Uma abordagem baseada em programação linear inteira para a geração de solos de guitarra

Nailson dos Santos Cunha, Anand Subramanian
Universidade Federal da Paraíba
Centro de Informática - Campus V, Av. dos Escoteiros - Mangabeira, 58055-000, João Pessoa - PB
nailsoncunha@gmail.com, anand@ct.ufpb.br

Dorien Herremans
Queen Mary University of London
Centre for Digital Music (C4DM) - School of Electronic Engineering and Computer Science -
Mile End Road, London, E1 4NS
d.herremans@qmul.ac.uk

RESUMO

Este trabalho trata da aplicação de métodos de otimização para a composição algorítmica, com foco na geração de solos de guitarra. Foi elaborada uma abordagem semiautomática, pois utiliza pequenos fragmentos melódicos (*licks*) criados a partir de modelos humanos. Os solos gerados possuem características do estilo *Blues* e são aplicados sobre um modelo denominado *12-Bar Blues*. Um banco de *licks* foi criado, onde são realizados sorteios de instâncias menores, diversificando os possíveis candidatos para o solo. É formulado um problema de otimização, que consiste no sequenciamento de forma otimizada de um subconjunto de *licks*, utilizando um modelo de programação linear inteira. Foram implementadas regras para a criação da matriz que define o custo de transição entre os *licks*. Os solos criados foram avaliados por músicos de diferentes níveis. Os resultados obtidos demonstram que estes foram estatisticamente mais bem avaliados que aqueles sequenciados aleatoriamente, possuindo um percentual de aceitação favorável.

PALAVRAS CHAVE. Otimização combinatória, Programação inteira, Solos de guitarra.

Tópicos. Outros, Otimização combinatória

ABSTRACT

This work deals with the application of optimization methods for algorithmic composition, focusing on the generation of guitar solos. A semiautomatic approach was developed, because it makes use of small melodic fragments (licks), created from human models. The solos generated are from the Blues style and they are applied over a model called 12-Bar Blues. A licks database was created, where smaller instances are randomly created from it, diversifying the possible candidates to be in the solo. An optimization problem that consists of determining the optimal sequence of a subset of licks by using a integer linear programming model is formulated. A set of rules was implemented for creating a matrix that defines the transition cost between the licks. The solos created were evaluated by musicians of different levels. The results obtained show that these solos were statistically much better evaluated than those randomly sequenced, with a favorable percentage of acceptance.

KEYWORDS. Combinatorial optimization. Integer programming. Guitar solos.

Paper topics. Others, Combinatorial optimization

1. Introdução

Composição algorítmica (CA) é o processo que envolve o uso de algoritmos para a criação de composições musicais de maneira automática. Tais algoritmos são geralmente desenvolvidos através da identificação e estudo de padrões em métodos tradicionais de composição e suas aplicações na criação de novas músicas [Freitas, 2011]. Alguns desses padrões podem ser implementados em forma de regras que o algoritmo deve seguir para o desenvolvimento de uma nova peça musical. Porém, apenas o uso de regras pode não ser suficiente para criar uma música de "qualidade", resultando algumas vezes apenas em músicas que, apesar de respeitarem algumas regras básicas de teoria musical, não possuem dinâmica, variações e outros elementos que enriquecem a música.

Miranda e Biles [2007] afirmam que os algoritmos para CA são comumente estocásticos, baseados em regras e/ou em inteligência artificial (IA). Em uma visão mais ampla, de acordo com Papadopoulos e Wiggins [1999] e Gonçalves et al. [2009], pode-se citar como principais métodos para a CA: modelos matemáticos, sistemas baseados em conhecimento, gramáticas, métodos evolucionários, sistemas baseados em aprendizado, sistemas distribuídos e sistemas híbridos.

O uso de algoritmos no processo de composição musical está longe de ser uma prática recente. O primeiro exemplo conhecido desta aplicação foi utilizado na peça *Musikalisches Würfelspiel* (Música de Dados), composta por Wolfgang Amadeus Mozart no século XVIII, de modo que os resultados obtidos jogando dados direcionavam a escolha de pequenas seções da música e as variações que seriam aplicadas e que juntas resultariam na peça musical completa [Feijão, 2004].

Já a composição auxiliada por computador (CAC, do inglês *Computer-aided composition*) é uma área de pesquisa relativamente nova [Herremans e Sörensen, 2012]. Os trabalhos de Hiller e Isaacson [1957, 1958] são considerados os primeiros a simular o processo de composição musical através de uma abordagem baseada em regras [Sandred et al., 2009].

Muitas pesquisas têm sido realizadas ultimamente nas áreas de CA e CAC. Grande parte envolve o uso de heurísticas e de meta-heurísticas aplicadas á criação de novas composições musicais, buscando, na maioria das vezes, alcançar uma realidade de composição próxima aos modelos usados pelos humanos, isto é, de tal forma que as composições geradas por algoritmos não careçam de dinamicidade e/ou criatividade. Várias técnicas podem ser aplicadas para se alcançar este objetivo. Langston [1989] descreve seis técnicas que foram utilizadas em um projeto onde era possível ouvir uma música gerada por computador através de uma ligação via o sistema de telefonia público.

Este trabalho apresenta a aplicação de uma abordagem de otimização em um *framework* com o objetivo de compor solos que tenham características do instrumento guitarra. O método desenvolvido modela a criação de solos de guitarra como um problema de otimização combinatória em conjunto com uma técnica conhecida na literatura como "*Riffology*" [Langston, 1989]. Tal técnica tem como unidade básica de construção o "*riff*", podendo também ser conhecido como "*lick*"ou "frase", que é um fragmento melódico pequeno, baseado em um modelo humano, aplicado na construção de solos e melodias.

A abordagem utilizada foi considerada semiautomática devido ao fato de utilizar como unidade fundamental para a construção dos solos *licks* já disponíveis criados a partir de modelos humanos. Entretanto, a escolha preferencial por *licks* de tamanhos maiores que dois compassos, fazendo assim a subdivisão desses *licks* em vários outros menores, de apenas um compasso, tornou possível a criação de inúmeras melodias que diferem bastante dos *licks* originais completos.

O trabalho está organizado como segue. Na seção 2, uma breve revisão da literatura explorando os trabalhos relacionados na área de CAC que fazer uso de técnicas de otimização, bem como heurísticas e metaheurísticas. Na seção 3 é realizada a apresentação do problema, bem como sua modelagem como um problema de otimização combinatória e a metodologia aplicada para a sua resolução. Na seção 4 são apresentados os resultados obtidos através do experimento. Por fim, a seção 5 traz algumas considerações finais a respeito do que foi apresentado neste trabalho.

2. Trabalhos Relacionados

Grande parte dos trabalhos realizados em CA faz uso de métodos baseados em computação evolutiva (CE), que são estratégias computacionais que aplicam os conceitos de evolução na criação de inteligência artificial, principalmente através do uso de algoritmos genéticos (AG). Miranda e Biles [2007] afirmam que essa área ficou conhecida como música evolutiva (ME). Uma aplicação de AG na criação de melodias é o GenJam (*Genetic Jammer*), criado por Biles [1994]. O GenJam é um AG desenvolvido para gerar solos de *Jazz* que, segundo seu criador, segue um modelo baseado na forma em que músicos de *Jazz* iniciantes aprendem a improvisar. Para criar a improvisação, o sistema utiliza alguns dados pré-definidos que são: tonalidade, tempo, ritmo, número de compassos e a progressão de acordes. Existe um mapeamento acorde/escala que determina as notas seguras de se usar para criar o improviso sobre determinado acorde. A avaliação do *fitness*[1] dos indivíduos gerados é feita de forma interativa, ou seja, através intervenção humana avaliando o solo criado, se este foi bom será utilizado para a criação de novos indivíduos, se foi ruim será descartado. Isso é o que o autor denomina de gargalo de *fitness* (*fitness bottleneck*).

Bäckman [2010] se baseou em métodos de *swarm intelligence*, que é uma disciplina de IA interessada na concepção de sistemas multi-agentes inteligentes, inspirando-se no comportamento coletivo dos insetos e animais sociais [Blum e Li, 2008], juntamente com uma variação do problema do caixeiro viajante (PCV), um problema clássico de otimização bastante conhecido e estudado na literatura, para criar solos de *Jazz* improvisados. O autor considerou as notas como os pontos a serem visitados durante o *tour* do caixeiro, sendo permitido visitar uma mesma nota mais de uma vez, algumas notas podendo não ser visitadas e também sem a necessidade de retornar á primeira nota. Este solo criado através do processo descrito anteriormente serve como entrada para um algoritmo evolucionário, baseado na meta-heurística colônia de formigas (ACO, do inglês *Ant Colony Optmization*), que faz seu aprimoramento. Um detalhe desta implementação foi o uso de uma lista tabu para permitir a repetição de algumas notas de acordo com regras definidas pelo autor. Esse trabalho é parte de um projeto maior, intitulado *Evolutionary Jazz Improvisation* que foca no uso de algoritmos evolucionários com avaliação automática de *fitness* aplicados á criação de improvisos de harmonias e melodias de *Jazz* [Bäckman, 2009, 2012, 2013].

Herremans e Sörensen [2012] desenvolveram um algoritmo baseado na meta-heurística *variable neighbourhood search* (VNS) para a composição automática de contraponto, que é a combinação de melodias ocorrendo simultaneamente de acordo com um conjunto de regras [Sadie e Tyrrell, 2001], de primeira espécie. A escolha do tipo de composição foi devido ao fato de existirem regras formais, escritas, que os autores consideraram não muito difíceis de quantificar e também por ser o contraponto de primeira espécie o mas restritivo das cinco espécies existentes. Em um trabalho seguinte Herremans e Sörensen [2013] conseguiram aprimorar seu algoritmo para gerar até a quinta espécie de contraponto.

Tanaka e Fujii [2015] apresentaram uma proposta com o intuito de criar composições musicais através do uso de programação inteira. Em seu estudo, eles consideram uma música como sendo sequências de padrões musicais, tais como padrões de acordes, padrões rítmicos e padrões melódicos. Como objetivo principal, foi abordado um modelo que pode dar especificações de características das estruturas musicais globais no contexto de geração de musical, podendo assim ajudar usuários a criar novas peças musicais e/ou obter as estruturas musicais desejadas. Com base nesse objetivo, eles realizaram um estudo sobre a peçã musical Op. 101 No.74 de Ferdinand Beyer, relacionando como os padrões de composição nela encontrados poderiam ser representados como restrições de programação inteira. No mesmo trabalho também foi realizada a análise de padrões rítmicos sobre a famosa melodia *Ode to Joy* de Beethoven. Entretanto, não houve uma implementação prática dos conceitos apresentados.

[1] *Fitness*, ou função de aptidão, são valores usados em um processo de seleção natural para escolher quais possíveis soluções ou indivíduos vão continuar para a próxima geração e quais serão descartados [Goldberg, 1989].

3. Definição e modelagem do problema

A metodologia que foi empregada consistiu em considerar a criação de solos como um problema de otimização combinatória, possibilitando a resolução do problema por meio da aplicação de um método exato.

Com o objetivo de reduzir o escopo no contexto musical, para este trabalho foi escolhido criar solos no estilo *Blues* utilizando uma harmonia de um modelo de *12-Bar Blues* como base.

Foi criado um banco de dados de *licks* de guitarra *Blues* onde os *licks* que compuseram este banco possuem o tamanho de 1 compasso obtidos como fragmentos de *licks* de tamanho maior, geralmente entre 3 e 5 compassos. Isto também possibilitou tirar proveito do fato de neles já estarem encapsuladas algumas das principais técnicas aplicadas à guitarra como, por exemplo, *bend*, *slide*, *hammer on* e *pull off*.

A estratégia adotada para a criação dos solos é através do sequenciamento de subconjuntos de *licks* sorteados do banco de dados criado. Dessa forma, foram definidas regras que ditam o que é uma boa transição entre dois *licks*. Estas regras foram inferidas empiricamente, após a audição e análise de uma grande quantidade de solos de *Blues*, observando características em comum possíveis de se representar em forma de regras. Para tal, foi adotada uma abordagem minimalista com a programação de poucas regras, porém que cobrissem o que foi catalogado durante a análise de solos, de modo a evitar que a utilização de muitas regras faça com que os solos criados deixem de pertencer a um conjunto mais generalizado e passem a ser de gosto mais específico ou mais pessoal.

O conjunto de regras implementadas pode ser dividido em dois grupos: individuais, Tabela 1, e de transição, Tabela 2. As regras individuais dizem respeito áquelas que consideram os *licks* individualmente e isso possibilitou que elas fossem programadas diretamente como restrições no modelo matemático. As regras de transição são aquelas que consideram sempre pares de *licks*, bonificando ou penalizando a ocorrência do *lick* B após o *lick* A, levando em conta características importantes de ambos. Tais regras foram aplicadas na criação, em tempo de execução, da matriz de penalidades de um subconjunto *licks* sorteado do banco de dados. Para a criação da matriz de transições, bonificar significa diminuir o custo de transição e penalizar significa aumentar o custo.

Tabela 1: Regras individuais implementadas

Regra	Descrição	Quantidade
1	Utilização de *licks* do tipo *turnaround*	$= 1$
2	Utilização de *licks* de repetição	$= 1$
3	Utilização de *licks* iniciados e/ou terminados em pausa	≤ 3

Tabela 2: Regras de transição implementadas

Regra	Transiccão	Tipo	Valor
1	De qualquer *lick* para um *lick* de repeticcão	bônus	-50
2	De *lick* de repeticcão para qualquer outro *lick*	bônus	-50
3	De *lick* não terminado em pausa para qualquer outro iniciado em pausa $\leq$ a 1 tempo	bônus	-15
4	De *lick* terminado em pausa $\leq$ a 1 tempo para qualquer outro não iniciado em pausa	bônus	-15
5	De qualquer *lick* A para qualquer *lick* B onde a primeira nota de B seja a imediata anterior ou posterior a última nota de A na escala pentatônica (maior ou menor)	bônus	-15
6	De qualquer *lick* A para qualquer *lick* B onde a primeira nota de B seja igual a última nota de A	bônus	-15
7	De qualquer *lick* que termine em pausa $>$ 1 tempo para outro que comece com pausa ¿ 1 tempo	penalidade	+25
8	De qualquer *lick* que termine em pausa $>$ 2 tempos para qualquer outro	penalidade	+25
9	De qualquer *lick* para outro que iniciado em pausa $>$ 2 tempos	penalidade	+25
10	De qualquer *lick* para qualquer outro do tipo *turnaround*	penalidade	+100000

Todas as transições da matriz são inicializadas com o valor 100. Em seguida, é realizada uma varredura completa dos *licks*, em pares, e verificado se a transição entre eles está de acordo com alguma das regras, caso sim, é subtraída sua bonificação ou adicionada sua penalidade. Estes valores foram definidos após vários testes com diferentes custos e audições dos solos gerados, de modo que a seleção dos *licks* não fique tendenciosa, existindo assim um equilíbrio entre os custos das transições.

3.1. Conceitos e terminologias

- *Blues* – é um estilo musical nascido no sul dos EUA sem uma data de origem precisa, mas consolidado mais fortemente durante as décadas de 1920 e 1930, período de atuação de grandes intérpretes como Charley Patton, Son House, Tommy Johnson e Robert Johnson [Carrijo, 2014]. Seu som, em sua origem, é bastante associado a um estado de depressão e melancolia. Segundo Lewis et al. [2005], *Blues* é um gênero musical que possui uma grande probabilidade de produzir uma emoção específica às pessoas familiarizadas com a cultura ocidental.
- *12-Bar Blues* – (*Blues* doze compassos), em sua forma mais básica, é um modelo de *Blues* que utiliza em sua progressão três acordes principais, que são os acordes I, IV e V da escala referente a tonalidade da música. Sua progressão de acordes preenche ciclos de doze compassos, daí a origem do nome [Krenz, 2009].
- Compasso – é a forma de divisão da música em pequenas partes, sendo que estas subdivisões podem ou não ter a mesma duração.
- *Licks* com pausas – A pausa é uma característica bastante presente em solos e improvisos de *Blues*. Pausas curtas causam, comumente, uma boa impressão de mudança de contexto no decorrer da melodia e pausas pr olongadas podem causar uma sensação de frustração ou expectativa que não se mostram tão interessantes na maioria dos casos.
- *Licks* de repetição – são *licks* que se baseiam na repetição de figuras melódicas mostrando-se como uma maneira bastante efetiva de construir e manter momentos importantes e/ou de grande intensidade na música. Foi percebido que tais *licks* podem ser aplicados como uma espécie de "coringa", de modo que sua ocorrência em qualquer momento do solo é geralmente uma boa alternativa.
- *Licks turnaround* – em termos de *Blues*, *turnaround* refere-se a figura musical tocada, normalmente, sob os acordes dos dois últimos compassos finalizando um ciclo de acordes, preparando para o início de um novo ciclo ou para o término da música.

3.2. Formulação matemática

O problema pode ser definido sobre um grafo $G = (L, A)$, em que L representa o conjunto de *licks* e A o conjunto de arcos conectando dois *licks*, tal que $A = \{(i, j) \mid i \neq j\}$. Seja p_{ij} a penalidade de transição entre o *lick* i e o *lick* j, c_i a quantidade de compassos do *lick* i, NC uma constante que define a quantidade de compassos disponíveis para o solo, LR uma constante que define a quantidade de *licks* de repetição que podem ser selecionados para o solo, LP uma constante que define a quantidade de *licks* que iniciam e/ou terminam em pausa que também podem ser selecionados para o solo, R o conjunto de *licks* de repetição, T o conjunto de *licks turnarounds*, P o conjunto de *licks* que se iniciam e/ou terminam em pausa. Defina x_{ij} como sendo uma variável binária que assume valor 1 se o arco $(i, j) \in A$ for utilizado e 0, caso contrário e y_i como sendo uma variável auxiliar que assume o valor 1 caso o *lick* $i \in L$ tenha sido utilizado e 0, caso contrário.

A formulação matemática proposta para o problema foi a seguinte:

$$\min \sum_{i \in L} \sum_{j \in L} p_{ij} x_{ij} \tag{1}$$

$$\text{s.a.} \quad \sum_{j \in L} x_{ij} = y_i \qquad \forall i \in L \tag{2}$$

$$\sum_{i \in L} x_{ij} = y_j \qquad \forall j \in L \tag{3}$$

$$\sum_{i \in L} c_i y_i = NC \tag{4}$$

$$\sum_{i \in R} y_i \leq LR \tag{5}$$

$$\sum_{i \in P} y_i \leq LP \tag{6}$$

$$\sum_{i \in T} x_{i0} = 1 \tag{7}$$

$$x_{ij} + x_{ji} \leq 1 \qquad \forall i \in L,\ \forall j \in L\ ,\ i < j \tag{8}$$

$$\sum_{i \in L} \sum_{j \in \overline{L}} x_{ij} \geq 1 \qquad \forall L \subseteq V',\ |S| > 2 \tag{9}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in L,\ \forall j \in L \tag{10}$$

$$y_i \in \{0, 1\} \qquad \forall i \in L \tag{11}$$

A função objetivo (1) minimiza o custo de transição entre os *licks*. As restrições (2) e (3) estabelecem, respectivamente, que caso o *lick* tenha sido utilizado, nele chega apenas um arco e dele sai apenas um arco. A restrição (4) define que o total da soma dos compassos dos *licks* utilizados tem que ser igual a quantidade de compassos disponíveis para o solo. A Restrição (5) assegura que o solo conterá, no máximo, LR *licks* de repetição. A restrição (6) garante que no existirão máximo, LP *licks* que iniciam e/ou finalizam em pausa. A restrição (7) estabelece a obrigatoriedade de utilização de um *lick* do tipo *tunaround*. As restrições (8) e (9) eliminam a ocorrência de *subtours*. Como existe um número exponencial de restrições (9), elas são adicionadas sob demanda através de um procedimento de geração de cortes. Por fim, (10) e (11) definem a natureza das variáveis.

Vale ressaltar que o problema aqui definido pode ser visto como sendo um Problema de Caminho Mínimo com Restrição de Recursos (SPPRC, do inglês *Shortest Path Problem with Resource Constraints*) [Irnich et al., 2005], considerado fortemente $\mathcal{NP}$-difícil. A visualização da geração de solos como um SPPRC está ilustrada na Figura 1.

Na representação, têm-se que os nós equivalem aos *licks* e os compassos referem-se a janela de tempo na qual os *licks* podem ser adicionados a solução. O *lick* "d" equivale a um *lick dummy*, que é utilizado como ponto de partida e de finalização do solo. Os nós com preenchimento listrado são representações de *licks* de repetição, que, assim como os *licks* comuns, podem ser selecionados em qualquer momento do solo. Os nós de cor cinza representam os *licks turnaround*, sendo que eles preenchem de 2 compassos e só podem ocorrer no final de cada ciclo de 12 compassos, ocupando os dois últimos compassos dos solos. O caminho encontrado como solução é ilustrado pelos nós preenchidos em preto, conectados pelos arcos que contêm os respectivos custos da penalidade de transição entre os *licks*.

A formulação matemática, bem como o procedimento de geração de cortes de *subtour*, foram implementados na linguagem programação C++ utilizando para sua resolução a biblioteca *concert* do *solver* CPLEX versão 12.4, que por sua vez faz uso de um algoritmo de B&C para resolução de problemas de programação linear inteira.

3.3. *Framework Proposto*

O *framework* proposto neste trabalho divide em três etapas o processo necessário para a criação de solos de guitarra, a saber: pré-processamento, otimização e pós-processamento. A visualização completa da arquitetura do *framework* é exibida na Figura 2.

Na Etapa 1 são recebidas como entrada informações referentes ao tamanho da instância que será sorteada, velocidade da música e a quantidade de compassos disponíveis para o solo.

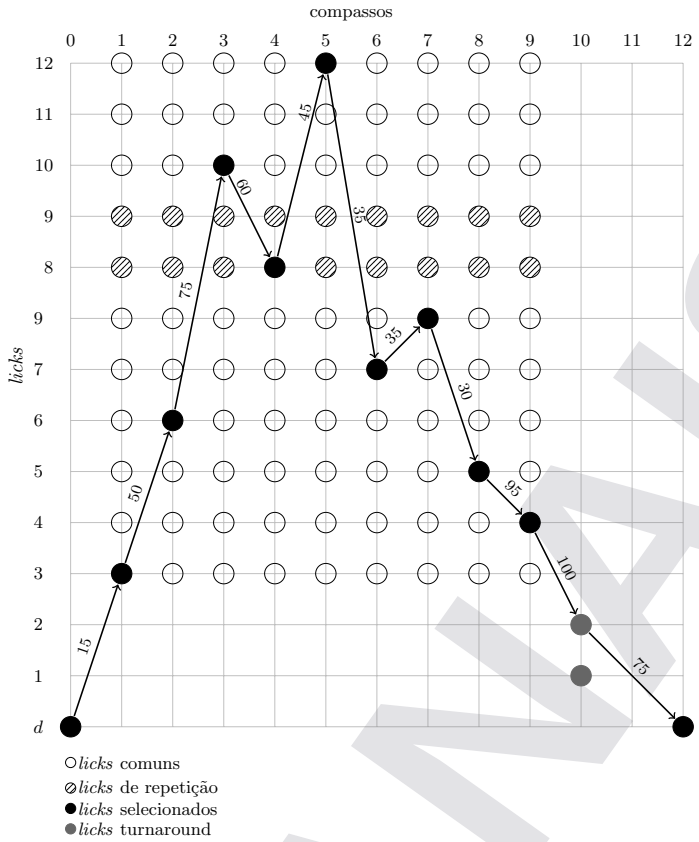

Figura 1: Visualização da geração de solos como variante do SPPRC

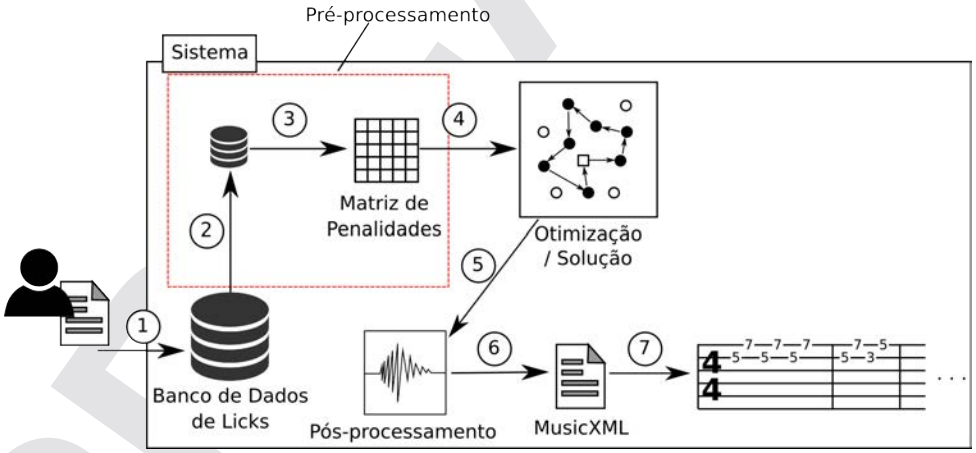

Figura 2: Fluxograma de funcionamento do sistema

Na Etapa 2 ocorre o sorteio do subconjunto de *licks*, de acordo com o parâmetro definido na inicialização. O sistema coleta aleatoriamente do banco de dados a quantidade de *licks* que serão candidatos a estarem na solução. Na Etapa 3 é gerada a matriz de penalidades referente ao subconjunto de *licks* sorteados. A Etapa 4 é onde o otimizador, fazendo uso da matriz de penalidades e das informações recebidas no primeiro passo, gera a solução encontrada para aquela instância dos *licks* escolhidos na Etapa 2. Na Etapa 5, já com os valores da solução, opcionalmente pode ser

feito o pós-processamento e refinamento do solo, ajustando e removendo as tensões indesejadas. Na Etapa 6, é exportada a saída da solução no formato MusicXML e, finalmente, a Etapa 7 a ilustra no formato de tablatura.

4. Resultados

O *framework* desenvolvido através da abordagem de otimização foi programado utilizando a linguagem *Python* 2.7 e a biblioteca *lxml* Versão 3.4 para a manipulação de textos e arquivos no formato XML. É importante frisar que durante a fase de otimização o tempo médio de execução do *solver* para resolver as instâncias utilizadas na geração dos solos, contendo cerca de 160 *licks*, é de aproximadamente 4,3 segundos.

Um dos objetivo da avaliação dos resultados é mostrar que os solos gerados através da otimização da sequência de *licks* soam mais interessantes que uma sequência aleatória e também avaliar a possível aceitação de uso dos solos, seja na criação de novas ideias e improvisos ou no seu uso em composições musicais.

Desta forma, a estratégia adotada para a avaliação dos solos foi submetê-los a um teste de audições, em que o ouvinte classifica o quanto gostou, ou acha que o solo está bom, usando o modelo de escala de Likert [Albaum, 1997].

A abordagem completa aconteceu da seguinte forma: foi desenvolvido um sistema em que músicos e/ou entusiastas pudessem fazer a audição dos solos via Internet; foram inseridos no sistema solos criados aleatoriamente e solos criados pelo *framework*; cada usuário cadastrado para votação recebeu seis solos para avaliar, sendo três aleatórios e três não-aleatórios, porém o ouvinte não era informado sobre isso; após um solo ser avaliado o próximo é selecionado randomicamente, podendo ser escolhido de qualquer um dos dois tipos; por fim, o usuário responde sim ou não a uma questão com o intuito de saber se ele concorda que o solo poderia ser aplicado em uma música ou apresentação ao vivo. Participaram desta etapa um total de 173 usuários avaliadores, que foram divididos em três grupos com base no nível musical em que eles se enquadram, sendo 75 iniciantes, 51 intermediários e 47 profissionais.

A primeira forma de verificação dos resultados das avaliações dos solos foi através da contabilização total de cada opção da escala de Likert, dividindo os solos em dois grupos, aleatórios e otimizados, sendo que o termo "otimizados" é apenas para explicitar que os solos foram gerados pelo *framework* desenvolvido. Nesta visualização, não há diferenciação entre os níveis dos avaliadores.

É possível perceber, através da análise dos gráficos mostrados nas Figuras 3 e 4, que os solos otimizados tiveram uma classificação melhor que os solos aleatórios. Na Figura 3 as parcelas que correspondem as avaliações que vão de terrível até moderado correspondem a $51,26\%$ do gráfico, enquanto que estas mesmas parcelas, para os otimizados, correspondem a apenas $36,43\%$, restando assim, aproximadamente, $63,57\%$ para as avaliações bom, muito bom e excelente.

A Tabela 3 mostra os valores da contabilização referentes aos solos gerados aleatoriamente em comparação aos solos otimizados, gerados pelo *framework* e que foram submetidos a avaliação dos músicos.

Tabela 3: Contabilização geral em "%"das avaliações dos solos

	Terrível	**Muito ruim**	**Ruim**	**Moderado**	**Bom**	**Muito Bom**	**Excelente**
Aleatórios	3,66	9,06	17,15	21,39	24,86	16,96	6,94
Otimizados	0,39	5,04	8,33	22,67	25,58	23,84	14,15

Outra parte da análise dos resultados do experimento foi através da comparação das médias dos solos aleatórios e otimizados. Essas diferenças foram submetidas ao testes de normalidade de Shapiro e Wilk [1965], Anderson e Darling [1954] e Lilliefors [1967] para verificar se

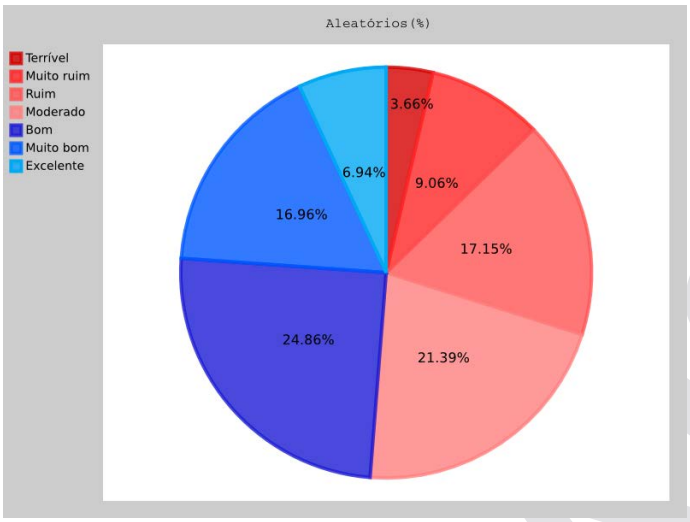

Figura 3: Visão geral. Solos aleatórios

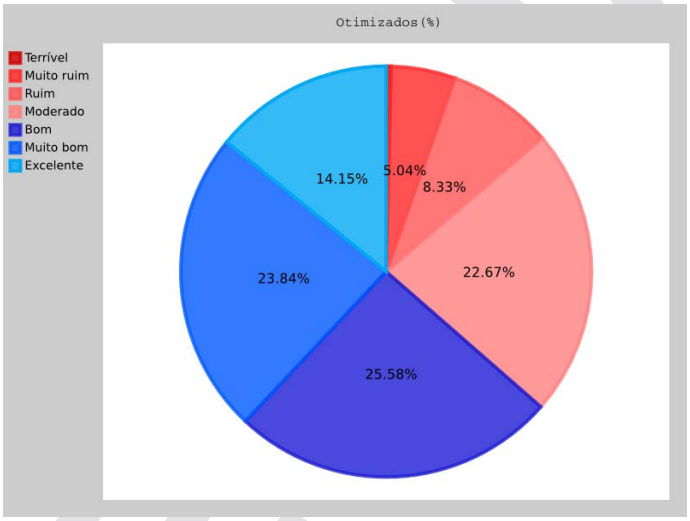

Figura 4: Visão geral. Solos otimizados

distribuição de probabilidade associada poderia ser aproximada pela distribuição normal. O resultado dos testes, conforme exibidos na Tabela 4, com nível de significância de 5%, mostra que os dados não seguem a distribuição normal.

Tabela 4: Resultado dos testes de normalidade

Teste	**p-Valor**
Shapiro-Wilk	0,03186
Anderson-Darling	0,002275
Kolmogorov-Smirnov	0,00005665

Uma vez que os dados não estão normalizados torna-se mais adequado a aplicação de um teste não-paramétrico, como é o caso do *Wilcoxon signed-rank test* [Woolson, 2008]. O resultado do teste, mostrado na Tabela 5, com nível de significância de 1%, mostrou que existe evidência de diferença estatística entre as amostras.

Tabela 5: Resultado do *Wilcoxon signed-rank test*

	p-Valor
Iniciantes	0,00005985
Intermediários	0,0001161
Profissionais	0,0000654

Os resultados dos testes confirmaram que, perceptivelmente, os solos otimizados foram considerados mais "agradáveis" do que os aleatórios, segundo as avaliações dos ouvintes participantes. De acordo com a avaliação das pontuações médias de cada músico, o p-*value* resultante, com nível de significância de 1%, indica que os solos otimizados tendem a ter, significativamente, uma melhor classificação em relação aos solos aleatórios.

Para cada solo de guitarra posto para a avaliação também foi computada a frequência com que a opção "Este solo poderia ser usado em uma música ou em uma *performance* ao vivo" foi marcada como "sim". O *Wilcoxon test*, obteve o resultado de $p = 0,0027$ mostrando novamente que existe diferença e relevância estatística entre os dados da amostra. Sendo assim têm-se que a média da distribuição da frequência de uso dos solos, otimizados e aleatórios, exibida na Figura 5, é significativamente diferente, de tal maneira que os solos otimizados são selecionados como possíveis de utilizar em uma média de 81% contra 68% para os solos aleatórios.

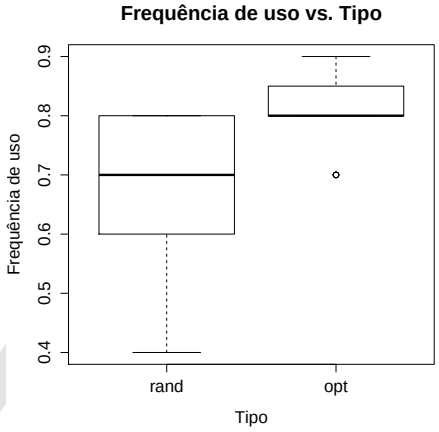

Figura 5: *Boxplot* da frequência de uso dos solos

5. Considerações Finais

Modelar a criação de solos de guitarra como um problema de otimização combinatória e resolvê-lo através da aplicação de programação matemática e técnicas de otimização mostrou-se um caminho bastante promissor para a CAC. As melodias geradas através da utilização dessa abordagem, foram avaliadas de forma favorável, de acordo com as classificações dadas pelos músicos que participaram da avaliação. O resultado do experimento mostrou uma melhor aceitabilidade dos solos otimizados do que os solos criados de maneira aleatória, mesmo ambos tendo sido criados a partir da mesma instância de um subconjunto do banco de dados de *licks*. Tal fato mostra como o foco na maneira como os *licks* são concatenados têm uma grande importância na criação dos solos, não bastando apenas que eles estejam na mesma tonalidade ou simplesmente sigam a mesma escala.

Vale ressaltar que a influência exercida pelas regras de transição nos solos pode sofrer algum impacto por parte das características dos *licks* que populam o banco de dados. Por exemplo, foi definido um pequeno conjunto de regras de transição, dentre elas a utilização de pequenas pausas e notas próximas entre a passagem de um *lick* para outro. Portanto, se os *licks* que compõem o banco

de dados, em sua natureza, possuem essas características, torna-se difícil perceber a diferença entre um solo com sequência aleatória e outro sequenciado de forma otimizada.

O solos gerados e submetidos à avaliação estão disponíveis no endereço http://nailsoncunha.net.

Agradecimentos

Este projeto foi parcialmente financiado pelo programa de pesquisa e inovação *European Union's Horizon 2020*, processo no. 658.914. Os autores também agradecem a colaboração dos músicos Zé Filho e Marcos Rosa.

Referências

Albaum, G. (1997). The likert scale revisited: an alternate version. *Journal of the Market Research Society*, 39(2):331–348.

Anderson, T. W. e Darling, D. A. (1954). A test of goodness of fit. *Journal of the American statistical association*, 49(268):765–769.

Bäckman, K. (2009). Automatic jazz harmony evolution. In *SMC: Proceedings of the 6th Sound and Music Computing Conference 23-25 July 2009 Casa da Música, Porto-Portugal*, p. 349–354.

Bäckman, K. (2010). Ant colony optimization and evolutionary algorithms applied to jazz solo improvisation. *Encontro Internacional de Música e Artes Sonoras (EIMAS)*.

Bäckman, K. (2012). Automatic fitness in generative jazz solo improvisation. In *EIMAS 2012 Conference Proceedings*, p. 1–10.

Bäckman, K. (2013). Evolutionary jazz improvisation-jazz harmony and solo improvisation created by means of evolutionary algorithms with automatic fitness. *Sport and Art*, 1(3).

Biles, J. A. (1994). Genjam: A genetic algorithm for generating jazz solos. *International Computer Music Conference*.

Blum, C. e Li, X. (2008). *Swarm intelligence in optimization*. Springer.

Carrijo, D. D. P. (2014). Cinema & blues: Representações audiovisuais do gênero no século xxi.

Feijão, P. C. (2004). Um algoritmo de criação de improvisos com harmonia de jazz.

Freitas, A. R. R. (2011). Música evolutiva: uma abordagem computacional para composição algorítmica.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition. ISBN 0201157675.

Gonçalves, A., Lopes, E., e Paiva, A. (2009). The music of paintings: a rhythmic perspective. *12th Generative Art Conference GA2009*.

Herremans, D. e Sörensen, K. (2013). Composing fifth species counterpoint music with a variable neighborhood search algorithm. *Expert Systems with Applications*, 40(16):6427 – 6437. ISSN 0957-4174. URL http://www.sciencedirect.com/science/article/pii/S0957417413003692.

Herremans, D. e Sörensen, K. (2012). Composing first species counterpoint with a variable neighbourhood search algorithm. *Journal of Mathematics and the Arts*, 6(4):169–189. URL http://dx.doi.org/10.1080/17513472.2012.738554.

Hiller, L. e Isaacson, L. (1957). *Illiac Suite*. Score, Theodore Presser Co, New York, USA.

Hiller, L. e Isaacson, L. (1958). Musical composition with a high-speed digital computer. *Journal of Audio Engineering Society 6*.

Irnich, S., Desaulniers, G., et al. (2005). Shortest path problems with resource constraints. *Column generation*, 6730:33–65.

Krenz, S. (2009). *Blues Guitar*. Legacy Learning Systems.

Langston, P. (1989). Six techniques for algorithmic music composition. In *15th International Computer Music Conference*. Citeseer.

Lewis, R., Wieczorkowska, A., e Ras, Z. W. (2005). Pentatonic harmonics in fourier transforms: Why the blues are blue.

Lilliefors, H. W. (1967). On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, 62(318):399–402.

Miranda, E. R. e Biles, J. A. (2007). *Evolutionary computer music*. Springer-Velarg. ISBN 1-84628-599-2.

Papadopoulos, G. e Wiggins, G. (1999). AI methods for algorithmic composition: A survey, a critical view and future prospects. *In AISB Symposium on Musical Creativity*.

Sadie, S. e Tyrrell, J. (2001). *The New Grove Dictionary of Music and Musicians*. Oxford University Press, Oxford. ISBN 0195170679, 978-0195170672.

Sandred, O., Laurson, M., e Kuuskankare, M. (2009). Revisiting the illiac suite - a rule based approach to stochastic processes. *Sonic Ideas/Ideas Sonicas*, p. 42–46.

Shapiro, S. S. e Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611.

Tanaka, T. e Fujii, K. (2015). Describing global musical structures by integer programming on musical patterns. In *Mathematics and Computation in Music*, p. 52–63. Springer.

Woolson, R. (2008). Wilcoxon signed-rank test. *Wiley Encyclopedia of Clinical Trials*.