

# GP を用いた JAZZ 実時間自動生成システムの試作

井上彩<sup>†1</sup> 城和貴<sup>†1</sup>

**概要:** 近年、個人がメディアを通じて発信することが盛んに行われるようになり、それに伴い著作権フリーの音楽への需要が高まっている。しかし、音楽的な専門知識や経験のない一般ユーザにとっては一から曲を創作することは困難である。そこで、計算機による自動作曲の一手法として、遺伝的プログラミングを用いた JAZZ 即興演奏の自動生成モデルを提案する。さらに、自作音楽の需要の高まりを考慮し、学術的であるこのモデルを実用的に使用可能とするためのシステムを構築する。また、遺伝的プログラミングにおける初期個体として利用する既存曲の選択を可能とし、生成される曲の幅を拡大させる。

**キーワード:** 遺伝的プログラミング, 自動作曲, MusicXML

## Prototype of JAZZ real-time automatic generation system using GP

AYA INOUE<sup>†1</sup> KAZUKI JOE<sup>†1</sup>

**Abstract:** In recent years, it has become popular for individuals to transmit through the media, and along with this, the demand for non-copyrighted music is increasing. However, it is difficult for general users who have no musical expertise or experience to create songs from scratch. Therefore, we propose an automatic generation model of JAZZ improvisation using genetic programming as a method of automatic composition by computer. Furthermore, we will construct a system that enables practical use of this academic model in consideration of the growing demand for self-made music. It also enables the selection of existing songs to be used as initial individuals in genetic programming, expanding the range of songs generated.

**Keywords:** Genetic Programming, Automatic composition, MusicXML

### 1. はじめに

近年、個人がメディアを通じて発信することが盛んに行われるようになり、それに伴い著作権フリーの音楽への需要が高まっている。しかし、音楽的な専門知識や経験のない一般ユーザにとっては一から曲を創作することは困難である。そこで、ユーザが簡単に曲を作ることができるように、計算機によって自動的に楽曲を生成させる試みが行われている。計算機による自動作曲の手法として、進化的計算 [1]を用いて楽曲を生成する手法 [2] [3] [4]とディープラーニング [5]を用いて楽曲を生成する手法 [6]がある。前者の進化的計算とは、生物の進化のプロセスをモデル化し、組合せ最適化問題などに応用する人工知能の一分野である。自動作曲を、膨大な数の音符の中から最適な組み合わせを探索する最適化問題であると考えれば、進化的計算は自動作曲に適した手法であると考えられる。後者のディープラーニングを用いた手法は、近年注目されている。ディープラーニングエンジンの deepjazz [6]に MIDI 形式のサンプルデータを渡すと、その曲に近いメロディが生成されるというものである。この手法では、ジャンルを問わず生成できるが、細部まで正確に表現することができない。そのため、過去の偉大なジャズ奏者である Charlie Parker や Miles Davis,

Oscar Peterson などのジャズ・ミュージシャンの再現は不可能である。また、計算過程を見ることができないという問題点もある。よって、本稿では遺伝的プログラミング (Genetic Programming, GP) を用いた JAZZ 即興演奏の自動生成モデルを提案する。遺伝的プログラミングは、進化的計算の一種である。さらに、自作音楽の需要の高まりを考慮し、学術的であるこのモデルを実用的に使用可能とするためのシステムを構築する。また、遺伝的プログラミングにおける初期個体となる既存曲の選択を可能とするため、楽譜データとして MusicXML [7]というファイルフォーマットを利用する。MusicXML は XML 形式で表記され、楽譜、パート、小節それぞれの情報が階層化され記述されている。既存曲の選択を可能とすることで、生成される曲の幅を拡大させる。

本稿の構成は以下の通りである。第 2 章で既存研究を紹介し、第 3 章では、既存曲の選択や楽譜作成ソフトに用いる MusicXML について述べ、第 4 章では本システムの概要を示す。

### 2. 既存研究

本章では、既存研究である GP を用いた JAZZ 即興演奏の自動生成モデル [8]について説明する。このモデルは、メ

<sup>†1</sup> 奈良女子大学  
Nara Women's University

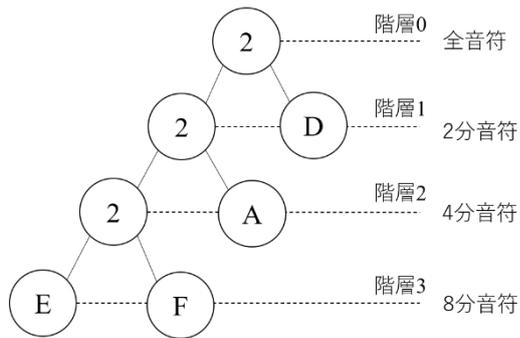


図 1 遺伝子個体の例



図 2 図 1 を楽譜にした結果

ロディ生成パート，コード進行生成パート，ベースライン生成パート，ドラム生成パートの順で，それぞれの音列を生成する．本来，ジャズの即興演奏は，原曲を1コーラス弾いた後，各パートのアドリブが数コーラス続く形態をとる．既存研究のモデルにおいても，ジャズ即興演奏の形態に則った流れをとる．このモデルでは，GPの持つ遺伝子個体を木構造で表現できるという特徴を生かして，階層の深さと音の長さを対応付けている．図1は遺伝子個体の例であり，図2は図1を音符に起こした結果である．階層が深くなればなるほど細かい音符を表すことが可能となり，音の長さや3連符などを表現することができる．図3は既存曲を元にして，4種類に分けられたパートを生成する流れである．それぞれ12小節毎の音列生成を試みる．このようにして， $i$ 番目に生成される12小節の音列を，それぞれ部分メロディ $i$ ，部分コード進行 $i$ ，部分ベースライン $i$ ，部分ドラム $i$ とする．これにより，適応度評価の際に部分メロディ $i-1$ ，部分コード進行 $i-1$ ，部分ベースライン $i-1$ と比較しやすくしている．

メロディ生成パートについて述べる．部分メロディは，テナーサックスの音域である3F~6Ebに加え，継続(～)と休符(\*)を合わせた36音を使用する音列とする．部分メロディ $i$ の適応度評価は次の項目に従って行う．

1-1. ジャズの特徴を満たしているかの評価

- (1) コード構成音の割合
- (2) ペンタトニック・スケールの割合
- (3) 使用禁止音の割合

1-2. 部分メロディ $i-1$ との比較

- (1) Music Entropy の比較
- (2) 平均音価の比較
- (3) 1小節に含まれる音数の比較

1-3. サックス奏者のアドリブ演奏譜との音高変化の比較

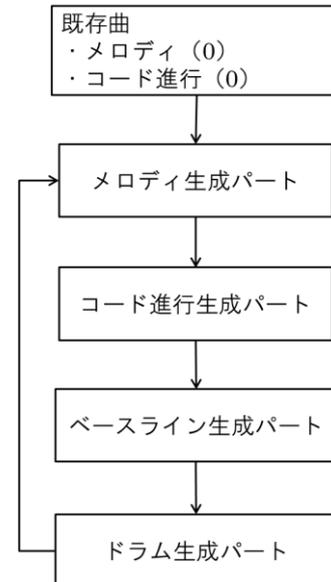


図 3 モデルの全体像

ジャズの特徴を満たしているか否かを評価する項目1-1について述べる．項目1-1は，コード進行に適したメロディを生成するために行う．各小節にコード構成音，ペンタトニック・スケール，使用禁止音が含まれている割合を算出し，評価する．ペンタトニック・スケールとは，5つの音階で構成されることを意味し，半音を構成する隣接音を使わない音階のことである．この音階を使用することで，様々なコード進行の中で使えるアドリブ演奏が可能になる．また，ペンタトニック・スケールには，メジャー・ペンタトニック・スケール，マイナー・ペンタトニック・スケールの2種類があるが，このモデルでは，これらの音が含まれている割合が高いものを適応度評価の高い個体とする．また，メロディ音が不協和になる音を使用禁止音とする．これらの音が含まれている割合が低いものを適応度評価の高い個体とする．

部分メロディとの比較を行う項目1-2は，曲の急激な変化を防ぐために行う．部分メロディにおける前後の音の遷移確率を用いて，曲の印象を Music Entropy として計算する．1小節に含まれる音数を $nN$ とし，小節ごとの比較を行う．14種類のメロディの終端要素からなる事象 $\{x_1, x_2, \dots, x_{14}\}$ とし，事象 $X_1$ が起こった条件下で事象 $X_k (k=1, 2, \dots, 14)$ が起こる確率を $Y_{kl}$ とする．このとき Music Entropy を以下のように定義する．

$$\text{Music Entropy} = -\sum_{kl=2}^{nN} Y_{kl} \times \log Y_{kl} \quad (1)$$

前12小節と Music Entropy の値の近いものほど適応度評価が高くなる．また，1小節内に含まれる1音の平均音価，つまり音の長さの平均を計算し，比較を行う．さら



図 4 既存曲のメロディの一部

に、各小節間に含まれる音数を比較する。これは、Music Entropy の計算の際に、音数とその値に大きく影響するため、各小節に含まれる音数を近いものにする必要がある。

項目 1-3 ではサクソ奏者のアドリブ演奏譜との音高変化の比較を行う。この項目では、実際の奏者の演奏事例と生成された音列の音高変化の比較を行い、音高変化の一致数の高い音列を優先的に選択する。音高変化は、メロディに対するコード進行が一致しているものと比較することで得られる。このように、実際の奏者のアドリブ演奏譜例から音高変化を抽出し、生成した音列と比較することで、アドリブ演奏の特徴を持つ音列生成が可能となる。

次に、コード進行生成パートについて述べる。このパートでは、3つのコードにヴォイスンクを行うことで、コード生成の種類を増やすことができる。ヴォイスンクとは、コード構成音の並び方を変更し、曲の印象をより適した流れに変えることができるものである。既存曲で使用されるコードをヴォイスンクしたものを終端要素の候補とし、1つのコードに対して44種類のヴォイスンクパターンを用意する。また、部分コード進行*i*の適応度評価は次の項目に従って行う。

- 2-1. 部分メロディ*i*の各小節におけるメロディとコード構成音の比較
- 2-2. 部分コード進行*i-1*との Music Entropy の比較
- 2-3. 休符の割合

項目 2-1 は、各小節のコード構成音についてメロディ音との比較を行い、メロディとの不協和な半音衝突をなるべく避けるためである。

項目 2-2 は部分コード進行*i-1*と部分コード進行*i*の印象の急激な変化を避けるために行う。このモデルでは、前後の小節間に類似性を持たせるため式(1)を用いて求められる Music Entropy の比較を行う。

休符の割合を決める項目 2-3 については、各コードを付点4分音符と8分音符のリズムで構成し、各小節において休符の長さの割合が25~50%となるような個体を優先的に選択するように設定を行う。

図 5 スコア生成例

次に、ベースライン生成モデルについて述べる。このパートでは4ビートのベースラインを生成するため、生成する音個体は4分音符とする。部分ベースライン*i*の適応度評価は次の項目に従って行う。

- 3-1. コード構成音との整合性
- 3-2. ペンタトニック・スケールの割合
- 3-3. 部分ベースライン*i-1*との Music Entropy の比較

コード構成音との整合性を評価する項目 3-1 は、与えられたコード進行に適したベースラインを生成するために、それぞれコードの構成音であるか否かの割合を計算

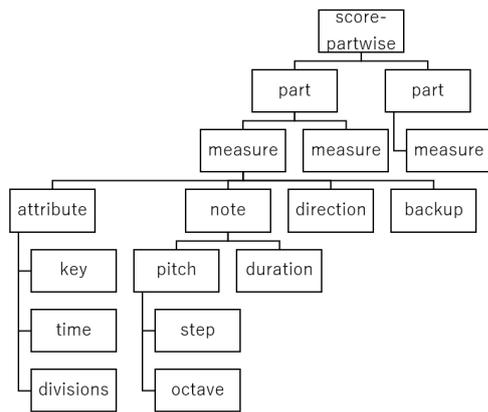


図 6 MusicXML の構造

し評価する。

項目 3-2 では、メロディ生成パートと同様に、ペンタトニック・スケールの割合を算出する。項目 3-3 についても、メロディ生成パート、コード進行生成パートと同様に、印象の急激な変化を避けるために式(1)を用いて求められる Music Entropy の比較を行う。

最後に、ドラム生成パートについて述べる。スネアドラムとハイハットから成るドラムのリズムを生成し、スネアドラムはアップビートのリズム生成、4 ハイハットは 8 ビートのリズム生成を行う。アップビートとは、各小節の 2 拍目と 4 拍目に強拍を置くスタイルである。このモデルでは、スネアドラムのみ GP による生成を行う。部分ドラム  $i$  のスネアドラムの適応度は次の項目に従って行う。

- 4-1. リズムがアップビートになっているか
- 4-2. コードの変化時にリズムの変化が起こる割合

項目 4-1 では、2 拍目と 4 拍目に強拍が置かれている個体を優先的に選択する。

項目 4-2 については、ドラムのリズムの変化はコード進行の変化時に起こりやすいため、コード進行の変化時に、3 連音符などが生成される個体の適応度を上げることで、リズム変化を生成するようにする。

このように、各パートでそれぞれジャズの理論に基づき評価を行う。結果として、アドリブ演奏の音高変化の特徴を持つメロディが生成され、メロディに適したコードが生成できる。ベースラインでは、ペンタトニック・スケールが多く含まれたコードに適した音列が生成され、ドラムではアップビートのリズムを生成できる。各パート間において、適した音列を持つジャズの生成が可能である。

図 4 は既存曲のメロディの一部であり、GP の初期個体として入力する楽譜情報である。図 5 は自動生成モデルに基づき生成された音列の一部であり、各パートで行う評価に基づいた音列が生成されている。

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE score-partwise PUBLIC
"-//Recordare//DTD MusicXML 3.1 Partwise//EN"
"http://www.musicxml.org/dtds/partwise.dtd">
<score-partwise version="3.1">
  <part-list>
    <score-part id="P1">
      <part-name>Music</part-name>
    </score-part>
  </part-list>
  <part id="P1">
    <measure number="1">
      <attributes>
        <divisions>1</divisions>
        <key>
          <fifths>0</fifths>
        </key>
        <time>
          <beats>4</beats>
          <beat-type>4</beat-type>
        </time>
        <clef>
          <sign>G</sign>
          <line>2</line>
        </clef>
      </attributes>
      <note>
        <pitch>
          <step>C</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        <type>whole</type>
      </note>
    </measure>
  </part>
</score-partwise>
```

図 7 MusicXML 記述例の一部



図 8 図 7 を楽譜にした結果

### 3. MusicXML

#### 3.1 MusicXML の概要

GP の初期個体として入力する楽譜のファイルフォーマットは MusicXML を用いる。MusicXML は、XML 形式の楽譜表記のためのオープンファイルフォーマットである。MusicXML は、Recordare L.L.C. [9]が開発した仕様と、ミュージカル・プラン社が開発した仕様が存在するが、互換性はない。今回は楽譜作成ソフトウェアにおいて互換性の高い前者の仕様の MusicXML を用いる。

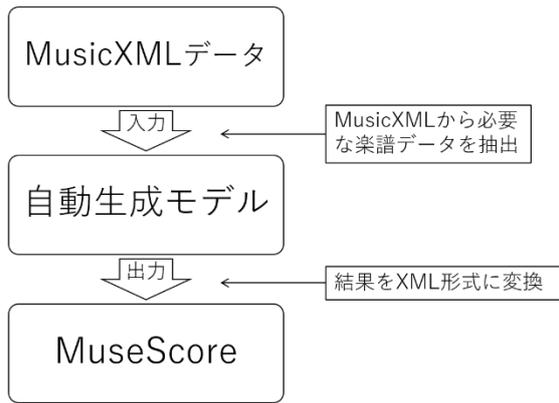


図 9 システムフロー

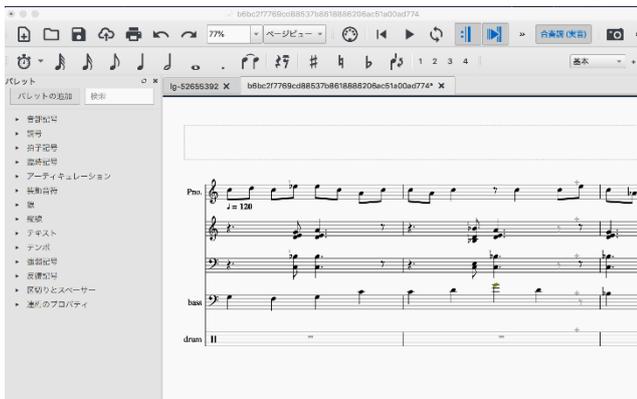


図 10 MuseScore で表記された生成結果

### 3.2 MusicXML の構成

MusicXML は、楽譜情報、パートリスト、小節の順に階層化されており、構造を図 6 に示す。また、図 7 [10] に MusicXML 記述例と図 7 に対応する楽譜を図 8 [10] に示す。それぞれのパートには小節情報を含む<measure>要素が記述されており、子要素には調・拍子・音符の最小長さを規定する<attribute>要素、音符や休符を規定する<note>要素、時間の巻き戻しを行う<backup>要素などの情報が記述されている。MusicXML では、音符や休符が記述されるごとにその長さ分の時間が進むため、右手で弾く音を記述した後、左手の音を記述したいときに<backup>要素を用いて時間の巻き戻しを表現する。<attribute>要素中にある<divisions>要素では、音符、休符の最小長さを規定する。また、<note>要素の子要素として<duration>要素があるが、これは<divisions>要素で決められた音符や休符の最小長さがいくつ分かを表している。

### 4. システムの概要

図 9 に本システムの概要を示す。初めに、MusicXML データから自動生成モデルにおいて計算に必要な楽譜情報を



図 11 Automator の仕組み



図 12 Automator アイコン

抽出する。抽出後、GP の初期個体に入力し、計算を行う。その後、結果を XML 形式のデータに再度変換し、MuseScore [11]に入力する。MuseScore は、オープンソースのソフトウェアとして提供されている楽譜作成ソフトであり、MusicXML と互換性があるため、本システムではこのソフトを使用する。図 10 は、MuseScore で表記された生成結果であり、楽譜表記に加え、作成した楽譜の再生を行うことができる。また、自動生成としてのユーザビリティを考慮し、このシステム全ての流れを Automator [12]を使用して自動化する。Automator とは、アップル社が開発するワークフロー構築ソフトであり、Mac OS X v10.4 から搭載されている。Mac OS の標準アプリケーションだけでなく、互換性のないアプリケーションやスクリプトを組み合わせ、全ての工程を自動化できるツールである。図 11 に Automator の仕組みを示す。図 12 の Automator のアイコンをクリックし起動後、既存曲の選択画面で選択を行う。その後は MusicXML からデータの抽出、GP の計算、楽譜再生まで自動で行う仕組みである。なお、楽譜データ入力から出力までの処理時間は、平均 20 秒前後であり、実時間自動生成システムとして処理に時間がかかっている。通常、実時間処理は 24 分の 1 秒を意味する。それは、映画の再生には 1 秒間に静止画 24 コマを記録する 24fps が採用されており、1 コマ 24 分の 1 秒だからである。そのため、24 分の 1 秒以内での処理を可能とすることが今後の課題であり、ハイエンドの GPU やメニーコアプロセッサを利用することで、本システムの高性能化を行う必要がある。また、配布され

ている MusicXML データは多岐にわたるため、様々な楽譜を入力することで、幅広いジャンルでの音楽自動生成が容易となる。しかし、得られる結果の特徴にばらつきが出ると考えられるため、評価式の見直しを行う必要がある。

## 5. おわりに

近年、人工知能を用いて計算機によって作曲をする研究は多数行われている。中でもディープラーニングを用いた作曲システムは注目されており、deepjazz [6]というジャズの作曲ツールが存在する。ディープラーニングエンジンにジャズの MIDI データを渡すと、ジャズに近いメロディが生み出されるというものである。しかし、ディープラーニングを用いたシステムには問題がある。それは、計算過程を見ることができないことである。そのため、ジャズの繊細な表現や、過去の偉人の再現は不可能である。しかし、GP を用いることで詳細な部分まで評価式を設定できるため最適な解を探索することができる。

本稿では、GP を用いた JAZZ 即興演奏の自動生成モデルをより実用的にするため、初期個体として用いる楽譜の選択を容易とし、全ての工程を自動化させるシステムを提案する。楽譜データとして MusicXML というファイルフォーマットを利用する。MusicXML は XML 形式で表記され、楽譜、パート、小節それぞれの情報が階層化され記述されている。MusicXML データから自動生成モデルにおいて計算に必要な楽譜情報を抽出し、GP の初期個体に入力する。GP の計算結果を XML 形式のデータに再度変換し、楽譜作成ソフト MuseScore に入力する。自動生成としてのユーザビリティを考慮し、このシステム全ての流れを Automator を使用して自動化している。このシステムの楽譜データ入力から出力までの処理時間は、平均 20 秒前後である。実時間自動生成システムの実時間処理は 24 分の 1 秒が望ましいため、今後の課題として、GPU やメニーコアプロセッサを利用することで、本システムの高性能化を行う。また、配布されている MusicXML データは多岐にわたり、幅広いジャンルでの音楽自動生成を可能にするためには、得られる結果の特徴にばらつきが出ないよう評価式の見直しを行う必要がある。

## 参考文献

- [1] 三宮信夫, 喜多一, 玉置久, 岩本貴司. 遺伝的アルゴリズムと最適化, 2006.
- [2] J. A. Biles. “GenJam: A Genetic Algorithm for Generating Jazz Solos,” In Proceeding of the 1994 International Computer Music Conference, ICMA, San Francisco, pp.3-4, 1994.
- [3] 山田拓志, 椎塚久雄. 遺伝的アルゴリズムを用いた自動作曲について, 音楽情報科学 27-2, pp.7-14, 1998.

- [4] 田中健, 外山史, 東海林健二. 遺伝的アルゴリズムを用いたメロディー進行とリズムの組み合わせによる自動作曲, 音楽情報科学 41-8, pp.43-48, 2001.
- [5] “ディープラーニング (深層学習) ”. <https://jp.mathworks.com/discovery/deep-learning.html>. (参照 2020-11-15).
- [6] J.-S. Kim, “deepjazz”. <https://deepjazz.io/>. (参照 2020-11-15).
- [7] “MusicXML”. <https://ja.wikipedia.org/wiki/MusicXML>. (参照 2020-11-15).
- [8] “A Music Composition Model with Genetic Programming - A Case Study of Chord Progression and Bassline-” Kanoe Kunimatsu, Yu Ishikawa, Masami Takata, Kazuki Joe,” In Proceedings of 2015 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA2015), VOL.1, pp.256-262 (2015.7).
- [9] Recordare L.L.C., “musicXML”. <https://www.musicxml.com/>. (参照 2020-11-15).
- [10] “musicXML, Hello World”. <https://www.musicxml.com/tutorial/hello-world/>. (参照 2020-11-15).
- [11] “musescore”. <https://musescore.org/ja>. (参照 2020-11-15).
- [12] “Automator”. <https://ja.wikipedia.org/wiki/Automator>. (参照 2020-11-15).