平成 30 年度 卒業論文

ジャズのベースラインの自動生成

指導教員 北原 鉄朗 准教授

日本大学文理学部情報科学科

志賀 あゆみ

2019 年 2 月 提出

概　　要

ジャズの楽譜にはベーシストが弾くべきベースラインが書かれていない場合がある. そのため, ベースを演奏するとき, ベーシストは楽譜のコード進行から自らベースラインを作成する必要がある. しかし初心者にとって, ベースラインを作ることは容易ではない. まず, 自分でベースラインを考えようとしても, コードを見ただけでは何の音を弾いたらいいかわからないという問題がある. また, インターネット上にベースラインの演奏例が公開されている場合もあるが, すべての曲に対応しているわけではなく, 仮に演奏例が存在したとしても, 難しいものは真似することが困難である. このことから本研究は, ユーザがベースを練習する際に生じる, “自分でベースラインの作成が困難” という問題と, “入手できるベースラインが演奏者の演奏レベルに合うとは限らない” という 2 つの問題の解消を狙ったものである.

まず, “自分でベースラインの作成が困難” という問題については, ユーザから与えられた任意のコード進行に対してジャズのベースラインの自動生成を行い, ベーシストが手本となるベースラインを探す, または生成する手間をなくすことで解決する. ベースラインの自動生成が難しいのは, 前後の音とコードのつながりなどを考慮に入れながら, 行わなければならないからである. そして, 単純なルールだけでは効果的な生成は困難である. そこで, まずベースラインの自動生成は隠れマルコフモデル (Hidden Markov Model : HMM) を用いて行う. HMM の場合, 「コード進行のみが与えられた場合, その背後に隠されたベースラインを推定する問題」という発想で考える. すなわち, 与えられたコード進行から適合するベースライン

を決定するのではなく, そのコード進行を決定したと考えられるベースラインを, 遡って推定する逆問題と考える. このような問題を解決するために, HMM が多く使われているため, この自動生成の問題も HMM を用いて定式化する. あるベースラインからコード進行の生成される過程を HMM を用いて確率モデル化し, 尤度最大の原理によって与えられたコード進行を生成する遷移系列の中で最も尤度の高い系列を Viterbi アルゴリズムによって求める. 本研究では, 隠れ状態が異なる 3 つの手法でベースラインの自動生成を行う. 手法 1 は各音名を隠れ状態とし, 手法 2 では各音高を隠れ状態に, 手法 3 では拍節位置ごとの各音名を隠れ状態と設定する. 出力記号は共通してコード進行とする. HMM を用いることで, 確率的に最ももっともらしいベースラインを生成することができる.

次に, “入手できるベースラインが演奏者の演奏レベルに合うとは限らない” という問題については, ユーザが生成するベースラインの難易度を選択することで解決する. ベースラインの自動生成には HMM の手法 3 が後述の実験から有効であるとわかったため, 手法 3 の考えを基本として, 難易度を追加したものを遺伝的アルゴリズム (Genetic Algorithm : GA) を用いて実装する. GA は, 適応度関数が最大の個体が得られるように個体の選択, 交叉, 突然変異を繰り返す最適化アルゴリズムである. GA はモデルに制約がなく, 適応度関数を自由に設計することができるため, 適応度関数さえ設計すればどんな問題も解決することができる. そのため, 確率的に最ももっともらしいベースラインを生成することに加え, 適応度関数に生成するベースラインの演奏上の難易度に関する制約を入れることで, ベースラインの難易度を制御することができる. 本研究では, 難易度を 3 つ用意し, 難易度ごとにベースラインの自動生成を行う. 難易度はリズムと指の移動量によって変化させる.

実装したベースラインの自動生成システムの有効性を確認するために, 生成されたベースラインが適切なものであるかどうか検証する. 103 個のコード進行に対して HMM の 3 つの手法でベースラインの自動生成を行い, 客観評価と専門家による

主観評価を行った. 3 手法それぞれの評価の結果, 手法 3 による生成結果が正解データとの一致率が 41.75%で最も高い結果であった. その他の客観評価も 8 項目中 6 項目で 3 つの手法の中で手法 3 が正解データに最も近い結果であった. また主観評価でも 4 つの項目すべてで手法 3 がいい結果となった. 以上のことから, 拍節を考慮した手法 3 によるベースラインの自動生成が最も有効な手法であるとわかった.

次に GA に対しても実験を行った. まず, 世代数を決定するために重み値をすべて 1 で固定し, 1, 10, 25, 50, 100 世代数ごとに自動生成を行い, 正解データとの一致率を求めた. その結果, 世代数を 25 から増やしても一致率があまり変わらなかったため, 世代数を 25 と決定した. 次に, 重み値を決定するために世代数を固定し, 様々なパラメータの組み合わせで自動生成を行い, 正解データとの一致率を求めた. 単体での精度が 43.58%と圧倒的に高い chord に加え, 難易度を変化させる move と off を含めた組み合わせの中で最も一致率が高いものを GA での生成に使用した. この実験の結果から, chord, move, off の 3 つのみを使ってベースラインを生成する手法が 36.2%で最も高かった. 次に, 実験で求めた世代数と適応度関数を使用して生成を行ったところ, 初級, 中級, 上級それぞれの難易度の定義 (8 分音符の割合, 指の移動量) に沿ったベースラインが生成された.

目　　次

目　　次 **v**

図 目 次 **ix**

表 目 次 **xi**

第 1 章　序　　論 **1**

1.1　背景 1

1.2　目的 2

1.3　論文の構成 2

第 2 章　関連研究 **3**

2.1　和声付けに関する関連研究 3

2.1.1　ルールベースでの和声付け 3

2.1.2　HMM を用いた自動和声付け [2] 3

2.1.3　事例ベース推論を用いた自動和声付け [4] 4

2.2　ベースライン生成に関する関連研究 4

2.2.1　フレーズを選択してベースライン生成を行うシステム [7] . . 4

2.2.2　ルールベースによるウォーキングベース生成 [8] 5

2.2.3　遺伝的プログラミングを用いたベースライン生成 [9] 5

2.3　演奏者のレベルに合わせる編曲に関する関連研究 5

2.3.1　ギター演奏者の習熟度に合わせたのタブ譜自動生成 [10] . . . 5

2.4 関連研究との類似点・相違点 6

第 3 章 ベースライン自動生成 7

3.1 HMM を用いたベースライン自動生成 7

3.1.1 問題設定 7

3.1.2 状態の設計 8

3.1.3 初期確率・出力確率・遷移確率 8

3.1.4 音高決定 9

3.2 GA を用いたベースライン自動生成 10

3.2.1 問題設定 10

3.2.2 難易度の定義 10

3.2.3 リズムの決定 10

3.2.4 初期個体の生成 11

3.2.5 適応度関数 11

第 4 章 実 験 15

4.1 使用データ 15

4.2 HMM を用いた場合のベースライン生成 16

4.2.1 実行例 16

4.2.2 客観評価 19

4.2.3 主観評価 21

4.3 GA を用いた場合のベースライン生成 24

4.3.1 GA の世代数の決定 24

4.3.2 GA の重み値の決定 24

4.3.3 実行例 28

4.3.4 客観評価 31

第 5 章 結 論 35

参考文献 **37**

図 目 次

4.1 コード D B♭ F♯ D 17
4.2 コード Em Em A Dm 17
4.3 コード Gm C F B♭ 17
4.4 世代数ごとの一致率 25
4.5 重み値が 1 つだけ 0 それ以外 1, またはすべて 1 の場合 25
4.6 重み値が 1 つだけ 1 それ以外 0 の場合 26
4.7 重み値が 3 と他 1 つは 1 それ以外 0 の場合 26
4.8 重み値が 3 と他 2 つは 1 それ以外 0 の場合 27
4.9 重み値が 345 は 1 それ以外で 01 の組み合わせの場合 27
4.10 コード D B♭ F♯ D 29
4.11 コード Em Em A Dm 29
4.12 コード Gm C F B♭ 29

表 目 次

3.1 難易度の定義 14

4.1 HMM の客観評価の結果 23
4.2 HMM の主観評価の結果 23
4.3 GA の客観評価の結果 34

第1章　序　　論

1.1　背景

音楽の楽しみ方について, 自ら楽器を演奏する方法がある. 多くの人が趣味やバンド演奏の一部として楽器演奏を楽しんでおり, その中で, ベースはバンドに欠かせない楽器の 1 つである.

バンドで曲を演奏する場合, 特にポップスなどはバンドスコアが存在する場合がある. バンドスコアにはそれぞれの楽器に対して演奏するべき旋律が書かれている. それに対して, ジャズの楽譜にはベーシストが弾くべきベースラインが書かれていない場合がある. そのため, ベースを演奏するとき, ベーシストは楽譜のコード進行から自らベースラインを作成する必要がある. しかし初心者にとって, ベースラインを作ることは容易ではない. まず, 自分でベースラインを考えようとしても, コードを見ただけでは何の音を弾いたらいいかわからないという問題がある. また, インターネット上にベースラインの演奏例が公開されている場合もあるが, すべての曲に対応しているわけではなく, 仮に演奏例が見つかったとしても, 演奏者の演奏レベルに合うとは限らない.

ベースラインの自動生成に関する研究は, 与えられたメロディーにコード進行を付与する和声付けの研究 [2, 3, 4, 5, 6] に比べ, 比較的少数である. あらかじめ決められたフレーズに沿ってベースラインを生成するシステム [7] や, ルールベースでベースラインを生成するもの [8], 遺伝的プログラミングを用いて自動生成するもの [9] などが開発されている. しかし, 上述の研究 [7] では, フレーズが固定されているためある程度決められたベースラインしか生成ができない. また, 研究 [7, 8, 9]

に共通して, ベースラインの難易度が一定であるという問題がある.

1.2 目的

1.1 節で述べたようにジャズのベースを練習するためには以下の 3 つの問題が存在する.

- コードを見ただけでは何の音を弾いたらいいかわからないという問題
- インターネット上にすべての曲に対してベースラインの演奏例があるとは限らないという問題
- 演奏例が演奏者の演奏レベルに合わないという問題

本研究は, これら 3 つの問題を解決するベースライン自動生成システムを実現することを目的とする. 具体的にはこのシステム内で「ユーザから与えられたコード進行からベースライン自動生成」, さらに「難易度ごとにベースラインの自動生成」を実現することを目的とする.

1.3 論文の構成

本論文の構成は次の通りである. 2 章では和声付けに関する研究事例, ベースラインの生成に関する研究事例, ユーザのレベルに合わせた編曲に関する研究事例についてそれぞれ述べる. 3 章では 2 章で設定した研究課題を達成するためのベースラインの自動生成手法について述べる. 4 章では本研究で提案した手法によって生成されたベースラインの評価とその結果について述べる. 5 章では本論文の結論を述べる.

第2章　関連研究

1.2 節で述べた 3 つの目的を達成するために, まず既存の和声付けに関する研究, ベースラインの生成に関する研究, ユーザのレベルに合わせて編曲する研究について述べる.

2.1　和声付けに関する関連研究

本節では, 和声付けについて述べられている研究事例を紹介する.

2.1.1　ルールベースでの和声付け

Pachet らの調査 [1] は, ルールベースの和声付けに関する様々な研究の調査である. 和声付けの規則には, 音符の遷移に関するものと, 同時音符または和音に関するもの, 和音の進行に関するものの 3 つがある. この調査の結果, 4 パート (ソプラノ, アルト, テノール, ベース) の和声付けの問題は, 制約充足問題 (CSP) を用いることで解決することができると述べた.

2.1.2　HMM を用いた自動和声付け [2]

川上らの研究 [2] は, 和音名を隠れ状態とする隠れマルコフモデルを用いて, 観測された旋律の背後にある和声を推定することである. 和声付けだけでなく, 転調の検出も可能であることを示した.

2.1.3　事例ベース推論を用いた自動和声付け [4]

平田らの研究 [4] は, 演繹オブジェクト指向データベースに基づく音楽知識表現体系に基づいて, ジャズ和声を生成するシステム「パーピープン」の開発である. まず事例ベースを作成し, ユーザはシステムと対話しながら, ユーザから与えられた単純な和音進行に対してユーザの意図を反映させ, 楽曲の流れやボイスリーディングを考慮に入れて演奏を構成する. システムを試用した結果, ユーザの意図を表現するのにケーデンス木が有効であると確認し, 多様で一貫した曲調の楽曲を生成することを実現している.

2.2　ベースライン生成に関する関連研究

本節では, ベースラインの自動生成システムの実装を行っている研究事例を紹介する.

2.2.1　フレーズを選択してベースライン生成を行うシステム [7]

小中らの研究 [7] は, 任意のコード進行からジャズのベースラインをインタラクティブに生成するシステム「LPB88」の開発である. このシステムはあらかじめ固定された複数のフレーズの候補からユーザが 1 つ選び, 各音をどのオクターブで弾くかを決定することでベースラインを生成している. しかし, ユーザは用意されたフレーズから選択するため, 決められた動きのベースラインしか生成できないという問題がある. また, フレーズによって難しさが変わる可能性があるが, それぞれに難易度が定められているわけではない.

2.2.2 ルールベースによるウォーキングベース生成 [8]

Rui Dias らの研究 [8] は, ルールベースによりジャズでよく使用されるベースラインのウォーキングベースを生成するシステムである. コードの 1 拍目はルート音とし, 次の小節のルート音の音高を決定する. その後, 現在のコードの 1 拍目から次のコードの 1 拍目の間のベースラインを埋めていく. 次の小節のルート音の音高とベースラインの音高の幅を変化させることで様々なベースラインの生成を実現している. 音高の幅を変化させることで難しさが変わる可能性があるが, 難易度が定められているわけではない.

2.2.3 遺伝的プログラミングを用いたベースライン生成 [9]

國松らの研究 [9] では, 遺伝的プログラミングを用いてブルースのコード進行を生成し, そのコード進行に対応するベースラインを生成するシステムの実装を行っている. テンション・ノートを付与することで様々な響きを表現できるようにしている. このシステムは, 生成したコード進行に対応するベースラインが生成されるため, ユーザが求めるコード進行とは異なる可能性がある. また, 難易度の設定は設けられていない.

2.3 演奏者のレベルに合わせる編曲に関する関連研究

本節では, ユーザのレベルに合わせて楽譜や旋律を編曲する研究事例を紹介する.

2.3.1 ギター演奏者の習熟度に合わせたのタブ譜自動生成 [10]

矢澤らの研究 [10] は, 実際のギターの演奏音から演奏者の習熟度に応じたタブ譜を自動生成することである. 具体的には, 初級者向けには音符の欠落を許容してで

も演奏が簡単なタブ譜を生成し, 上級者向けには音高を正確に再現するタブ譜をそれぞれ生成するシステムである. まず, 既存の多重基本周波数推定法を用いて音高推定を行い, その後その結果をもとに運指推定を行う. さらに, 推定された最適運指を用いて後処理を行うことで, 元の音高推定結果に含まれるギターで演奏不可能な音高の組み合わせを排除する. その結果, 音高推定の適合率を保ったまま運指の簡略化を実現した.

2.4　関連研究との類似点・相違点

本研究は 1.2 節で述べたように, ユーザから与えられたコード進行からベースラインを自動生成することと, 難易度ごとに生成することを目的としている. 2.2 節であげた既存研究のうち, 遺伝的プログラミングを用いてベースラインを自動生成する研究 [9] では, コード進行はユーザが指定することができず, さらに難易度を選択することも可能ではない. また, 固定されたフレーズからベースラインをユーザが選択する研究 [7] や, ルールベースでベースラインを生成する研究 [8] で, 「任意のコード進行からのベースラインの自動生成の問題」は解決されている. しかし, これらは難易度については触れていないので「難易度ごとのベースラインの自動生成の問題」は解決できていない. このように, 先に挙げた研究で「任意のコード進行からのベースラインの自動生成の問題」と「難易度ごとのベースラインの自動生成の問題」の両方に関連したシステムは存在しなかった.

以上のことから, ベースラインを自動生成するシステムを実装する上で以下のことに着目したシステムを考える.

- ユーザが与えた任意のコード進行からベースラインの自動生成を行うこと.
- 難易度ごとにベースラインの自動生成を行うこと.

第3章　ベースライン自動生成

上述の通り, ベースラインの自動生成が難しいのは, 前後の音とコードのつながりなどを考慮に入れながら, 行わなければならないからである. そして, 単純なルールだけでは効果的な生成は困難である. この問題を解決するために, 本研究ではHMM を用いるベースラインの自動生成手法と GA を用いてベースラインを自動生成する手法の 2 つを提案する. HMM はあるベースラインからコード進行の生成される過程を, 大量のベースラインのデータから確率モデル化するため, 前後の音のつながりやコードとの関係を確率で表すことができる. GA は初期個体を複数個生成し, 適応度関数が最大の個体が得られるように個体の選択, 交叉, 突然変異を繰り返すことによって求める. 適応度関数に音のつながりなどの条件を含めることで, 最適なベースラインを生成することができる.

3.1　HMM を用いたベースライン自動生成

3.1.1　問題設定

本システムは, 任意のコード進行が読み込まれると, ベースラインを自動的に生成する. 生成するベースラインは 4/4 拍子とし, 調はハ長調と設定する. ユーザから与えられるコード進行は m 小節 (現在の実装では $m = 4$) とし, 1 小節に 1 つのコードが対応しているものとする. 入力可能なコードの種類は, C∼B の 12 個のメジャーコードと Cm∼Bm の 12 個のマイナーコードの合計 24 個である. また, 生成されるベースラインは 4 分音符のみで構成される.

3.1.2　状態の設計

本研究では, 以下の手法ごとに HMM の隠れ状態を設定する. 出力記号は 3 つの手法に共通してコード進行とする.

(1) 1 オクターブにまとめる (以下, 手法 1).

各状態 h_i は, 0 以上 11 以下の整数をとるものとし, 各々の値が音名 (C, C$^\sharp$, ..., B) に対応するものとする.

(2) そのままの MIDI ノートナンバーで数える (以下, 手法 2).

各状態 h_i は, 0 以上 32 以下の整数をとるものとし, 各々の値が音高 (E1, F1, ..., C4) に対応するものとする.

(3) 拍節を考慮する (以下, 手法 3).

各状態 h_i は, 0 以上 47 以下の整数をとるものとし, 音名 h'_i および拍節位置 b_i を用いて $h_i = h'_i + 12b_i$ と定義する. ここで, h'_i(=0, ..., 11) は各音名 (C, C$^\sharp$, ..., B) に対応し, b_i(=0, 1, 2, 3) は小節内の拍節位置に対応する. このようにすることで, 拍節位置ごとに別々の出力確率や遷移確率を学習できる.

上記の 3 つの手法で, ベースラインの $4m$ 個の音名 $H = (h_1, ..., h_{4m})$ を決定する. ただし, 手法 2 の場合は音高を決定する.

3.1.3　初期確率・出力確率・遷移確率

- 初期確率

  $P(h_i)$ は初期確率である. 1 音目の各状態の確率を求める.

- 出力確率

  $P(c|h_i)$ は状態が h_i のときにコード c が出力される確率である.

- 遷移確率

  $P(h_i|h_{i-1})(i \geq 2)$ はベースラインにおける遷移確率である. h_{i-1} の各状態から h_i の各状態へ遷移する確率を求める.

3.1.4 音高決定

HMM から得られる結果が $4m$ 個の音名 $H = (h_1, ..., h_{4m})$ の場合 (手法 1, 3), 音高 n_i を決定する必要がある. ただし, 音高は MIDI ノートナンバーで 28~60 の範囲とする. 手法 2 の場合は HMM の結果から音高が得られるため, h_i をそのまま出力する音高とする. 以下, 音高を決める方法を述べる.

1. n_1 は上述の範囲内で, h_1 の音名の最低音と決める.

2. $n_2, ..., n_{4m}$ は滑らかなベースラインにするために, n_{i-1} の音高から n_i への音高の差が小さくなるようにする.

具体的には, n_i のオクターブ位置におけるドの音のノートナンバーを o_i とする $(n_i = o_i + h_i)$ と, 次の式により n_i を計算する.

- $i = 1$ のとき, $n_1 = o_1 + h_1$

$$o_1 = \begin{cases} 24 & (h_1 \geq 4) \\ 36 & (h_1 \leq 3) \end{cases}$$

- $i \geq 2$ のとき, 次式により o_i を求め, $n_i = o_i + h_i$ を決定する.

$$o_i = \begin{cases} \max(o_{i-1} + 12, 24) & (h_i - h_{i-1} < -5) \\ \min(o_{i-1} - 12, 48) & (h_i - h_{i-1} > +5) \\ o_{i-1} & (\text{otherwise}) \end{cases}$$

ただし, n_i が 28~60 の範囲に入らない場合, o_i を 12 ずつ変化させ, 範囲内に収まるようにする.

3.2　GAを用いたベースライン自動生成

3.2.1　問題設定

生成するベースラインは3.1.1節と同様に, 4/4拍子とし, 調はハ長調と設定する. ユーザから与えられるコード進行はl小節 (現在の実装では$l = 4$) とし, 1小節に1つのコードが対応しているものとする. ただし, ベースラインはMIDIノートナンバーで28~50の範囲で, 4分音符と8分音符で構成される. また, 難易度は, 初級・中級・上級の3つから選択可能である. ここで用いるGAでは初期個体を複数個生成し, 適応度関数が最大の個体が得られるように個体の選択, 交叉, 突然変異を繰り返す.

後述の実験の結果からHMMの手法3が有効であるとわかったため, GAでは手法3の考え方を基本とし, その応用として難易度を追加する.

3.2.2　難易度の定義

難易度は3段階とし, 1小節に対してそれぞれ表3.1のように定義する. 表の指の移動距離は, 弦を移動する場合, 隣り合う弦の同じフレットへの移動ならば1フレットとする.

3.2.3　リズムの決定

8分音符でベースラインを生成すると同時にMIDIファイルを作成する. 表拍と裏拍の音高が等しい場合, MIDIファイル生成時に8分音符2つを4分音符1つとみなし, 4分音符の長さで記録することでリズムを変化させる.

3.2.4 初期個体の生成

MIDI ノートナンバーをランダムに選び, 初期個体を 2000 個分用意する.

3.2.5 適応度関数

適応度関数は, 生成されるベースラインが次の条件を満たすときに高い値を返すように設計する：

- 既存の音楽でも頻繁に出現する音高の並びである.
- ユーザが指定したコード進行に沿っている.
- ユーザが指定した難易度にあっている.
- 同じ音高が何度も続くなどの単純な並びではない.
- 実際に演奏可能な音高の並びである.

これらの条件を満たすように, GA から得られる $8l$ 個の音高列 $G = (g_1, ..., g_{8l})$ に対する適応度関数 $F(G)$ を次のように定義する：

$$\begin{aligned} F(G) =& w_1 \ \mathrm{first}(G) + w_2 \ \mathrm{seq}(G) + w_3 \ \mathrm{chord}(G) + w_4 \ \mathrm{move}(G) \\ &+ w_5 \ \mathrm{off}(G) + w_6 \ \mathrm{ent}(G) + w_7 \ \mathrm{reg}(G). \end{aligned}$$

適応度関数で用いる確率

GA でのベースライン生成に用いる確率を求める.

- 初期確率・出力確率・遷移確率
  手法 3 と同様に拍節位置を考慮して確率を求める (以下, 拍節データ P_h).
- MIDI ノートナンバーの出現率
  各 MIDI ノートナンバーの出現率を求める (以下, 出現率データ P_r).

各パラメータの計算式

適応度関数 $F(G)$ の各パラメータの計算式ついて述べる．ただし, g_i の音名を t_i で表す．また, $P_h(t_1)$, $P_h(t_i|t_{i-1})$, $P_h(c|t_i)$ は拍節データから, $P_r(g_i)$ は出現率データから学習する．

- first(G)：初期確率

$$\mathrm{first}(G) = \log P_h(t_1).$$

- seq(G)：bigram 確率

$$\mathrm{seq}(G) = \sum_{i=2}^{8l} \log P_h(t_i|t_{i-1}).$$

- chord(G)：出力確率

  対応するコードを c とする．

$$\mathrm{chord}(G) = \sum_{i=1}^{8l} \log P_h(c|t_i).$$

- move(G)：フレットの移動距離

$$\mathrm{move}(G) = \sum_{i=2}^{8l} \log s(g_{i-1}, g_i).$$

  1. 初級

$$s(g_{i-1}, g_i) = \begin{cases} 1.0 & (\text{距離が}2\text{以下のとき}) \\ 0.25 & (\text{距離が}3\text{のとき}) \\ 0.125 & (\text{距離が}4\text{のとき}) \\ 0.0625 & (\text{距離が}5\text{以上のとき}) \end{cases}$$

2. 中級・上級

$$s(g_{i-1}, g_i) = \begin{cases} 1.0 & (\text{距離が } 2 \text{ 以下のとき}) \\ 0.5 & (\text{距離が } 3 \text{ のとき}) \\ 0.25 & (\text{距離が } 4 \text{ のとき}) \\ 0.125 & (\text{距離が } 5 \text{ 以上のとき}) \end{cases}$$

- off(G)：表拍と裏拍の関係

$$\text{off}(G) = \sum_{i=2,4,6,\ldots,8l} \log b(g_i).$$

$$b(g_i) = \begin{cases} 0.99 & (g_i = g_{i-1} \text{ かつ初級のとき}) \\ 0.7 & (g_i = g_{i-1} \text{ かつ中級のとき}) \\ 0.015 & (g_i = g_{i-1} \text{ かつ上級のとき}) \\ 0.01 & (\text{otherwise}) \end{cases}$$

- ent(G)：エントロピー

  $g_1, ..., g_{8l}$ の各音名の出現確率を P_i とする. P_0 は「ド」, P_1 は「ド♯」, ..., P_{11} は「シ」の出現確率を表す.

$$\text{ent}(G) = -\sum_{i=0}^{11} P_i \log P_i$$

- reg(G)：各音高の出現率

$$\text{reg}(G) = \sum_{i=1}^{8l} \log P_r(g_i).$$

重み値 $w_1, ..., w_7$ はあらかじめ定めた値を用いる.

表 3.1: 難易度の定義

要素＼難易度	初級	中級	上級
リズム	4分音符	4分音符+ 8分音符 (0~2個)	4分音符+ 8分音符 (0~4個)
音数	4個	4~5個	4~6個
指の移動距離	フレット3つ	フレット4つ	フレット4つ

第4章　実　　験

本研究は，ユーザによって与えられた任意のコード進行に対してベースラインを自動生成することが目的である．提案する手法によって生成したベースラインがどの程度の割合で適切であったかどうかを検証する必要がある．本章では，行った実験の手順・考察について述べる．

4.1　使用データ

本研究で使用するデータは,『ジャズ・ベース・ランニング 104 実例集』というジャズのベースラインの実例集 [11, 12, 13] と, インターネット上で公開されているジャズのベースラインの演奏例 [14] のうち, 4 拍子のものから選ぶ.

上記の実例集と演奏例のうち, 音符が 4 つ以下で, かつ音高が MIDI ノートナンバーで 60 以下で, さらにコードが 1 つのみ存在する小節が 4 つ連続する部分をデータとして使用する. 音高は生成する範囲を MIDI ノートナンバーで 60 までと設定したため, 使用するデータもその範囲内であるものを使用する. なお, コード進行とベースラインはあらかじめハ長調に移調しておく. コードの条件に関して, 例えば,

|Am　|D　　|Gm　|C　　|

(|は 1 小節ごとの区切りを表す) は 1 小節の中でコードの変化がないため, この 4 小節をデータとして使用するが,

|F　　|Em A |D　　|D　　|

は2小節目に2つのコードが存在するため使用しない.

以上の条件を満たす4小節に対応するベースラインを抽出し, 4分音符に直す. 以下, その方法について述べる.

- 全音符のように, 音符の長さがちょうど拍の先頭と終わりのものは, 1拍ごとに切り4分音符にする.
- 8分音符のように, 音符の長さが半拍のもので2つ続いている場合は, 拍の先頭にある音符の長さを2倍にして4分音符にする. 2つ目の音符は無視する. もし8分音符と付点4分音符の組み合わせの場合は, それぞれの音符の長さを4分音符にする.
- 休符は0とする.
- 同様にGAで使用するために8分音符に直す.

このようにして作成した206種類のコード進行のファイルを103個ずつ, 学習データとテストデータに分ける.

4.2 HMMを用いた場合のベースライン生成

4.2.1 実行例

HMMで自動生成を行った実行例を図4.1~4.3に示す. コード進行D B$^{\flat}$ F$^{\sharp}$ D, Em Em A Dm, Gm C F B$^{\flat}$をそれぞれ入力として与え, 生成を行った結果である.

- 図4.1

  手法1, 2 1小節目の手法1では2音目, 手法2では1, 2, 3音目で非コードトーンが生成されたが, それ以外はコードトーンで構成されている. 手

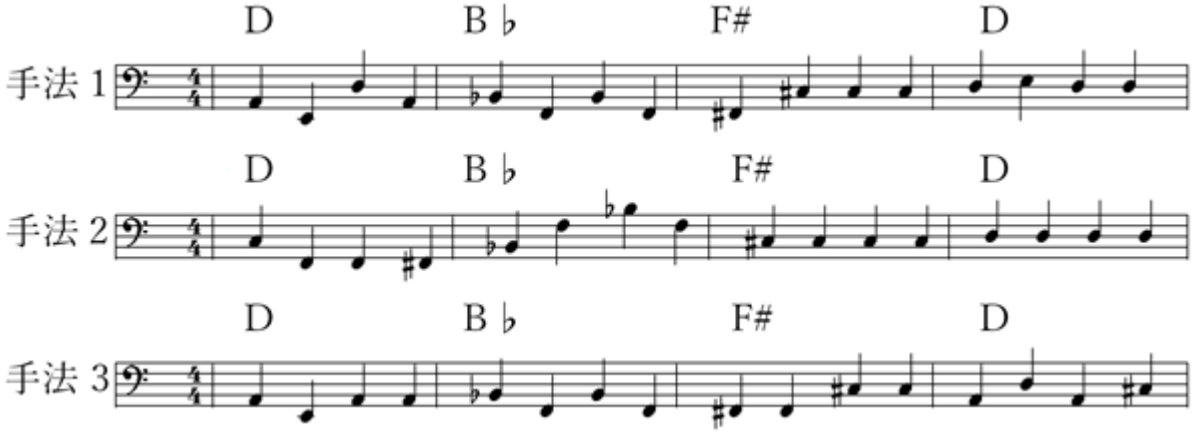

図 4.1: コード D $B^{\flat}$ $F^{\sharp}$ D

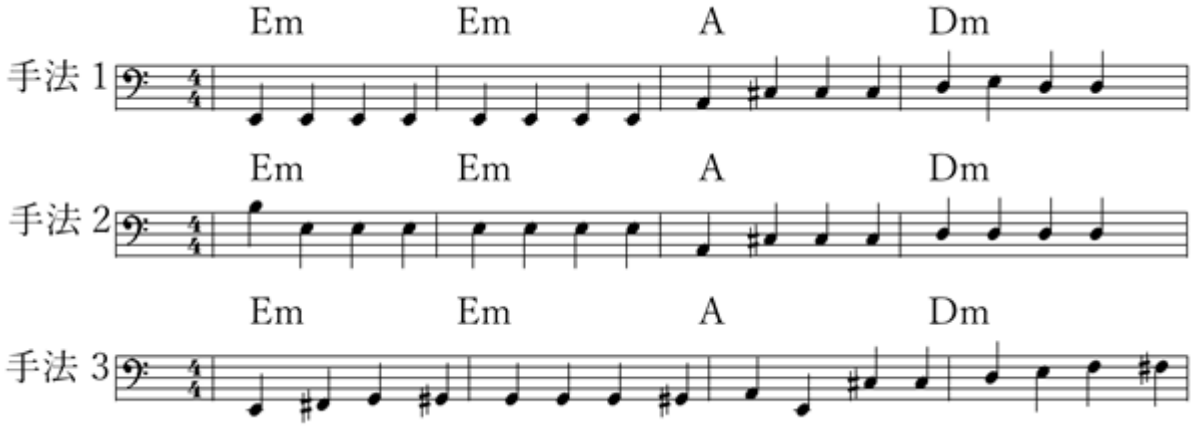

図 4.2: コード Em Em A Dm

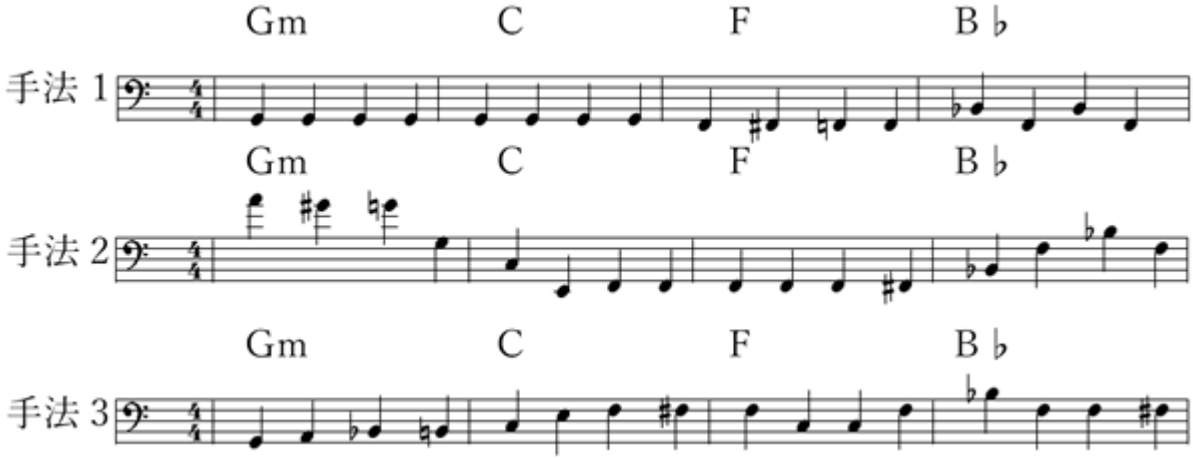

図 4.3: コード Gm C F $B^{\flat}$

法1, 2ともに2小節目までは動きがあるが, 3, 4小節目は同じ音が連続している.

手法3 1小節目の2音目で非コードトーンが生成されたが, 全体的に滑らかなベースラインが生成された.

- 図4.2

手法1 4小節目の2音目で非コードトーンが生成されたが, それ以外はコードトーンで構成されている. すべての小節で同じ音が3つ以上存在している.

手法2 すべてコードトーンで生成されたが, すべての小節で同じ音が3つ以上存在している.

手法3 各小節に非コードトーンがあるが, これらは経過音とみなすことができる.

- 図4.3

手法1 3小節目の2音目で非コードトーンが生成されたが, それ以外はコードトーンで構成されている. 1, 2小節目で同じ音が連続 しており, 3, 4小節目も同じ音が複数回生成されている.

手法2 1小節目の音高が高く, 2小節目にかけて音高の幅が大きくなってしまった. 特に1小節目は1音目が非コードトーンである. 2小節目から3小節目にかけて, 同じ音が続いている.

手法3 1拍目がすべて対応するコードのルート音で構成された. 各小節の非コードトーンは経過音とみなすことができる.

4.2.2 客観評価

評価方法

提案する手法によって生成したベースラインがどの程度の割合で適切に生成されたか, 検証を行う. 正解データは 4.1 節で作成したテストデータ 103 個とする. また, 正解データが休符 (0) の場合は 1 つ前の音と同じ音として計算する. 1 つのコード進行に複数の正解データがある場合は, 最も正解率が高いものをそのコード進行の正解率とし, 103 個の平均を求める.

以下の項目に対して評価を行う. ただし, 評価 2~9 は, 手法 1~3 で生成されたベースラインと正解データそれぞれに対して評価を行う.

評価 1 生成されたベースラインと正解データの音名が一致する割合を求める.

評価 2 対応するコードのルート音の割合を求める.

評価 3 各小節の 1 拍目が, 対応するコードのルート音の割合を求める.

評価 4 各小節の 1 拍目が, 対応するコードのコードトーンの割合を求める.

評価 5 不協和音 (対応するコードのコードトーンに対して短 2 度) の割合を求める.

評価 6 同音進行の割合を求める.

評価 7 順次進行の割合を求める.

評価 8 跳躍進行の割合を求める.

評価 9 出現する音名の個数を求める.

これらの検証は, HMM を用いた手法 1, 2, 3 の 3 通りのベースラインと正解データに対して行う. なお, 評価 1 は値が高いほうが良い結果であり, それ以外は正解データの値に近いほど良い結果とみなす.

評価結果・考察

HMM で客観評価を行った結果を表 4.1 に示す. 結果は小数点以下第三位を四捨五入する.

評価 1 手法 3 が最も高く, 手法 2 が最も低い結果となった. 手法 3 は拍節を考慮したため, 同じ音が連続せず, より正解データに近づいたと考えられる.

評価 2, 9 評価 9 の結果, 手法 3 が正解の値に近く, 多くの音名を使用してベースラインを生成している. 反対に手法 1, 2 は平均 4.20 個と 5.40 個の音名を使用して生成している. 評価 2 の結果も合わせると, 生成されたベースラインは各コードのルート音が多く, 単調になっていると考えられる. また, 実際の正解データを見ると順次進行が多いものと, ルート音を繰り返すものが存在していたため, 評価 2 の正解データの結果が高くなったと考えられる.

評価 3, 4 手法 1, 2 は評価 3 と評価 4 の差が大きく出たことから, 1 拍目にコードのルート音が出現する確率は低いが, コードのコードトーンは多いとわかった. 手法 3 は評価 4 で 96.36%という結果が出ており, 正解データとほとんど同じ値であるため, 1 拍目はコード進行に対応した生成ができていると考えられる.

評価 5 手法 3 が最も不協和音が多い結果になった. しかし, 手法 3 は手法 1, 2 と比べ, 単調ではないベースラインが生成され, 経過音が不協和音になってしまったと考えられる. よって, 手法 3 は より動きがあるベースラインが生成できた結果となったといえる.

評価 6, 7, 8 手法 1, 2 は評価 6 の値が大きいため, 同じ音が連続する単調なベースラインが生成されたと考えられる. 評価 7, 8 の結果より, 大きな

動きが少なく, 滑らかなベースラインであるといえる. 一方, 手法 3 は評価 6 が最も低く, 評価 7 の割合が半分以上であるため, 滑らかで単調でないベースラインが生成されたと考えられる.

よって, 生成結果が単調なベースラインではなく, 滑らかで動きがあるベースラインが生成された手法 3 を用いてベースラインを自動生成する手法が, 最も有効であると考えられる.

4.2.3 主観評価

評価方法

提案する手法によって生成したベースラインがどの程度の割合で適切に生成されたか, ジャズのベース経験者 1 名に評価を行ってもらう. テストデータの 50 種類のコード進行 (50 種類× 3 手法+正解 1 個：合計 200 個のベースライン) を評価対象とし, 50 個の平均を求める. 1 つのコード進行に複数の正解データがある場合は, 休符が含まれていないものを 1 つ選び, そのコード進行の正解率とする. 評価の際には, ベースラインとコード進行が書かれた楽譜を見て, さらにコード楽器とベースラインが記録された音源を聴いて行う.

以下の項目に対して評価を行う.

評価 1 ベースラインに対して, 不適切と思われる音符に丸を付ける.

評価 2 全体的な質を 5 段階で評価する.

評価 3 全体的な滑らかさを 5 段階で評価する.

評価 4 コードとの調和を 5 段階で評価する.

なお, 評価 1 は値が小さいほうが良い結果であり, それ以外は値が大きいほど良い結果とみなす.

評価結果・考察

HMM で主観評価を行った結果を表 4.2 に示す. 結果は小数点以下第三位を四捨五入する.

評価 1 正解データより手法 3 のほうが不適切な音が少ないという結果となった. 手法 2 はすべてのベースラインの中で最も多い 9 個の不適切な音が生成されたため, 平均が高くなったと考えられる. これは, 同じ音が連続して生成されることがあり, 不適切な音が続いてしまったためである. 手法 1 に関しても同様のことがいえる.

評価 2 手法 3 が最も高く, 手法 2 が最も低い結果となった. 手法 2 が低い結果になった原因は, 跳躍進行や同音進行が多いベースラインが他の手法より多く生成されたことが考えられる.

評価 3 手法 3 が最も高く, 手法 2 が最も低い結果となった. 同音進行が多い場合は「どちらともいえない」が選択される傾向があったため, 評価 2, 4 より全体的に低い結果となったと考えられる.

評価 4 手法 3 が最も高く, 手法 2 が最も低い結果となった. 他の評価より比較的値が高く, コードの情報を HMM に組み込んだためと考えられる.

よって, 生成結果が不適切な音が少なく, すべての評価で最も高い結果となった手法 3 を用いてベースラインを自動生成することが, 最も有効であると考えられる.

表 4.1: HMM の客観評価の結果

方法 / 項目	手法 1	手法 2	手法 3	正解
評価 1	38.47%	35.32%	41.75%	-
評価 2	54.79%	56.55%	32.89%	51.33%
評価 3	53.88%	52.91%	76.46%	90.29%
評価 4	85.68%	81.07%	96.36%	98.05%
評価 5	11.10%	17.05%	24.33%	25.36%
評価 6	51.13%	49.19%	17.67%	41.88%
評価 7	31.59%	28.67%	53.07%	66.21%
評価 8	17.28%	22.14%	29.26%	43.04%
評価 9	4.20 個	5.40 個	7.78 個	8.68 個

表 4.2: HMM の主観評価の結果

方法 / 項目	手法 1	手法 2	手法 3	正解
評価 1	1.40 個	1.88 個	0.60 個	0.98 個
評価 2	3.52	3.00	3.70	3.96
評価 3	3.40	2.96	3.80	3.96
評価 4	3.54	3.16	3.82	4.10

4.3 GA を用いた場合のベースライン生成

4.3.1 GA の世代数の決定

GA で生成するときの世代数を決定するために, 重み値をすべて 1 で固定し, 複数の世代数ごとに 10 回ずつ生成を行う. 4.1 節のテストデータとの一致率を求め, その平均を求める. その結果を図 4.4 に示す.

この結果から, 世代数を 25 以上に増やしても一致率の平均はほとんど横ばいであるため, 世代数は 25 と設定する.

4.3.2 GA の重み値の決定

GA で生成するときの重み値を決定するために, 4.3.1 節で決定した世代数で固定し, 様々な組み合わせごとに生成を行う. 4.1 節のテストデータとの一致率を求め, その平均を求める. ただし, 1 ～ 7 の数字は 3.2.5 節の各パラメータに対応するものとする. よって, 1 は first, 2 は seq, 3 は chord, 4 は move, 5 は off, 6 は ent, 7 は reg とする. その結果を図 4.5, 4.6 に示す. 図の横軸は重み値が 1 のパラメータを示している.

この結果から, chord のみを 1 とした場合が最も一致率が高いため, 次に chord との組み合わせが最も高くなるものを探す. その結果を図 4.7, 4.8 に示す.

この結果から, chord と ent の重み値を 1 の場合が最も一致率が高い. しかし, GA で実装する難易度にかかわる move と off が含まれていないため, 次に chord と move と off との組み合わせが最も高くなるものを探す. その結果を図 4.9 に示す.

この結果から, 最も一致率が高いのは図 4.8 より, chord, move, off のみの重み値が 1 の場合である. よって, chord, move, off の重みを 1, それ以外は 0 と

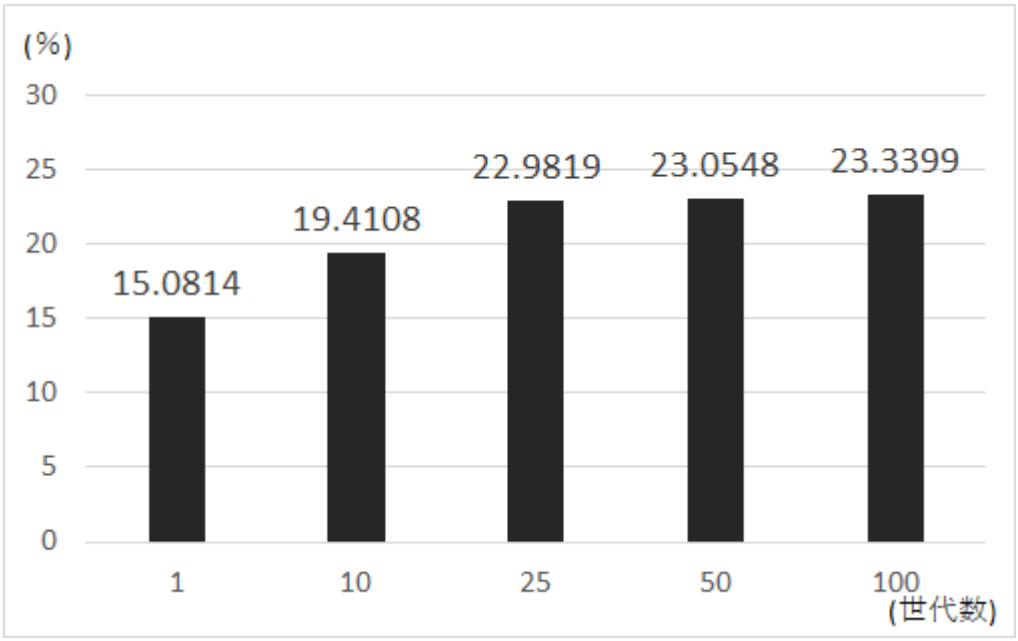

図 4.4: 世代数ごとの一致率

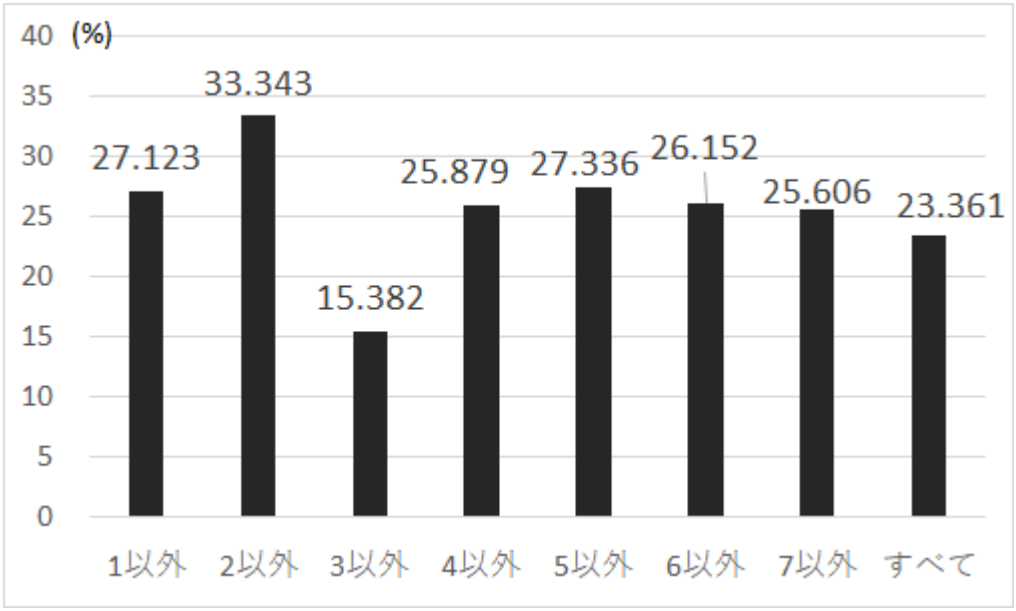

図 4.5: 重み値が 1 つだけ 0 それ以外 1, またはすべて 1 の場合

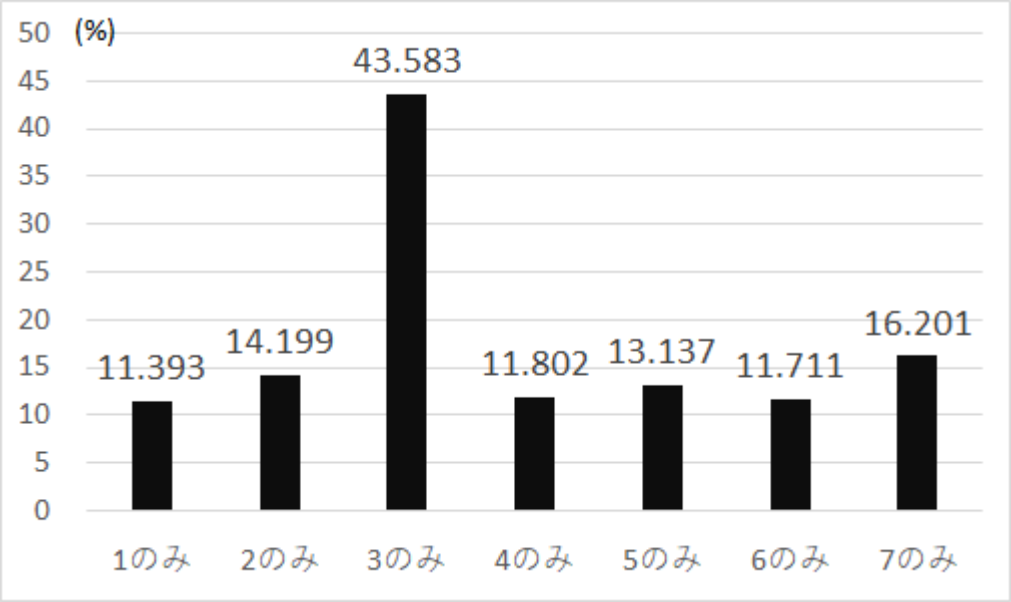

図 4.6: 重み値が 1 つだけ 1 それ以外 0 の場合

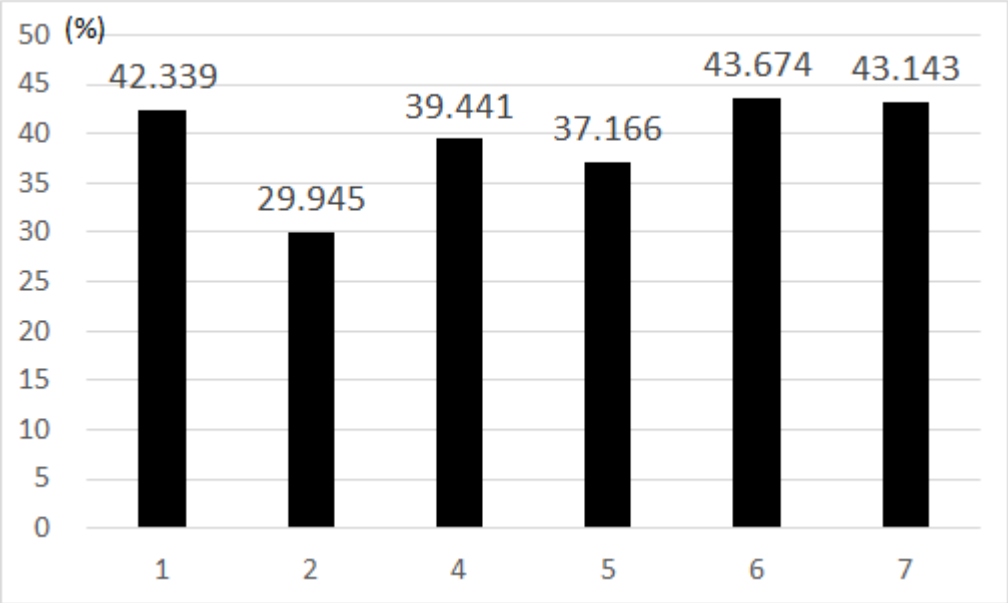

図 4.7: 重み値が 3 と他 1 つは 1 それ以外 0 の場合

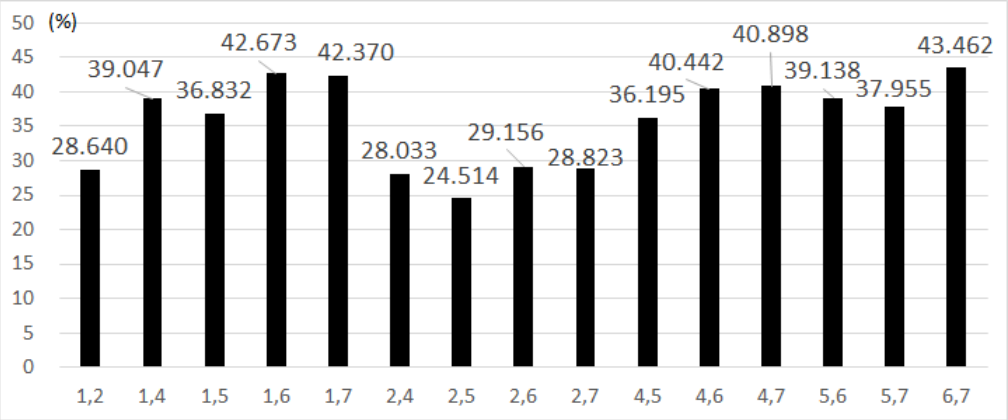

図 4.8: 重み値が 3 と他 2 つは 1 それ以外 0 の場合

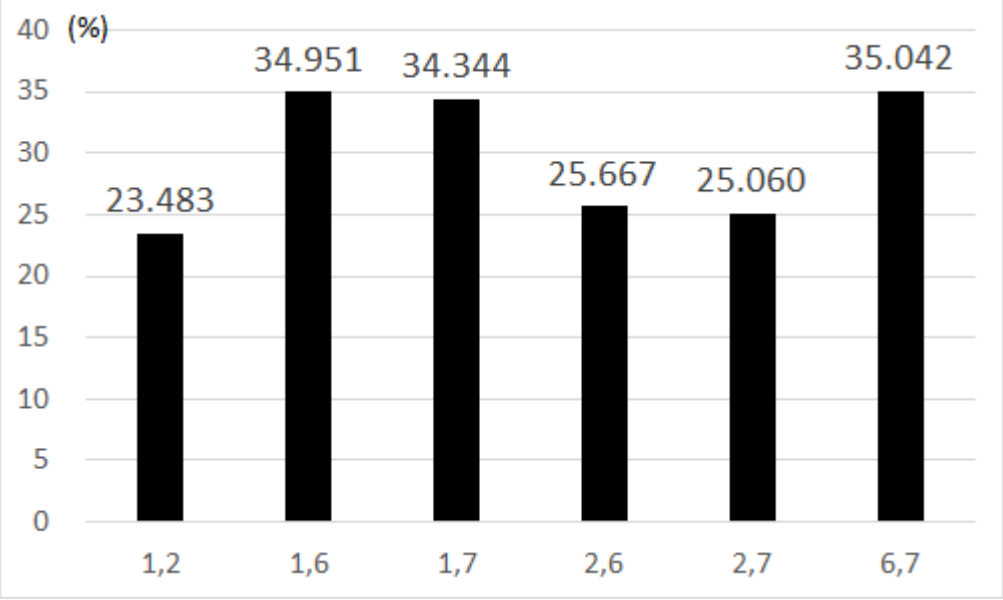

図 4.9: 重み値が 345 は 1 それ以外で 01 の組み合わせの場合

設定し, GA で生成を行う.

4.3.3 実行例

GA で自動生成を行った実行例を図 4.10~4.12 に示す. 4.2.1 節と同様に, コード進行 D $B^{\flat}$ $F^{\sharp}$ D, Em Em A Dm, Gm C F $B^{\flat}$ をそれぞれ入力として与え, 生成を行った結果である. ただし, 重み値は 4.3.2 節で設定したものとする.

- 図 4.10

  初級 すべて 4 分音符で生成され, 各小節の 1 拍目は対応するコードのルート音で構成された. 3 小節目の 1 拍目から 2 拍目以外は滑らかなベースラインが生成された. 実際に弾いてみると, 3 小節目の 1 拍目から 2 拍目への移動が同じフレット上の 4 弦から 2 弦への移動であったため, 弦の移動量が大きく演奏が難しくなっていた.

  中級 すべて 4 分音符で生成され, 各小節の 1 拍目は対応するコードのコードトーンで構成された. すべての小節で指の移動量が多い結果となった. 実際に弾いてみると, 2 小節目の 1 拍目から 2 拍目への移動が 4 弦から 1 弦への移動であり, 2 小節目の 4 拍目から 3 小節目の 2 拍目までフレットと弦の移動量が大きくなっている. さらに, 3 小節目の 3 拍目から 4 拍目もフレットの移動が多く, 演奏が難しくなっていた.

  上級 各小節の 1 拍目は対応するコードのルート音で構成された. 2, 3, 4 小節目は 4 分音符と 8 分音符で生成されたが, 2, 3 小節目には非コードトーンも生成された. 実際に弾いてみると, 次に弾く音を考えて押さえる場所を工夫すると演奏しやすいベースラインとなっていた.

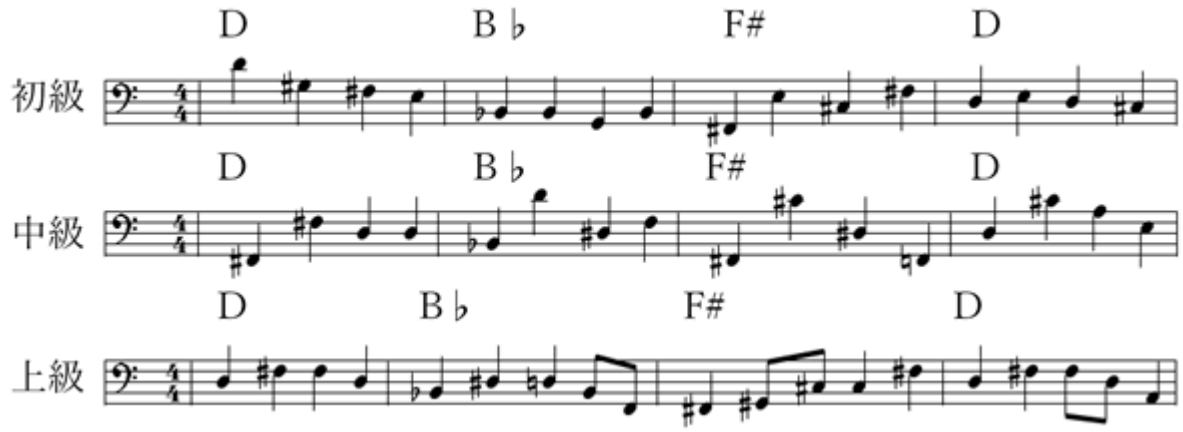

図 4.10: コード D $B^{\flat}$ $F^{\sharp}$ D

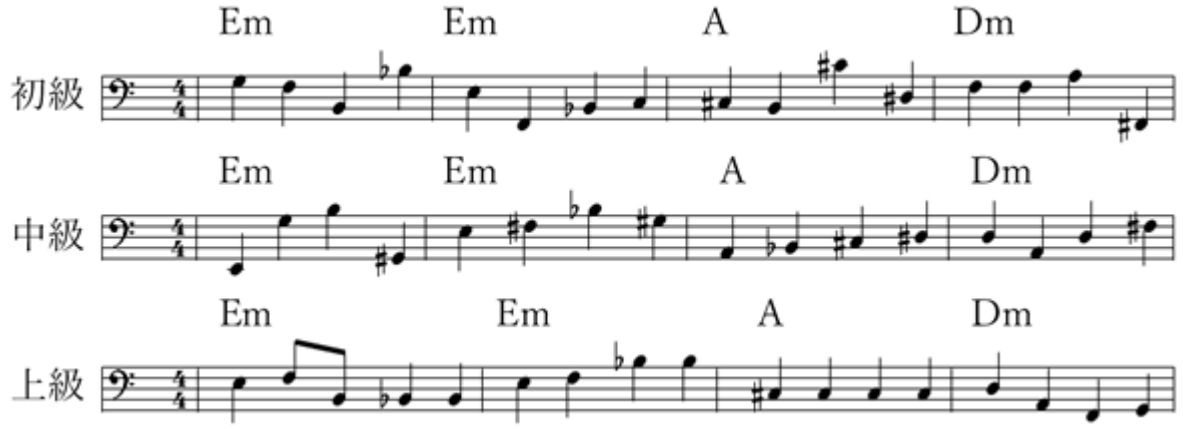

図 4.11: コード Em Em A Dm

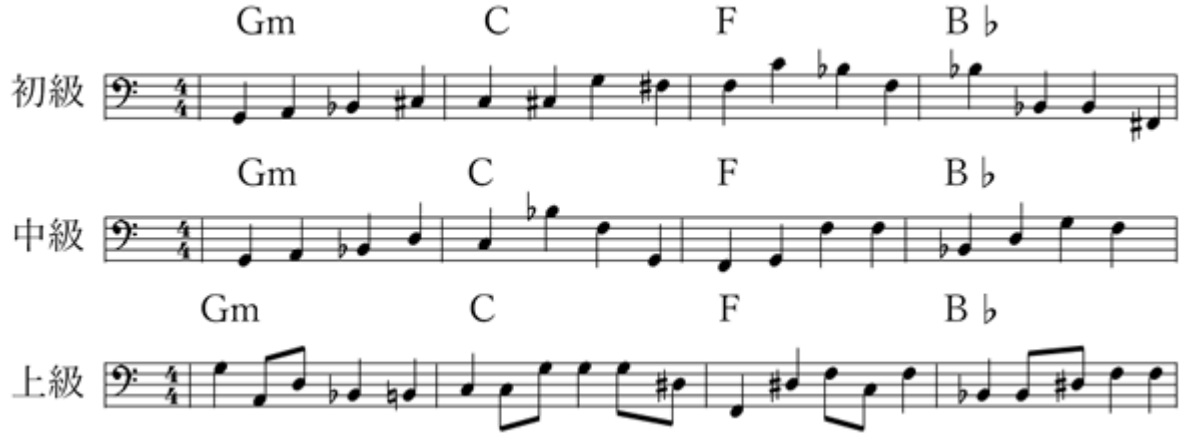

図 4.12: コード Gm C F $B^{\flat}$

– 図 4.11

初級 すべて 4 分音符で生成され, 各小節の 1 拍目は対応するコードのコードトーンで構成された. 指の移動量が多く, すべての小節で非コードトーンが生成されたが, 2 小節目では特に多く生成された. 実際に弾いてみると, 3 小節目の 2 拍目から 3 拍目が 3 弦から 1 弦の移動に加え, フレットの移動量も大きいため, 演奏が困難である. さらに, 4 小節目の 3 拍目から 4 拍目が 1 弦から 4 弦への移動が生成された.

中級 すべて 4 分音符で生成され, 各小節の 1 拍目は対応するコードのルート音で構成された. 1 小節目に 1 オクターブ以上の動きが 2 回も生成された. 1, 2, 3 小節目で非コードトーンが生成されたが, 2 小節目が特に多くなった. 実際に弾いてみると, 1 小節目の 3 拍目から 2 小節目の 1 拍目への移動が 1 弦から 4 弦そして 4 弦から 2 弦で大きく, 演奏が難しい.

上級 各小節の 1 拍目は対応するコードのコードトーンで構成された. 3 小節目が単調であり, 1, 2, 4 小節目では非コードトーンが生成された. 実際に弾いてみると, 大きな指の移動はなく演奏しやすいベースラインを生成することができた.

– 図 4.12

初級 すべて 4 分音符で生成され, 各小節の 1 拍目は対応するコードのルート音で構成された. すべての小節に非コードトーンが存在し, 4 小節目は 1 拍目から 2 拍目で 1 オクターブの動きが生成されたが, 実際に弾いてみると大きな移動がなく演奏がしやすいベースラインであった.

中級 すべて 4 分音符で生成され, 各小節の 1 拍目は対応するコードの

ルート音で構成された. すべての小節に非コードトーンが生成された. 実際に弾いてみると, 2 小節目の 1 拍目から 2 拍目, 3 拍目から 4 拍目, 3 小節目の 2 拍目から 3 拍目で同じフレット上の弦の動きが大きく, 演奏がやや困難なベースラインとなった.

上級 各小節の 1 拍目は対応するコードのルート音で構成された. すべての小節に 8 分音符が存在し, 2 小節目には 8 分音符が 4 つ生成された. さらに, すべての小節に非コードトーンが存在するが, 1 小節目の 4 拍目は経過音とみなせる. 実際に弾いてみると, 2 小節目の 4 拍目から 3 小節目の 2 拍目までの「レ♯ファレ♯」の動きが押さえる位置によっては演奏しづらいが, その他は演奏しやすいベースラインであった.

– 生成結果を見ると, 全体的に 1 オクターブ以上の移動がよく生成されている. これは, 適応度関数の遷移確率などの多くの計算でオクターブを考慮していないためと考えられる. オクターブの情報を保持したままにするためには多くのデータが必要となるが, 本研究でのデータ数では足りなかったためその情報をなくしたことがこのような生成結果となった原因と推測される.

4.3.4 客観評価

評価方法

4.2.2 節と同様に, 生成したベースラインの検証を行う. 評価方法と評価項目も 4.2.2 節と同様である.

この検証は, GA を用いて難易度ごとにベースラインを生成した 3 通りのベースラインと正解データに対して行う. なお, 評価 1 は値が高いほうが良い結

果であり, それ以外は正解データの値に近いほど良い結果とみなす. ただし, 重み値は 4.3.2 節で設定したものとする.

評価結果・考察

GA で客観評価を行った結果を表 4.3 に示す. 結果は小数点以下第三位を四捨五入する.

評価 1 上級が最も一致率が高い結果となった. 初級, 中級が変わらず, 上級のみ差があったことから, 適応度関数の off のパラメータを大幅に変え, 4 分音符だけでなく 8 分音符も生成されたことが一致率が上がった原因と考えられる.

評価 2 上級が最も良い結果であった. 評価 1 と同様, 上級だけ高くなったことから, 適応度関数のパラメータの調節により, このような結果になったと考えられる.

評価 3, 4 評価 3 では上級が正解を上回る結果となった. さらに, 評価 4 ではすべての結果が正解より良い結果であり, 上級は 1 拍目が 100%コードトーンで生成された. これは適応度関数の chord のパラメータがうまく作用したためと考えられる.

評価 5 初級が最も不協和音が多い結果となった. 初級はすべて 4 分音符で生成されるため, 不協和音が生成された場合も必ず 8 分音符が 2 つ生成されることが他の難易度より割合が高くなった原因であると考えられる.

評価 6 GA の正解データの結果が HMM の結果と大きく違うのは, GA の正解データは 8 分音符で構成されており, 正解データが 4 分音符の場合は 8 分音符が 2 つ連続することになるからである. よって, 同音進行の割合が大幅に上昇したと考えられる. また, 中級が最も高い結果となり, そ

の原因は適応度関数の off のパラメータの調整が足りず, 8 分音符が少なかったためと考えられる.

評価 7, 8 両方とも上級が最も良い結果であった. しかし, すべての難易度で評価 8 のほうが値が高く, 滑らかではないベースラインが生成された. 適応度関数の move のパラメータの調節が原因と考えられる.

評価 9 初級が一番高くなった. 4.3.2 節の結果より, 適応度関数のパラメータに ent を含めなかったことで, エントロピーの制御ができず, 上級より初級が高くなったと考えられる.

よって, GA を用いて 3 つの定義に沿ったベースラインを生成することができた. しかし, 難易度が上がるにつれて生成されたベースラインの音楽的な良さがどのように変化したのか, 本研究では検証できていない.

表 4.3: GA の客観評価の結果

難易度 / 項目	初級	中級	上級	正解
評価 1	36.20%	36.47%	40.66%	-
評価 2	37.17%	35.13%	41.02%	52.00%
評価 3	80.58%	76.94%	94.66%	90.29%
評価 4	98.30%	99.51%	100.0%	98.06%
評価 5	18.45%	16.99%	13.90%	25.30%
評価 6	58.88%	59.41%	53.59%	71.59%
評価 7	17.44%	16.16%	20.26%	32.16%
評価 8	23.68%	24.43%	26.15%	20.98%
評価 9	7.66 個	7.55 個	7.61 個	8.71 個

第5章　結　　論

本稿は, ユーザがジャズのベースを練習する際に生じる, “コードを見ただけでは何の音を弾いたらいいかわからない”, “インターネット上にすべての曲に対してベースラインの演奏例があるとは限らない”, “演奏例が演奏者の演奏レベルに合うとは限らない” の 3 つの問題の解消を狙ったものである.

まず, コードを見ただけでは何の音を弾いたらいいかわからない問題, インターネット上にすべての曲に対して演奏例があるとは限らない問題については, ベースラインの自動生成を HMM を用いて行った. 3 つの手法でベースラインの生成を行い, それぞれのベースラインがどの程度適切であったかを検証した. 学習データとテストデータをそれぞれ 103 コード分用意し, テストデータを正解データとして使用した. 客観評価を行ったところ, 手法 3 が正解データとの一致率が 41.75%と最も高い結果となった. その他の客観評価も 8 項目中 6 項目で 3 つの手法の中で手法 3 が正解データに最も近い結果であった. また主観評価でも 4 つの項目すべてで手法 3 が良い結果となった. 以上のことから, 拍節を考慮した手法 3 によるベースラインの自動生成が最も有効であるとわかった.

次に, 演奏例が演奏者の演奏レベルに合うとは限らない問題については, 3 つの難易度からユーザに選択してもらい, その難易度にあったベースラインを HMM の手法 3 の考え方を基本として, GA を用いて実装する手法を提案した. まず世代数を決定するために, 適応度関数の重み値をすべて 1 で固定し, 複数の世代数ごとに自動生成を行い, 正解データとの一致率を求めた. その結果, 世代数を 25 から増やしても一致率があまり変わらなかったため, 世代数を 25 と決定した. 次に, 重み値

を決定するために世代数を固定し, 様々なパラメータの組み合わせで自動生成を行い, 正解データとの一致率を求めた. 単体での精度が 43.58%と圧倒的に高い chord に加え, 難易度を変化させる move と off を含めた組み合わせの中で最も一致率が高いものを GA での生成に使用した. この実験の結果から, chord, move, off の 3 つのみを使ってベースラインを生成する手法が 36.2%で最も高かった. 次に, 実験で求めた世代数と適応度関数を使用して生成を行ったところ, 初級, 中級, 上級それぞれの難易度の定義 (8 分音符の割合, 指の移動量) に沿ったベースラインが生成された. しかし, 難易度が上がるにつれて生成されたベースラインの音楽的な良さがどのように変化したのか, 本研究では検証できていない.

今後は, GA の難易度ごとの自動生成の音楽的良さに対しての検証を行いたい. さらに, 入力できるコードの種類を増やすことで, より初心者が練習しやすいシステムにしていきたい. また, 実装したベースライン自動生成システムを利用して, ユーザがベースラインを練習することができるか検証していく予定である. さらに, 主観評価をより多くの専門家にお願いをする予定である.

参考文献

[1] Francois Parchet, Pierre Roy：“Musical Harmonization with Constraints: A Survey”, Constraints Journal, Vol.6, No.1, pp.7–19, 2001.

[2] 川上 隆, 中井 満, 下平 博, 嵯峨山 茂樹:“隠れマルコフモデルを用いた旋律への自動和声付け”, 情報処理学会研究報告音楽情報科学 (MUS), Vol.2000, No.19, pp.59–66, 2000.

[3] 菅原 啓太, 西本 卓也, 嵯峨山 茂樹：“HMM と音符連鎖確率を用いた旋律への自動和声付け”, 情報処理学会研究報告音楽情報 (MUS), Vol.2003, No.127, pp.49–54, 2003.

[4] 平田 圭二, 青柳 龍也：“パーピープン：ジャズ和音を生成する創作支援ツール”, 情報処理学会論文誌, Vol.42, No.3, pp.633–641, 2001.

[5] 後藤 真孝, 平田 圭二：“ハービー君：演繹オブジェクト指向に基づいてジャズらしいコードにリハーモナイズするシステム”, 情報処理学会研究報告音楽情報 (MUS), Vol.1996, No.75, pp.33–38, 1996.

[6] 三浦 雅展, 青山 容子, 谷口 光, 他：“ポップス系の旋律に対する和声付与システム：AMOR”, 情報処理学会論文誌, Vol.46, No.5, pp.1176–1187, 2005.

[7] 小中 裕喜, 田中 英彦：“ICOTone：立派なベース LPB88”, 情報処理学会全国大会講演論文集, Vol. 第 37 回, No. 応用システム, pp.1964–1965, 1988.

[8] Rui Dias, Carlos Guedes : “A CONTOUR-BASED JAZZ WALKING BASS GENERATOR”, SMC 2013, Stockholm, Sweden, pp.305–308, 2013.

[9] 國松 香苗, 石川 由羽, 高田 雅美, 城 和貴 : “GP を用いた音楽自動生成-コード進行とベースラインの一事例-”, 情報処理学会研究報告数理モデル化と問題解決 (MPS), Vol.2015-MPS-104, No.7, pp.1–4, 2015.

[10] 矢澤 一樹, 糸山 克寿, 奥乃 博 : “ギター演奏者の習熟度に合わせた音響信号からのタブ譜自動生成”, 情報処理学会研究報告音楽情報科学 (MUS), Vol.2013-MUS-100, No.17, pp.1–6, 2013.

[11] 中村新太郎 : “ジャズ・ベース・ランニング 104 実例集 1”, サーベル社, 2017.

[12] 中村新太郎 : “ジャズ・ベース・ランニング 104 実例集 2”, サーベル社, 2017.

[13] 中村新太郎 : “ジャズ・ベース・ランニング 104 実例集 3”, サーベル社, 2017.

[14] http://www.projazzlab.com/study-tools/

謝　　辞

本論文を作成するにあたり, 指導教員の北原鉄朗准教授から, 丁寧かつ熱心なご指導を賜りました. 的確な指導や多くの議論によって, 自身の研究がより良いものとなりました. ここに感謝の意を表します.

また, 生成したベースラインの主観評価を快く受けてくださった鈴木唯思氏, 困っている時に手を差し伸べていただいた北原研究室の先輩である石山先輩, 切磋琢磨し研究に励んだ北原研究室の同期, 後輩の皆様に感謝いたします.