da árvore e cada terminal uma folha. Funções podem ser condições, sensores, operações aritméticas ou lógicas, entre outras. Programação genética é geralmente utilizada quando a solução buscada é um programa ou função que deve executar uma certa tarefa desejada. A programação genética tem algumas vantagens em relação a algoritmos genéticos usuais em aplicações musicais: os indivíduos não possuem tamanho fixo, e é possível gerar funções que criam estrutura musical muito mais facilmente.

No algoritmo de Laine e Kuuskankare, a melodia é considerada uma função com domínio (tempo) e imagem (notas musicais) discretos, cujo valor determina a nota da melodia em um dado instante de tempo [99]. Para ilustrar com um simples exemplo, uma melodia que sobe um passo da escala a cada instante pode ser representada pela função $f(k) = f(k-1) + 1$, com k discreto e um valor inicial definido. Melodias mais complexas podem ser geradas usando combinações de funções não-lineares. Usando programação genética, é possível encontrar funções que gerem melodias similares a um conjunto inicial de entrada, e gerar variações sobre o conjunto de entrada. O maior problema deste método é que a produção de peças mais longas é muito difícil, pois as funções necessárias para gerar melodias grandes se tornam muito complexas.

Spector e Alpern apresentam um sistema de composição que utiliza programação genética e a avaliação do *fitness* é baseada em uma rede neural treinada com melodias do saxofonista Charlie Parker, expandindo um sistema anterior baseado em regras [134]. O objetivo é a criação de uma melodia de 1 compasso, em resposta a uma melodia de entrada também de 1 compasso. Os indivíduos do sistema são *músicos*: funções que modificam a melodia de entrada através de variações musicais comuns, como transposição, inversão, mudança rítmica, etc. Cada indivíduo recebe o *fitness* de acordo com a avaliação feita pela rede neural de sua melodia de saída e o processo evolutivo segue, aplicando-se operadores genéticos padrão para chegar à próxima geração, e assim sucessivamente. Percebeu-se que a rede neural conseguiu capturar alguns aspectos abstratos de estilo e estrutura musical, que dificilmente são modelados por regras formais. Ainda assim, a rede neural atribuiu *fitness* alto a algumas melodias consideradas pelos autores como sem interesse musical. Uma forma de melhorar a saída é a combinação de críticos neurais e simbólicos (regras). Dessa forma foram obtidos resultados melhores, pois as regras garantem um mínimo de estrutura. Outra forma de melhorar a saída é sofisticar a rede neural ou o seu treinamento, pois ficou claro para os autores que o conjunto de treinamento foi muito

restrito.

Johanson e Poli também criaram melodias simples como o seu sistema GP-Music [88]. Os indivíduos são representados por árvores onde os nós terminais são notas musicais, e os outros nós são funções com um ou mais parâmetros que aplicam variações musicais como inversão, duplicação, concatenação, entre outras. O *fitness* de cada indivíduo é obtido de duas formas: a primeira é pela avaliação do usuário. Algumas melodias agradáveis foram obtidas dessa forma. A segunda forma, numa tentativa de diminuir o gargalo de *fitness*, é uma rede neural treinada de forma supervisionada para tentar avaliar as melodias da mesma forma que o usuário. As melodias geradas com este tipo de avaliação não foram tão boas quanto as anteriores.

A principal dificuldade na implementação de um sistema de composição que utiliza métodos evolutivos é a avaliação do *fitness*. Os casos onde o *fitness* é avaliado pelo usuário geralmente apresentam melhores resultados, mas sofrem com a ineficiência (gargalo de *fitness*). Além disso, as peças geradas refletem diretamente o gosto do usuário, com pouco espaço para a generalização ou mudança de estilo. No caso *fitness* avaliado por regras simbólicas formais, as dificuldades são semelhantes às encontradas em sistemas baseados em regras (seção 2.3), especialmente a dificuldade em encontrar as regras. Redes neurais têm apresentado indícios promissores, e tem a vantagem do aprendizado. Mas, até o momento, ainda não se conseguiu aplicá-las com bons resultados. O uso combinado de regras formais e interação humana para a avaliação do *fitness*, como no Vox populi [113], é uma boa alternativa a esse problema.

2.7 Aprendizado de máquina

O termo aprendizado de máquina é utilizado para designar sistemas que, em geral, não possuem nenhum conhecimento *a priori* do domínio, e aprendem através de exemplos. Podemos classificar este tipo de sistema de acordo com a forma de armazenamento do conhecimento; conhecimento *subsimbólico* (redes neurais) ou simbólico. Em música, sistemas que usam aprendizado de máquina adquirem conhecimento através da análise de peças musicais existentes. Esse conhecimento pode ser sobre algum aspecto isolado, como melodia, variações rítmicas, ou sobre vários aspectos simultaneamente. O conhecimento adquirido pode ser utilizado como ferramenta para a análise musical, ou como a base para um sistema de composição e/ou performance. A grande motivação para o uso de sistemas de aprendizado é que não é necessário encontrar regras complexas *a priori*; as regras serão formuladas através do treinamento.

2.7.1 Redes Neurais Artificiais

Redes neurais artificiais (RNA) são modelos que procuram simular as redes neurais naturais do organismo humano. Neste tipo de abordagem procura-se construir um sistema que modele os circuitos cerebrais e espera-se a emergência de um comportamento inteligente. Por ser baseado em redes de elementos e nas conexões entre eles, esta abordagem é também denominada *conexionismo*. Uma explicação resumida de redes neurais naturais será apresentada, para em seguida ser apresentado o conceito de redes neurais artificiais.

A célula básica de uma rede neural é o neurônio. Um neurônio tem um corpo celular chamado *soma*, e recebe sinais através de ramificações denominados *dendritos*, transmitindo sinais através de

uma outra ramificação, geralmente única, conhecida como *axônio*. A transmissão de sinais entre neurônios ocorre quando se tem uma perturbação da membrana do neurônio, causando uma pequena variação de tensão elétrica, conhecida como *potencial de ação*. A formação de um potencial de ação pode ser causada por vários tipos de estímulos, por exemplo, elétricos ou químicos. Esse potencial vai percorrendo o axônio até chegar à região da sinapse, o que causa a liberação de moléculas conhecidas como *neurotransmissores*, que irão se colar nos dendritos do outro neurônio. Os neurotransmissores causam uma modificação na membrana do neurônio receptor, o que pode gerar um novo potencial de ação, excitando o neurônio, ou dificultar o aparecimento de um potencial, ou seja, inibindo o neurônio. A transmissão de sinais entre os neurônios depende do tipo dos neurotransmissores, da sua abundância e da sensibilidade da membrana a excitações. Todos esses fatores determinam como cada neurônio pode excitar ou inibir outro neurônio. Modificando esses fatores, todo o comportamento da rede neural é modificado. Geralmente essas mudanças estão relacionadas ao aprendizado da rede neural. Em uma rede neural natural, os dendritos de um neurônio podem receber informações de fontes externas (dos órgãos dos sentidos, por exemplo) ou de axônios de outros neurônios. O axônio, por sua vez, pode estar ligado a dendritos de outros neurônios ou a outros órgãos (nos motoneurônios, por exemplo, a saída dos axônios excita os músculos). Os neurônios que recebem excitação do exterior são *neurônios de entrada*; os que têm sua saída direcionada para alterar o exterior são os *neurônios de saída*; e os neurônios que têm as suas entradas e saídas conectadas a outros neurônios são os *neurônios internos* ou *ocultos*.

2.7.2 Componentes Básicos das Redes Neurais Artificiais

Em 1943, McCullogh, neurobiólogo, e Pitts, estatístico, propuseram o primeiro modelo de um neurônio artificial em seu artigo "*A logical calculus of ideas imminent in nervous activity*" [105]. Motivado por esse artigo, Frank Rosenblatt apresentou o perceptron em seu artigo "*The Perceptron: a Perceiving and Recognizing Automaton*" [131], considerado a primeira geração de redes neurais. Nesta seção serão apresentados brevemente os conceitos básicos de redes neurais artificiais.

Neurônio Artificial

Vários modelos diferentes de redes neurais já foram propostos, mas praticamente todos possuem como característica comum uma célula básica - o neurônio artificial. O modelo mais utilizado é uma generalização do modelo de McCullogh e Pitts e está ilustrado na figura 2.3.

Neste modelo de neurônio artificial, os dendritos são representados pelas entradas sinápticas y_i, que são números reais. Os valores w_i, também reais, são denominados *pesos sinápticos* e determinam como as entradas são combinadas para calcular a entrada efetiva u, também chamada *confluência*. A função comumente utilizada para calcular u é a soma ponderada das entradas, dada pela fórmula

$$u = \sum_{i=1}^{N} w_i y_i + \theta \tag{2.3}$$

O parâmetro θ é chamado *bias*. A entrada efetiva u é avaliada por uma função f, chamada *função de ativação*, para gerar a saída do neurônio, o valor a. Essa função, na maioria dos casos, é a função sigmóide, por ter a vantagem de se poder calcular a derivada em um ponto diretamente com o valor da

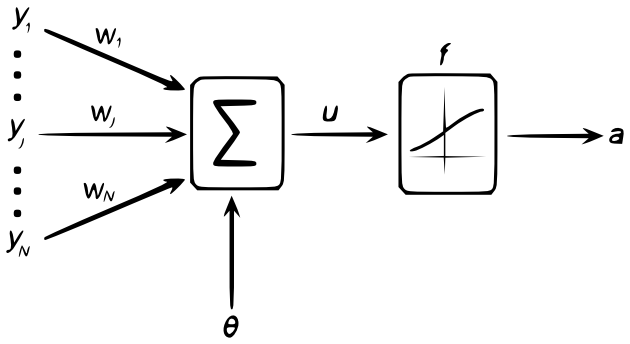

Figura 2.3: Modelo de um neurônio artificial

função nesse ponto. Outras funções também utilizadas são a tangente hiperbólica, tangente inversa, funções gaussianas, funções lineares por partes e função limite.

Topologia das RNAs

Em uma RNA, os neurônios estão interconectados em uma rede de forma a facilitar a computação distribuída. A configuração dessas conexões pode ser descrita eficientemente com um grafo direcionado, que consiste em um conjunto de vértices (os neurônios) e arcos direcionados(as entradas sinápticas). Essa rede é usualmente denominada *topologia da RNA*. Uma rede neural é uma rede direta (*feed forward* ou *acyclic*) se o grafo correspondente não tem ciclos. Se o grafo tiver ciclos, a rede é recorrente(*recurrent* ou *cyclic*). Devido ao ciclo recorrente, tais redes levam a sistemas dinâmicos não lineares e exibem um comportamento bastante complexo. A figura 2.4 mostra uma exemplo de rede direta e rede recorrente.

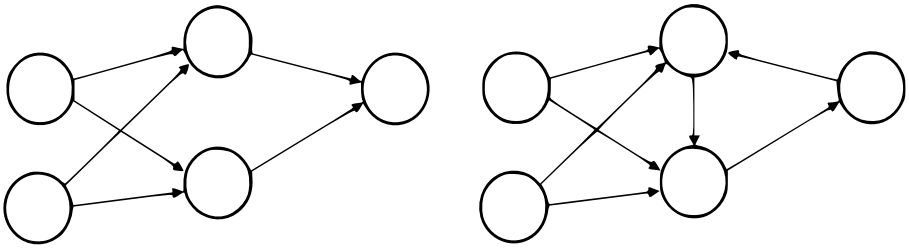

Figura 2.4: Uma rede direta e uma rede recorrente

O modelo mais utilizado de redes neurais é o *multilayer perceptron model* (MLP), uma rede neural direta em camadas. Para introduzir o MLP, primeiro será apresentado o conceito de *perceptron*.

Perceptron

O *perceptron* é um modelo de introduzido por Rosenblatt em 1957. O modelo consiste em um único neurônio, com soma ponderada das entradas e função de ativação dada por uma função limite. Dado um vetor de entrada $\overline{y} = (y_1, y_2, \ldots, y_N)$, a entrada efetiva u é dada pela fórmula 2.3 e a saída a é calculada pela função limite

$$a = f(u) = \begin{cases} 1, & u \geq 0 \\ 0, & u < 0 \end{cases} \tag{2.4}$$

O *perceptron* pode ser usado para a classificação de um conjuto de vetores de entrada em dois grupos. Para isso, é necessário treinar o perceptron para ajustar os seus pesos sinápticos. Um conjunto de treinamento $\{(\bar{y}_i, d_i); i \in I_r\}$ e um conjunto de testes $\{(\bar{y}_j, d_j); j \in I_t\}$ devem ser utilizados. Os valores d_i são as saídas desejadas da rede aos vetores $\bar{y}$, com $d_i \in \{0, 1\}$, e I_r e I_t são os conjuntos contendo os índices dos pares $(\bar{y}, d)$, e devem ser disjuntos. O conjunto de treinamento será utilizado para ajustar os valores dos pesos sinápticos e o conjunto de testes é utilizado para avaliar se o perceptron está classificando corretamente os vetores.

Os pesos sinápticos w_i serão ajustados por um algoritmo iterativo de aprendizado, onde os vetores do conjunto de treinamento são alimentados à rede e para cada vetor temos uma iteração k, os pesos são ajustados de acordo com a fórmula

$$\bar{w}_{k+1} = \bar{w}_k + \eta(d_k - a_k)\bar{x}_k \tag{2.5}$$

onde $\bar{w}_k = (w_1^k, w_2^k, \ldots, w_N^k)$, e w_i^k é o valor do peso sináptico w_i na iteração k. Da mesma forma, a_k é o valor de a na iteração k, e a é calculado com as equações 2.3 e 2.4. O parâmetro η é denominado *taxa de aprendizado* e deve ser fornecido *a priori*. É possível provar que, se os vetores de entrada forem linearmente independentes, este algoritmo fará os pesos sinápticos convergirem para soluções factíveis em um número finito de iterações. Esta é na verdade a principal desvantagem deste modelo, pois ele não convergirá para um conjunto de vetores que não é linearmente independente. Outra desvantagem é que, mesmo convergindo, não é possível prever se haverá convergência rápida. Essas desvantagens motivam a criação de modelos mais complexos, como é o caso do MLP.

Multilayer Perceptron Model(MLP)

Uma rede neural MLP consiste em uma rede de neurônios direta e em camadas. Cada neurônio na rede possui uma função de ativação não-linear, geralmente contínua e diferenciável. As funções mais utilizadas são a sigmóide e a tangente hiperbólica. A figura 2.5 mostra um exemplo de uma rede MLP, com 1 camada de entrada, 2 camadas intermediárias (também chamadas camadas ocultas) e 1 camada de saída. Este é o tipo mais comum de rede neural, principalmente por existirem métodos de treinamento bastante difundidos e fáceis de usar.

2.7.3 Aprendizado

"Aprender é o ato que produz um comportamento diferente a um estímulo externo devido a excitações recebidas no passado e é, de uma certa forma, sinônimo de aquisição de conhecimento" [14].

As redes neurais possuem capacidade de aprendizado através de exemplos, ou seja, são capazes de modificar o seu comportamento se ajustando aos exemplos apresentados, além de conseguir generalizar o conhecimento adquirido. A forma mais comum de classificar o aprendizado é segundo a realimentação explícita durante o aprendizado.

Quando há um professor que assinala regularmente os erros e os acertos da rede neural, temos um *aprendizado supervisionado*. Neste tipo de aprendizado as correções apresentadas pelo professor são

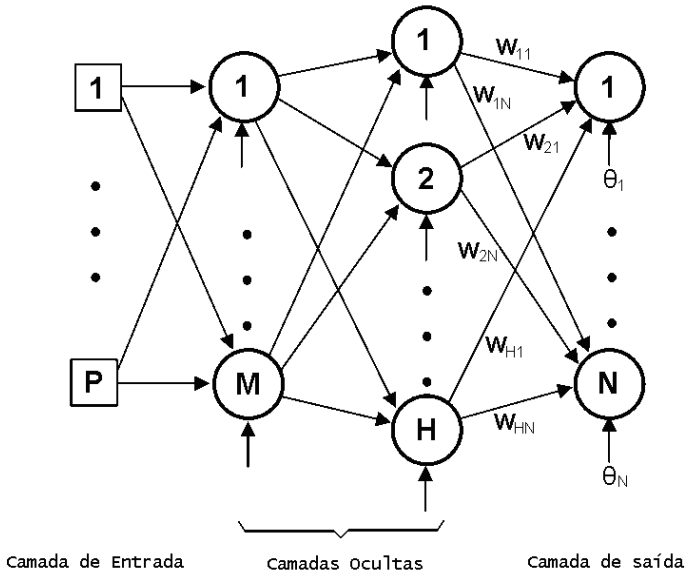

Figura 2.5: Exemplo de uma rede MLP

utilizadas para modificar as conexões sinápticas da rede e diminuir o erro da saída da rede. Este modo de aprendizado pode ser considerado um problema de minimização do erro entre a saída da rede e a saída desejada. Quando não há um professor explícito, temos o *aprendizado não-supervisionado*. Este tipo de aprendizado é utilizado geralmente quando se usam redes neurais para problemas de *clustering*, reconhecimento de padrões e estimação de densidade.

Aprendizado Supervisionado - Retropropagação

A forma mais utilizada (mas não única) de aprendizado supervisionado em redes MLP é a retropropagação (*backpropagation*). O objetivo da retropropagação é minimizar o erro da rede neural modificando os seus pesos sinápticos. Para isso, apresenta-se à rede um exemplo do conjunto de treinamento e se obtêm o erro da rede, comparando a saída da rede com a saída esperada. É calculado o gradiente do erro em relação aos pesos sinápticos da camada de saída, que é atualizada por um tamanho de passo escolhido. Com isso é possível calcular o erro da saída da penúltima camada, repetindo o processo por todas as camadas. Esta foi a primeira regra de aprendizado inventada para efetuar o treinamento supervisionado de redes diretas com mais de duas camadas. Apesar de ser bastante utilizado, possui algumas deficiências críticas, como a convergência muito lenta se os parâmetros de aprendizado não estão bem ajustados e a convergência para mínimos locais no espaço do erro. Várias melhorias têm sido sugeridas nos últimos anos, como métodos para ajuste ótimo dos parâmetros e alternativas ao método de otimização de máxima descida (gradiente), como o método de gradientes conjugados ou método de Newton.

Aprendizado não-supervisionado e redes auto-organizadas

O aprendizado supervisionado se baseia em treinar a rede oferecendo um conjunto de treinamento que possui os exemplo de entrada e a saída desejada, utilizando métodos de minimização do erro

de saída. Mas existem problemas onde não é fornecida a saída desejada, o conjunto de treinamento consiste apenas no exemplos de entrada e as informações necessárias ao aprendizado devem ser retiradas apenas desses exemplos. Esse é o *aprendizado não-supervisionado*. Este tipo de aprendizado é utilizado para problemas de *clustering* e reconhecimento de padrões, por exemplo.

Redes auto-organizadas são uma classe de redes neurais que se utilizam do aprendizao não-supervisionado. Existem vários tipos de redes auto-organizadas aplicáveis a uma grande variedade de problemas. Serão expostos brevemente 3 tipos comuns de redes auto-organizadas: as redes de *aprendizado competitivo*, apresentadas por Rumelhart e Zipser [132], as *redes de Kohonen* [93] e as *redes ART* apresentadas por Carpenter e Grossberg [34].

(a) Aprendizado Competitivo

No aprendizado competitivo, neurônios pertencentes a uma única camada competem entre si para representar o vetor de entrada. Apenas o neurônio vencedor tem os seus pesos sinápticos alterados para ficar ainda mais perto do vetor de entrada. Esta abordagem é conhecida como winner-takes-all. Vetores de entrada similares terão como vencedor o mesmo neurônio, o que faz com que redes com aprendizado competitivo sejam consideradas algoritmos de clustering. A figura 2.6 mostra uma rede neural competitiva.

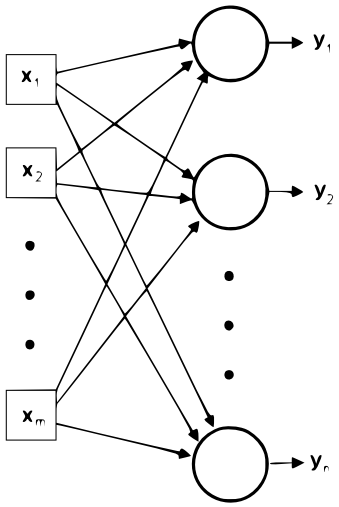

Figura 2.6: Rede Neural Competitiva

Cada uma das m entradas está conectada aos n neurônios de saída. Para determinar o neurônio vencedor, geralmente é utilizada a regra da distância euclidiana, onde o neurônio vencedor k é o que fica a uma menor distância euclidiana do vetor de entrada $\bar{x} = (x_1, x_2, \ldots, x_m)$, ou seja,

$$||\bar{w}_k - \bar{x}|| \leq ||\bar{w}_i - \bar{x}|| \quad i = 1, 2, \ldots, n \quad i \neq k \tag{2.6}$$

onde $\bar{w}_i$ é o vetor de pesos sinápticos do neurônio i. Os pesos serão atualizados de acordo com a equação

$$\bar{w}_k^{(t+1)} = \bar{w}_k^{(t)} + \eta(\bar{x}^{(t)} - \bar{w}_k^{(t)}) \tag{2.7}$$

onde $\bar{w}_k^{(t)}$ é o vetor de pesos sinápticos do neurônio k na iteração t do processo de treinamento, $\bar{x}^{(t)}$ é o k-ésimo vetor de entrada. Logo, os pesos sinápticos se tornarão cada vez mais próximos dos

vetores de entrada, como mostra a figura reffig:clustering para o caso de três neurônios, com vetores de treinamento normalizados.

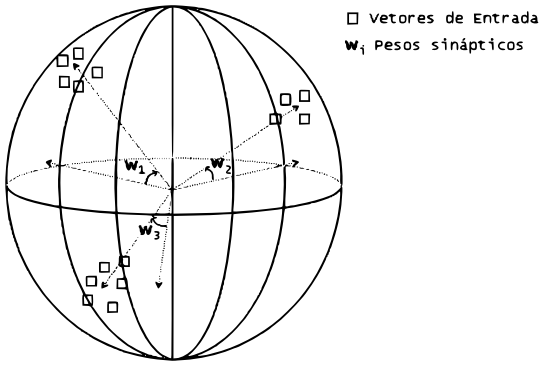

Figura 2.7: Pesos sinápticos se deslocam após o treinamento

Um ponto importante neste tipo de implementação é a iniciação dos pesos sinápticos. Pode-se iniciá-los aleatoriamente, mas, especialmente em espaços com dimensão muito grande, é possível que alguns pesos nunca sejam escolhidos vencedores e dessa forma nunca sejam utilizados. Uma saída para esse problema é escolher os pesos com valores de vetores do conjunto de treinamento, escolhidos aleatoriamente. Outras formas mais sofisticadas de contornar esse problema já foram propostas, como o *frequency sensitive competitive learning* [2] e o *leaky learning* [97].

(b) Redes de Kohonen

As Redes de Kohonen, ou *self-organizing maps*, também são consideradas redes competitivas, mas com uma topologia diferente. Este modelo também é formado por apenas duas camadas, a de entrada e de saída. Os neurônios na camada de saída estão dispostos em um mapa, geralmente uma matriz 2x2, chamado SOM (*self-organizing map* - mapa auto-organizador). Esse mapa determina também a vizinhança de cada neurônio. A figura 2.8 mostra um exemplo de uma rede de Kohonen, com os neurônios da camada de saída organizados em uma matriz 6x5.

Como no caso anterior há a competição entre os neurônios, e vetores semelhantes terão como vencedor o mesmo neurônio. A diferença principal é que ao invés de atualizar os pesos sinápticos apenas do vencedor, são atualizados também os pesos dos neurônios pertencentes à sua vizinhança. Dessa forma, vetores com características semelhantes vão se agrupando em regiões próximas no mapa. As redes de Kohonen são úteis para diversos problemas de *clustering* e análise de dados, como classificação de caracteres na escrita manual, identificação de assinaturas e reconhecimento de voz, por exemplo [141].

(c) Redes ART(*adaptive resonance theory*)

As redes ART constituem uma família de arquiteturas neurais desenvolvidas especialmente para resolver o dilema da *estabilidade-plasticidade*. Esse dilema pode ser resumido na pergunta "como aprender coisas novas (plasticidade), certificando-se que o conhecimento já adquirido não é apagado ou corrompido?". Este tipo de arquitetura possui apenas duas camadas, uma de entrada e uma de saída, e o aprendizado se dá de forma competitiva e não-supervisionada, assim como as redes apresentadas

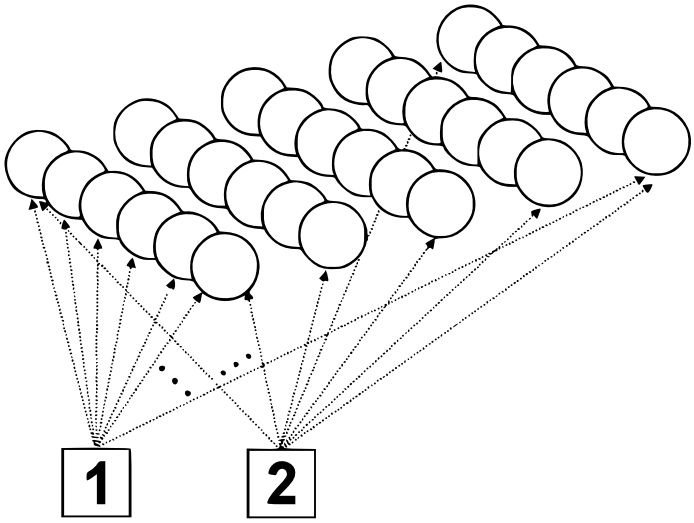

Figura 2.8: Rede auto-organizadora de Kohonen

nas 2 seções anteriores. A diferença principal na arquitetura é que as redes ART são redes recorrentes, ou seja, o grafo que corresponde à sua topologia contêm ciclos, como foi definido na página 23.

As redes ART são bastante utilizadas em problemas de *clustering* e reconhecimento de padrões, quando não se tem uma idéia precisa da quantidade de padrões distintos que irão ocorrer. A rede contém um número necessário de neurônios de saída, que não são utilizados até que seja necessário. Ao analisar uma nova entrada, a rede verifica a similaridade entre o neurônio vencedor e a entrada. Se forem suficientemente similares, a entrada é classificada na categoria referente ao neurônio vencedor, que tem os seus pesos ajustados. Caso contrário, uma nova categoria é iniciada (um novo neurônio de saída é ativado) e essa entrada passa a ser o primeiro representante dessa nova categoria. Se não existem neurônios disponíveis para aceitar uma nova categoria, a entrada é ignorada pela rede. O grau de similaridade necessário para criar uma nova categoria ou classificar a entrada em uma categoria existente é controlado pelo *parâmetro de vigilância* ρ. Alterando esse parâmetro é possível ter um controle no número de categorias criadas pela rede. Para alguns exemplos de aplicações que utilizam redes ART, ver os trabalhos de [35], [95] e [94].

2.7.4 RNAs em Aplicações Musicais

Uma das principais motivações de se utilizar RNA em música é a sua capacidade de aprendizado. Como o conhecimento musical é bastante abstrato, de forma a ser difícil formalizá-lo simbolicamente, com redes neurais é possível introduzir o conhecimento no sistema de forma subsimbólica, com o treinamento através de exemplos. Uma das aplicações mais comuns de RNA em música é a avaliação da qualidade musical. A rede é treinada de forma supervisionada para avaliar a qualidade de peças ou trechos musicais da mesma forma que o um usuário humano o faria. Várias aplicações que utilizam métodos evolutivos calculam o *fitness* através de uma rede neural. Algumas dessas aplicações já foram apresentadas na seção 2.6.1(aplicações com *fitness* neural).

Outro uso frequente é treinar uma RNA em composições de um determinado estilo musical, ou o estilo de um autor específico, tentando capturar a sua essência. A rede geralmente é treinada de

forma a conseguir prever os passos seguintes de uma composição. Por exemplo, se a rede é treinada utilizando-se como entrada o conteúdo harmônico das peças, a sua função será prever o próximo acorde; se for treinada sobre a melodia, deverá prever a próxima nota ou conjunto de notas. Dessa forma, uma rede treinada pode ser utilizada para compor novas peças (melodias, harmonias, etc.) dentro do estilo capturado.

Todd usou uma rede neural direta com treinamento supervisionado para a geração de melodias, em uma das primeiras utilizações de RNA para aplicações musicais [139]. A topologia sugerida por Todd é utilizada em boa parte das aplicações para previsão/geração de melodias, na forma original ou com pequenas alterações [119].

O sistema híbrido *NetNeg* [67] também utiliza RNA para a composição de melodias, ou mais especificamente, contraponto a duas vozes no estilo do século XVI. O sistema é dividido em duas partes: uma rede neural, semelhante à utilizada por Todd, para a previsão da nota seguinte mais provável. A rede foi treinada em quatro melodias do compositor Knud Jeppesen. A segunda parte é formada por dois *agentes inteligentes* [146], um conceito em Inteligência Artificial que basicamente representa um módulo de *software* independente programado para receber uma tarefa e decidir como resolvê-la. Esses agentes podem apresentar uma série de características como adaptação ao meio, aprendizado, comunicação e cooperação entre diferentes agentes e até entre agente e ser humano. No caso do *NetNeg*, cada um representando uma das vozes do contraponto. Os agentes recebem a lista de notas mais prováveis, de acordo com a rede neural, e decidem dinamicamente qual é o par de notas a ser adicionado. O conhecimento musical dos agentes é modelado utilizando regras de contraponto tiradas de um livro de Jeppesen.

Mozer também gerou melodias através de uma RNA [114]. A RNA foi treinada com a intenção de substituir métodos anteriores onde se utilizam tabelas de transição nota a nota para compor linhas melódicas. A rede funciona como um preditor, onde a cada nota da melodia é apresentada, uma a uma, e sua tarefa é prever qual será a próxima nota. Mozer conclui que, apesar de gerar melodias melhores do que as geradas através de uma tabela de transição de 3ª ordem, ainda falta "coerência global". Esse problema se origina principalmente do fato de que foram utilizadas as transições nota a nota para o treinamento da rede, dificultando a aquisição de informações de níveis musicais mais elevados. Por outro lado, com a arquitetura da rede neural utilizada é difícil capturar informações locais e ao mesmo tempo informações mais globais. Algumas direções possíveis para tentar solucionar esse problema são comentadas.

Franklin usou uma rede neural com treinamento em duas fases no seu sistema *CHIME* (*computer human interacting musical entity*), para improvisação em jazz [52, 54, 53]. O objetivo é alternar solos humanos com a resposta do computador, um processo conhecido em música popular como *trading fours*, pois usualmente cada músico, durante a sua vez, improvisa durante quatro compassos. NA primeira fase, a rede passa por treinamento supervisionado para aprender a reproduzir três melodias do músico de jazz Sonny Rollins. Na segunda fase é aplicado o aprendizado reforçado, por meio de regras obtidas com a ajuda de um músico profissional. O sistema continua aprendendo a cada execução, enquanto alterna solos com um humano. Apesar de algumas limitações, como o pequeno conjunto de regras utilizado, o *CHIME* é um exemplo interessante de interação homem-máquina.

O problema clássico da harmonização em quatro vozes também foi tratado com redes neurais. O HARMONET [72] é um sistema híbrido que harmoniza melodias usando uma combinação de redes

neurais com técnicas de CSP (ver seção 2.3). A tarefa de harmonização é dividida em 3 partes: o esqueleto harmônico, o preenchimento dos acordes e os ornamentos. Na primeira, a mais importante, a estrutura harmônica da peça é criada, com um acorde por tempo. Para isso, uma rede neural foi treinada com backpropagation em um conjunto de peças de J.S. Bach. A segunda tarefa envolve o preenchimento das notas dos acordes tendo como entrada a melodia dada e o esqueleto harmônico da fase anterior. Primeiro são gerados todos os acordes possíveis que se encaixam na melodia dada e o esqueleto harmônico gerado. Depois, regras simbólicas testam os acordes em questões básicas como a proibição de oitavas e quintas paralelas e a tessitura de cada voz, suavidade e equilíbrio das vozes, entre outras. O melhor acorde é escolhido. Na última fase, são adicionados ornamentos. Uma outra rede neural decide onde podem ser adicionadas colcheias entre os tempos, e em que vozes. O HARMONET tem a vantagem de ser mais flexível em relação a algoritmos de harmonização que utilizam apenas regras, pois apenas as regras básicas de harmonização estão formalizadas, grande parte do conhecimento está embutido na rede neural. Para simular o estilo de outro compositor bastaria mudar o conjunto de treinamento. Já no caso da abordagem por regras, grande parte das regras teria de ser reescrita.

Alguns anos mais tarde, Feulner se juntou a Hörnel para desenvolver o MELONET, um sistema de criação de variações melódicas baseadas na harmonia [50, 78]. Hörnel é um dos mais férteis pesquisadores na área de aplicações musicais utilizando RNA. Um ano antes, ele apresentou o SYSTHEMA [77], para a análise e síntese de melodias clássicas onde a representação utilizada não leva em consideração apenas notas individuais, mas *motivos*. Essa representação mais completa é capaz de captar informação musical sobre níveis hierárquicos mais elevados, e foi utilizada no sistema MELONET. Além disso, várias redes neurais trabalhando em escalas diferentes de tempo, combinando aprendizado supervisionado e não-supervisionado, são utilizadas para que seja possível capturar macroestruturas musicais. O sistema funciona em duas etapas: na primeira, uma melodia de entrada é harmonizada pelo sistema HARMONET. Em seguida, uma rede neural é responsável pela aplicação de variações sobre uma das vozes. Essa rede neural foi treinada com exemplos de variações de J. Pachelbel. As composições foram consideradas de boa qualidade, e o resultado positivo foi atribuído especialmente à forma de representação das melodias.

Em artigos posteriores o MELONET e o HARMONET foram melhorados através de um processo evolutivo de otimização [81, 82]. A ferramenta ENZO utiliza algoritmos genéticos para a otimização de redes neurais, adicionando e removendo pesos e unidades. Os dois sistemas foram otimizados pelo ENZO, que ajustou pesos e tamanhos das redes utilizadas, melhorando a sua performance. Os dois sistemas são reapresentados mais tarde, com algumas modificações e melhorias [79]. Em um artigo seguinte de Hörnel e Olbrich foi apresentado um método de análise e comparação de estilos, utilizando redes neurais do HARMONET [80]. Treinando o HARMONET em diferentes estilos, é possível usar o HARMONET como um identificador de estilos. Em outro artigo, o sistema foi modificado para a harmonização em tempo real [83]. O novo sistema foi denominado HARMOTHEATER. O funcionamento é semelhante ao do HARMONET. A diferença é que alguns dados de entrada usados para o HARMONET não estarão disponíveis para o HARMOTHEATER, pois só se tem a informação até o tempo atual. Qualquer informação futura, como informações de fraseado e desenvolvimento melódico posterior, não estará disponível. Além disso, melodias tocadas em tempo real podem não estar fixas a uma tonalidade, podendo modular ao longo do tempo. Para abordar esses problemas, foi desenvolvido

um método para a previsão dinâmica do centro tonal. O problema da predição do conteúdo harmônico já foi abordado por vários pesquisadores [47, 138, 15, 62, 125]. No caso do HARMOTHEATER, a performance foi melhorada sensivelmente ao utilizar a predição do centro tonal.

Gang e Lehmann também desenvolveram um sistema para a predição de acordes em 1995, para depois apresentar uma versão em tempo real [63]. O sistema é semelhante à rede neural do HARMONET. A rede foi treinada com 18 músicas folclóricas com harmonias simples, até que a rede pudesse reproduzir os exemplos. Depois, a rede foi testada com novas melodias. As sequências de acordes geradas foram consideradas de boa qualidade por músicos profissionais. A rede em tempo real teve uma performance um pouco pior, o que já era esperado, pelo fato de não possuir informações da melodia adiante. Em um artigo posterior, o sistema foi testado com várias configurações diferentes de parâmetros da rede, para poder ajustá-los para valores ótimos [64].

Um sistema de predição de acordes baseado na tensão musical foi apresentado por Melo e Wiggins [108]. Em um artigo anterior, Melo usou duas redes neurais cooperativas em níveis hierárquicos diferentes para capturar a tensão musical de uma peça [107]. As RNA's foram treinadas tendo como entrada uma curva média de tensão dada por 10 ouvintes que escutaram o último movimento da 1ª sinfonia de Prokofiev. O sistema pôde prever bem a parte desconhecida da peça (80% foi usado no treino), e podia também gerar música dada uma curva de tensão, embora os resultados não tenham sido tão bons. Usando essa pesquisa como base, o trabalho foi sofisticado, utilizando-se como entrada a tensão musical e também a sequência harmônica. Dessa forma espera-se que a rede seja capaz de prever sequencialmente o conteúdo harmônico de uma peça, dada a sua curva de tensão. Consequentemente é possível também compor novas harmonias no estilo do conjunto de treinamento, que se encaixem na curva de tensão desejada. Os resultados indicaram que existe uma correlação entre tensão musical e conteúdo harmônico, e que a tensão pode ser utilizada para a produção de novas harmonias, e provavelmente para outros elementos musicais também.

Buzzanca busca um método de reconhecimento de estilo musical [28]. Ele já havia tentado resolver esse problema com gramáticas [27] (ver seção 2.4), mas são apresentadas várias razões para mostrar que gramáticas não são satisfatórias para o reconhecimento de um estilo. É muito difícil encontrar um conjunto de regras que consiga capturar um estilo e ao mesmo tempo ter a capacidade de generalização. Buzzanca conclui que redes neurais são muito mais apropriadas para essa tarefa, e apresenta uma rede neural treinada para o reconhecimento de estilos musicais, baseada na técnica de pesos compartilhados (*shared weights*) de LeCun [100].

Redes neurais também começaram a ser utilizadas em reconhecimento de padrões, nas mais diversas áreas [87]. Na seção 2.8 serão apresentadas algumas aplicações que utilizam RNAs em reconhecimento de padrões em um contexto musical.

Sistemas de aprendizado, como redes neurais, oferecem formas alternativas de composição algorítmica, que de certa forma se parecem com a atividade cerebral do ser humano. Redes neurais têm se mostrado métodos eficientes para a cognição musical. Pelos exemplos apresentados percebe-se que a sua qualidade melhora quando combinada com formas de conhecimento simbólico. Apesar disso, RNAs também apresentam algumas desvantagens [119]:

- Redes neurais são ótimas para cognição, mas a composição é um processo muito mais intelectual, mais simbólico. As redes neurais capturam com sucesso a estrutura superficial de uma melodia e reproduz novas melodias de acordo com esse conhecimento recém-adquirido, mas

não é capaz de absorver aspectos mais abstratos, como fraseado e tonalidade, por exemplo.

- A representação do tempo não pode ser tratada eficientemente mesmo com redes neurais recorrentes.

- A sua grande vantagem é ser um sistema que aprende através de exemplos, mas nem sempre isso é completamente verdadeiro, pois em muitos casos um humano precisa filtrar exemplos contraditórios do conjunto de treinamento para que o treino prossiga como o desejado. Além disso, o conjunto de treinamento pode acabar limitando a saída do algoritmo.

- Pesquisadores da área apresentam como uma grande vantagem de RNA's o fato de que elas podem aprender nos exemplos coisas que não podem ser representadas por regras, como as exceções. Entretanto ainda não se viu nenhuma implementação onde isso ocorre comprovadamente.

2.7.5 Raciocínio Baseado em Casos (*Case-Based Reasoning* - CBR)

CBR é uma técnica recente de raciocínio automatizado (*automated reasoning*) e aprendizado de máquina. Em CBR, um problema é resolvido comparando-o com um conjunto de problemas já resolvidos, buscando os problemas similares e adaptando as suas soluções para o problema atual. A característica mais importante de sistemas CBR é a capacidade de utilizar o conhecimento de situações semelhantes já resolvidas. Além disso, os sistemas CBR estão continuamente aprendendo, pois cada novo problema resolvido pode ser adicionado à sua biblioteca de casos resolvidos, aumentando o seu conhecimento.

CBR é mais uma alternativa, na área de aprendizado de máquina, a sistemas com conhecimento baseado em regras, sendo especialmente apropriado em casos onde o número de regras necessárias para capturar um determinado conhecimento é intratável, ou o conhecimento sobre o domínio é incompleto ou muito difícil de ser formalizado.

Um sistema CBR pode ter várias formas de aplicação: pode adaptar e combinar soluções existentes para encontrar a solução de um novo problema; explicar uma nova situação através da experiência adquirida em situações já experimentadas; criticar novas soluções com base em soluções antigas. Todos esses diferentes aspectos podem ser classificados em dois tipos principais: *interpretative* CBR e *problem solving* CBR. No primeiro, o aspecto principal é discutir se uma nova situação pode ser tratada como situações anteriores através das suas diferenças e similaridades. No segundo, o objetivo é construir uma solução para um novo caso através de adaptações das soluções de casos já resolvidos. Esta divisão é teórica, pois na prática vários problemas têm componentes dos dois casos, e um sistema CBR eficiente certamente tem que utilizar uma combinação dos dois tipos.

Em geral, dado uma situação a ser resolvida, um sistema CBR utiliza os seguintes passos[103]:

1. Recuperar casos relevantes da memória (o que requer uma organização apropriada da memória de acordo com as características dos problemas)

2. Selecionar os melhores casos

3. Derivar uma solução

4. Avaliar a solução - para assegurar que soluções ruins não sejam repetidas

5. Armazenar a nova solução na memória

Uma forma comum de descrever os passos de um sistema CBR, primeiro utilizada por Aamodt e Plaza [1], são os quatro R's - *retrieve* (passos 1 e 2), *reuse* (3), *revise* (4) and *retain*(5).

O primeiro problema a ser resolvido para a implementação de um sistema CBR é a indexação-recuperação-seleção de casos, que corresponde ao passo *retrieve*. Os passos seguintes só serão bem sucedidos se os casos selecionados forem relevantes, e a recuperação dos casos relevantes por sua vez dependerá de uma boa indexação. Para o problema da indexação as mais diversas técnicas estão sendo pesquisadas. O foco dessas pesquisas é desenvolver métodos que encontrem as características mais importantes de cada caso, descartando as características irrelevantes, para poder efetuar uma indexação precisa. Para o passo de recuperação e seleção dos casos relevantes, são necessárias técnicas de comparação, como *nearest neighbor*, e a definição de uma métrica eficiente. Como algumas características são mais importantes do que outras, várias métricas estudadas incorporam pesos diferentes para cada característica.

Outro problema fundamental é a organização da memória. Uma boa indexação não é suficiente quando a memória de casos é muito grande. Uma organização linear, como uma lista, por exemplo, é uma forma ineficiente para a recuperação. Uma forma muito mais eficiente seria uma estrutura hierárquica, onde nós internos são generalizações de casos individuais, como em sistemas baseados no modelo de memória dinâmica de Schank [133]. Outra forma bastante utilizada é a apresentada por Bareiss, Porter e Wier em seu sistema PROTOS [12]. A grande maioria dos sistemas CBR utiliza uma das duas formas, ou até uma combinação das duas. Um problema que deve ser levado em consideração é o crescimento exagerado da memória, que pode levar à degradação da performance. Um sistema robusto deve ser capaz de tratar este problema. Soluções comuns são a escolha rigorosa dos casos que devem entrar na memória, a introdução de um limite superior no número de casos e até a eliminação ocasional de casos existentes.

Após escolher o(s) problema(s) relevante(s), é necessária a adaptação para encontrar a solução procurada. Uma boa adaptação de casos existentes ao caso atual pode reduzir significativamente o trabalho necessário para a resolução. O interesse neste aspecto do CBR tem aumentado nos últimos anos. Várias técnicas de adaptação estão sendo apresentadas, desde técnicas de interpolação a algoritmos genéticos.

Como boa parte dos sistemas em inteligência artificial, existem sistemas que utilizam CBR em conjunto com outras técnicas. Existem diversos estudos que apresentam sistemas que combinam CBR com outros tipos de raciocínio artificial: CBR com lógica *fuzzy*, CBR com sistemas baseados em regras, CBR com *argumentation-based reasoning* e CBR com *Bayesian reasoning*, para citar alguns exemplos. Lopes de Mantaras faz uma descrição de vários sistemas que utilizam CBR, em conjunto com outras técnicas ou sozinho [103].

Aplicações Musicais

Uma aplicação musical que utiliza CBR é o *SaxEx* [9, 8]. *SaxEx* é um sistema CBR para a geração de performances expressivas de melodias baseado em performances expressivas humanas. A análise e síntese do som são feitas através de um modelo espectral SMS (*spectral modeling synthesis*). SMS

é um conjunto de técnicas para análise, transformação e síntese de som. Através da análise espectral do som monofônico de entrada, o SMS consegue extrair atributos complexos como rubato, dinâmica, vibrato, frequência média e articulação, por exemplo. Esses atributos podem ser modificados e novamente adicionados à representação espectral sem perda de qualidade. Isso torna o SMS ideal para o SaxEx. O SMS calcula os atributos do som de entrada e o SaxEx os modifica, devolvendo novamente para que o SMS monte a saída. O *SaxEx* vai modificar esses parâmetros através do CBR. Em um artigo posterior [7], o *SaxEx* recebeu algumas melhorias, como o uso de lógica *fuzzy* e a interatividade com usuário. Utilizando a experiência adquirida com o SaxEx, Grachten, Arcos e López de Mántaras desenvolveram o *Tempo-Express*, um sistema CBR para a aplicação de transformações de tempo no contexto de melodias de jazz [68]. Diferentemente do *SaxEx*, a entrada do *Tempo-Express* é uma gravação expressiva que terá o seu tempo de execução alterado, acelerando ou retardando a gravação da forma que um músico humano o faria, e não apenas uma transformação uniforme.

Em outra abordagem, Ramalho e Ganascia utilizaram CBR como uma das parte de um sistema para simular criatividade, no domínio da improvisação e acompanhamento em *jazz* [127]. Segundo os autores, a criatividade de um músico de *jazz* durante o improviso vem de dois aspectos: regras e memória. As regras foram aprendidas usualmente através de professores e métodos de ensino. Essas regras não são difíceis de formalizar. Mas o conhecimento musical não se resume a tais regras. A maior parte do conhecimento musical é a memória, obtida ao escutar e imitar os grandes mestres de *jazz*. A memória é muito difícil de ser formalizada; como criar regras que transformam conceitos como tensão, *swing* e estilo, por exemplo, em uma performance musical? Outro aspecto abordado pelos autores é que as ações musicais são executadas pelos músicos levando em consideração um contexto que vai sendo criado durante a apresentação. A interação entre os instrumentistas e destes com o público vai alterando o contexto forçando os músicos a expressarem o seu conhecimento como uma resposta rápida aos eventos que estão acontecendo, ao invés de uma busca precisa pela melhor resposta musical. A criatividade ocorre dento do confronto contínuo entre o conhecimento do músico e o contexto da performance ao vivo.

Os autores utilizam o conceito de PACTs (*Potential ACTions*), proposto por Pachet [116], que representam as ações ou intenções que os músicos podem tomar durante a performance. As PACTs podem atuar em uma ou mais dentre três dimensões musicais: ritmo, intensidade e altura. Uma PACT pode ter desde um nível de abstração bem alto, como "toque com ritmo em síncope" (atuando no ritmo) ou "use a escala maior" (atuando na altura), passando por níveis mais inferiores, como "toque forte" (atuando na intensidade) e "toque um arpejo ascendente" (atuando na altura), chegando ao nível mais baixo, onde todas as dimensões estão determinadas e não há nenhum grau de liberdade, como no exemplo "toque esta frase um tom acima". Uma PACT no nível mais baixo é denominada PACT executável (*playable* PACT). As PACTs podem ser combinadas para que se chegue a um nível cada vez mais baixo, podendo chegar eventualmente a uma PACT executável.

Mas, só com combinações de PACTs não se garante que se chegue sempre a PACTs executáveis. Para isso foi desenvolvida uma *memória musical*, que utiliza CBR. A memória acumula casos de performances de músicos reais, e está codificada como PACTs de baixo nível.

Durante a execução, existe um *contexto*, formado pelos acordes do trecho e os eventos que vão ocorrendo, como a interação entre os músicos e a platéia; esse contexto será analisado para que se monte a *memória a curto-prazo*. Essa memória será utilizada para que se montem as notas que serão

executadas pelo músico (uma PACT executável).

Em um artigo posterior, Ramalho, Rolland e Ganascia aplicaram o sistema proposto para simular o comportamento de um baixista de jazz durante uma performance ao vivo [128]. Apesar de algumas limitações levantadas pelos autores, o modelo de criatividade do sistema é bastante interessante, baseado na experiência acumulada pelo músico e o contexto musical da performance ao vivo.

2.8 Reconhecimento e Extração de Padrões

Psicólogos e analistas musicais enfatizam que a identificação de repetições relevantes em música é um processo pelo qual um ouvinte treinado alcança uma interpretação complexa de um trabalho musical [110]. Esse fato justifica a utilização de métodos de reconhecimento e extração de padrões relevantes, muitos deles derivados de outras áreas, por este ser um problema bastante estudado na área de engenharia. Reconhecimento de padrões é o problema de, dado um padrão determinado, encontrar no conjunto desejado exemplos desse padrão, exatos ou aproximados. A extração de padrões é um problema pouco mais complexo, pois exige que, dado o conjunto desejado, se encontrem padrões que se repetem frequentemente.

Um método de reconhecimento e/ou extração de padrões pode ser utilizado de diversas formas. Pode ser um componente em um sistema de cognição musical, uma ferramenta para a análise musical, quebrando a estrutura em blocos menores, ou ainda ser um assistente para a composição. Um ponto fundamental ao funcionamento deste tipo de algoritmo é a codificação da informação musical. Já existem vários métodos na literatura de engenharia para o reconhecimento e extração de padrões em sequências de símbolos (*strings*). Boa parte das abordagens em música computacional aproveita esses métodos codificando eventos musicais como *strings*.

Carpinteiro usa uma rede neural para o reconhecimento de sequências musicais codificadas como *strings* [36]. O sistema é uma extensão do rede auto-organizadora (SOM - *self-organizing map*, veja seção 2.7.1) de Kohonen, com duas redes SOM trabalhando em conjunto de forma hierárquica. A melodia escolhida para testar o modelo foi a terceira voz da 16ª fuga do cravo bem temperado de J.S. Bach. Foram designados dois padrões de referência, e a rede foi capaz de discriminar se outros trechos da melodia eram exemplos de um dos padrões.

Conklin e Anagnostopoulou descrevem uma técnica de reconhecimento de padrões utilizando a noção de *múltiplos pontos de vista* (*multiple viewpoint*) [41], apresentada por Conklin anteriormente [42]. Nessa nova codificação, um trecho musical é representado por um certo número de *strings*, cada uma representando um aspecto musical (ponto de vista) diferente. Exemplos de pontos de vista são o contorno melódico, a altura das notas e o intervalo de tempo entre notas consecutivas, por exemplo. A estrutura musical em um nível hierárquico mais elevado também é representada, de uma forma rudimentar. Em alguns pontos de vista, por exemplo, só a primeira nota de cada tempo é representada, ou só a primeira nota de cada compasso. Um algoritmo eficiente foi desenvolvido para encontrar padrões em um conjunto de obras, e não em uma única obra, como costuma ser feito em outros algoritmos. Um método estatístico é utilizado para restringir a busca apenas a padrões que ocorrem muito mais vezes do que o esperado. A forma de codificação foi sofisticada em um artigo posterior [40], para poder representar também estruturas verticais (harmonia), além das estruturas musicais horizontais já representadas (linhas melódicas). Esta é uma abordagem nova e aparentemente bastante promissora.

A grande maioria dos métodos de reconhecimento de padrões em música analisa as peças musicais horizontalmente, e essa nova visão pode trazer bons resultados.

Rolland desenvolveu um algoritmo de extração de padrões melódicos denominado *FlExPat* (*Flexible Extraction of Patterns*) [130]. O algoritmo é dividido em 2 etapas: na primeira, *passage comparison*, todos os trechos da melodia são comparados 2 a 2 para calcular uma distância bastante conhecida, a *edit distance*, que é o número de alterações necessárias para que os 2 padrões se tornem iguais. Além das alterações básicas, inclusão, exclusão e troca, novas operações com significado musical são incluídas. Essa distância determinará a similaridade entre dois trechos. Nessa fase é montado um *grafo de similaridade*, onde cada trecho será representado por um vértice. Se a similaridade entre dois trechos ultrapassa um determinado limiar, é criado um arco entre os dois vértices correspondentes, com um peso proporcional à similaridade. Com o grafo montado, passa-se à segunda fase, de categorização. Nessa fase serão extraídos padrões do grafo de similaridade através de um algoritmo conhecido como *Star Centre*. Cada vértice terá um valor denominado *proeminência* (*prominence*), que é a soma dos pesos de todos os seus arcos adjacentes. Os vértices serão ordenados decrescentemente em relação à proeminência, indicando quais são os padrões mais relevantes da melodia de entrada.

Cambouropoulos e colegas apresentam dois algoritmos para o reconhecimento de padrões aproximados [30]. Cada um dos algoritmos tem a sua definição da distância entre dois padrões. Em outro artigo são discutidos vários aspectos importantes para a implementação de um algoritmo de reconhecimento e extração de padrões, como a forma de representação, técnicas para a representar estruturas musicais mais elevadas e pré-segmentação para melhorar a eficiência [29]. Em outros artigos subsequentes foi apresentado um algoritmo de *clustering* [31, 32]: tendo como entrada uma melodia pré-segmentada, o algoritmo constrói uma representação adequada de cada segmento em vários níveis musicais e organiza esses segmentos em categorias relevantes, determinando automaticamente a quantidade necessária de categorias distintas.

Uma nova representação que não utiliza *strings* foi proposta por Meredith, Lemström e Wiggins [109, 110]. Os autores apresentam o SIA (*structure induction algorithm*), um algoritmo que descobre os padrões mais repetidos em qualquer conjunto de pontos em um espaço Cartesiano de qualquer dimensão, de forma eficiente. O SIATEC é uma extensão do SIA que gera um conjunto de TECs (*translational equivalent classes*). Dois elementos são *translationally equivalents* quando um pode ser obtido através do outro por meio de uma translação. Portanto, uma TEC é uma classe onde todos os seus elementos são *translationally equivalents*. O SIA é então aplicado no conjunto de TECs, ao invés do conjunto original. Para aplicar o SIATEC em música, os dados serão representados de forma multidimensional. Em um primeiro momento, as notas musicais foram codificadas como um par (altura musical, tempo de início), mas podem ser adicionadas outras dimensões como timbre, intensidade e duração, por exemplo, pois o SIATEC trabalha em qualquer dimensão. O SIATEC foi aplicado em algumas peças musicais, e os resultados indicam que, apesar de conseguir encontrar algumas TECs com significado musical, encontra também várias que não seriam normalmente classificadas como tendo algum significado musical. Para resolver esse problema, estão sendo desenvolvidas heurísticas que selecionem apenas TECs com significado musical.

David apresentou o sistema *Sorcerer* para a descoberta de alusões musicais [46]. A intenção não é apresentar um novo algoritmo de reconhecimento de padrões, mas sim mudar o objetivo de aplicar um algoritmo desse tipo. O objetivo é tentar encontrar, numa determinada composição que se pretende

analisar, alusões a padrões existentes em um outro conjunto de obras, de onde se imagina que o autor da primeira composição possa ter sido influenciado. O *Sorcerer* pode ser utilizado por musicólogos para revelar alusões desconhecidas, ou confirmar algumas já suspeitadas, sendo um ótimo recurso para o estudo do estilo de um compositor. O programa pode ser utilizado também por compositores que desejam verificar até que ponto suas próprias composições são influenciadas por algum outro compositor ou estilo.

2.9 Performance Expressiva

A palavra expressividade é usada, na maioria dos estudos em performance musical, para indicar uma presença sistemática de desvios da notação musical [148]. Foram propostas várias formas de representar esses desvios, que podem ser classificados de acordo com a abordagem utilizada para construir o modelo de representação. Um dos métodos mais relevantes é o de *analysis-by-measurement*, que é baseado na análise da medição dos desvios praticados em performances humanas. Essa análise visa reconhecer padrões nos desvios praticados e descrevê-los através de fórmulas numéricas, como no trabalho de Canazza e colegas [33].

Outro método consiste em derivar modelos, descritos por um conjunto de regras, pelo método de análise por síntese (*analysis-by-synthesis*). Essas regras definem quantitativamente os desvios a serem aplicados em uma partitura musical, de forma a produzir uma performance mais humana. Cada regra modela um princípio musical utilizado por músicos humanos em sua performance. Nesta abordagem, primeiro as regras são obtidas, através das indicações de músicos profissionais; depois, as performances geradas aplicando as regras são avaliadas, o que permite um ajuste das regras. O conjunto de regras mais reconhecido é o *KTH rule system* [58, 57], onde cada regra tem um peso controlado por parâmetros k, permitindo que se modele a saída adaptando as regras para diferentes situações. Uma implementação em LISP dessa regras, o *Director Musices*, foi apresentado por Friberg [59]. O sistema foi sofisticado mais tarde por Bresin e Friberg com a adição de emoções na performance [24]. Nesse artigo, o objetivo dos autores é aumentar a flexibilidade do *Director Musices* para produzir performances incorporem qualidade emocional. Gabrielsson [61, 60] e Juslin [91, 92] propuseram uma lista de sinais expressivos (*expressive cues*) que são características de cada emoção básica (medo, raiva, felicidade, tristeza, solenidade e ternura). Os sinais estão descritos de forma quantitativa e dizem respeito elementos como tempo, volume, articulação, timbre e *ritardando* final, entre outros. O método utilizado foi análise por síntese: as regras e seus parâmetros modelando uma determinada emoção foram escolhidos dentre as regras KTH, e após escutar o resultado os parâmetros são alterados e o processo é repetido. Após um grande número de testes, um consenso foi obtido, resultando em um conjunto de regras para cada emoção. Outras melhorias são introduzidas por Sundberg, Friberg e Bresin [137], que analisam a performance das regras de variação rítmica, comparando com a performance de pianistas profissionais. Em outra abordagem, Bresin implementou uma rede neural capaz de aprender as regras KTH, com generalização suficiente para aplicar variações expressivas em peças diferentes daquelas do conjunto de treinamento [23, 22].

Um trabalho recente de Zanon e Poli também utiliza as regras KTH, e o foco é desenvolver um método ótimo de estimação dos parâmetros k [148].

Outras formas de se chegar a regras que imprimam expressividade em performances computacio-

nais incluem diversas técnicas de aprendizado de máquina [84, 144, 143, 142, 145].

Capítulo 3

O Algoritmo Proposto

3.1 Introdução

O algoritmo proposto, *Bebopper*, é um sistema de composição e performance em tempo real. Ele cria um improviso melódico e uma seção rítmica composta de bateria, baixo e piano, em tempo real. O usuário, através da interface do sistema, escolhe a sequência de acordes a ser executada, definindo o número de repetições e a se a música terá alguma forma definida (AABA, por exemplo). A saída do sistema é enviada para um *buffer* MIDI, que pode ser direcionado para qualquer dispositivo MIDI presente no computador onde está sendo executado. Tipicamente, a saída MIDI será enviada para a placa de som ou para um instrumento MIDI conectado ao computador. As decisões durante a criação do improviso são tomadas por meio de um novo modelo estocástico, que busca simular o pensamento de um músico durante o improviso. Durante a execução, o usuário pode avaliar a qualidade do improviso que está sendo executado, clicando em alguns botões próprios da interface. As notas dadas pelo usuário serão utilizadas para ajustar alguns parâmetros do modelo estocástico, melhorando a sua saída a cada execução, em um processo de aprendizado supervisionado. O objetivo desta pesquisa é a criação de um novo modelo estocástico mais adequado a simular as escolhas de um músico durante a execução de seu improviso, além da implementação de regras que consigam capturar o estilo de improvisação em questão, o *bebop*. Podemos dividir o algoritmo em três partes principais: a criação do improviso melódico, a criação da seção rítmica e o modelo estocástico proposto. Nas seções seguintes serão apresentadas detalhadamente cada uma dessas partes do algoritmo, para em seguida apresentar o funcionamento do algoritmo.

3.2 O improviso melódico

O improviso melódico a ser criado pelo algoritmo é baseado no estilo de *jazz* conhecido como *bebop*. Esse estilo surgiu por volta de 1940-50 e, entre outras inovações, mudou a forma de se pensar o improviso em *jazz*, influenciando solistas de *jazz* e de vários outros estilos até hoje. Várias circunstâncias históricas contribuíram para o surgimento do *bebop*. A segunda guerra mundial foi um dos principais elementos. Com o alistamento de diversos músicos profissionais e os cortes de gastos impostos pelas novas taxas de entretenimento, a situação ficou cada vez mais difícil para as *big bands* tradicionais da época, com menos profissionais disponíveis e menos clubes oferecendo contra-

tos. Com isso grupos menores, tipicamente de 4 a 6 músicos, foram sendo formados. Sem os arranjos elaborados das *big bands*, esses grupos menores abriram espaço para a exploração de elementos de improvisação e interação entre os instrumentistas. Harmonias mais complexas, com substituições e acordes alterados, foram criadas; frases mais elaboradas, com cromatismos e novas escalas, em tempos bastante acelerados, que exigiam uma excelente técnica do instrumento. Thelonious Monk, Charlie Parker e Dizzy Gillespie são alguns dos músicos mais importantes dessa fase de transição. O *bebop* é considerado o momento em que o *jazz* deixou de ser uma simples forma de entretenimento, uma "música para dançar", para se tornar uma forma de arte.

Por ter causado tamanha revolução no mundo do *jazz* e pelo seu mais de meio século de vida, existe uma considerável biblioteca sobre esse estilo, desde estudos históricos, teóricos, até coletâneas de clichês, frases musicais e técnicas que os grandes músicos representantes do *bebop* utilizam em seus solos. Todas essas características tornam o *bebop* uma excelente escolha de estilo a ser implementado.

O *Bebopper* cria um improviso simulando algumas técnicas musicais utilizadas por músicos de *bebop* através de um conjunto de regras. Ele está dividido em 2 partes principais: a criação do ritmo do improviso e a escolha das notas musicais para o ritmo criado, que serão detalhadas nas seções seguintes.

3.3 A Implementação Rítmica

O ritmo é parte importantíssima de qualquer peça musical, talvez até mais importante que a própria escolha dos tons. Essa importância justifica o fato de que o ritmo de cada trecho do improviso seja escolhido pelo algoritmo antes da escolha dos tons. Escolher o ritmo independentemente dos tons torna o sistema mais robusto e flexível em vários aspectos. Operações como deslocamento rítmico, ostinatos, *appoggiaturas* e repetição de parte do ritmo em outros trechos, por exemplo, são facilmente implementadas dessa forma. Além disso, grande parte da comunicação que ocorre entre os músicos é baseada no ritmo. Tratar o ritmo de forma independente facilitará bastante a implementação da comunicação entre os instrumentos do programa em implementações futuras. As seções seguintes detalharão cada componente da implementação rítmica do algoritmo.

3.3.1 Resolução mínima (RM)

Seria muito complexo permitir ao programa a geração de qualquer padrão rítmico, em qualquer resolução. A saída usual é limitar a resolução mínima (RM), ou seja, a nota mais curta que pode surgir em um padrão rítmico. Esse processo é conhecido como *quantização*. O algoritmo implementado aceita qualquer resolução mínima, e a escolha é feita pelo usuário. Para o *bebop*, a resolução mais indicada é a de colcheia. Para ritmos brasileiros e latinos, usualmente a semicolcheia. A nota de RM será utilizada para a criação da célula rítmica básica, como veremos. Em casos especiais, como *appoggiaturas*, poderão ser criados padrões rítmicos com notas de metade da RM.

3.3.2 Notas do Acorde (NA) e Notas de Aproximação (NP)

O algoritmo de escolha do ritmo, além de definir a quantidade de notas e a sua distribuição rítmica no trecho atual, define também o tipo de cada nota. Existem 2 tipos possíveis, *notas do acorde* (NA)

e *notas de aproximação* (NP).

No estilo *bebop*, as notas pertencentes ao acorde sendo tocado são notas importantes que devem ser enfatizadas, ao mesmo tempo que se incluem outras notas entre elas, para tornar o solo mais complexo. Essa idéia é um ponto fundamental deste algoritmo. O improviso será construído como uma sequência de notas pertencentes ao acorde, colocadas em pontos fundamentais. Essas notas são denominadas *notas do acorde* (NA). Entre as NA's serão inseridas notas de ligação, para que se crie um caminho e seja construídas frases musicais. Essas notas de ligação serão denominadas *notas de aproximação* (NP), que podem também pertencer ao acorde sendo tocado.

Portanto, ao definir o ritmo de um trecho, o algoritmo inclui as NA's e depois decide quantas NP's deve colocar entre as NA's, e onde. A forma mais óbvia de escolher a posição das NA's de modo a enfatizá-las é colocando-as nos tempos fortes do compasso. Por exemplo, usando como RM a colcheia, em um compasso 4/4, os tempos fortes são o primeiro e o terceiro. Os outros espaços são preenchidos com NP's, como mostra a figura 3.1.

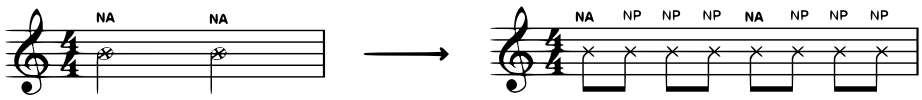

Figura 3.1: NA's nos tempos fortes, e os espaços restantes preenchidos com NP's

As NA's serão ligeiramente acentuadas em relação às NP's, durante a execução do improviso, para destacá-las. Esse acento é implementado através do parâmetro *velocity* do padrão MIDI. O padrão rítmico à direita, na figura 3.1, é a *célula rítmica básica*, e sobre ela podem ser aplicadas uma série de variações, como omissões e deslocamentos, para que se construam outros padrões rítmicos. Nas seções seguintes veremos as variações possíveis.

3.3.3 Omissões

A variação mais simples possível sobre a célula rítmica básica apresentada na seção anterior é a omissão, que significa omitir uma ou mais NP's. No estágio atual do *Bebopper*, as NA's não podem ser omitidas, já que todo o algoritmo de criação está fortemente baseado nelas, como veremos adiante. Portanto, a partir de uma célula básica podem ser obtidas $2^6 = 64$ variações rítmicas. Uma delas está indicada na figura 3.2.

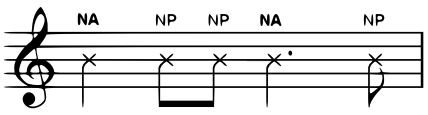

Figura 3.2: Um exemplo de variação por omissão

3.3.4 Antecipação da NA

Outra variação que pode ser aplicada sobre a célula básica é a antecipação da NA. Este é um recurso bastante utilizado por músicos durante o improviso, antecipando uma resolução melódica ou harmônica para o contratempo anterior. No programa, isto significa adiantar uma ou mais NA's em uma colcheia (no caso de RM de colcheia), como mostra o exemplo da figura 3.3, onde a NA do 3º tempo do compasso atual e a NA do 1º tempo do compasso seguinte estão antecipadas.

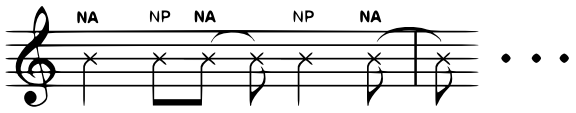

Figura 3.3: Compasso criado com antecipação de NA's e algumas omissões

3.3.5 Variações nas posições das NA's

Os padrões rítmicos expostos até o momento prendem as NA's nos tempos fortes ou, no caso de antecipações, no tempo imediatamente anterior a um tempo forte. O programa também permite que se criem células sem essas restrições. Podemos distribuir as NA's de modo a enfatizar algum ritmo desejado, como na figura 3 4, e aplicar omissões e/ou outras variações para alterar o ritmo das NP's. Podemos também deixar as NA's totalmente livres, uniformemente distribuídas ao longo do tempo, sem posição fixa. Com isso ganha-se em flexibilidade, mas é possível que o improviso fique menos estruturado. De qualquer forma, as escolhas que o algoritmo de geração do ritmo faz são baseadas em um modelo estocástico que pode ter os seus parâmetros ajustados por meio de treinamento supervisionado. Portanto, o gosto musical do usuário que treinar o programa vai decidir quais os padrões rítmicos que serão utilizados mais frequentemente.

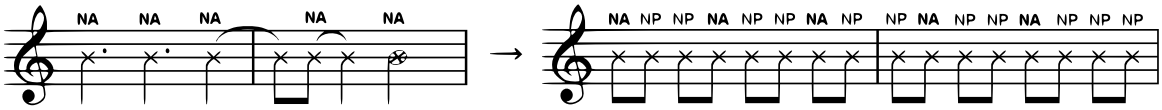

Figura 3.4: NA's num ostinato de semínimas pontuadas criando uma nova célula básica

3.3.6 Articulação das frases e Duração das Notas

As frases em *bebop* são tocadas geralmente em *legato*. Por isso, a duração básica das notas é sempre exatamente até o início da próxima nota, causando o efeito de articulação em legato. Para que uma nota não fique muito longa, existe um limite superior da duração. O limite pode ser variado pelo usuário, e foi utilizado nos testes o limite de 4 vezes a duração da RM. Outro fator importante da articulação característica de frases de *bebop* é o uso de síncope através de acentuações em notas no contratempo. Essa acentuação ocorre na maioria dos casos após um intervalo ascendente, e em notas que tem uma duração um pouco maior do que as outras notas da frases. Ao se decidir o ritmo de um

trecho, o algoritmo decide também quais serão as notas acentuadas. Essas notas terão um acento maior através do parâmetro MIDI *velocity*, e serão tocadas em *staccato*. As outras notas terão articulação em *legato*, como foi dito, com duração máxima de 4 RM's. A figura 3.5 mostra um exemplo de uma frase gerada, com a articulação indicada.

Figura 3.5: Exemplo de uma articulação possível, com *legato* e *staccato acentuado* (*marcato*)

3.3.7 *Appoggiaturas*

Appoggiaturas são notas de passagem que precedem tons essenciais, com função de embelezamento. São utilizadas praticamente em qualquer estilo musical, e certamente muito utilizadas em *bebop*.

As Appoggiaturas foram implementadas como notas com duração de metade da RM, que se situam entre duas notas consecutivas com duração de 1 RM, como mostra a figura 3.6, onde a appoggiatura está indicada. Outras formas de appoggiaturas estão sendo estudadas para futura implementação.

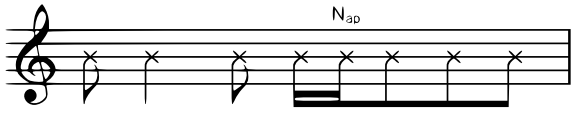

Figura 3.6: Um exemplo de *appoggiatura*

As notas de *appoggiatura* recebem o rótulo N_{ap} para que seja possível identificá-las durante a fase seguinte, de preenchimento dos tons musicais. As técnicas para o preenchimento das appoggiaturas serão apresentadas posteriormente.

3.3.8 Velocidade

Todas as variações apresentadas até aqui utilizam ritmos que tem como base a RM, no caso dos exemplos apresentados, a colcheia. Essa forma de montar o ritmo teve como motivação o fato de que os improvisos em *bebop* são em grande parte constituídos por frases formadas basicamente de séries de colcheias, com algumas variações rítmicas. Entretanto, o algoritmo também permite que sejam criadas frases que têm como base notas diferentes da RM. Para isso, foi criado um parâmetro multiplicador sp chamado de velocidade, que pode assumir como valor qualquer potência de 2. Ao multiplicar a duração das notas de uma célula básica de ritmo por esse parâmetro, estaremos gerando uma nova célula básica, com uma velocidade diferente, sobre a qual podemos aplicar variações da mesma forma que fizemos até agora. Os valores típicos de sp são 1, para o ritmo original, 2 ou 4, para ritmos mais lentos, e até ½, para trechos rápidos. Quando $sp > 1$, além do operador de omissão,

pode-se aplicar também a antecipação nas NP's. Com isso, uma grande variedade de padrões rítmicos pode ser construída. A figura 3.7 ilustra esse processo: (a) célula básica com $sp = 1$; (b) mudando para $sp = 2$; (c) Exemplo de padrão rítmico sobre a célula básica com $sp = 2$, aplicando antecipações e omissões nas NA's e NP's. Para os testes foi limitado o valor de sp em ½ , 1 ou 2.

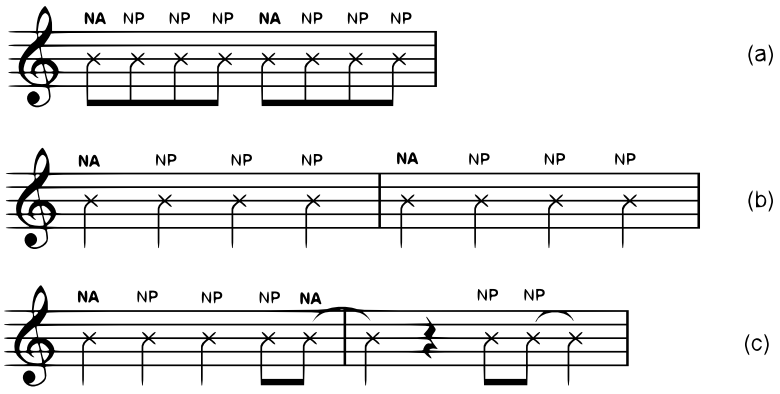

Figura 3.7: Mudança de velocidade (sp)

3.4 A Implementação Melódica

Após a escolha do ritmo, o programa deve escolher os tons musicais para preenchê-lo. Para tanto, foram implementadas regras que simulam várias técnicas utilizadas por músicos de *bebop* em seus solos. As técnicas foram inferidas de transcrições de solos como os de Charlie Parker [120] e métodos de improvisação [11, 38], bem como da minha experiência musical. As seções seguintes explicam os elementos que formam a base para o uso dessas técnicas. A lista de todas as técnicas melódicas implementadas será apresentada no apêndice B.

3.4.1 Harmonia

Os tons musicais serão escolhidos de acordo com a harmonia do trecho. Para isso será utilizada uma abordagem tradicional na improvisação, a relação *escala-acorde*: cada acorde possui uma ou mais escalas apropriadas para a improvisação. A tabela 3.4.1 mostra os tipos de acordes permitidos pelo algoritmo. Praticamente todos os tipos de acordes mais utilizados em harmonia de *jazz* podem ser substituídos por um dos acordes incluídos.

Para cada um dos 10 tipos de acorde, existem várias escalas possíveis. O *Bebopper* utiliza apenas 1 escala por acorde durante a montagem do improviso, portanto é necessário escolher a relação escala-acorde antes da execução. Em futuras implementações o algoritmo será capaz de utilizar mais de uma escala por acorde, mudando de escala quando for apropriado, em tempo real. A seção seguinte trata das escalas utilizadas pelo algoritmo, e da introdução de novas escalas sintéticas.

Acordes Codificados	Outros acordes possíveis	Escala Correspondente
$C^{7M(9)}$	C, C^{6}, C^{7M}	Maior
$C^{7M(9)(\#11)}$	$C^{7M(\#11)}$, $C^{(\#11)}$	Lídio
$Cm^{7(9)}$	Cm, Cm^{7}, Cm^{11}, Cm^{6}	Dórico
$Cm^{7(b5)}$	$Cm^{7(b5)(9)}$	Lócrio 9M
$C^{7(9)}$	C^{7}, $C^{7(9)(13)}$	Mixolídio
$C^{7(9)(\#11)(13)}$	$C^{7}(\#11)$, $C^{7(\#11)(13)}$	Mixolídio $\#11$
$C^{7(b9)(13)}$	$C^{7(\#9)(13)}$, $C^{7(b9)(\#11)(13)}$, $C^{7(\#9)(\#11)(13)}$	Dominante Diminuta
$C^{7(b9)(b13)}$	$C^{7(b13)}$, $C^{7(b9)}$	Mixolídio $b11$ $b13$
C^{alt7}	C^{7} com as tensões $\#5, b5, \#9, b9$	Alterada
C^{o}	$C^{o(13)}$, $C^{o(7M)}$	Diminuta

Tabela 3.1: Acordes codificados no *Bebopper* e sua relação com outros acordes possíveis

3.4.2 Escalas *bebop*

Uma das novidades introduzidas no *bebop* é o uso das chamadas *escalas bebop*. As escalas tradicionais, como as escalas maiores, menores e modos gregos, são escalas de 7 tons. As escalas *bebop* são obtidas através da introdução de uma aproximação cromática entre dois tons de uma dessas escalas de 7 notas, tornando-a uma escala de 8 notas. Essa nota adicional é escolhida de forma que, ao tocar uma escala *bebop*, descendente ou ascendentemente, sobre um acorde apropriado, haverá uma alternância entre notas do acorde (NA's) e notas fora do acorde (NP's). Com isso, ao tocar a escala, se começarmos em uma NA no tempo(ou uma NP no contra-tempo), sempre continuaremos com uma NA no tempo e uma NP no contra-tempo. Dessa forma serão enfatizadas as notas do acorde, mas teremos uma passagem com mais notas do que apenas as notas do acorde, inclusive com uma nota fora da escala "tradicional", a aproximação cromática. A criação de frases mais complexas que, por sua vez, exigem técnica mais apurada, é uma das grandes diferenças do *bebop* em relação aos estilos anteriores de improvisação.

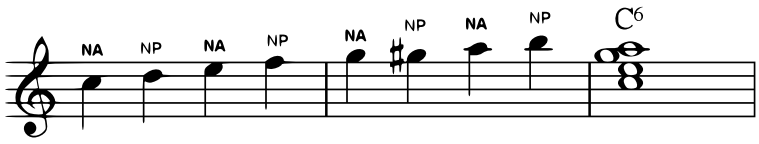

Figura 3.8: A escala de Dó maior *bebop*, obtida com a adição do G# na escala de Dó maior

Na literatura sobre o bebop é comum encontrar 3 escalas *bebop*: a maior *bebop*, a menor *bebop* e a dominante *bebop*, utilizadas em acordes maiores, menores ou dominantes, respectivamente. A figura 3.8 mostra a escala de Dó maior bebop, construída com a adição da quinta aumentada ($G\#$) na

escala de Dó Maior. A escala menor bebop é obtida incluindo-se a sétima maior no modo dórico, e a escala dominante bebop incluindo-se também a sétima maior, mas no modo mixolídio.

A idéia de alternar tons do acorde e tons fora do acorde é muito boa. Outras escalas sintéticas que satisfaçam tal restrição poderiam ser criadas. Mas, até o limite do meu conhecimento, não se encontram livros na área de improvisação que apresentem escalas que satisfaçam essa restrição, além das 3 escalas *bebop* comuns. Essas 3 escalas bebop enfatizam os tons básicos do acorde: a tônica, a terça, a quinta e a sétima (ou sexta, no caso da maior bebop). E se quisermos enfatizar as tensões do acorde, como a nona, décima primeira e décima terceira? Charlie Parker, considerado um dos melhores saxofonistas da era *bebop*, fazia isso em seus improvisos [38]. Podemos pensar em construir escalas que satisfaçam as restrições das escalas *bebop* (alternância de tons do acorde e fora do acorde) enfatizando outras tensões do acorde. De fato, isso é possível, e neste estudo são introduzidas várias novas escalas bebop, para diversos tipos diferentes de acordes, e em alguns casos mais de uma escala para o mesmo acorde, enfatizando tons diferentes. A figura 3.9 mostra uma nova escala de Dó maior bebop, que com a adição do D# enfatiza a terça, quinta, sétima e nona do acorde de Dó maior. Sobre essas escalas é que será construído o improviso. No apêndice A estão listadas todas as escalas utilizadas pelo *Bebopper*.

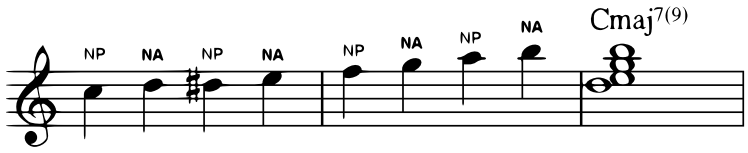

Figura 3.9: Uma nova escala de Dó maior bebop, com a adição do D#

3.4.3 Aproximação das notas-alvo

Uma forma de criar caminhos entre as NA's usando NP's é considerar as NA's como *notas-alvo* das NP's, ou seja, as NP's serão tons cromáticos ou diatônicos, imediatamente acima ou abaixo da NA que será a nota-alvo. Essa técnica é comumente chamada de *targeting*, ou aproximação. Uma aproximação comum é a aproximação cromática, por exemplo, onde a nota-alvo é aproximada por um tom cromático adjacente. A figura 3.10 mostra alguns exemplos desse tipo de aproximação para um acorde de Dó Maior.

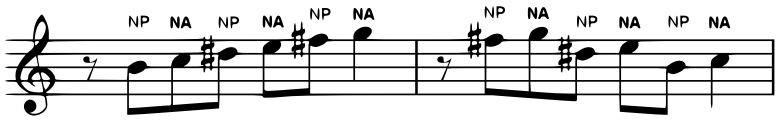

Figura 3.10: Aproximações cromáticas para NA's do acorde de Dó Maior

Outra aproximação bastante comum é o que comumente se chama de cercamento, ou *enclosure*[38], onde a nota-alvo é cercada por duas NPs, uma delas em um intervalo cromático ou diatônico imediatamente acima e outra abaixo. A figura 3.11 mostra exemplos desse tipo de aproximação.

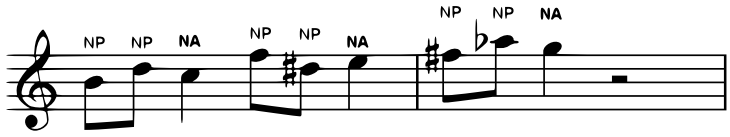

Figura 3.11: Cercamento em notas do acorde de Dó maior, notas cromáticas e diatônicas

De forma semelhante, aproximações de 3 notas, combinando notas diatônicas, cromáticas e até outras NA's adjacentes podem ser implementadas. No *Bebopper* serão utilizadas aproximações de 1, 2 ou 3 notas.

3.4.4 Arpejos

Arpejo é uma sequência, geralmente monotônica, de notas de um acorde. O arpejo é uma técnica bastante utilizada em vários estilos e também no bebop. A característica principal do arpejo utilizado em um improviso bebop é que é que suas notas são geralmente formadas pelas tensões do acorde(7ª, 9ª, 11ª, e 13ª), ao invés das notas básicas do acorde (tônica, 3ª e 5ª). As novas escalas introduzidas na seção 3.4.2 garantem que tais tensões sejam utilizadas.

Ao se utilizar um arpejo, as NP's se tornarão NA's, pois serão notas do acorde sendo utilizado. Na figura 3 12, temos um exemplo de um arpejo utilizando a nova escala *bebop* da figura 3.9, enfatizando a 3ª, 5ª, 7ª, e 9ª de Dó.

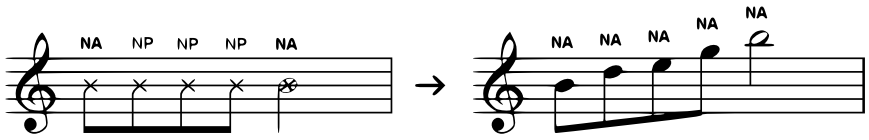

Figura 3.12: Um arpejo em Dó maior usando a 3ª, 5ª, 7ª, e 9ª

Serão utilizadas algumas variações sobre o tipo básico de arpejo da figura 3.12, como mudanças na direção do arpejo e saltos de mais do que 1 NA. O único cuidado a ser tomado com arpejos é não exagerar o seu uso, o que resulta em um solo mecânico, semelhante a um exercício de estudo. Como o algoritmo pode ser treinado através treinamento supervisionado, como veremos adiante, o usuário será capaz de evitar o uso excessivo de arpejos.

3.4.5 Clichês de Bebop

Técnicas usadas repetidamente por alguns músicos, que se transformaram em assinaturas do estilo ou de um determinado músico. Foram implementados alguns clichês utilizados por músicos como Miles Davis e John Coltrane, bem como clichês que se tornaram de uso comum em jazz. A figura 3.13 ilustra um exemplo de clichê utilizado no algoritmo, que foi utilizado bastante por Miles Davis e continua sendo utilizado comumente até hoje em dia. A idéia básica é começar a frase numa NA no tempo forte e percorrer descendentemente os tons da escala, intercalando com notas em intervalo de 3ª menor descendente em relação à nota anterior. Com isso algumas notas estranhas à escala surgem na

frase, mas devido à repetição do intervalo descendente e as notas da escala nos tempos fortes, a frase é agradavelmente dissonante. Além disso, ela propriedade das escalas bebop, a frase terminará também em uma NA no tempo forte. Outros intervalos além da 3ª menor são utilizados, como 2ª maior, 3ª maior e 4ª justa.

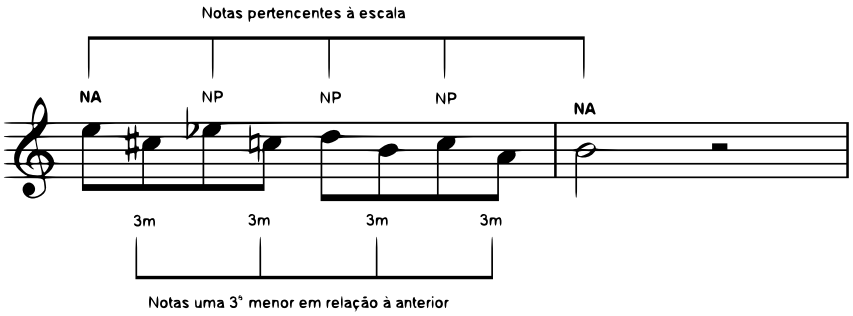

Figura 3.13: Um clichê utilizado por Miles Davis

3.4.6 *Appoggiaturas*

Como vimos, ao escolher o padrão rítmico utilizado, o algoritmo também escolhe onde serão adicionadas *appoggiaturas*, sempre entre duas notas consecutivas de duração igual à RM (ver figura 3.6). Para designar uma nota para a *appoggiatura*(N_{ap}), o algoritmo em um primeiro momento a ignora, preenchendo as outras notas como se a *appoggiatura* não existisse. Ela será designada posteriormente, em função das duas notas entre as quais se encontra. A regra para se encontrar a N_{ap} é a seguinte:

- Se a distância entre as duas notas é de 1 tom, a N_{ap} é a nota entre elas;
- Caso contrário, a N_{ap} é a nota cromática abaixo da 2ª nota.

A *appoggiatura* com tons cromáticos é bastante comum em *bebop*, justificando a escolha da regra acima. A figura 3.14 mostra alguns exemplos de *appoggiaturas* que podem surgir durante o improviso.

3.4.7 Desenvolvimento temático

Criar motivos e frases que são variações de motivos já utilizados é uma técnica utilizada em praticamente qualquer estilo musical, e no *bebop* não é diferente. O algoritmo, durante a geração do improviso, mantêm uma memória com todas as técnicas utilizadas, na ordem em que foram executadas. Em um dado momento da geração do improviso, o algoritmo pode se utilizar novamente de uma mesma sequência de técnicas recentemente utilizadas, uma ou até mais vezes, na mesma ordem da primeira vez, gerando desenvolvimento temático. O interessante nesta implementação é que serão repetidas as mesmas técnicas musicais, e não as mesmas notas; as notas serão outras, pois a harmonia do trecho e a nota sendo tocada no momento da repetição são outras, gerando notas diferentes, mas fortemente relacionadas com as primeiras. Há ainda a possibilidade de alteração rítmica nas repetições. Ou seja, não haverá apenas a repetição pura e simples das notas, mas uma verdadeira variação de um tema anteriormente executado. Na figura 3.15 vemos como isso pode ocorrer.

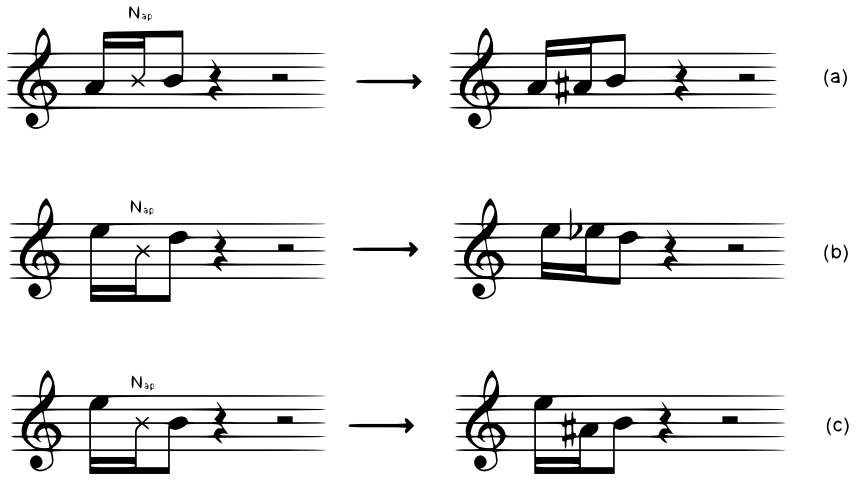

Figura 3.14: Appoggiaturas. (a) e (b) 1 tom entre as notas (c) Distância diferente de 1 tom

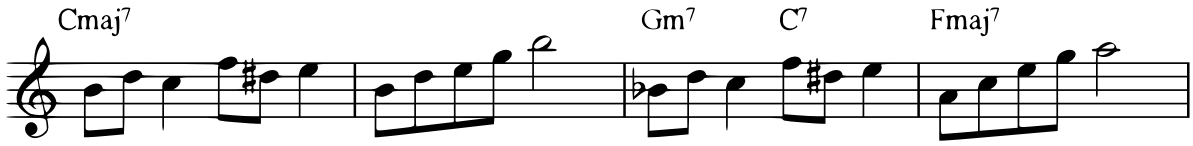

Figura 3.15: Um exemplo de um possível desenvolvimento temático

Os dois primeiros compassos foram montados utilizando as técnicas que foram apresentadas nas seções anteriores. No primeiro compasso, um cercamento utilizando notas diatônicas e também uma nota cromática, sobre as NA's do acorde de C^{7M}.O segundo compasso foi montado com um arpejo sobre esse mesmo acorde. Os dois últimos compassos são um desenvolvimento temático dos dois primeiros, utilizando as mesmas técnicas, mas sobre uma harmonia diferente. No 3º compasso, o mesmo cercamento gera uma nota diferente, o si bemol ao invés do si. No 4º compasso, o arpejo é composto de notas completamente diferentes do arpejo do 2º compasso, pois é montado sobre outro acorde, F^{7M} ao invés de C^{7M}. Apesar de gerar notas diferentes, é fácil perceber que a frase do 3º e 4º compassos é um desenvolvimento da 1ª frase, causando um interesse maior ao ouvinte.

3.5 Forma de utilização das técnicas apresentadas

Como vimos, no momento da escolha das notas musicais, já foi definido o padrão rítmico do trecho em questão e cada nota foi rotulada como NA ou NP. O algoritmo escolherá qual a técnica que preencherá as NA's e NP's com notas musicais. Cada técnica é responsável por designar notas musicais a todas as NP's desde o instante atual até a próxima NA, e esta inclusive. A próxima técnica escolhida designará as NP's dessa NA até a próxima, e assim por diante até o final do trecho atual. A figura 3.16 ilustra esse processo: (a) A última nota designada é sempre uma NA; (b) A técnica escolhida preenche todas as NP's até a próxima NA, inclusive ; (c) A técnica seguinte preenche até a próxima NA e termina o trecho atual.

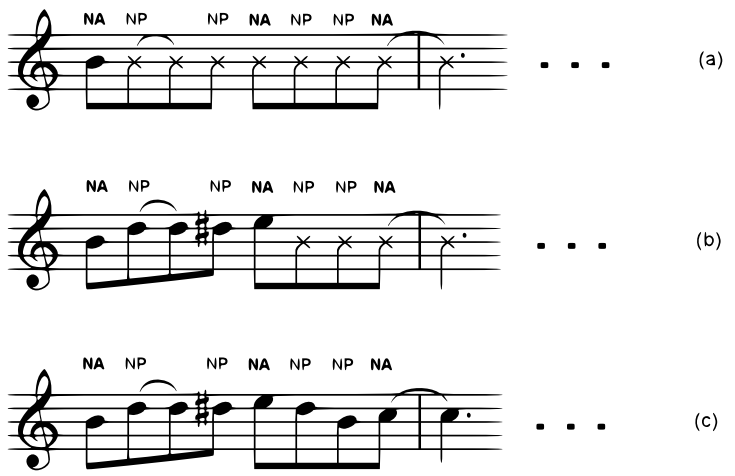

Figura 3.16: Designando notas musicais ao padrão rítmico do trecho atual

Portanto, em uma iteração, uma técnica é responsável por designar 1 NA e tantas NP's quantas estiverem entre essa NA e a escolhida anteriormente, tipicamente entre 1 ou 3, ou até mesmo nenhuma, em casos especiais.

3.5.1 Classificação das técnicas de escolha das notas

Como vimos, no momento da aplicação de uma técnica de escolha das notas, a última nota escolhida (pela última técnica) é sempre uma NA. Essa NA será denominada *NA atual*. A próxima técnica escolhida será responsável por preencher todas as NP's entre a NA atual e a próxima NA, esta inclusive, que será chamada de *NA seguinte*.

As técnicas são classificadas de acordo com o momento da escolha da NA seguinte. Se a NA seguinte é designada **antes** das NP's, é uma *técnica de chegada*; se a NA seguinte é designada **depois**, é uma *técnida de partida*. As técnicas de partida têm esse nome porque a escolha das NP's parte da NA atual, e a NA seguinte será escolhida após as NP's. Já nas técnicas de chegada, primeiro é escolhida a NA seguinte, em função da NA atual, e as NP's são escolhidas de forma a chegar na NA. A figura 3.17 e a figura 3.18 mostram o processo de cada técnica. Na figura 3.17, para técnicas de chegada, temos: (a) A NA atual (b) Primeiro se escolhe a NA seguinte, depois (c) as NP's. Na figura 3.18, para técnicas de partida: (a) A NA atual (b) Primeiro se escolhem as NP's, depois (c) a NA seguinte.

3.5.2 Técnicas de partida

As técnicas de partida usam a NA atual como ponto de partida para a escolha das NP's, e a escolha da NA é feita após a escolha das NP's, e em função destas. Os principais elementos constitutivos desta categoria de técnicas são as escalas (seção 3.4.2) e os arpejos (seção3.4.4). Por exemplo, se for escolhida uma determinada escala para ser utilizada, a NA atual será a nota de partida, e as NP's e NA seguinte serão as notas adjacentes dessa escala, ascendente ou descendentemente, como mostra a figura 3.19.

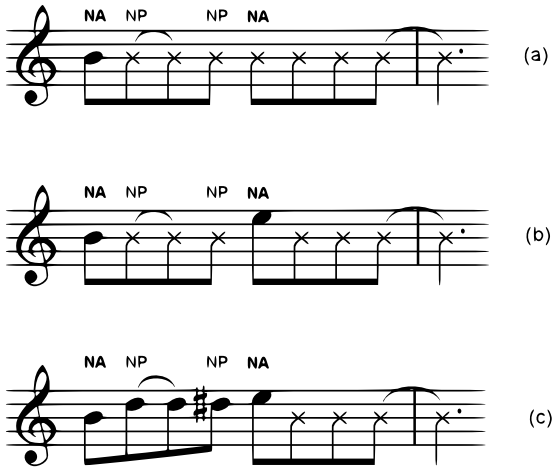

Figura 3.17: Um exemplo de uma técnica de chegada

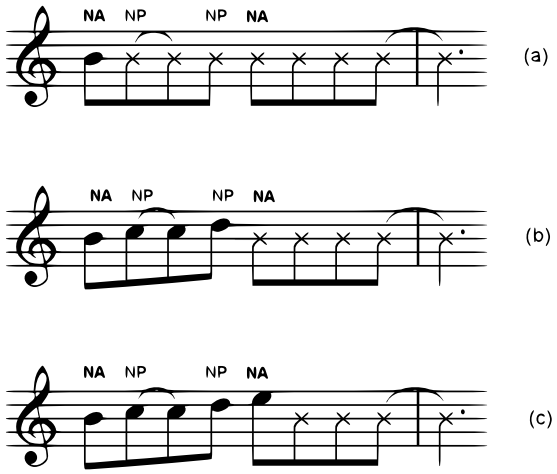

Figura 3.18: Um exemplo de uma técnica de partida

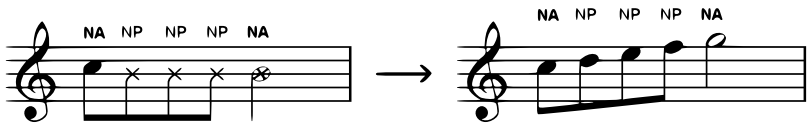

Figura 3.19: Aplicação de uma escala de Dó Maior bebop ascendentemente

3.5.3 Técnicas de chegada

As técnicas de chegada primeiro escolhem a NA seguinte, em função da NA atual, para depois escolher as NP's, em função dessa NA seguinte. A aproximação das notas-alvo (seção 3.4.3), por exemplo, pertence à categoria de técnicas de chegada. Na figura 3.20 abaixo, uma aproximação de 2 notas, uma nota cromática abaixo e uma nota diatônica acima, é aplicada. Primeiro é escolhida a NA seguinte, para que em seguida seja possível escolher as notas de aproximação.

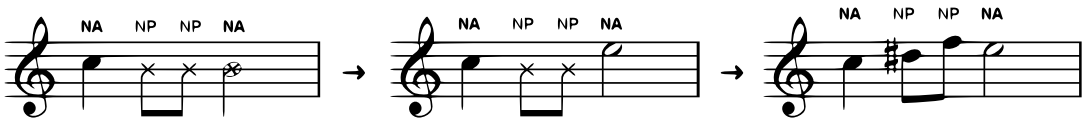

Figura 3.20: Aplicação de uma aproximação de 2 notas

3.5.4 A escolha da NA seguinte

Nas técnicas de partida, a NA seguinte é designada diretamente pela técnica escolhida. No caso das técnicas de chegada, a NA seguinte deve ser designada antes da aplicação da técnica, pois esta depende da NA seguinte. Para isso, foi desenvolvido um pequeno algoritmo que determina a NA seguinte em função da NA atual.

Em *bebop*, as frases usualmente contêm poucos saltos, ou seja, os intervalos entre as notas que formam as frases são pequenos, em sua maioria segundas e terças. Os intervalos grandes são pouco comuns. Por isso, a NA seguinte poderá ser apenas a primeira NA acima ou abaixo da NA atual, salvo um caso especial de deslocamento de oitava, que veremos a seguir. Por exemplo, se estivermos usando a escala de Dó Maior bebop (Figura 3.8), e a NA atual for a nota mi, a NA seguinte poderá ser ou uma NA abaixo, a nota dó, ou uma NA acima, a nota sol, como mostra a figura 3.21.

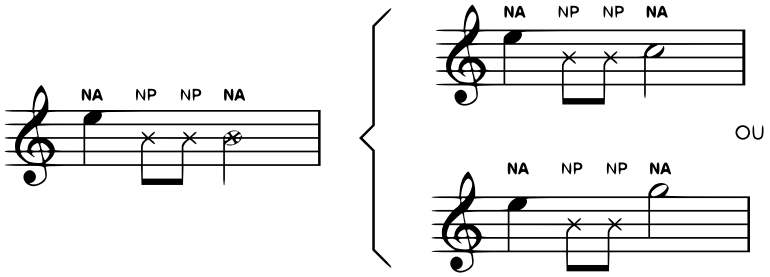

Figura 3.21: Escolha da NA seguinte: uma NA acima ou abaixo da NA anterior

Para escolher que direção tomar para a escolha da NA seguinte, foi criada uma função que, tendo como entrada a NA atual, retorna a probabilidade de que a NA seguinte seja escolhida acima da NA atual. Essa função foi denominada *função de direção* e está desenhada na figura 3.22. A intenção dessa função é fazer com que se tenha maior probabilidade de escolher a NA seguinte na direção da região central, para evitar que o improviso permaneça muito tempo em uma região muito grave ou muito aguda.

Os parâmetros a, d, N_{MIN} e N_{MAX} regulam o comportamento da função de direção. N_{MIN} e N_{MAX} são, respectivamente, a nota mínima e nota máxima permitida pelo algoritmo, usando o padrão MIDI. O parâmetro d define o tamanho da região próxima às extremidades onde a probabilidade é fixa, e o valor dessa probabilidade fixa é definida por a. Com essa gama de parâmetros é possível moldar a função de direção de varias maneiras. Desde uma função totalmente uniforme, até uma função que force que as NA's se concentrem na região central entre N_{MIN} e N_{MAX}, ou qualquer função intermédiaria entre essas duas.

Por exemplo, em um caso extremo, com $a = 0.5$ e $d = 0$, a função de direção é constante, com

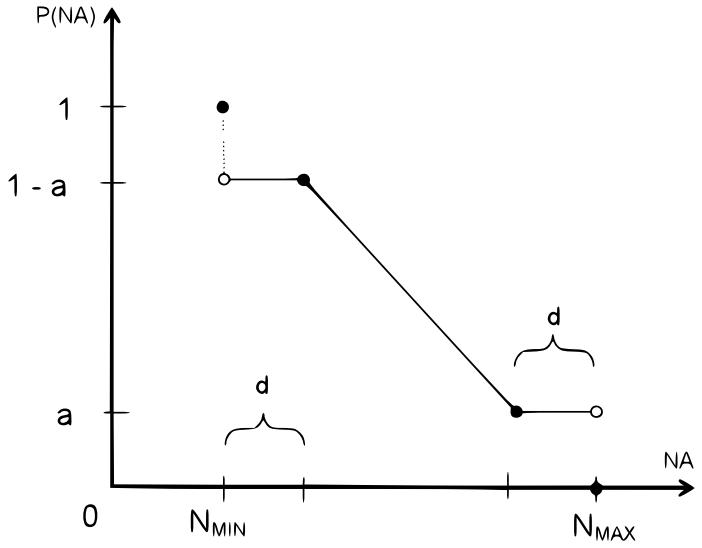

Figura 3.22: Probabilidade de escolher a NA seguinte acima da NA atual

valor igual a 0.5. Isso significa uma distribuição de probabilidade uniforme, ou seja, independentemente da posição da NA atual, teremos a mesma probabilidade de escolher a próxima NA acima ou abaixo da atual, resultando em um solo com alta probabilidade de utilizar bastante notas muito agudas e muito graves, podendo permanecer bastante tempo nas extremidades, além de ter grande chance de mudar rapidamente de uma extremidade para a outra rapidamente. Isso pode gerar um solo com maior variedade de notas, mas por outro lado, ficar por muito tempo em uma extremidade também pode tornar o solo desinteressante.

No outro caso extremo, com $a = 0$ e $d = (N_{MIN} + N_{MAX})/2$, a função de direção escolherá sempre a próxima NA na direção da nota $(N_{MIN} + N_{MAX})/2$, ou seja, na região central. Isso deixará o solo sempre na mesma região, possivelmente tornado-o desinteressante.

A idéia da função de direção é escolher os parâmetros a e d de forma que a função fique em um caso intermediário entre os dois casos extremos, fazendo com que o solo explore ao máximo possível toda a região entre as notas N_{MIN} e N_{MAX}, mas sem permitir que permaneça por muito tempo nas extremidades. O usuário pode modificar esses parâmetros e perceber a mudança causada no estilo do improviso, até chegar a um conjunto de parâmetros que o satisfaça.

A função de direção pode ser definida formalmente pela equação 3.1.

$$f(NA) = \begin{cases} 1, & NA = N_{MIN} \\ 1 - a, & N_{MIN} < NA \leq N_{MIN} + d \\ \frac{2a-1}{N_{MAX}-N_{MIN}} NA + \frac{N_{MAX}-d-a(N_{MAX}+N_{MIN})}{N_{MAX}-N_{MIN}-2d}, & N_{MIN} + d < NA < N_{MAX} - d \\ a, & N_{MAX} - d \leq NA < N_{MAX} \\ 0, & NA = N_{MAX} \end{cases} \quad (3.1)$$

Uma variação que pode ser aplicada ao processo de escolha da NA é o *deslocamento de oitava*, uma técnica frequente nos improvisos de jazz. Após a escolha da NA, se o algoritmo decidir aplicar o deslocamento de oitava, essa NA é deslocada em uma oitava na direção da NA anterior. Logo, a distância entre a NA anterior e a atual, que acabou de ser escolhida, será sempre menor do que uma oitava. A figura 3.23 mostra um exemplo do que pode acontecer, usando a escala de Dó Maior *bebop*.

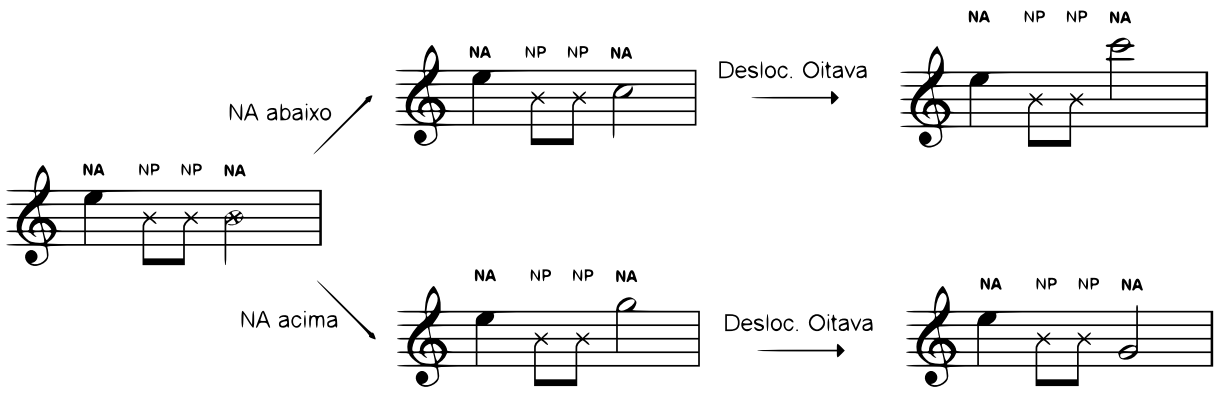

Figura 3.23: Exemplos de deslocamento de oitava

3.6 Introduzindo Estrutura: A Forma Musical

A forma musical é basicamente como se distribuem diferentes seções de uma peça ao longo de sua execução. Por exemplo, em *jazz*, uma forma comum é a AABA, onde temos duas seções, A e B, e a forma de execução é simplesmente tocar a seção A duas vezes, a B uma vez, e mais uma vez a seção A. Essa forma pode se repetir quantas vezes se quiser (ou estiver combinado com a banda), para improvisos, por exemplo, ou para a repetição do tema (melodia) principal.

Foi introduzida no *Bebopper* uma maneira simples de se obter estrutura musical, baseada na forma. O usuário pode definir seções onde serão utilizadas as mesmas técnicas para a geração do improviso. Cada seção pode ter o tamanho de compassos que se queira, podendo ser repetida em qualquer trecho do improviso. Essas seções podem ser definidas em toda a duração do improviso, ou só em algumas partes. Por exemplo, pode-se definir uma seção de duração de 4 compassos, posicionada nos primeiro 4 compassos e também nos últimos quatro, sem definir seção alguma nos compassos intermediários. Dessa forma, será executado um improviso livre, em que a última frase, de quatro compassos, é semelhante à primeira, pois serão utilizadas as mesmas técnicas para a sua criação.

Com isso é possível criar uma peça musical em uma forma pré-determinada, seja ela uma forma clássica do *jazz*, ou qualquer outra que se tenha em mente.

3.7 A Seção Rítmica

A seção rítmica é composta pelos instrumentistas que acompanham o solista durante o seu improviso. No estágio atual, a seção rítmica está implementada de forma bem simples, apenas para servir de acompanhamento ao aspecto principal do *Bebopper*, o improviso. A seção rítmica está composta de bateria, baixo e piano. A bateria é montada através de um padrão simples de jazz, conduzindo no *hihat*, com algumas variações na utilização da caixa através de uma distribuição aleatória simples. O baixo fica alternando a tônica e a quinta, como um baixo de *swing* simples. Uma melhoria que está sendo testada é a utilização de um subconjunto das técnicas utilizadas para a criação do improviso para a criação das linhas de baixo no estilo *walking bass*, bastante utilizado em *jazz*. O piano é único responsável pelo acompanhamento harmônico do solista. São utilizados *voicings* sem a fundamental, em terças, escolhidos e distribuídos ao longo do tempo por uma distribuição aleatória simples.

A seção rítmica tem ainda muito espaço para melhorias, algumas propostas na conclusão deste trabalho.

3.8 O modelo estocástico

Nesta seção será apresentado o modelo estocástico implementado no *Bebopper*. Ele é utilizado pelo algoritmo para escolher qual o padrão rítmico a ser utilizado, quais notas devem ser NP's ou NA's, e posteriormente quais técnicas serão utilizadas para designar notas musicais aos padrões rítmicos. O modelo foi inspirado no raciocínio de um músico durante a execução de seu improviso, do ponto de vista de que, à medida que o improviso progride, um contexo é criado e os parâmetros do modelo devem variar ao longo do tempo tentando simular esse contexto. Uma vantagem do modelo proposto é que pode ser treinado por aprendizado supervisionado, através da avaliação da saída algoritmo pelo usuário, em tempo-real.

3.8.1 Funcionamento

A cada iteração, o algoritmo tem um trecho da peça a ser preenchido pelas técnicas musicais apresentadas. Sempre que o algoritmo tiver de fazer uma escolha entre várias técnicas disponíveis, recorrerá ao modelo estocástico. As escolhas são, basicamente, qual o padrão rítmico a ser escolhido, ou qual a técnica a ser aplicada para a escolha das notas.

Seja $R^{(n)}$ o conjunto dos índices de todas as técnicas possíveis de serem escolhidas em um dado instante $t_n\,, n \in \mathbb{N}$ durante a execução do algoritmo, com

$$t_0 < t_1 < t_2 < \dots$$

Logo, $R^{(n)}$ é função de t_n:

$$\begin{aligned} R^{(0)} &= R(t_0) \\ R^{(1)} &= R(t_1) \\ &\vdots \end{aligned} \tag{3.2}$$

Vamos definir o parâmetro $p_i^{(n)}, i \in R^{(n)}$, como a probabilidade atual da opção i ser escolhida no instante t_n, ou seja

$$\begin{aligned} p_i^{(0)} &= p_i(t_0) \\ p_i^{(1)} &= p_i(t_1) \\ &\vdots \end{aligned} \tag{3.3}$$

No início da execução, temos $p_i^{(0)} = p_i$, onde p_i é o parâmetro que chamaremos de *probabilidade inicial* da técnica i. As probabilidades $p_i^{(n)}$ devem estar normalizadas, ou seja, obedecem à equação

$$\sum_{i \in R^{(n)}} p_i^{(n)} = 1 \tag{3.4}$$

A forma de alterar as probabilidades ao longo do tempo busca simular o pensamento de um músico, onde cada escolha durante um improviso vai influenciar as escolhas seguintes. Para explicar esse processo, primeiro será apresentado um modelo simplificado, de onde se retiram algumas idéias básicas. Em seguida será apresentado o modelo completo implementado.

3.8.2 Modelo Simplificado

Durante um improviso, é comum o músico repetir continuamente um determinado grupo de notas, ou padrão rítmico, com a intenção de criar uma tensão, que mais à frente será resolvida, com a alteração desse padrão ou a tomada de uma direção inesperada. Em outro caso, algumas técnicas devem ser utilizadas esporadicamente, pois apesar de terem um grande efeito, seu uso exagerado pode empobrecer um solo.

Para tentar gerar essas situações, foi implementado um modelo que altera as probabilidades das técnicas ao longo do tempo, como uma forma de simular um contexto musical.

Como vimos, a probabilidade de se escolher uma técnica k no instante atual t_n é $p_k^{(n)}$. Ao ser escolhida uma determinada técnica k, a probabilidade de escolher esta mesma técnica no próximo instante t_{n+1} é alterada por um fator positivo e constante, s_k, denominado *ajuste condicional de probabilidade*. Cada técnica $i \in R^{(n)}$ possui um fator s_i distinto. Poderíamos calcular a probabilidade para o próximo instante, t_{n+1}, de acordo com a equação

$$p_k^{(n+1)} = s_k \cdot p_k^{(n)} \tag{3.5}$$

Se $s_k > 1$, a probabilidade da técnica k para o próximo instante t_{n+1} aumenta. Se $s_k < 1$, diminui. Mas, dessa forma, seria possível que a probabilidade $p_k^{(n+1)}$ se tornasse maior do que 1, o que não é factível. Portanto, para $s_k > 1$, é necessário encontrar outra forma de atualização. Precisa-se encontrar uma função monotônica $f(p, s)$ que satisfaça as condições

$$\begin{aligned} f(p,1) &= p \\ \lim_{s\to\infty} f(p,s) &= 1 \end{aligned} \tag{3.6}$$

Uma função que satisfaz estas condições é

$$f(p,s) = 1 - (1-p)e^{(1-s)} \tag{3.7}$$

Claramente, existem infinitas outras funções, esta foi a escolhida por sua simplicidade. Com isso podemos definir a *função de atualização de probabilidade*:

$$f_{at}(p,s) = \begin{cases} p \cdot s\,, & s \leq 1 \\ 1 - (1-p)e^{(1-s)}\,, & s > 1 \end{cases} \tag{3.8}$$

Logo, quando o algoritmo escolhe a técnica k no instante t_n, a probabilidade da mesma técnica ser escolhida no próximo instante t_{n+1} é definida pela fórmula

$$p_k^{(n+1)} = f_{at}(p_k^{(n)}, s_k) \tag{3.9}$$

O gráfico da figura 3.24 ilustra a função de atualização, com $p_k^{(n+1)}$ em função de s_k, e $p_k^{(n)} = 0.3$. Ao se alterar a probabilidade de uma técnica, as probabilidades de todas as outras devem ser alteradas

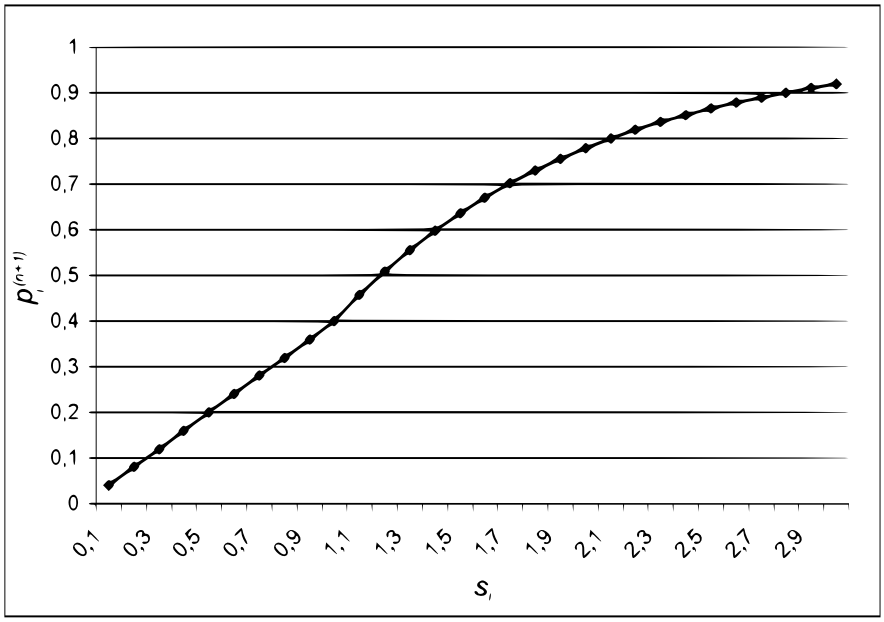

Figura 3.24: Gráfico da função de atualização de probabilidade

para que se mantenha a condição da equação 3.4, a normalização das probabilidades. Poderíamos simplesmente somar novamente todas as probabilidades e dividi-las por esse fator de normalização encontrado. Mas, dessa forma, a variação causada pelo parâmetro s_k seria modificada também, e acabaria sendo 'diluída", especialmente quando é s_k grande e $p_k^{(n)}$ é pequeno, ou s_k é pequeno e $p_k^{(n)}$ é grande. Por exemplo, se temos $p_k^{(n)} = 0.4$ e $s_k = 3$, aplicando a equação 3.9 chega-se a $p_k^{(n+1)} \approx 0.92$, uma probabilidade bastante alta. Ao normalizar pelo processo descrito acima, teríamos de dividir todas as probabilidades por $p_k^{(n+1)} - p_k^{(n)} + 1 = 1.52$, resultando em $p_k^{(n+1)} = 0.61$, uma probabilidade bem menor do que a original. De fato, com esse tipo de normalização é difícil fazer com que qualquer probabilidade se torne bastante elevada, pois sempre há a atenuação causada pela divisão pelo fator de normalização. Logo, chega-se à conclusão que seria melhor que as probabilidades fossem normalizadas sem que se modifique o valor da probabilidade $p_k^{(n+1)}$. Impondo a condição de que as probabilidades sejam alteradas proporcionalmente pelo mesmo fator, chegamos à equação

$$p_i^{(n+1)} = \frac{1 - p_k^{(n+1)}}{1 - p_k^{(n)}} p_i^{(n)} \, , \forall i \neq k \, , i \in R^{(n)} \tag{3.10}$$

O valor $\frac{1-p_k^{(n+1)}}{1-p_k^{(n)}}$ é o *fator de normalização*. Apesar de ser mais caro computacionalmente, é mais fácil de prever o efeito dos parâmetros s_i, além de ser possível causar variações mais drásticas com valores muito elevados ou muito pequenos de s_i.

O conjunto de probabilidades $p_i^{(n+1)} \, , i \in R^{(n+1)}$ continua satisfazendo a condição de normalização das probabilidades (equação 3.4), desde que o conjunto $p_i^{(n)} \, , i \in R^{(n)}$ satisfaça. Sendo k o índice da técnica escolhida no instante n, temos:

$$\begin{aligned}
\sum_{i \in R^{(n)}} p_i^{(n+1)} &= \\
&= \sum_{i \neq k} p_i^{(n+1)} + p_k^{(n+1)} \\
&= \frac{1 - p_k^{(n+1)}}{1 - p_k^{(n)}} \sum_{i \neq k} p_i^{(n)} + p_k^{(n+1)} \\
&= \frac{1 - p_k^{(n+1)}}{1 - p_k^{(n)}} \left(1 - p_k^{(n)}\right) + p_k^{(n+1)} \\
&= 1 - p_k^{(n+1)} + p_k^{(n+1)} \\
&= 1
\end{aligned}$$

Além da atualização das probabilidades que ocorre a cada instante t_n, de acordo com as equações 3.9 e 3.10, existe uma atualização adicional, que ocorre apenas em alguns instantes, mais especificamente ao final de cada compasso do improviso, chamada *atualização de convergência*. Esta atualização tem como objetivo fazer com que as probabilidades atuais retornem gradativamente à sua probabilidade inicial $p_i^{(0)}$. Para isso será introduzido um parâmetro, λ_i, com $0 < \lambda_i < 1$, denominado *velocidade de convergência*. A atualização é dada pela equação 3.11. Ela consiste em um passo na direção da probabilidade inicial, proporcional a λ_i e à distância entre a probabilidade inicial e $p_i^{(0)}$ a probabilidade no instante atual $p_i^{(n)}$.

$$p_i^{(n+1)} = p_i^{(n)} + \lambda_i \left(p_i^{(0)} - p_i^{(n)}\right) \quad \forall i \in R^{(n)} \tag{3.11}$$

Podemos ainda enxergar essa atualização como uma combinação convexa entre a probabilidade inicial, $p_i^{(0)}$, e a probabilidade no instante atual, $p_i^{(n)}$, reescrevendo a equação acima como

$$p_i^{(n+1)} = \lambda_i p_i^{(0)} + (1 - \lambda_i)\, p_i^{(n)} \quad \forall i \in R^{(n)} \tag{3.12}$$

Quanto maior for λ_i, mais rapidamente a probabilidade da técnica i convergirá para a sua probabilidade inicial. Através desse parâmetro é possível calcular por quanto tempo vão atuar as alterações causadas pelos ajustes condicionais s_i.

Para evitar que uma determinada técnica tenha sua probabilidade tão elevada que domine todas as outras, será incluído um último parâmetro, p_i^{MAX}, que é o teto de probabilidade de cada técnica i. Se a probabilidade de uma técnica i, $p_i^{(n)}$, ao ser atualizada pelas equações 3.9 e 3.10 ultrapassar este teto, imediatamente é reduzida. Podemos reduzí-la ao valor de sua probabilidade inicial, $p_i^{(0)}$, ou até a um valor ainda menor, por exemplo $p_i^{(0)}/2$. Após esse ajuste é necessário aplicar a equação 3.10 novamente para garantir a normalização das probabilidades. Podemos interpretar essa redução como resultado da "saturação" do uso dessa técnica, e é necessário ficar um tempo sem utilizá-la. Ou, como na situação apresentada no início da seção, o algoritmo utiliza uma certa técnica repetidas vezes, causando uma tensão, que será resolvida ao se reduzir a probabilidade dessa técnica.

Escolhendo os parâmetros do modelo de forma apropriada, várias situações musicais diferentes podem ser simuladas. Alguns gráficos da probabilidade de uma técnica ao longo do tempo indicam

algumas dessas situações. Na figura 3.25, temos $p_i^{(0)} = 0.1$, $s_i > 1, \forall n$ e $\lambda_i = 0.4$. A técnica foi escolhida quando t = 2, e teve a sua probabilidade aumentada, pois $s_i > 1$. Nos instantes seguintes, a técnica teve uma chance maior de ser escolhida, até que a sua probabilidade fosse retornando gradualmente à probabilidade inicial, $p_i^{(0)} = 0.1$. Na figura 3.26 temos um caso semelhante, mas com $p_i^{(0)} = 0.05$, $s_i \gg 1, \forall n$ e $\lambda_i = 0.8$. Logo, a chance de escolher essa técnica durante o improviso é baixa, mas ao ser escolhida a sua probabilidade aumenta drasticamente. Como $\lambda_i = 0.8$ é um valor alto, a sua probabilidade retorna rapidamente ao valor inicial $p_i^{(0)}$. Essa configuração de parâmetros pode servir para técnicas que devem ser utilizadas poucas vezes durante o improviso, mas, quando utilizadas, têm uma grande chance de serem utilizadas repetidamente, perdendo rapidamente a probabilidade se não forem utilizadas logo. Na figura 3.27 temos a situação do uso repetitivo de uma técnica. Os parâmetros são similares ao caso anterior, com a diferença de que a técnica foi escolhida repetidamente, aumentando cada vez mais a sua probabilidade, até que o seu valor ultrapassou o teto p_i^{MAX} , tendo sido reduzida a probabilidade para $p_i^{(0)}/2$, praticamente forçando o algoritmo a escolher uma outra técnica. Este conjunto de parâmetros serve muito bem para simular o caso onde o músico utiliza repetidamente um determinado padrão, com a intenção de criar uma tensão, para logo depois resolvê-la. Um caso diferente é apresentado na figura 3.28. Agora temos $p_i^{(0)} = 0.8$, $s_i < 1, \forall n$ e $\lambda_i = 0.1$. Como $p_i^{(0)}$ é grande, a técnica será escolhida muitas vezes. Porém, ao ser escolhida, como $s_i < 1$ ela terá a sua probabilidade reduzida, e com a velocidade de convergência muito baixa, $\lambda_i = 0.1$, a probabilidade voltará muito lentamente ao valor inicial. Logo, esse caso simula técnicas que serão utilizadas muitas vezes ao longo do improviso, mas que raramente serão utilizadas muito próximas entre si, evitando o uso exagerado dessa técnica.

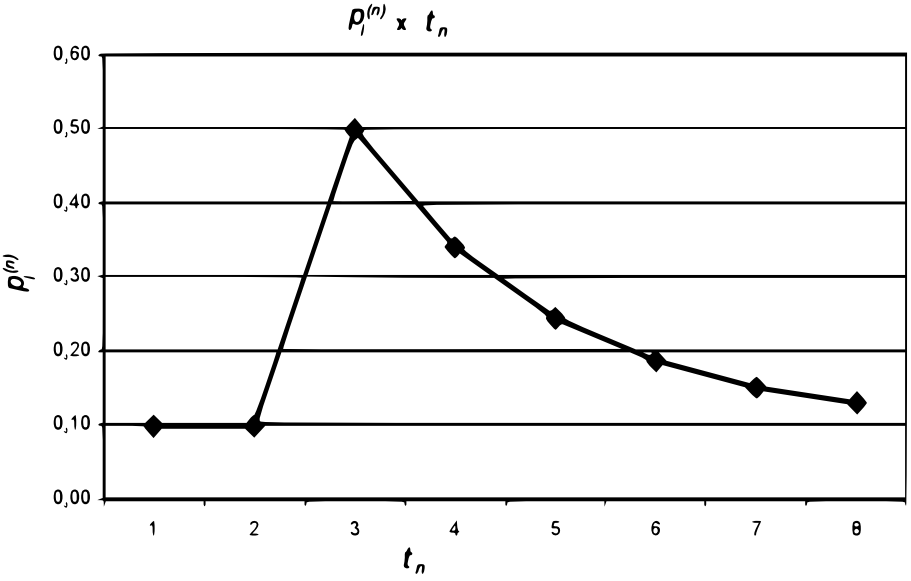

Figura 3.25: Exemplo de um gráfico da probabilidade de uma técnica ao longo do tempo

O modelo simplificado exige que uma determinada técnica seja escolhida para que sua probabilidade seja alterada por um fator s_i. E se, ao se escolher uma técnica qualquer, não só a sua probabilidade como a de outras técnicas for alterada? As situações dos gráficos apresentados acima poderiam ocorrer sem que a técnica em questão tivesse necessariamente que ser escolhida; na situação do gráfico da figura 3.26, uma outra técnica escolhida poderia alterar a probabilidade da técnica do gráfico,

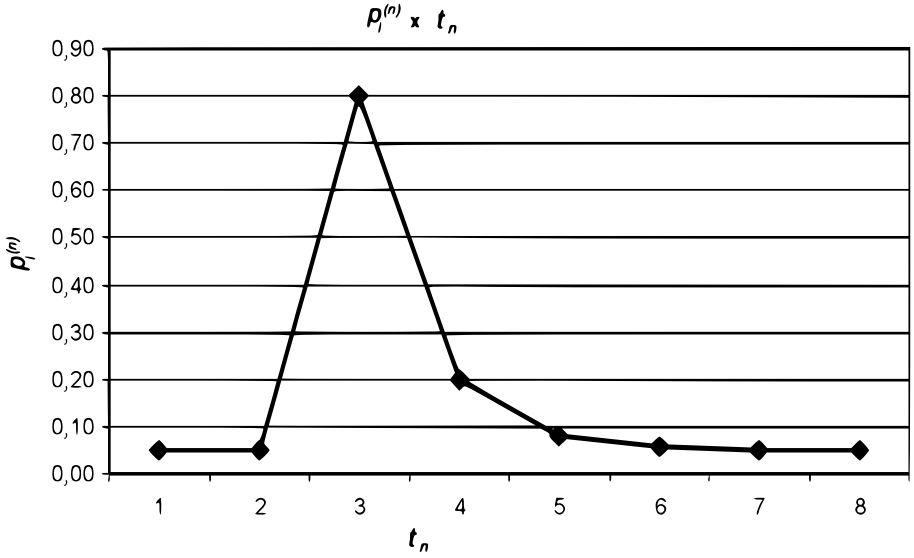

Figura 3.26: Exemplo de um gráfico da probabilidade de uma técnica ao longo do tempo

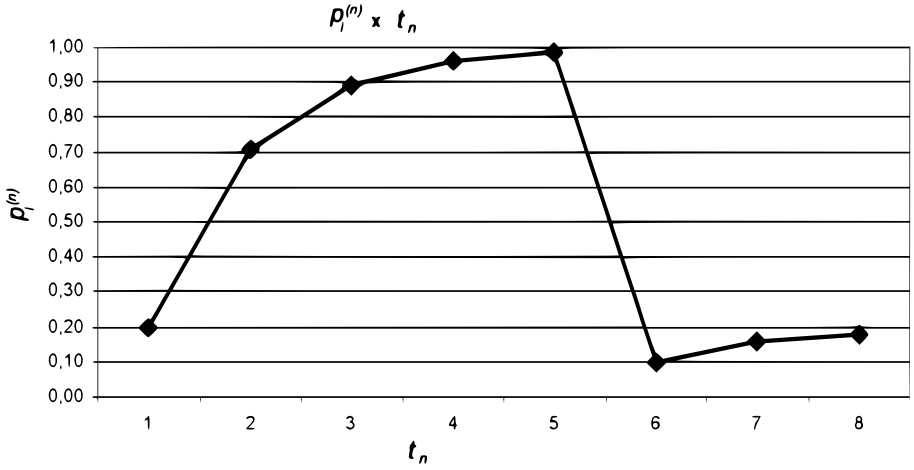

Figura 3.27: Exemplo de um gráfico da probabilidade de uma técnica ao longo do tempo

fazendo com que se tenha uma grande chance de que seja escolhida logo após a primeira, criando um *caminho provável*. Todas as outras situações podem ter generalizações semelhantes. Incluindo essa modificação, onde uma técnica escolhida pode alterar a sua própria probabilidade e a de todas as outras técnicas, temos o modelo estocástico extendido.

3.8.3 Modelo Estendido

O modelo estocástico completo tem apenas uma modificação em relação ao modelo simplificado apresentado anteriormente. No modelo anterior, ao ser escolhida uma técnica k no instante t_n, apenas a probabilidade $p_k^{(n+1)}$ era alterada pelo *ajuste condicional de probabilidade*, s_k, e as probabilidades restantes $p_i^{(n+1)}, i \in R^{(n)}, i \neq k$ eram alteradas proporcionalmente apenas para manter a normalização. No modelo completo, as probabilidades de todos os eventos podem ser alteradas no momento da escolha de uma técnica. Para isso, o parâmetro s_i será estendido em uma matriz $\mathbf{S}_{ac}$, a *matriz de ajuste condicional de probabilidade*, onde os seus elementos s_{ij} indicam o ajuste aplicado na probabilidade

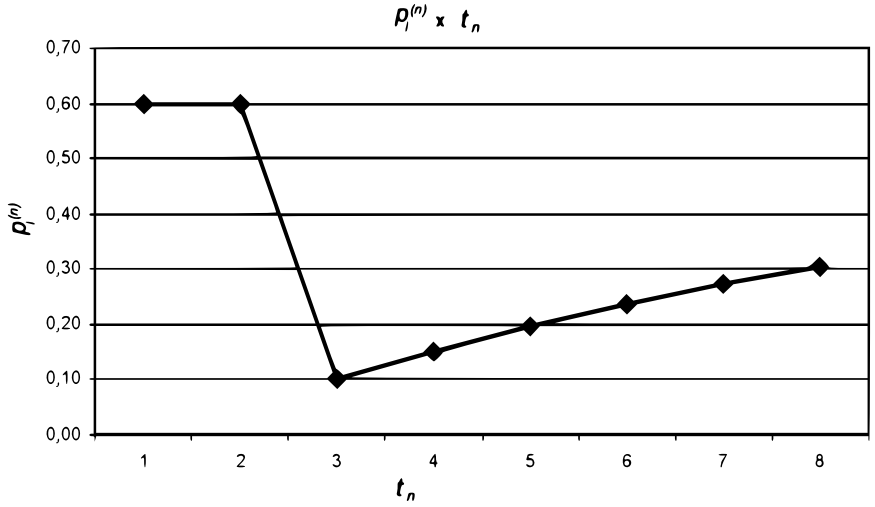

Figura 3.28: Exemplo de um gráfico da probabilidade de uma técnica ao longo do tempo

da técnica j ao ser escolhida a técnica i. A matriz $\mathbf{S}_{ac}$ pode ser definida como

$$\mathbf{S}_{ac} = \begin{pmatrix} s_{11} & s_{12} & \dots & s_{1N} \\ s_{21} & s_{22} & \dots & s_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ s_{N1} & s_{N2} & \dots & s_{NN} \end{pmatrix}$$

onde N é o número de técnicas existentes. Quando, em um dado instante t_n, a técnica k for escolhida, as probabilidades de todas as técnicas $i \in R^{(n)}$ para o instante t_{n+1} serão geradas de acordo com a fórmula

$$p_i^{(n+1)} = f_{at}(p_i^{(n)}, s_{ki}) \quad \forall i \in R^{(n)} \tag{3.13}$$

onde a função $f_{at}(p, n)$ é a *função de atualização de probabilidade*, definida na equação 3.8.

Portanto, utiliza-se a mesma função f_{at} do modelo simplificado, apenas com a diferença que as probabilidades de **todas** as técnicas i pertencentes ao conjunto $R^{(n)}$ são atualizadas no momento da escolha de uma técnica. Como ocorria no modelo anterior, a cada aplicação da equação 3.13 deverá ser aplicada também a equação 3.10 para garantir a normalização das probabilidades. O processo de *atualização de convergência* da equação 3.11 ou 3.12 também ocorre da mesma forma que no modelo simplificado.

Com a matriz $\mathbf{S}_{ac}$ de ajuste se consegue uma possibilidade maior de alterar as probabilidades das técnicas durante a criação do improviso.

Alterar as probabilidades que cada técnica tem de ser escolhida ao longo do tempo faz com que o improviso não seja estático, previsível e sem intenção, como em algumas aplicações que utilizam métodos estocásticos. De acordo com a evolução do improviso, as probabilidades se alteram, fazendo com que as direções que serão tomadas estejam sempre variando, e dependam bastante dos instantes anteriores do improviso. Essa é uma boa forma de tentar simular o pensamento de um músico durante o improviso e a criação de um contexto musical.

Resta ainda a questão de como encontrar esses parâmetros. Seria muito trabalhoso ter que entrar manualmente com todos esses parâmetros, especialmente os valores da s_{ij} da matriz $\mathbf{S}_{ac}$. Por isso, a forma encontrada o treinamento supervisionado durante a execução, com feedback humano em tempo real, descrito na seção seguinte. É importante ressaltar que não deve existir apenas 1 conjunto de parâmetros que produza bons resultados. Cada conjunto de parâmetros pode ser considerado como um estilo diferente, com possivelmente vários desses estilos produzindo resultados agradáveis.

3.9 Treinamento supervisionado

O treinamento foi feito de forma semelhante ao *GenJam* [16]. Durante a execução do solo, o usuário tem a opção de qualificar o trecho tocado a partir de dois botões na tela, "bom" ou "ruim". Pode-se clicar mais de uma vez seguida no mesmo botão, indicando uma nota ainda maior, ou pior. A partir das notas dadas para o trecho, são ajustados os parâmetros dos recursos utilizados nesse trecho. A intenção de treinar o modelo estocástico não é chegar ao melhor músico possível. Ao invés disso, podem ser criados vários conjuntos de parâmetros (músicos), cada qual treinado com um enfoque diferente. Com isso, é possível ter vários músicos que, em futuras implementações, podem ser utilizados em conjunto, cada um em trechos diferentes, para conseguir um melhor resultado.

3.9.1 Atualização dos parâmetros

Ao final da execução do improviso, após o armazenamento de todas as notas do usuário, devemos atualizar os parâmetros do modelo estocástico, refletindo o aprendizado adquirido. Isso pode ser feito das mais diversas formas, e ainda está sendo estudado mais a fundo. É difícil chegar a uma forma de ajuste de alguns parâmetros, como o λ_i , por exemplo. Mas, para os parâmetros s_{ij} , existe uma solução proposta. Se em um determinado trecho foram utilizados, por exemplo, as técnicas a,b,c e d, nessa sequência, então os parâmetros que devem ser ajustados são s_{ab}, s_{bc} e s_{cd}. Se esse trecho recebeu uma avaliação positiva, devem ser aumentados; se a avaliação foi negativa, devem ser diminuídos. Além disso, as probabilidades iniciais p_i, para $i = a, b, c, d$ também são ajustadas, com um fator um pouco menor.

Vamos definir $T^{(c)}$ como o conjunto das técnicas utilizadas no compasso c do improviso. A nota desse compasso, g_c, é calculada como o número de vezes que botão *bom* foi pressionado menos o número de vezes que botão *ruim* foi pressionado durante o compasso c. Ao final do improviso, as probabilidades iniciais serão alteradas de acordo com a fórmula

$$p_i = p_i \cdot \alpha^{g_c} \quad \forall i \in T^{(c)} \quad c = 1, 2, 3, \ldots \tag{3.14}$$

onde α é o *fator de ajuste das probabilidades iniciais*. Os parâmetros s_{ij} serão atualizados de acordo com a equação

$$s_{ij} = s_{ij} \cdot \beta^{g_c} \quad \forall i, j \in T^{(c)} \text{ onde a técnica } i \text{ é imediatamente anterior a } j \quad c = 1, 2, 3, \ldots \tag{3.15}$$

onde β é o *fator de ajuste dos parâmetros condicionais*. Tipicamente, teremos $\alpha < \beta$, ou seja, as probabilidades iniciais são ajustadas de forma mais lenta do que os parâmetros condicionais s_{ij}.

Para clarear, um exemplo: se no compasso 1, a nota dada foi 2 (botão "bom" pressionado 2 vezes, e as técnicas utilizadas foram a,b,c e d, nessa ordem, as probabilidades iniciais serão ajustadas da seguinte forma:

$$p_a = p_a \cdot \alpha^2$$
$$p_b = p_b \cdot \alpha^2$$
$$p_c = p_c \cdot \alpha^2$$
$$p_d = p_c \cdot \alpha^2$$

e os parâmetros s_{ij} serão atualizados com as fórmulas

$$s_{ab} = s_{ab} \cdot \beta^2$$
$$s_{bc} = s_{ab} \cdot \beta^2$$
$$s_{cd} = s_{ab} \cdot \beta^2$$

Os valores utilizados para os testes foram $\alpha = 1{,}05$ e $\beta = 1{,}2$.

3.10 Funcionamento do algoritmo

O *Bebopper* foi implementado em C++. A comunicação com o usuário é feita através de sua interface gráfica, mostrada na figura 3.29. O usuário pode configurar a saída MIDI para qualquer saída disponível no computador. Pode também inserir a harmonia, com número de compassos e número de *chorus* (repetições), e armazená-la em um arquivo, para utilização futura. A performance executada pode opcionalmente ser gravada em um arquivo MIDI. Os volumes da melodia(improviso) e cada instrumento podem ser alterados em tempo real. Ao iniciar a performance, os botões de avaliação são ativados, para que o processo de aprendizado supervisionado possa ocorrer.

O funcionamento do algoritmo pode ser resumido pelo seguinte pseudo-código:

```
Inicie a Interface
Aguarde a entrada dos parâmetros pelo usuário
Início da execução:
Enquanto não chegar ao fim da peça, faça:
- Escolha o ritmo do improviso no trecho atual
- Escolha técnicas para preencher o ritmo escolhido
- Crie a seção rítmica do trecho
- Envie para o buffer MIDI o trecho montado
- Armazene as notas dadas pelo usuário para o trecho
- Próximo trecho
Fim (Enquanto)
Gravar arquivo MIDI de saída
Ajustar os parâmetros do modelo estocástico
Voltar à Interface
```

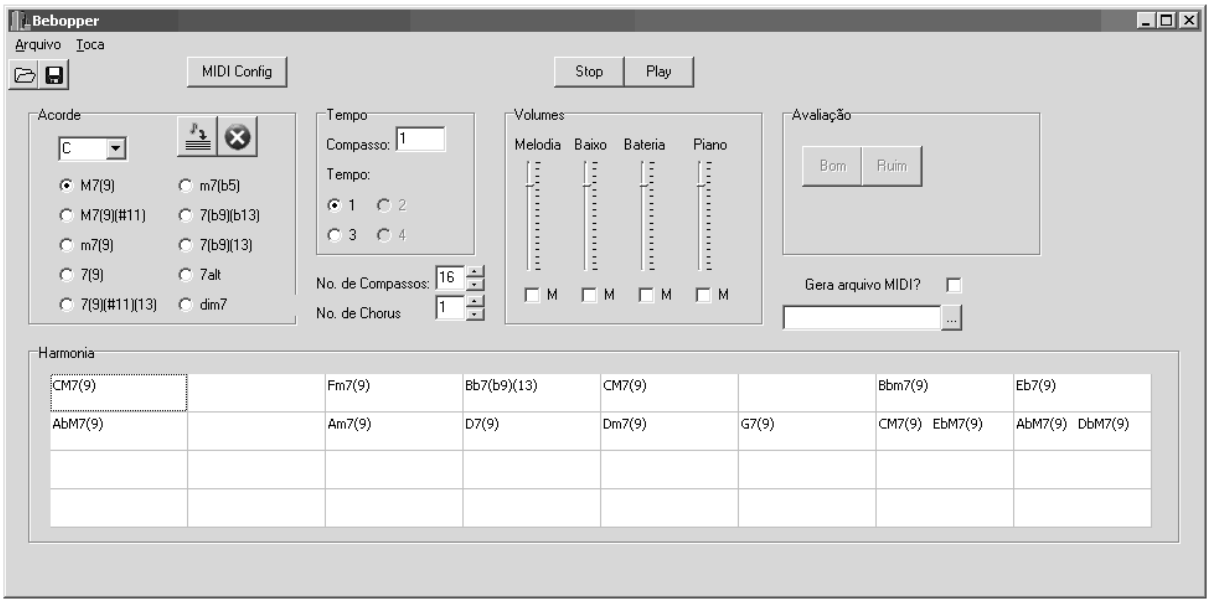

Figura 3.29: A interface gráfica do Bebopper

Capítulo 4

Conclusões e Perspectivas

Uma das motivações para a criação deste algoritmo foi o fato de que vários algoritmos existentes criam melodias de uma entre duas formas: probabilidades de transição nota a nota, com distribuições aleatórias, cadeias de Markov ou mesmo redes neurais treinadas para isso; a outra forma é aplicando variações sobre temas dados. A primeira possui muito pouca informação musical. Mesmo métodos mais sofisticados como redes neurais não conseguem capturar macroestrutura musical, ficando apenas na superfície: apenas as transições locais se assemelham com as de uma melodia humana. A segunda forma, de aplicação de variações sobre temas dados, obtêm mais sucesso estético, mas é difícil conseguir balancear o dilema estrutura-criatividade[140]: se são permitidas poucas variações, garante-se que o material gerado terá bastante estrutura, pois terá semelhanças com o tema inicial, mas será pouco criativo; se são permitidas variações excessivas, teremos mais criatividade, mas é possível que as frases geradas já não tenham nenhuma ligação entre si, e com isso perde-se em estrutura.

A abordagem tomada no *Bebopper* se assemelha com a primeira, mas com uma diferença fundamental: ao invés de usar notas individuais, são implementadas técnicas que músicos de *bebop* utilizam para montar conjuntos de notas. Adicionando essa informação musical, as frases geradas têm muito mais sentido. Ao invés de tentar montar frases com letras, as frases serão montadas com sílabas, e até palavras, do vocabulário do *bebop*.

As frases montadas dessa forma fazem muito mais sentido musical. De fato, logo nas primeiras iterações, antes mesmo de treinar os parâmetros do método estocástico, em algumas frases geradas se reconhecia nitidamente o estilo de um *jazzman*.

Outra contribuição deste trabalho é a proposta de um método estocástico diferente para a tomada de decisões, baseado no raciocínio de um músico. Analisando os resultados, concluímos que, apesar de suas limitações atuais, o método se mostrou uma alternativa possível a outras abordagens.

Na seção seguinte apresentamos alguns resultados musicais e mais detalhes sobre a performance do modelo método estocástico.

4.1 Resultados

Na figura 4.2 temos um exemplo de saída do Bebopper. Após uma pequena fase de treinamento, para ajustar levemente os parâmetros, forçamos o parâmetro $s_{45,45}$ a um valor bem alto e $p_i^{MAX} = 0{,}95\,\forall i$. Fizemos também $s_{i,45}$ para alguns valores de i, ou seja, diminuindo a probabilidade da técnica caso ela não seja escolhida logo. Com isso, queríamos testar a situação apresentada na figura 3.27, onde uma técnica é utilizada repetidamente, criando tensão, que será resolvida quando o

músico utilizar outra técnica. Foi utilizada a técnica 45 nesse exemplo, um clichê de John Coltrane(ver Apêndice). Na partitura da figura 4.2 estão indicados os momentos onde a técnica 45 é utilizada. Na figura 4.1 está desenhado o gráfico da probabilidade dessa técnica em função do tempo. Pode-se ver que cada utilização da técnica no improviso corresponde a um aumento brusco na sua probabilidade, facilitando a sua escolha repetidamente. Em dois momentos a técnica foi utilizada seguidamente, sendo utilizada 3 vezes seguidas no compasso 13, ultrapassando o valor p_{45}^{MAX}, tendo o seu valor reduzido a $p_{45}/2$.

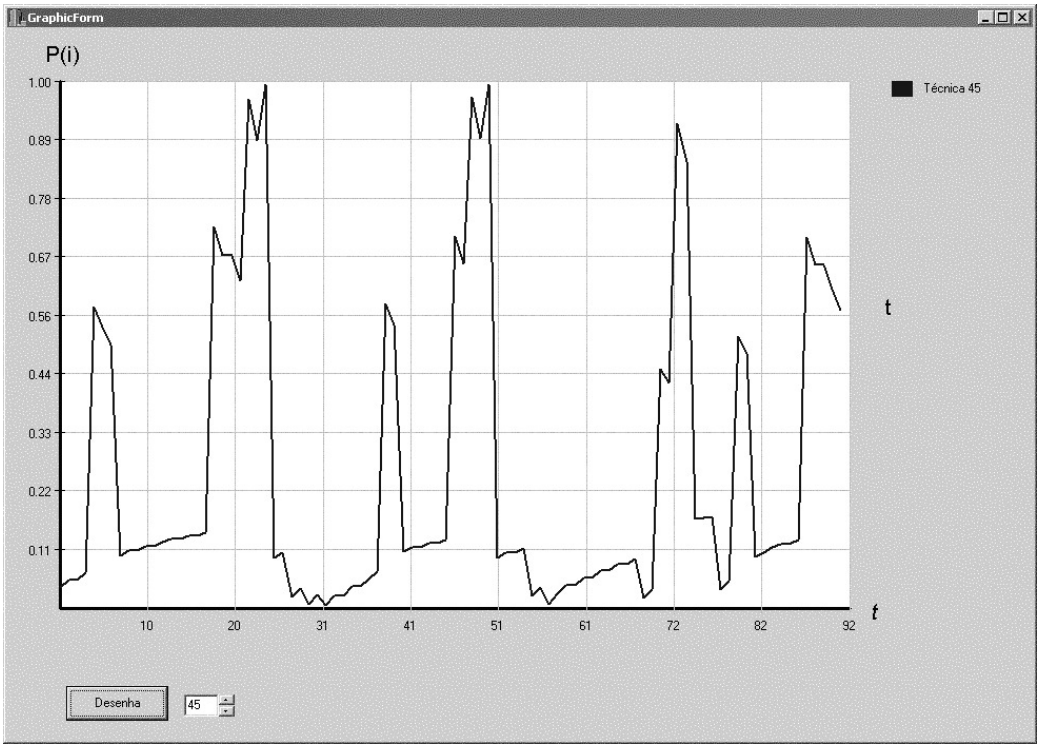

Figura 4.1: Gráfico da probabilidade da técnica 45 em função do tempo, gerado pelo *Bebopper*

Na figura 4.3 temos a partitura de um outro exemplo gerado pelo *Bebopper*. Agora queríamos simular a situação da figura 3.28, onde uma técnica tem uma probabilidade alta de ser utilizada, mas o seu parâmetro de ajuste condicional reduz bastante a sua probabilidade, fazendo com que a técnica não seja utilizada exageradamente. A técnica desse exemplo é a de número 42, o clichê de Miles Davis apresentado na figura 3.13. Na partitura estão indicados os pontos onde a técnica foi utilizada, e a sua probabilidade é dada na figura 4.1. Pelo gráfico vemos que a técnica não deveria ter sido usada repetidamente, mas como o método é estocástico, sempre há essa chance. Vemos na partitura que a técnica foi escolhida em lugares próximos, como no 2º e 3º compassos, além de ser utilizada seguidamente no 9º compasso. Os outros dois exemplos ficaram mais espalhados, no 5º e no 11º compassos. De fato, utilizando este modelo estocástico, na maioria dos casos consegue-se garantir essas situações específicas, da figura 3.27 e da figura 3.28. Mas não existe uma garantia, pois por menor que seja uma probabilidade, se ela é maior do que zero pode ser escolhida.

Uma opção que começou a ser testada para atenuar esse problema é limitar o número de escolhas possíveis em um dado instante para apenas as técnicas com maior probabilidade, eliminando a possibilidade de escolher uma técnica com probabilidade baixa. Essa opção melhorou levemente a qualidade dos improvisos, pois eram escolhidos aqueles com maior probabilidade muito mais vezes, mais em conseqüencia diminuiu a criatividade, pois certas técnicas acabaram sendo repetidas demasiadamente.

Outro problema que percebemos durante os testes é que, devido ao fato de os parâmetros do processo estocástico da geração do ritmo estarer separados dos parâmetros da geração dos tons, algumas vezes a técnica mais provável apontada pelo modelo estocástico não podia ser escolhida, pois o ritmo do trecho não permitia. Com isso, vários "caminhos" indicados pelo modelo não podem ser seguidos. Uma atualização possível no *Bebopper* pode ser a integração dos parâmetros da criação do ritmo e dos tons. A complexidade adicional seria compensada pela maior robustez do modelo resultante.

Apesar do problema acima, o modelo foi capaz de absorver o treinamento de uma forma bem razoável. Após algumas execuções foi possível notar o surgimento de algumas frases bem elaboradas, e do aumento da freqüência das técnicas consideradas melhores durante a avaliação. Um dos maiores problemas é a quantidade de parâmetros, especialmente a matriz de ajuste condicional. Com 54 técnicas atuais, a matriz possui 2916 elementos. Considerando que a cada compasso são geralmente utilizadas 2 técnicas, seriam necessários 1458 compassos, na melhor das hipóteses, para alterar cada um deles, algo em torno de 45 melodias de 32 compassos. Na prática, as alterações começam a ser percebidas mais fortemente a partir de de 100 melodias, aproximadamente.

Outro ponto que poderia ser melhorado é a ordem do modelo. Para modificar as probabilidades só é levada em consideração a técnica que acabou de ser escolhida. Ou seja, pode ser considerado de 1º ordem, mesmo não sendo rigoromente, pois, ao escolher uma determinada técnica, as probabilidades dificilmente serão as mesmas de quando se escolheu essa mesma técnica anteriormente, devido ao caráter dinâmico do sistema. Ainda assim, poderíamos aumentar a sua ordem considerando não só a técnica atual mais um certo número de técnicas anteriores para aplicar o ajuste condicional. O problema é que o número de parâmetros cresce exponencialmente, o que provavelmente tornaria o treinamento supervisionado um processo muito lento de ajuste dos parâmetros. Uma solução seria codificar uma biblioteca de improvisos de jazz como técnicas do sistema e utilizá-la para o treinamento. A fase de treinamento supervisionado serviria para um ajuste fino posterior.

4.2 Perspectivas

Dentre as melhorias em vista, podemos citar o aumento da quantidade de técnicas melódicas e rítmicas, para que sejam possíveis maiores possibilidades na criação do improviso.

Para o modelo estocástico e processo de treinamento, já vimos que é preciso fundir os parâmetros de criação rítmica e melódica em um só conjunto, para evitar incongruências no modelo. Outra possibilidade é aumentar a ordem do modelo, e treiná-lo em uma biblioteca de improvisos.

Uma parte que será melhorada futuramente é a seção rítmica, atualmente muito simples, com a intenção apenas de acompanhar o improviso. O passo seguinte é estudar formas de interação entre os instrumentos da seção rítmica e o instrumento solista, e entre os próprios instrumentos da seção rítmica, para que esta se torne viva, e não apenas baseada em padrões fixos. Cada instrumento se comunicará com os outros musicalmente, ao longo da execução. Além disso, qualquer um dos instrumentos poderá ser substituído por um músico humano, em tempo real, via MIDI. O músico pode ser o solista, e a seção rítmica deverá acompanhá-lo de acordo; pode ser o acompanhamento de piano, e perceber a comunicação entre o seu acompanhamento e o resto da seção rítmica, e consequentemente com o solista. Essa execução de forma interativa seria uma forma muito mais interessante para a prática musical (em alternativa a programas já existentes, como o *Band-in-a-Box*^{TM}) e até para apresentações ao vivo.

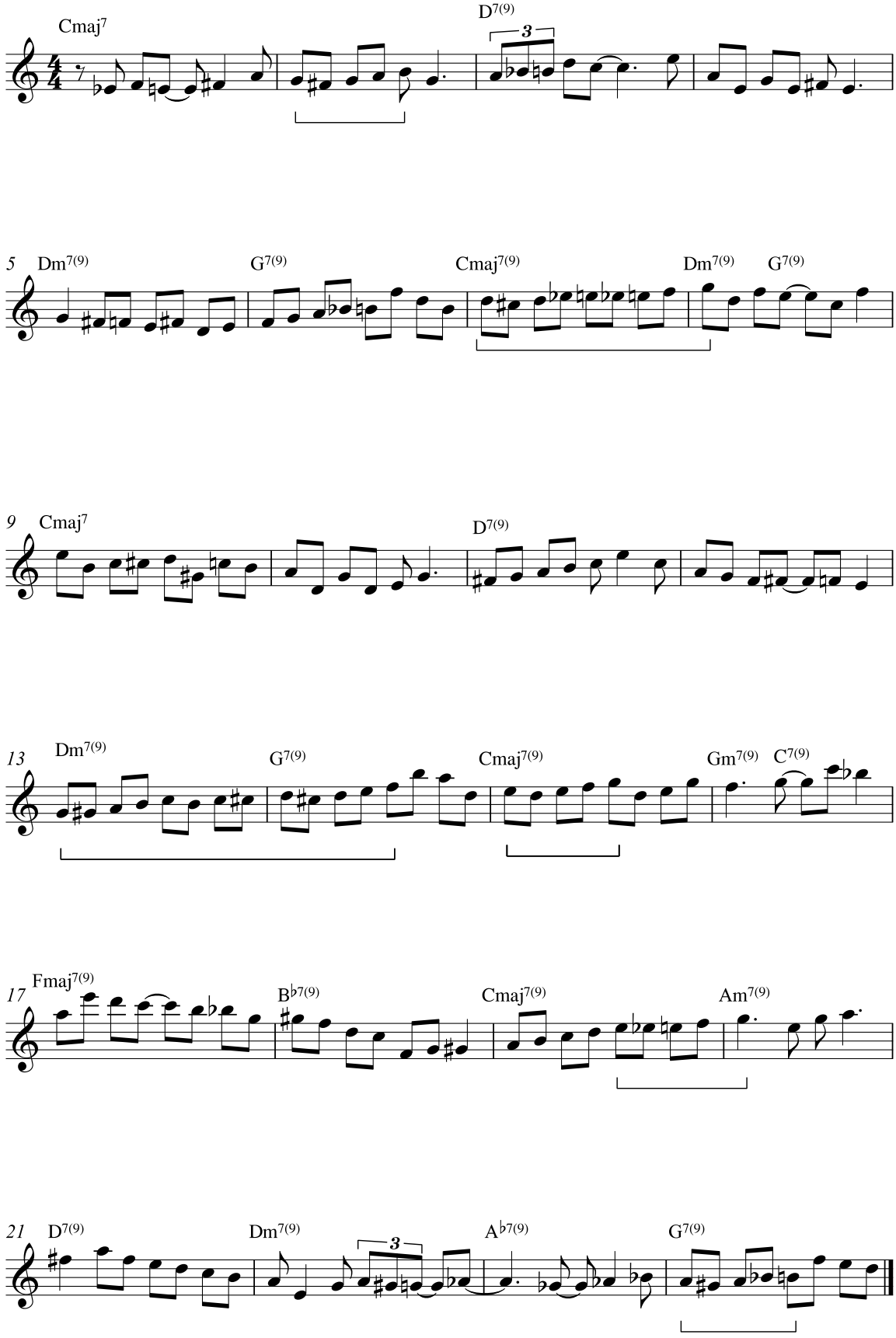

Figura 4.2: Um improviso gerado pelo *Bebopper* com a técnica 45 indicada

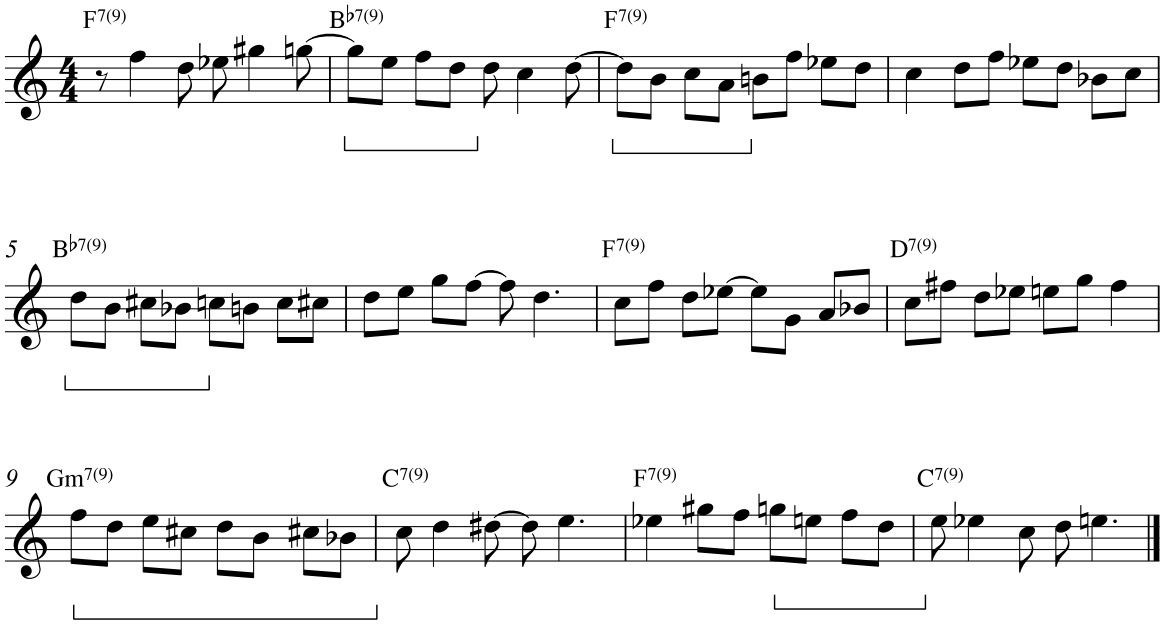

Figura 4.3: Um improviso gerado pelo *Bebopper* com a técnica 45 indicada

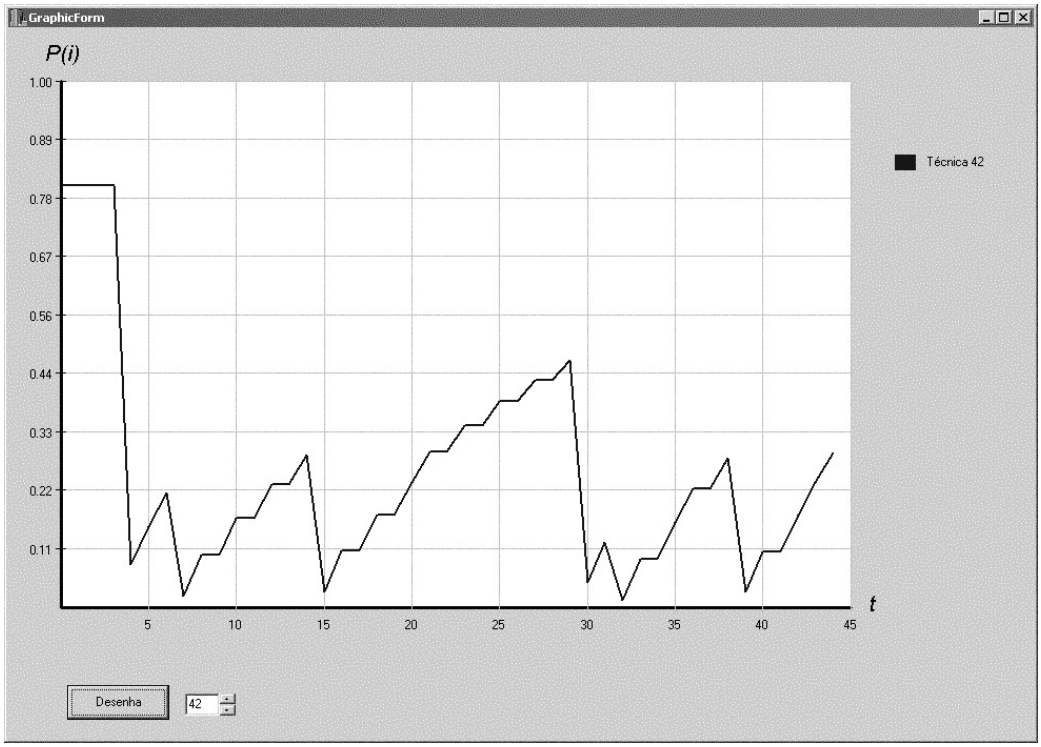

Figura 4.4: Gráfico da probabilidade da técnica 45 em função do tempo, gerado pelo *Bebopper*

Bibliografia

[1] Aamodt, A. and Plaza, E. [1994]. Case-based reasoning: foundational issues, methodological variations and system approaches, *Artificial Intelligence Communications* **7**(1): 39–59.

[2] Ahalt, S. C., Krishnamurthy, A. K., Chen, P. and Melton, D. E. [1990]. Competitive learning algorithms for vector quantization, *Neural Networks* **3**(3): 277–290.

[3] Allan, M. [2002]. *Harmonising chorales in the style of johann sebastian bach*, Master's thesis, University of Edinburgh.

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

[5] Ames, C. [1995]. Thresholds of confidence: An analysis of statistical methods for composition, part 1: Theory, *Leonardo Music Journal* **5**: 33–38.

[6] Ames, C. [1996]. Thresholds of confidence: An analysis of statistical methods for composition, part 2: Applications, *Leonardo Music Journal* **6**: 21–26.

[7] Arcos, J.-L. and Lopez de Mantaras, R. [2000]. Combining ai techniques to perform expressive music by imitation, *in* W. Birmingham, G. Widmer and R. Dannenberg (eds), *Artificial Intelligence and Music: Towards formal models for composition, performance and analysis*, Seventh National Conference on Artificial Intelligence, pp. 41–47.

[8] Arcos, J.-L., Lopez de Mantaras, R. and Cañamero, D. [1999]. Imitating human performances to automatically generate expressive jazz ballads, *Proceedings of the AISB'99 Symposium on Imitation in Animals and Artifacts*, pp. 115–120.

[9] Arcos, J.-L., Lopez de Mantaras, R. and Serra, X. [1998]. Saxex: A case-based reasoning system for generating expressive musical performances, *Journal of New Music Research* **27**(3): 194–210.

[10] Back, T. and Schwefel, H.-P. [1996]. Evolutionary computation: an overview, *Proceedings of the Third IEEE Conference on Evolutionary Computation*.

[11] Baker, D. N. [1989]. *How to Play Bebop, for all instruments*, Vol. 1-3, Alfred Publishing Co. Inc.

[12] Bareiss, E. R., Porter, B. W. and Wier, C. C. [1991]. Protos: an exemplar-based learning apprentice, *in* Y. Kodratoff and R. Michalski (eds), *Machine Learning: An Artificial Intelligence Approach*, Vol. 3, Morgan Kaufmann, pp. 112–139.

[13] Baroni, M., Dalmonte, R. and Jacoboni, C. [1999]. *Le regole della musica. Indagine sui meccanismi della comunicazione*, EDT, Torino.

[14] Barreto, J. M. [2002]. Introdução as redes neurais artificiais, Florianópolis: Universidade Federal de Santa Catarina. Disponível em <http://www.inf.ufsc.br/ barreto> Acesso em: 21 dez. 2003.

[15] Bharucha, J. J. and Todd, P. M. [1989]. Modeling the perception of tonal structure with neural nets, *in* P. Todd and D. Loy (eds), *Music and Connectionism*, MIT Press, Cambridge, MA, pp. 128–137.

[16] Biles, J. A. [1994]. Genjam: A genetic algorithm for generating jazz solos, *Proceedings of the 1994 International Computer Music Conference*, International Computer Music Association, San Francisco, pp. 131–137.

[17] Biles, J. A. [1998]. Interactive genjam: Integrating real-time performance with a genetic algorithm, *Proceedings of the 1998 International Computer Music Conference*, San Francisco: ICMA.

[18] Biles, J. A. [2001]. Autonomous genjam: Eliminating the fitness bottleneck by eliminating fitness, Publicação obtida pela internet no endereço http://www.it.rit.edu/ jab/. GECCO-2001 workshop - Non-Routine Design with Evolutionary Systems.

[19] Biles, J. A., Anderson, P. G. and Loggi, L. W. [1996]. Neural network fitness functions for a musical iga, *Proceedings of the International ICSC Symposium on Intelligent Industrial Automation (IIA'96) and Soft Computing (SOCO'96)*, ICSC Academic Press, Reading, UK, pp. B39–B44.

[20] Biletskyy, A. [2000]. Doctor webern: A visual environment for computer-assisted composition based on linear thematism, *Computer Music Journal* **24**(3): 34–37.

[21] Bilotta, E. and Pantano, P. [2000]. In search of musical fitness on consonance, *Electronic Musicological Review, Special Issue 5.3*, Curitiba: Universidade Federal do Paraná.

[22] Bresin, R. [1998]. Artificial neural networks based models for automatic performance of musical scores, *Journal of New Music Research* **27**(3): 239–270.

[23] Bresin, R. [2000]. *Virtual Virtuosity: Studies in Automatic Music Performance*, PhD thesis, Dept. of Speech Music and Hearing, Royal Institute of Technology - KTH, Stockholm.

[24] Bresin, R. and Friberg, A. [2000]. Emotional coloring of computer-controlled music performances, *Computer Music Journal* **24**(4): 44–63.

[25] Burton, A. R. and Vladimirova, T. [1997]. Genetic algorithm utilising neural network fitness evaluation for musical composition, *in* G. Smith, N. Steele and R. Albrecht (eds), *Proceedings of the 1997 International Conference on Artificial Neural Networks and Genetic Algorithms*, Vienna: Springer-Verlag, pp. 220–224.

[26] Burton, A. R. and Vladimirova, T. [1999]. Generation of musical sequences with genetic techniques, *Computer Music Journal* **23**(4): 59–73.

[27] Buzzanca, G. [2001]. A rule-based expert system for musical style recognition, *Proceedings of the 1st International Conference: Understanding and Creating Music*, Caserta.

[28] Buzzanca, G. [2002]. A supervised learning approach to musical style recognition, *in* C. Anagnostopoulou, M. Ferrand and A. Smaill (eds), *Additional Proceedings of the Second International Conference*, ICMAI 2002, Edinburgh, Scotland.

[29] Cambouropoulos, E., Crawford, T. and Iliopoulos, C. S. [1999]. Pattern processing in melodic sequences: Challenges, caveats & prospects, *Proceedings of the AISB'99 Convention (Artificial Intelligence and Simulation of Behaviour*, pp. 42–47.

[30] Cambouropoulos, E., Crochemore, M., Iliopoulos, C. S., Mouchard, L. and Pinzon., Y. J. [1999]. Algorithms for computing approximate repetitions in musical sequences, *Proceedings of the10th Australasian Workshop On Combinatorial Algorithms*.

[31] Cambouropoulos, E., Smaill, A. and Widmer, G. [1999]. A clustering algorithm for melodic analysis, *Proceedings of the Diderot'99 Forum on Mathematics and Music: Computational and Mathematical Methods in Music*.

[32] Cambouropoulos, E. and Widmer, G. [2000]. Automated motivic analysis via melodic clustering, *Journal of New Music Research* **29**(4): 303–317.

[33] Canazza, S., Poli, G. D., Rodà, A. and Vidolin, A. [2003]. An abstract control space for communication of sensory expressive intentions in music performance, *Journal of New Music Research* **32**(3): 281–294.

[34] Carpenter, G. A. and Grossberg, S. [1987]. A massively parallel architecture for a self-organizing neural pattern recognition machine, *Computer Vision, Graphics, and Image Processing* **37**(1): 54–115.

[35] Carpenter, G. A., Grossberg, S. and Mehanian, C. [1989]. Invariant recognition of cluttered scenes by a self-organizing art architecture: Cort-x boundary segmentation, *Neural Networks* **2**(3): 169–181.

[36] Carpinteiro, O. A. S. [1998]. Self-organizing map model for analysis of musical time series, *Proceedings of the 5th Brazilian Symposium on Neural Networks*, IEEE Computer Society, pp. 140–145.

[37] Chomsky, N. [1957]. *Syntactic Structures*, The Hague: Mouton.

[38] Christiansen, C. [2001]. *Essential Jazz Lines: The Style of Charlie Parker*, Mel Bay Publications.

[39] Clement, B. J. [1998]. Learning harmonic progression using markov models, *Technical report*, The University of Michigan.

[40] Conklin, D. [2002]. Representation and discovery of vertical patterns in music, *Music and Artificial Intelligence: Lecture Notes in Artificial Intelligence*, Vol. 2445, Springer- Verlag, pp. 32–42.

[41] Conklin, D. and Anagnostopoulou, C. [2001]. Representation and discovery of multiple viewpoint patterns, *Proceedings of the International Computer Music Conference*, International Computer Music Association, pp. 478–485.

[42] Conklin, D. and Witten, I. H. [1995]. Multiple viewpoint systems for music prediction, *Journal of New Music Research* **24**(1): 51–73.

[43] Cope, D. [1987]. An expert system for computer-assisted composition, *Computer music Journal* **11**(4): 30–46.

[44] Cope, D. [1992]. Computer modeling of musical intelligence in emi, *Computer Music Journal* **16**(2): 69–83.

[45] Cope, D. [1999]. Facing the music: Perspectives on machine-composed music, *Leonardo Music Journal* **9**: 79–87.

[46] Cope, D. [2003]. Computer analysis of musical allusions, *Computer Music Journal* **27**(1): 11–28.

[47] Cunha, U. S. and Ramalho, G. [2000]. Combining off-line and on-line learning for predicting chords in tonal music, *Proceedings of the Workshop on Artificial Intelligence and Music*, AAAI Press, pp. 6–10.

[48] Dugad, R. and Desay, U. B. [1996]. A tutorial on hidden markov models, *Technical Report SPANN-96.1*, Signal Processing and Artificial Neural Networks Laboratory, Department of Electrical Engineering, Indian Institute of Technology.

[49] Ebcioglu, K. [1988]. An expert system for harmonizing four-part chorales, *Computer Music Journal* **13**(3): 43–51.

[50] Feulner, J. and Hörnel, D. [1994]. Melonet: Neural networks that learn harmony-based melodic variations, *Proceedings of the 1994 International Computer Music Conference*, International Computer Music Association.

[51] Forney, G. D. [1973]. The viterbi algorithm, *Proceedings of the IEEE* **61**(3): 268–278.

[52] Franklin, J. A. [2001]. Multi-phase learning for jazz improvisation and interaction, *Proceedings of the Eighth Biennial Symposium on Arts and Technology (Perception and Interaction in the Electronic Arts)*.

[53] Franklin, J. A. [2002]. Improvisation and learning, *in* T. G. Dietterich, S. Becker and Z. Ghahramani (eds), *Advances in Neural Information Processing Systems 14*, MIT Press, Cambridge, MA.

[54] Franklin, J. A. and Manfredi, V. [2002]. Nonlinear credit assignment for musical sequences, *Proceedings of the Second International Workshop on Intelligence Systems Design and Applications*, pp. 245–250.

[55] Franz, D. M. [1998]. *Markov chains as tools for jazz improvisation analysis*, Master's thesis, State University, Virginia.

[56] Freeman, J. A. and Skapura., D. M. [1991]. *Neural Networks: Algorithms, Applications and Programming Techniques*, Addison-Wesley.

[57] Friberg, A. [1991]. Generative rules for music performance: a formal description of a rule system, *Computer Music Journal* **15**(2): 56–71.

[58] Friberg, A. [1995]. *A Quantitative Rule System for Musical Performance*, PhD thesis, Department of Speech, Music and Hearing, Royal Institute of Technology, Stockholm.

[59] Friberg, A. [2000]. Generating musical performances with director musices, *Computer Music Journal* **24**(3): 23–29.

[60] Gabrielsson, A. [1994]. Intention and emotional expression in music performance, *Proceedings of the Stockholm Music Acoustics Conference*, Royal Swedish Academy of Music, pp. 108–111.

[61] Gabrielsson, A. [1995]. Expressive intention and performance, *in* R. Steinberg (ed.), *Music and the Mind Machine: The Psychophysiology and the Psychopathology of the Sense of Music*, Springer Verlag, pp. 35–47.

[62] Gang, D. and Berger, J. [1997]. A neural network model of metric perception and cognition in the audition of functional tonal music, *Proceedings of the 1997 International Computer Music Conference*, International Computer Music Association.

[63] Gang, D., Lehmann, D. and Wagner, N. [1997]. Harmonizing melodies in real-time: the connectionist approach, *Proceedings of the 1997 International Computer Music Conference*, International Computer Music Association.

[64] Gang, D., Lehmann, D. and Wagner, N. [1998]. Tuning neural network for harmonizing melodies in real-time, *Proceedings of the 1998 International Computer Music Conference*, International Computer Music Association.

[65] Gibson, P. M. and Byrne, J. A. [1991]. Neurogen, musical composition using genetic algorithms and cooperating neural networks, *Proceedings of the 2nd International Conference in Artificial Neural Networks*, pp. 309–313.

[66] Gogins, M. [1991]. Iterated functions systems music, *Computer Music Journal* **15**(1): 40–48.

[67] Goldman, C. V., Gang, D., Rosenschein, J. S. and Lehmann, D. [1999]. Netneg: A connectionist-agent integrated system for representing musical knowledge, *Annals of Mathematics and Artificial Intelligence*, pp. 69–90.

[68] Grachten, M., Arcos, J.-L. and Lopez de Mantaras, R. [2003]. Extracting performers' behaviors to annotate cases in a cbr system for musical tempo transformations, *Case-Based Reasoning Research and Development* **LNAI 2689**: 20–34.

[69] Gwee, N. [2002]. *Complexity and Heuristics in Rule-Based Algorithmic Music Composition*, PhD thesis, Louisiana State University.

[70] Heins, L. G. and Tauritz, D. R. [1995]. Adaptive resonance theory (art): An introduction, *Technical Report 95-35*, Leiden University.

[71] Henz, M., Lauer, S. and Zimmermann, D. [1996]. Compoze – intention-based music composition through constraint programming, *Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence*, pp. 118–121.

[72] Hild, H., Feulner, J. and Menzel, W. [1992]. Harmonet: A neural net for harmonizing chorales in the style of j.s. bach, *in* R. P. Lippmann, J. E. Moody and D. S. Touretzky (eds), *Advance in Neural Information Processing 4 (NIPS 4)*, Morgan Kaufmann, pp. 267–274.

[73] Holland, J. [1975]. *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI.

[74] Horner, A. and Ayers, L. [1995]. Harmonization of musical progressions with genetic algorithms, *Proceedings of the 1995 International Computer Music Conference*, International Computer Music Association, Alberta, Canada, pp. 483–484.

[75] Horner, A. and Goldberg, D. E. [1991]. Algorithms and computer-assisted composition, *Proceedings of the Fourth International Conference on Genetic Algorithms*.

[76] Horowitz, D. [1994]. Generating rhythms with genetic algorithms, *Proceedings of the 1994 International Computer Music Conference*, International Computer Music Association, Aarhus, Denmark.

[77] Hörnel, D. [1993]. Systhema - analysis and automatic synthesis of classical themes, *Proceedings of the 1993 International Computer Music Conference*, International Computer Music Association.

[78] Hörnel, D. and Degenhardt, P. [1997]. A neural organist improvising baroque-style melodic variations, *Proceedings of the 1997 International Computer Music Conference*, International Computer Music Association.

[79] Hörnel, D. and Menzel, W. [1998]. Learning musical structure and style with neural networks, *Computer Music Journal* **22**(4): 44–62.

[80] Hörnel, D. and Olbrich, F. [1999]. Comparative style analysis with neural networks, *Proceedings of the 1999 International Computer Music Conference*, International Computer Music Association.

[81] Hörnel, D. and Ragg, T. [1996a]. A connectionist model for the evolution of styles of harmonization, *Proceedings of the 1996 International Conference on Music Perception and Cognition*, Faculty of Music, McGill University, Montreal. ISBN 0-7717-0484-4.

[82] Hörnel, D. and Ragg, T. [1996b]. Learning musical structure and style by recognition, prediction and evolution, *Proceedings of the 1996 International Computer Music Conference*, International Computer Music Association.

[83] Höthker, K. and Hörnel, D. [2000]. Harmonizing in real-time with neural networks, *Proceedings of the 2000 International Computer Music Conference*, International Computer Music Association.

[84] Ishikawa, O., Aono, Y., Katayose, H. and Inokuchi, S. [2000]. Extraction of musical performance rules using a modified algorithm of multiple regression analysis, *Proceedings of the International Computer Music Conference*, Computer Music Association, pp. 348–351.

[85] Jacob, B. L. [1995]. Composing with genetic algorithms, *Proceedings of the 1995 International Computer Music Conference*, San Francisco: International Computer Music Association, pp. 452–455.

[86] Jacob, B. L. [1996]. Algorithmic composition as a model of creativity, *Organised Sound* **1**(3): 157–165.

[87] Jain, A. K., Mao, J. and Mohiuddin, K. [1996]. Artificial neural networks: A tutorial, *Computer* **29**(3): 31–44.

[88] Johanson, B. and Poli, R. [1998]. GP-music: An interactive genetic programming system for music generation with automated fitness raters, *in* J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba and R. Riolo (eds), *Genetic Programming 1998: Proceedings of the Third Annual Conference*, Morgan Kaufmann, University of Wisconsin, Madison, Wisconsin, USA, pp. 181–186.
URL: *http://citeseer.ist.psu.edu/336573.html*

[89] Jones, D. E. [2000]. A computational composer's assistant for atonal counterpoint, *Computer Music Journal* **24**(2): 33–43.

[90] Järveläinen, H. [2000]. Algorithmic musical composition, *Technical report*, Helsinki University of Technology. Tik-111.080 Seminar on content creation.

[91] Juslin, P. [1997a]. Emotional communication in music performance: A functionalist perspective and some data, *Music Perception* **14**(4): 383–418.

[92] Juslin, P. N. [1997b]. Perceived emotional expression in synthesized performances of a short melody: Capturing the listener's judgment policy, *Musicae Scientiae* **1**(2): 225–256.

[93] Kohonen, T. [1997]. *Self-Organizing Maps*, 2nd edition edn, Springer-Verlag, Berlin.

[94] Kolodzy, P. J. [1987]. Multidimensional machine vision using neural networks, *Proceedings of the IEEE First International Conference on Neural Networks*, San Diego, CA, pp. II–747–II–758.

[95] Kolodzy, P. J. and van Alien, E. J. [1987]. Application of a boundary contour neural network to illusions and infrared imagery, *Proceedings of the IEEE First International Conference on Neural Networks*, San Diego, CA, pp. IV–193–IV–202.

[96] Koza, J. R. [1992]. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA.

[97] Krose, B. and van der Smagt, P. [1996]. *An Introduction to Neural Networks*, The University of Amsterdam. Disponível em http://carol.wins.uva.nl/ krose/col/nn/neuro-intro.ps.gz.

[98] Laine, P. [2000]. *A method for Generating Musical Motion Patterns*, PhD thesis, University of Helsinki, Institute for Art Research, Musicology, Faculty of Arts.

[99] Laine, P. and Kuuskankare, M. [1994]. Genetic algorithms in musical style oriented generation, *Proceedings of the International Conference on Evolutionary Computation*, pp. 858–862.

[100] LeCun, Y. [1989]. Generalization and network design strategies, *in* R. Pfeifer, F. Fogelman-Soulie, L. Steels and Z. Schreter (eds), *Connectionism in Perspective*, Elsevier Science Inc.

[101] Lerdahl, F. and Jackendoff, R. [1983]. *A generative theory of tonal music*, MIT Press, Cambridge.

[102] Levine, M. [1989]. *The Jazz Piano Book*, Sher Music.

[103] Lopez de Mantaras, R. [2001]. Case-based reasoning, *in* G. PALIOURAS, V. KARKALETSIS and C. D. SPYROPOULOS (eds), *Lecture Notes in Artificial Inteligence: Machine Learning and its aplications*, Springer-Verlag, pp. 127–145.

[104] McAlpine, K., Miranda, E. and Hoggar, S. [1999]. Making music with algorithms: A case-study system, *Computer Music Journal* **23**(2): 19–30.

[105] McCulloch, W. S. and Pitts, W. [1943]. A logical calculus of the ideas imminent in nervous activity., *Bulletin of Mathematical Biophysics* **5**: 115–133.

[106] McIntyre, R. A. [1994]. Bach in a box: The evolution of four part baroque harmony using the genetic algorithm, *Proceedings of the IEEE Conference on Evolutionary Computation*, pp. 852–857.

[107] Melo, A. F. [1998]. *A connectionist model of tension in chord progressions*, Master's thesis, Department of Artificial Intelligence, University of Edinburgh.

[108] Melo, A. F. and Wiggins, G. [2003]. A connectionist approach to driving chord progressions using tension, *Proceedings of the AISB'03 Symposium on Creativity in Arts and Science*, SSAISB Publications.

[109] Meredith, D., Lemström, K. and Wiggins, G. A. [2001]. Pattern induction and matching in polyphonic music and other multidimensional datasets, *Proceedings of the Fifth World Multi-conference on Systemics, Cybernetics and Informatics*, International Institute of Informatics and Systemics, pp. 61–66.

[110] Meredith, D., Lemström, K. and Wiggins, G. A. [2002]. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music, *Journal of New Music Research* **31**(4): 321–345.

[111] Moon, T. K. [1996]. The expectation-maximization algorithm, *IEEE Signal Processing Magazine* **13**(6): 47–60.

[112] Moroni, A. [2003]. *ArTEbitrariedade: Uma Reflexão sobre a Natureza da Criatividade e sua Possível Realização em Ambientes Computacionais*, PhD thesis, DCA/FEEC/Unicamp.

[113] Moroni, A., Manzolli, J., Zuben, F. V. and Gudwin, R. [2000]. Vox populi: An interactive evolutionary system for algorithmic music composition, *Leonardo Music Journal* **10**: 49–54.

[114] Mozer, M. C. [1994]. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multiscale processing, *Connection Science* **6**(2–3): 247–280.

[115] Nigrin, A. [1993]. *Neural networks for pattern recognition*, MIT Press, Cambridge, MA, USA.

[116] Pachet, F. [1994]. Representing knowledge used by jazz musicians, *Proceedings of the International Computer Music Conference*, International Computer Music Association, pp. 285–288.

[117] Pachet, F. and Roy, P. [1998]. *Formulating Constraint Satisfaction Problems on Part-Whole relations: the Case of Automatic Harmonization*, Workshop at ECAI'98. Constraint Techniques for Artistic Applications, Brighton, UK.

[118] Papadopoulos, G. and Wiggins, G. [1998]. A genetic algorithm for the generation of jazz melodies, *STeP'98, Human and Artificial Information Processing Finnish Conference on Artificial Intelligence*, Jyväskylä.

[119] Papadopoulos, G. and Wiggins, G. [1999]. Ai methods for algorithmic composition: A survey, a critical view and future prospects, *Proceedings of the AISB'99 Symposium on Musical Creativity*.

[120] Parker, C. [1978]. *Charlie Parker Omnibook: C instruments*, Atlantic Music.

[121] Pearce, M., Meredith, D. and Wiggins, G. [2002]. Motivations and methodologies for automation of the compositional process, *Musicae Scientiae* **6**(2): 119–147.

[122] Phon-Amnuaisuk, S., Smaill, A. and Wiggins, G. [2002]. A computational model for chorale harmonisation in the style of j.s. bach, *Proceedings of ICMPC7 (the 7th International Conference on Music Perception and Cognition)*, Sydney, Australia.

[123] Phon-Amnuaisuk, S., Tuson, A. and Wiggins, G. A. [1999]. Evolving musical harmonisation, *ICANNGA*.

[124] Phon-Amnuaisuk, S. and Wiggins, G. A. [1999]. The four-part harmonisation problem: A comparison between genetic algorithms and a rule-based system, *Proceedings of the AISB'99 Symposium on Musical Creativity*.

[125] Ponsford, D., GeraintWiggins and Mellish, C. [1999]. Statistical learning of harmonic movement, *Journal of New Music Research* **28**(2): 150–177.

[126] Rabiner, L. R. and Rader, C. M. [1972]. *Digital signal processing*, IEEE Press selected reprint series, IEEE, New York. ISBN 0879420189.

[127] Ramalho, G. and Ganascia, J.-G. [1994]. Simulating creativity in jazz performance, *Proceedings of the twelfth national conference on Artificial intelligence*, AAAI Press, pp. 108–113.

[128] Ramalho, G., Rolland, P.-Y. and Ganascia, J.-G. [1999]. An artificially intelligent jazz performer, *Journal of New Music Research* **28**(2): 105–129.

[129] Roads, C. [1985]. Grammars as representations for music, *in* C. Roads and J. Stawn (eds), *Foundations of Computer Music*, MIT Press, pp. 403–442.

[130] Rolland, P.-Y. [1999]. Discovering patterns in musical sequences, *Journal of New Music Research* **28**(4): 334–350.

[131] Rosenblatt, F. [1957]. The perceptron: A perceiving and recognizing automaton, *Technical Report 85-460-1, Project PARA*, Cornell Aeronautical Laboratory, Ithaca, New York.

[132] Rumelhart, D. E. and Zipser, D. [1985]. Feature discovery by competitive learning, *Cognitive Science* **9**: 75–112.

[133] Schank, R. C. [1982]. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*, Cambridge University Press.

[134] Spector, L. and Alpern, A. [1995]. Induction and recapitulation of deep musical structure, *Proceedings of International Joint Conference on Artificial Intelligence, IJCAI'95 Workshop on Music and AI*, Montreal, Canada: IJCAII, pp. 41–48.

[135] Steedman, M. [1984]. A generative grammar for jazz chord sequences, *Music Perception* **2**: 52–77.

[136] Steedman, M. [1996]. The blues and the abstract truth: Music and mental models, *in* A. Garnham and J. Oakhill (eds), *Mental Models In Cognitive Science*, Erlbaum, Mahwah, NJ, pp. 305–318.

[137] Sundberg, J., Friberg, A. and Bresin, R. [2003]. Attempts to reproduce a pianist's expressive timing with director musices performance rules, *Journal of New Music Research* **32**(3): 317–325.

[138] Thom, B. [1995]. Predicting chords in jazz: the good, the bad, and the ugly, *IJCAI-95, Music and AI Workshop*, IJCAI.

[139] Todd, P. M. [1989]. A connectionist approach to algorithmic composition, *Computer Music Journal* **13**(4): 27–43.

[140] Todd, P. M. and Werner, G. M. [1999]. Frankensteinian methods for evolutionary music composition, *in* N. Griffith and P. Todd (eds), *Musical networks: Parallel distributed perception and performance*, MIT Press/Bradford Books.

[141] Veelenturf, L. P. [1995]. *Analysis and Applications of Artificial Neural Networks*, Prentice Hall, New York.

[142] Widmer, G. [1992]. Qualitative perception modeling and intelligent musical learning, *Computer Music Journal* **16**(2): 51–68.

[143] Widmer, G. [2000]. Large-scale induction of expressive performance rules: First quantitative results, *Proceedings of the International Computer Music Conference*, International Computer Music Association.

[144] Widmer, G. [2002]. Machine discoveries: A few simple, robust local expression principles, *Journal of New Music Research 2002, Vol. 31* **31**(1): 37–50.

[145] Widmer, G. and Tobudic, A. [2003]. Playing mozart by analogy: Learning multi-level timing and dynamics strategies, *Journal of New Music Research* **32**(3): 259–268.

[146] Wooldridge, M. J. and Jennings, N. R. (eds) [1995]. *Intelligent Agents - Theories, Architectures, and Languages*, Vol. 890 of *Lecture Notes in AI*, Springer-Verlag.

[147] Xenakis, I. [2001]. *Formalized Music: Thought and Mathematics in Composition*, Harmonologia Series No.6, Pendragon Press, Hillsdale, NY. ISBN 1576470792.

[148] Zanon, P. and Poli, G. D. [2003]. Estimation of parameters in rule systems for expressive rendering of musical performance, *Computer Music Journal* **27**(1): 29–46.

[149] Zicarelli, D. [1987]. M and jam factory, *Computer Music Journal* **11**(4): 13.

Apêndice A

Lista das escalas *bebop* utilizadas

Nesta seção serão apresentadas todas as escalas, usuais ou sintéticas, utilizadas para os 10 tipos de acordes permitidos pelo *Bebopper*, conforme apresentados na tabela da seção 3.4.1. As propriedades necessárias a essas escalas estão detalhadas na seção 3.4.2.

A.1 Acorde $C^{7M(9)}$

Para este acorde, ao invés da escala maior *bebop* usual, que considera como NA a tônica, 3ª, 5ª e 6ª do acorde, foi construída uma escala onde a 3ª, 5ª e 7ª e 9ª são as NAs. Para isso, a escala maior foi utilizada como molde. Como entre a 3ª e a 9ª não existe uma NP, é preciso adicionar a 3ª menor à escala, chegando assim à escala sintética que foi denominada escala de maior *add* b3, mostrada na figura A.1.

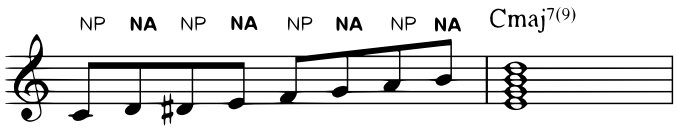

Figura A.1: Escala de C maior *add* b3

A.2 Acorde $C^{7M(9)(\#11)}$

Para este acorde seguimos o raciocínio do acorde anterior, apenas trocando a escala molde de maior para o modo lídio, alterando assim a 4ª justa para 4ª aumentada. A escala sintética resultante foi denominada de lídio *add* b3 (figura A.1).

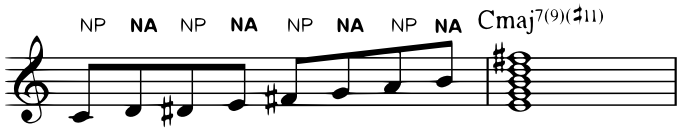

Figura A.2: Escala de C lídio *add* b3

A.3 Acorde $Cm^{7(9)}$

Neste acorde não é possível escolher a 3ª menor, 5ª e 7ª menor e 9ª como NAs, seguindo o raciocínio dos dois acordes anteriores, pois a distância entre a 3ª menor e a 9ª é de 1 semitom, impossibilitando a existência de uma NP entre essas 2 NAs. Temos, portanto, 2 escolhas: a escala menor *bebop* usual(figura A.3, que considera a tônica, a 3ª menor, 5ª e 7ª menor como NAs, ou uma nova escala, denominada menor *add* 3M(figura A.4), onde são NAs a 5ª, 7ª menor, 9ª e 11ª. Na primeira, o improviso soa mais "tradicional", pois são evidenciados os tons básicos do acorde. Na segunda, o solo é mais "moderno", pois as tensões são enfatizadas e também pelo fato de 3ª menor ficar apenas como NP, chegando em alguns momentos em uma sonoridade ambígua.

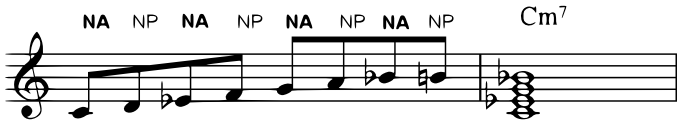

Figura A.3: Escala de C menor *bebop*

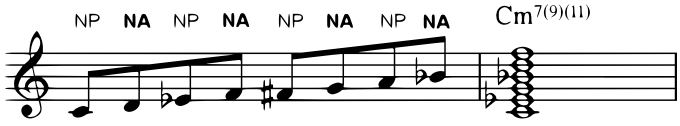

Figura A.4: Escala de C menor *add* 3M

A.4 Acorde $Cm^{7(b5)}$

A escolha usual para acordes deste tipo é a escala lócrio #2, derivada do sexto modo da escala menor melódica([102], pp. 71). Como o caso anterior, não é possível escolher a 3ª menor, 5ª diminuta, 7ª menor e 9ª como NAs, devido à 9ª e 3ª menor serem adjacentes. Logo, escolhendo a tônica, 3ª menor, 5ª diminuta e 7ª menor, basta adicionar a 7ª maior para chegar à escala sintética utilizada, denominada lócrio #2 *add* 7M (figura A.5).

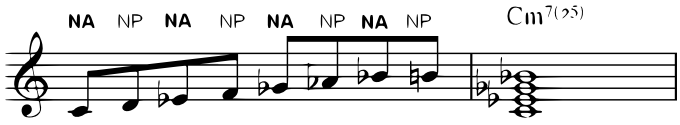

Figura A.5: Escala de C lócrio #2 *add* 7M

A.5 Acorde $C^{7(9)}$

A escala usual para este acorde é o modo mixolídio. Mais uma vez, escolhendo a 3ª, 5ª,7ª menor e 9ª como NAs, vemos que é necessário adicionar a 3ª menor a esta escala para garantir a alternância entre NAs e NPs. Com isso chegamos à escala mixolídio *add* 3m, exemplificada na figura A.6.

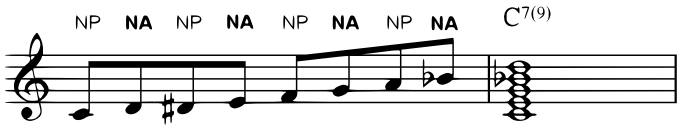

Figura A.6: Escala de C mixolídio *add* 3m

A.6 Acorde $C^{7(9)(\#11)(13)}$

Similar ao caso anterior, mas a tensão #11 sugere o uso da escala mixolídio #11, o quarto modo da escala menor melódica. A escala resultante foi denominada mixolídio #11 *add* 3m.

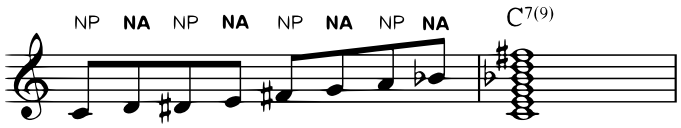

Figura A.7: Escala de C mixolídio #11 *add* 3m

A.7 Acorde $C^{7(b9)(13)}$

Este tipo de acorde pode ser derivado da escala diminuta da nota um semitom acima, ou seja, para o acorde em C, pode-se usar a escala C# diminuta. Começando a escala C# diminuta em C, temos uma escala que é usualmente denominada dominante-diminuta, ou *dom-dim*(figura A.8). Essa escala não precisa de nenhuma nota adicional, pois já é composta por 8 notas.

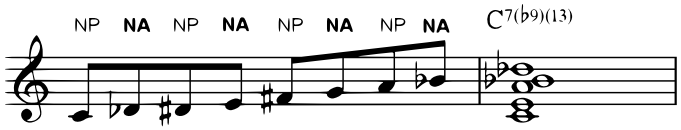

Figura A.8: Escala de C Dominante-Diminuta

A.8 Acorde $C^{7(b9)(b13)}$

Para este tipo de acorde, uma escala bastante utilizada é o quinto modo da a escala menor harmônica, que, embora não tenha um nome padrão definido, pode ser denominada de mixolídio b9 b13, por ser semelhante ao modo mixolídio com a nona e a décima terceira menores. Adicionando a nona aumentada, e considerando a 3ª, 13ª menor,7ª menor e 9ª menor como NAs, temos a escala utilizada, mixolídio b9 b13 *add* #9 (figura A.9).

A.9 Acorde C^{alt7}

Um acorde é denominado dominante alterado quando tanto a nona quanto a quinta são aumentadas ou diminutas, ou qualquer combinação das duas. Neste caso, a escala usual é a escala alterada,

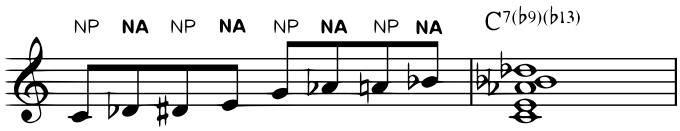

Figura A.9: Escala de C mixolídio b9 b13 *add* #9

o sétimo modo da menor melódica. Adicionando a essa escala a 4ª justa, com as NAs a 3ª, 4ª diminuta,7ª menor e 9ª menor temos a escala sintética denominada escala alterada *add* 4J, desenhada na figura A.10.

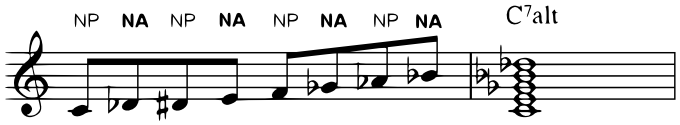

Figura A.10: Escala de C alterada *add* 4J

A.10 Acorde C^o

A escolha óbvia neste caso é a escala diminuta que, como a *dom-dim*, tem 8 notas e não precisa de nenhuma nota adicional.

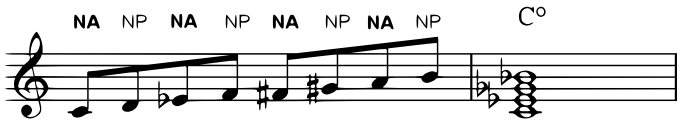

Figura A.11: Escala de C diminuta

Apêndice B

Lista das técnicas melódicas utilizadas

Nesta seção serão apresentadas todas as técnicas melódicas utilizadas para a criação dos improvisos no *Bebopper*. Como vimos, as técnicas estão divididas em 2 categorias: *técnicas de chegada* e *técnicas de partida*(seção 3.5.1).

B.1 Técnicas de chegada

As técnicas de chegada escolhem NP's para uma NA já definida. Elas estão subdivididas de acordo com a quantidade de NP's que antecedem a NA, uma, duas ou três. Para os exemplos desta seção será utilizada a escala C maior *add* b3, apresentada no apêndice A.

B.1.1 Técnicas com 1 NP

Neste caso, o ritmo escolhido determina que apenas 1 NP deve ser tocada antes da NA. Dois exemplos dessa situação estão demonstrados na figura B.1. Foram implementadas 10 técnicas nesta categoria, escolhendo a NP em relação à NA de acordo com intervalos diatônicos de 2ª até 5ª, ascendentes e descententes, e também intervalos de 2ª cromáticos ascentente e descendente. Os intervalos apresentados nesta seção serão abreviados para facilitar a apresentação das técnicas de 2 e 3 NPs.

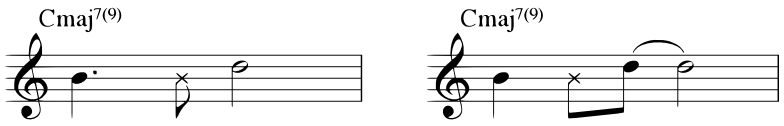

Figura B.1: 2 exemplos de 1 NP de chegada

As técnicas desta categoria estão exemplificadas na figura B.3.

1. Intervalo diatônico de 2a descendente (2D descendente)
2. Intervalo diatônico de 2a ascendente (2D ascendente)
3. Intervalo de 2a menor (cromático) descendente (2C descendente)
4. Intervalo de 2a menor (cromático) ascendente (2C ascendente)

5. Intervalo diatônico de 3a descendente (3D descendente)
6. Intervalo diatônico de 3a ascendente (3D ascendente)
7. Intervalo diatônico de 4a descendente (4D descendente)
8. Intervalo diatônico de 4a ascendente (4D ascendente)
9. Intervalo diatônico de 5a descendente (5D descendente)
10. Intervalo diatônico de 5a ascendente (5D ascendente)

B.1.2 Técnicas com 2 NPs

Neste caso, o ritmo escolhido determina que 2 NPs devem ser executadas antes da NA. Dois exemplos dessa situação estão demonstrados na figura B.2. Nesta categoria foram implementadas 16 técnicas, 10 delas combinando os intervalos da seção anterior (fig. B.4) e 6 usando NAs da escala (fig.ç B.5).

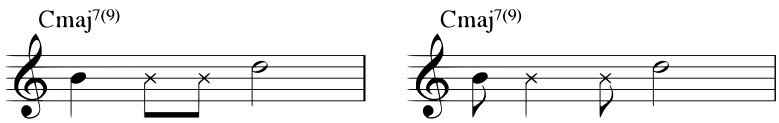

Figura B.2: 2 exemplos com 2 NPs de chegada

11. 2D descendente e 2D ascendente
12. 2D ascendente e 2D descendente
13. 2D descendente e 2C ascendente
14. 2C ascendente e 2D descendente
15. 2C descendente e 2D ascendente
16. 2D ascendente e 2C descendente
17. 3D descendente e 2D descendente
18. 2D descendente e 3D descendente
19. 3D ascendente e 2D ascendente
20. 2D ascendente e 3D ascendente
21. NA ascendente e NA descendente
22. NA descendente e NA ascendente
23. 2ª NA ascendente e NA ascendente
24. NA ascendente e 2ª NA ascendente
25. 2ª NA descendente e NA descendente
26. NA descendente e 2ª NA descendente

B.1.3 Técnicas com 3 NPs

Neste seção serão apresentadas as técnicas utilizadas quando há 3 NPs a incluir antes da NA. Foram implementadas 13 técnicas nesta categoria (fig. B.6).

27. 2C descendente com a próxima NP, 3D descendente com NA e 2D descendente com NA
28. 2C descendente com a próxima NP, 2D descendente com NA e 3D descendente com NA
29. 4D descendente, 3D descendente e 2D descendente com NA
30. 4D ascendente, 3D ascendente e 2D ascendente com NA
31. 3 notas cromáticas abaixo da NA
32. 3 notas cromáticas acima da NA
33. NA ascendente, 2D descendente e 2C descendente
34. 2D descendente, NA ascendente, e 2C descendente
35. 2ª NA descentende, NA descendente e 2D descendente
36. 2ª NA ascendente, NA ascendente e 2D ascendente
37. 2ª NA ascendente, 2D descendente com a nota anterior e NA ascendente
38. 2ªD ascendente, 2ª Maior descendente e 2ªC descendente
39. 2ªC descendente, 3ªD ascendente e 2ªD ascendente

B.2 Técnicas de partida

As técnicas de partida escolhem as NP's de um trecho e a próxima NA, a partir da NA atual. Para os exemplos desta seção será utilizada a escala C maior *add* b3, apresentada no apêndice A. Exemplos de todas as técnicas de partida implementadas são enumerados nas figuras B.7 e B.8.

40. Sobe a escala
    Partindo da NA atual, sobe a escala até próxima NA
41. Desce a escala
    Partindo da NA atual, desce a escala até próxima NA
42. Clichê Miles Davis em 4ª justa
    Apresentado na seção 3.4.5, usando intervalos de 4ª J
43. Clichê Miles Davis em 3ª menor
    O mesmo do anterior, usando intervalos de 3ª M

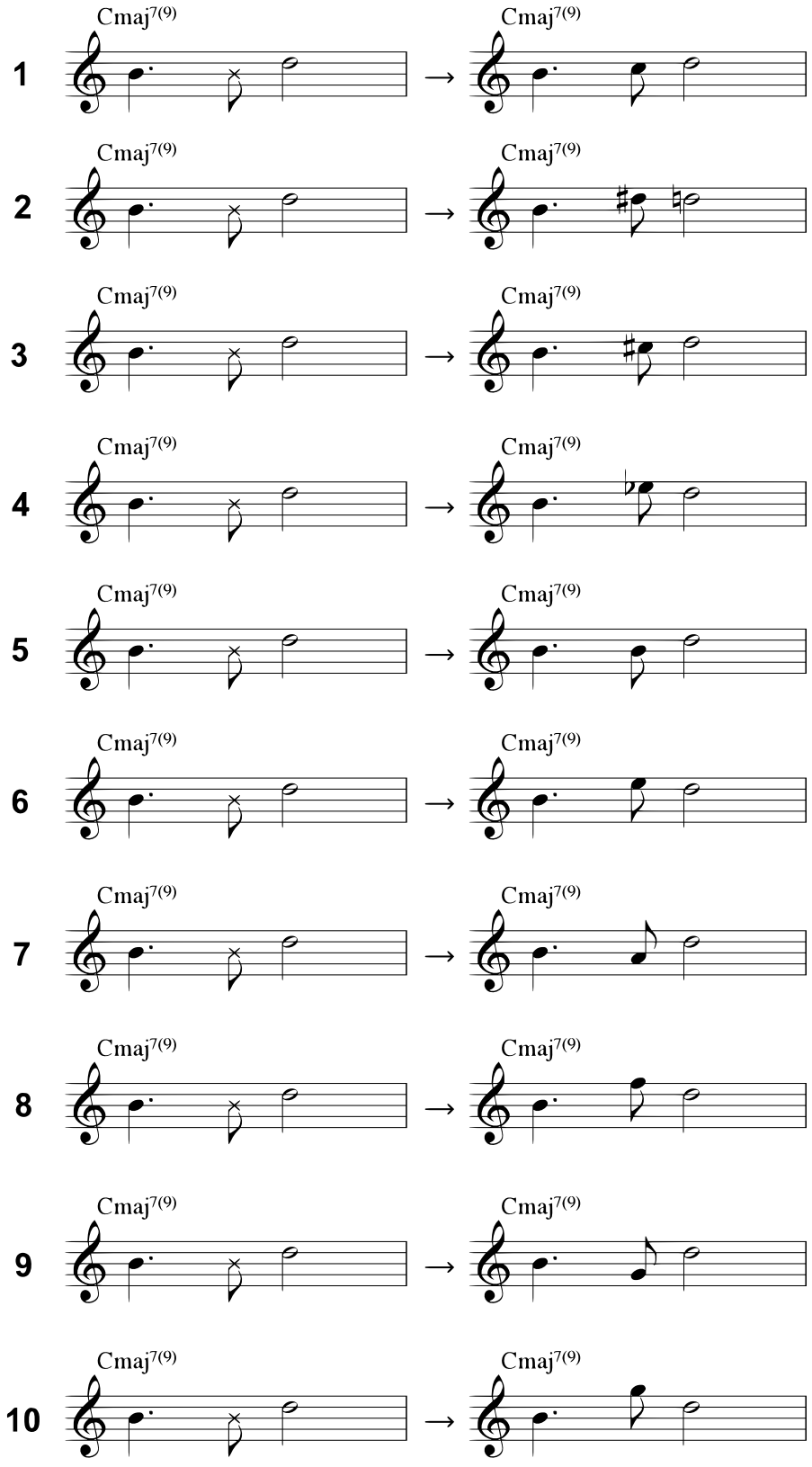

Figura B.3: Exemplos das técnicas de chegada 1 a 10, com 1 NP

44. Frase cromática
Esta é uma frase bastante utilizada em *bebop*, onde a intenção é partir da NA atual e chegar na

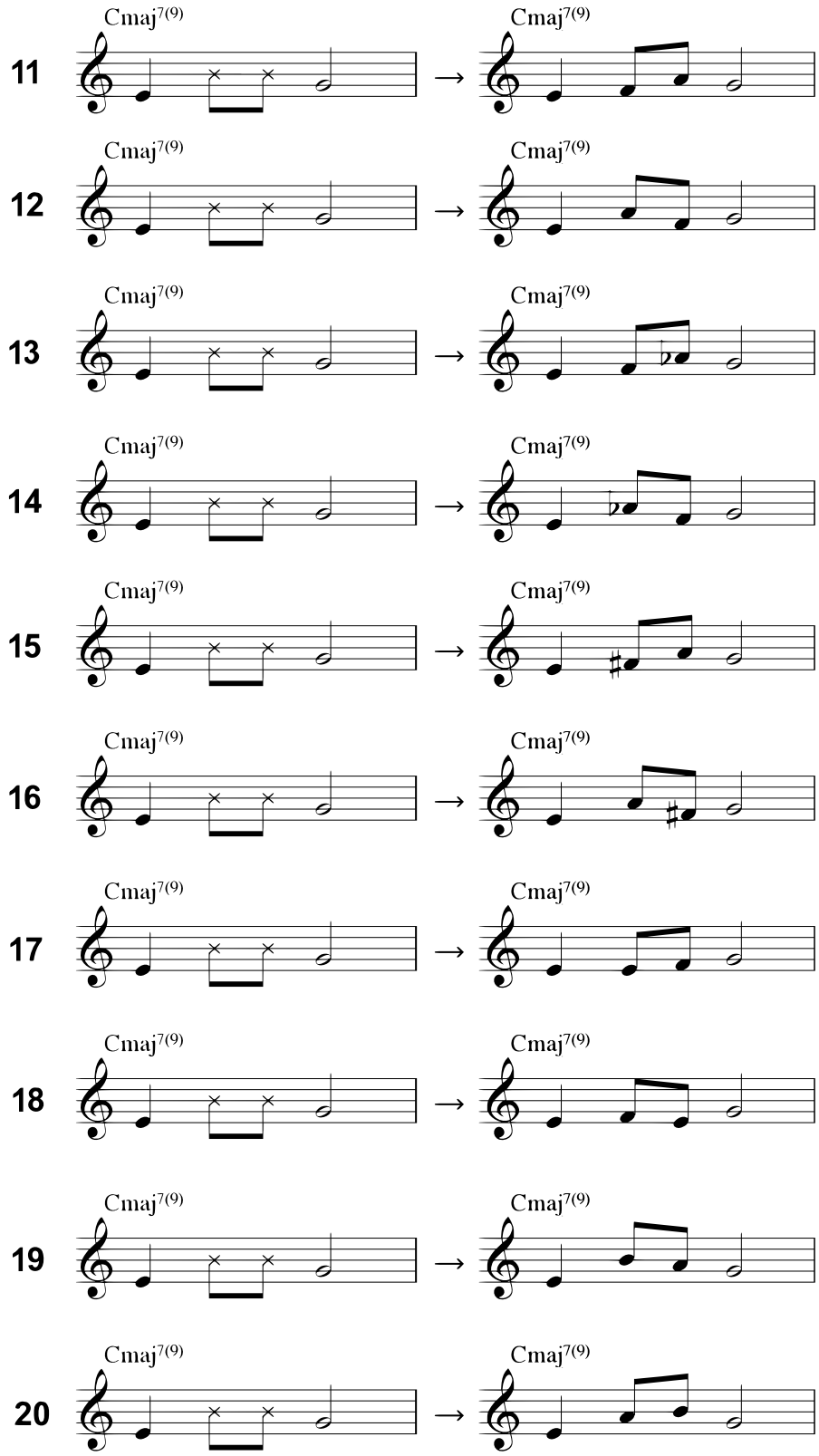

Figura B.4: Exemplos das técnicas de chegada 11 a 20, com 2 NPs

próxima NA descendente, descendo por 2 tons cromáticamente e aproximando-se da próxima NA por uma 2ªD descendente.

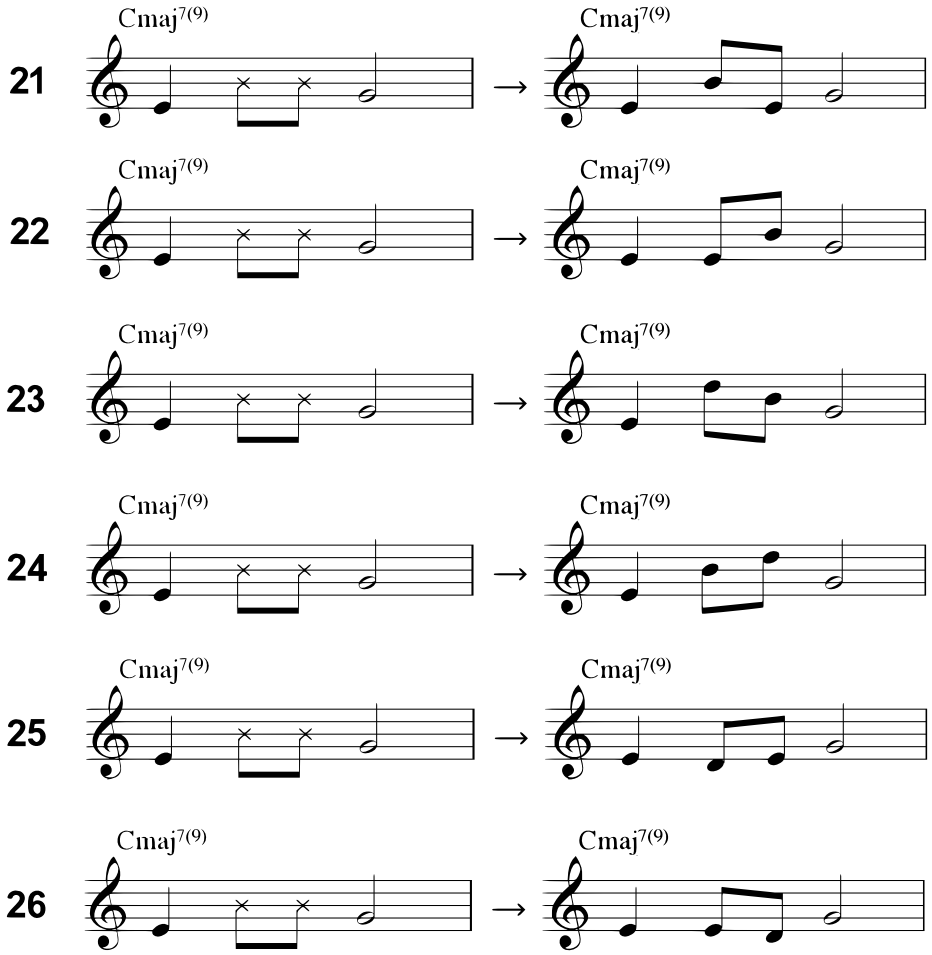

Figura B.5: Exemplos das técnicas de chegada 21 a 26, com 2 NPs

45. Clichê John Coltrane
    Bastante utilizado por Coltrane, este clichê é formado por uma 2ªC descendente em relação à NA atual, depois sobe a escala da NA atual até a próxima NA.

46. Sobe e desce
    Partindo na NA atual, sobe 2 tons da escala de depois volta para a NA atual.

47. Desce e sobe
    Partindo na NA atual, desce 2 tons da escala de depois volta para a NA atual.

48. Arpejo ascendente
    Arpejo ascendente utilizando apenas NAs.

49. Arpejo descendente
    Arpejo descendente utilizando apenas NAs.

50. Arpejo *loopback* oitava acima
    Um tipo de arpejo onde a 1ª nota é a NA atual 1 oitava acima e o arpejo segue descendentemente

51. Arpejo *loopback* oitava abaixo
    Um tipo de arpejo onde a 1ª nota é a NA atual 1 oitava abaixo e o arpejo segue ascendentemente

52. Nota presa em 2ªD descendente
    Uma frase que busca a próxima NA ascendente partindo da NA atual. A primeira nota desta frase é a 2ªD descentende em relação à NA atual, depois a 2ªD ascendente em relação à NA atul, então repete-se a 1ª nota (nota presa) para finalizar com a próxima NA ascendente.

53. Nota presa em 2ªD ascendente em relação à proxima NA
    Mesma frase do item anterior, com a diferença de que a nota presa agora é a 2ªD ascendente em relação à proxima NA ascendente.

54. Direções opostas para NA ascendente
    Partindo da NA atual, as notas desta frase são: 2ªD descendente, 2ªD ascendente, 3ªD descendente e próxima NA ascendente.

55. Direções opostas para NA descendente
    O inverso da frase anterior, ou seja: 2ªD ascendente, 2ªD descendente, 3ªD ascendente e próxima NA descendente.

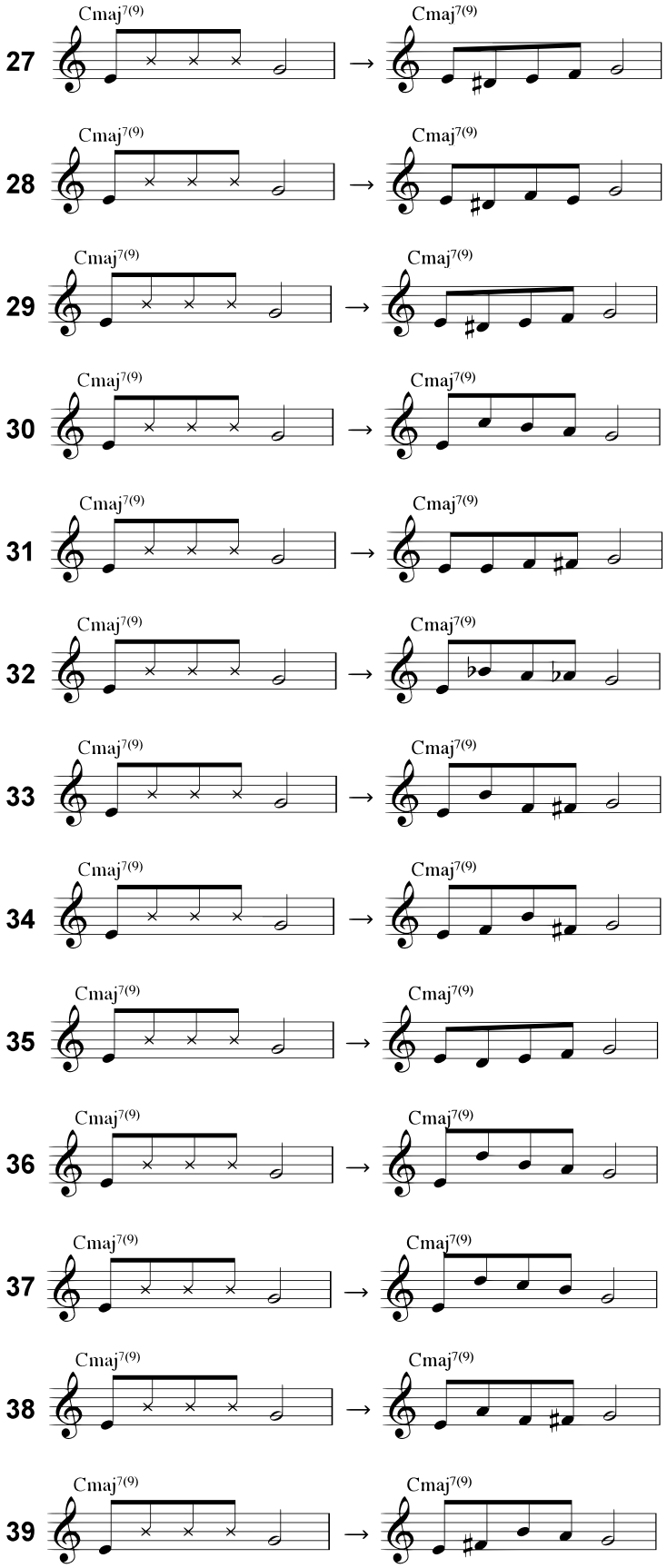

Figura B.6: Exemplos das técnicas de chegada 27 a 39, com 3 NPs

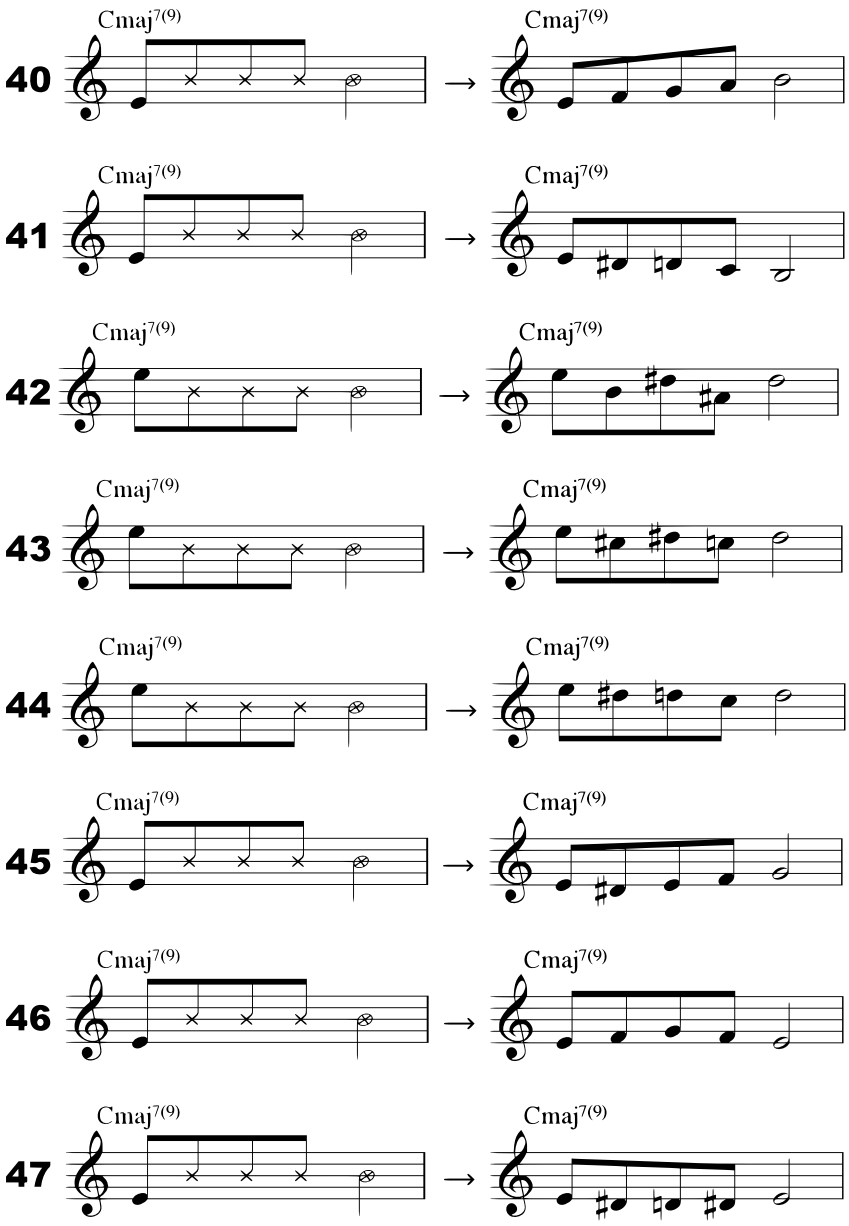

Figura B.7: Exemplos das técnicas de partida 40 a 47

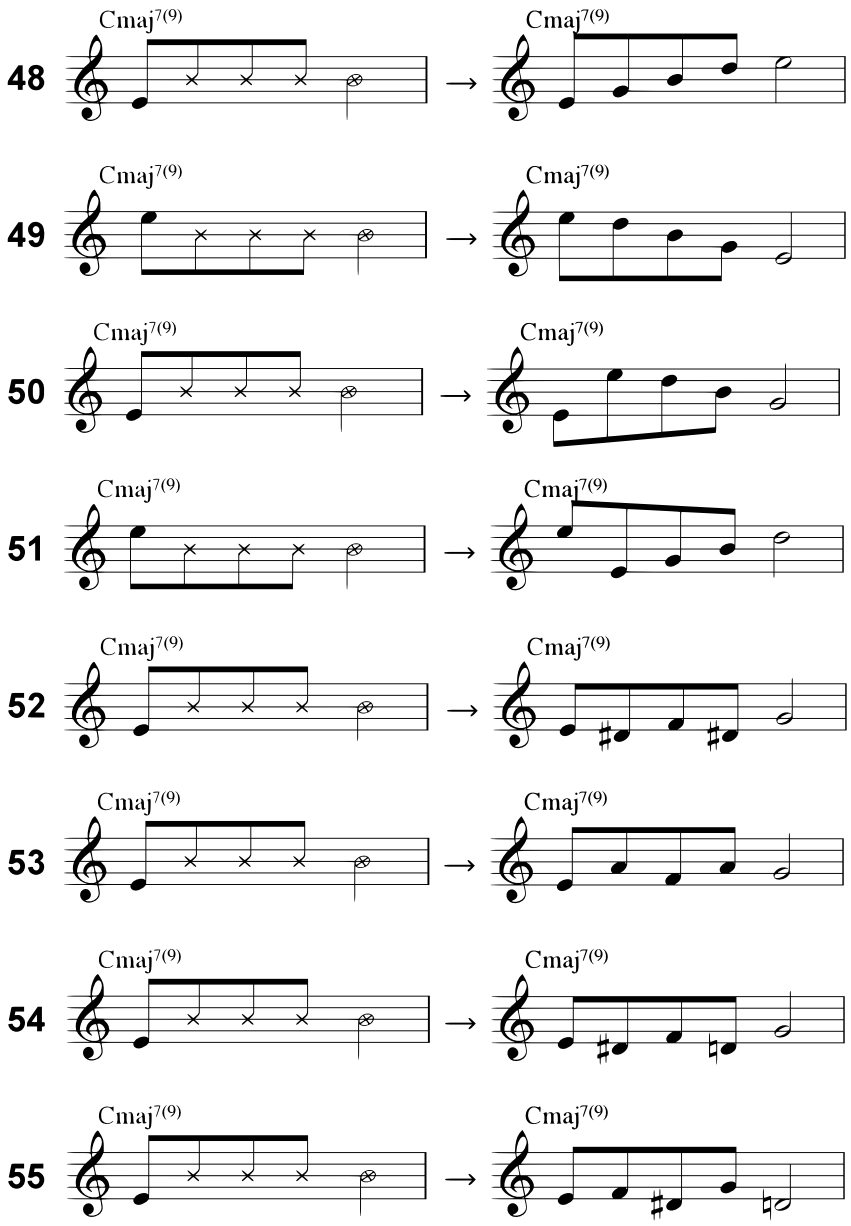

Figura B.8: Exemplos das técnicas de partida 48 a 55