社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－MUS－74 (28)
2008－SLP－70 (28)
2008／2／9

対話型作曲システム i-Sonneteer の他楽曲形式への応用

蓮井洋志* 小倉久和†

*室蘭工業大学情報工学科 †福井大学大学院応用理工部門知能システム工学専攻

我々は対話型進化的計算手法による作曲システム i-Sonneteer を作成している。HMM はメロディーを学習し、そのデータをもとに生成する。HMM が生成したメロディーをユーザが GUI で自分の好みに合うように編集し、編集したメロディーをうまく選択して HMM が学習する。その結果として、HMM をユーザの好みのモデルに最適化でき、最終的に好みを表したメロディーが生成できると考える。進化的計算手法においては、HMM を個体とし、HMM が学習するメロディーを遺伝子とする。染色体は 7 つの遺伝子を持つ。HMM が生成したメロディーをユーザが評価する。その評価値を HMM の適応度とする。そのため、適応度評価関数を考える必要が無く、簡単に多くの楽曲形式に応用することができる。本研究では、このシステムに伴奏定義ファイルを組みこんで、他楽曲形式に応用した。ショパン風のワルツやニューミュージックに応用した例について述べる。

Application of Interactive Composing System, i-Sonneteer, to Other Musical Styles

Hiroshi Hasui* Hisakazu Ogura†

*Department of Computer Science and Systems Engineering, Muroran Institute of Technology

†Graduate School of Human and Artificial Intelligent Systems, Fukui University

We implement the composing system, i-Sonneteer, with the interative evolutionary computation. HMM learns melodies and generates the melody based on learned data. The user edits the melodies that the HMM generated to favored one, and the HMM learns the melodies that selected from improved melodies well. The HMM is optimized into the model of user's favorite, and can generate the pleasant melody, finally. In the evolutionary computation, an individual is HMM and a gene is a melody which the HMM learns. A chromosome has 7 genes. The user estimates the melody that the HMM generated. Since the estimation value is the fitness of the HMM generated it, the system does not need the fitness function. As a result, the system is applied to many musical styles easily. In this study, we apply the system to other musical styles with the accompaniment definition file. We state about the event that it was applied to the waltz style in Chopin's fashion and the newmusic style.

1 まえがき

多くの楽曲形式に応用できる自動作曲システムは多い。Band-In-a-Box などは作曲する楽曲形式を表すスタイル数が 100 を越えている。しかし、これらのシステムの作る音楽は好みにあった楽曲形式の伴奏を生成できるが、すべてのユーザにとって、好みのメロディーを作ることはできていない。

自動作曲システムには、学習型のシステムと進化型のシステムがある。学習型のシステムは、学習する音楽の特徴を受け継ぐために、異なった雰囲気のメロディーを作るのが難しい。学習する音楽がその日の気分に合わなければ、好みのメロディーは作れない。進化型のシステムは音楽理論にあわないメロディーの作曲が実現難しい。多くの人間の好みのメロディーは音楽理論に合致している例は少なく、進化型のシステムは少数の人間の好みに合ったメロディーしか作れない。

我々は、対話型作曲システム i-Sonneteer を作成している。このシステムは、HMM をユーザの好みのモデルに最適化することを目的としている。i-Sonneteer は対話的進化的計算手法を用いて作成した。個体として HMM を使い、遺伝子としてメロディーを考えた。HMM は遺伝子の指定するメロディーを学習し、その学習データを用いてメロディーを生成する。ユーザは HMM の生成するメロディー

を好みのメロディーに修正する。HMMはユーザの修正したメロディーからうまく選択して学習することで、ユーザの好みのメロディーを生成できると考える。

i-Sonneteerは一種類の単純な伴奏パターンを用いていた。そのため、ユーザの好みにあったメロディーを生成できても、伴奏が好みではないために、文献[1]の実験では20世代までのすべての音楽の適応度が0.90以下と、本当に満足できる音楽の作曲ができなかった。多くの楽曲形式に対応した伴奏をつけることができれば、この問題を乗り越えることができると考えた。

本研究では、i-Sonneteerにさまざまな伴奏をつける機能を追加した。この機能を利用してショパン風のワルツ、ニューミュージックを作曲する。伴奏定義ファイルの設定を代えれば、様々な楽曲形式に応用できる。演奏にはMIDI音源装置SC-88Proを利用した。MIDI音源装置を使うことで臨場感を増すことに成功した。

本論文では、2ではi-SonneteerによるHMMの最適化方法について、3では他楽曲形式への応用方法について述べ、4で実際に作った楽曲について述べる。最後に5で考察する。

2 i-SonneteerによるHMMの最適化方法

2.1 対話型進化的計算手法

i-Sonneteerは、対話型進化的計算手法を用いてHMMを最適化する対話型作曲システムである。対話はGUIで行なう。ユーザがi-Sonneteerの生成したメロディーを評価、修正する。評価は0から1の数字で行ない、その値が評価したメロディーを生成したHMMの適応度となる。

HMMは遺伝子によって指定されたメロディーを学習し、その結果を利用してメロディーを生成する。一つの個体は7個の遺伝子で構成される。遺伝子はメロディーを表し、音楽番号でメロディーを指定する。7つのメロディーはユーザが修正したメロディーの集合から選択する。この選択を最適化することで、好みのメロディーを生成するHMMを作りあげる。

進化的計算手法には、適応度比例戦略とエリート保存戦略をあわせた遺伝的アルゴリズムを用いる。個体の数は10個とした。適応度の高い1個の個体はエリート解となり、1個体はエリート解の複製をし、残りの8個の個体が適応度比例戦略で交叉する。

突然変異率は0.1である。突然変異では遺伝子をランダムで1世代前に修正したメロディーの音楽番号に変更する。複製された個体は突然変異の対象となるが、エリート解はならないものとした。その他の遺伝的な操作として、親の生成したメロディーが持つ音楽番号が子の染色体に1つだけ残るようにした。

2.2 HMMによるメロディーの学習

HMMには、逐次的リズム音程生成モデル[2]によるリズム生成方法を参考に、音高の生成もできるように応用したモデルを用いる。これをk-measure HMMと呼ぶ。

k-measure HMMの学習方法を提案する。k小節ごとにメロディーの学習、生成を行なう。8小節のメロディーの場合、$(8/k)$回学習する。

音符の属性としては、開始半拍数、音高、音符の長さの3つがある。開始半拍数iを$(k\times l+1)$小節目の最初からの半拍数とし、音高をp、音符の長さをdと記述する。dの単位は半拍である。例えば、音高pが4オクターブ目のCである場合、$p=C4$と書く。また、休符も音高で$p=REST$と記述する。音の強さは一定とする。例えば、$k=2$の場合は、開始拍数が3小節目の3拍目で音高は4オクターブ目のDの4分音符は、$l=1$、$i=4$、$p=D4$、$d=2$である。

開始半拍数をiとしたとき、状態のラベルをn_iとする。この状態がn_iからn_jに遷移した場合、音符の長さは$d=j-i$で、この状態の記号出力は音高pである。

学習は、HMMの状態遷移確率などを適応化することによって行なう。学習手順を簡単に説明する。音符の開始半拍数がi、音高がp、音符の長さがdの場合、状態はn_iからn_{i+d}に遷移する。また、状態n_iのときの記号はpである。$i+d>8\times k$の場合は、$n_{8\times k+1}$に遷移する。メロディーの全ての音休符に対して、各状態から各状態への遷移した音符の数、各状態における各記号の音符の数を数える。k-measure HMMの学習パラメータは以下のように算出する。

$$P_i(i)=\frac{(k\times l+1)\text{小節目の最初の音符の状態が}\ n_i\text{の数}}{\text{学習する全小節数を}\ k\ \text{で割った数}} \tag{1}$$

$$P_i(j|i)=\frac{n_i\text{から}\ n_j\text{に遷移する音休符の数}}{n_i\text{の音休符の数}} \tag{2}$$

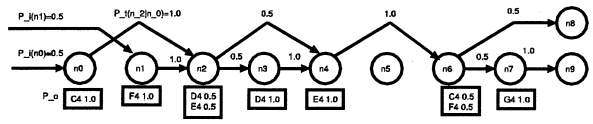

図 1: 学習した 1-measure HMM

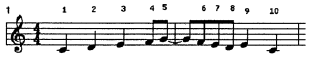

図 2: 学習するメロディーの楽譜

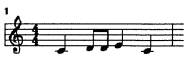

図 3: 1-measure HMM が生成したメロディー

$$P_o(o|i) = \frac{n_i\text{の記号が } o \text{ である音休符の数}}{n_i\text{の音休符の数}} \quad (3)$$

これらを、k-measure HMM の初期状態確率 $P_i(i)$、状態遷移確率 $P_t(j|i)$、記号出力確率 $P_o(o|i)$ とする。

図 2 の楽譜を 1-measure HMM で学習する様子を説明する。音符 j は楽譜の最初から見て j 番目の音符を示す。音符 1 は $l=0$、$p=C4$、$i=0$、$d=2$ である。この小節の最初の音符であるために初期状態が n_0 となる。

音符 1 は状態はノード n_0 から n_2 に遷移する。状態 n_0 にときの記号は $C4$ である。音符 2 は $l=0$、$p=D4$、$i=2$、$d=2$ である。状態はノード n_2 から n_4 に遷移する。状態 n_2 にときの記号は $D4$ である。音符 3、4 は同様に学習する。音符 5 は次の小節にタイでつながる音符である。この音符の場合は n_9 に遷移する。n_9 は次の小節にタイでつながる音符の遷移するノードである。状態が n_7 から n_9 に遷移し、状態 n_7 の記号は $G4$ となる。

音符 6 は、2 小節目の第 1.5 拍で、各パラメータは $l=1$、$p=F4$、$i=1$、$d=1$ である。この小節の最初の音符であるために初期状態が n_1 となる。初期状態が n_0 と n_1 の 2 つになった。両方の状態とも 1 音符だけしか存在しないので初期状態確率は、両方とも 0.5 となる。

音符 6 は状態はノード n_1 から n_2 に遷移する。状態 n_1 の記号は $F4$ である。音符 7 は $l=1$、$p=E4$、$i=2$、$d=1$ で n_2 から n_3 に遷移する。n_2 から遷移する音符は音符 2 と音符 7 があり、それぞれ状態 n_4、n_3 に遷移する。2 つあるために各々の状態への遷移確率は 0.5 となる。記号も $D4$ と $E4$ の 2 つがあり両方とも記号出力確率は 0.5 となる。音符 11 は小節のくぎりで終るために n_8 に遷移する。n_8、n_9 は出力記号を持たない。

図 2 の楽譜を 1-measure HMM が学習した結果、図 1 のようになる。長方形に囲まれた数字が記号出力確率、2 つのノードを結ぶ矢印の上の数字が状態遷移確率である。初期状態確率は左端の矢印の上の数字である。

2.3 HMM によるメロディーの生成

k-measure HMM は学習したデータに基づいて、k 小節のメロディーを生成する。k-measure HMM は 1 音符あたりの平均の状態遷移確率が一番高いリズムを探索する。探索には Dijkstra 法を用いた。各々の状態遷移確率の対数の逆数を取って $value$ を減算したものを距離とし、距離の和が一番小さいノード列を探索する。n_0 から $n_{8\times k}$ への最小距離を持つノード列がリズムを表す。

$value$ は、0 ではない状態の数を n、0 ではない状態遷移確率を $p_{state,i}$ とすると以下の式で表される。

$$value = -\frac{\sum_i log(p_{state,i})}{n}$$

音符の開始半拍数だけを考えた場合の 1 音符あたり生起確率の対数を $value$ とした。音符列のリズムの生起確率は状態遷移確率の積で計算されるために、すべての確率の積を計算する。確率が 0 の状態遷移には音符がないために生成される可能性がない。これは $value$ の計算の対象外とした。距離の和を計算するときに、$value$ を減算しないと、音符数の少ないメロディーが生成されてしまう。

前節で学習した 1-measure HMM では、n_0, n_2, n_3, n_4, n_6, n_8 が最小距離のノード列である。これ

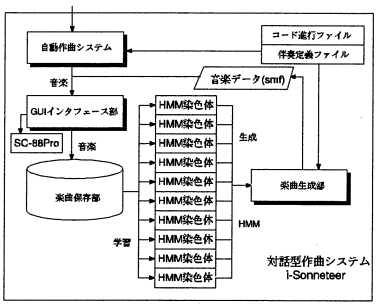

図 4: 対話型作曲システムの構成

は、4 分音符、8 分音符、8 分音符、4 分音符、4 分音符のリズムを表す。

音高は、音符の長さの列を決定したときの各状態における最大の記号出力確率を持つ記号を使った。もし同じ確率であれば、音高の低いものを使った。

図 3 に前節で学習した 1-measure HMM が生成したメロディーを示す。

3 対話型作曲システムの実現

3.1 システム構成

図 4 に対話型作曲システム i-Sonneteer の構成図を書く。この作曲システムは C++を用いて Linux 上で開発した。MIDI を用いて音楽を表現し、演奏には MIDI 音源装置 SC-88Pro を用いる。デバイスドライバにはフリーウェアの `srgplay-0.80` を用いた。

このシステムは自動作曲システム Sonneteer と GUI 部、楽曲生成部、楽曲保存部、対話型進化的計算部からなる。対話型進化的計算部には 10 の HMM 染色体がある。本研究の HMM 染色体は 8-measure HMM で学習する。

第 1 世代目に、自動作曲システムで作曲したメロディーをユーザが GUI 部で編集し、楽曲保存部に置く。対話型進化的計算部が、編集したメロディーを選択した初期個体を生成する。その結果出来た個体のメロディーを HMM 染色体が学習し、楽曲生成部でメロディーを生成する。それをユーザが GUI 部で編集、評価する。編集、評価時にメロディーを演奏する。

第 2 世代以降は、ユーザが評価した適応度から選択、交叉、突然変異を行い、HMM 染色体がメロディーを生成し、そのメロディーをユーザが編集、評価することをくり返す。ユーザが編集、評価時に満足なメロディーになったときに終わる。

```
alltrack: 6
channel: 1
banknumber: 0
tone: 39
channel: 2
banknumber: 0
tone: 12
channel: 3
banknumber: 0
tone: 1b
channel: 4
banknumber: 0
tone: 19
channel: 5
banknumber: 0
tone: 21
channel: 9
banknumber: 0
tone: 0
tempo: 100
measure: 8
beat: 4
drum: 8
# base pattern
.....
```

図 5: 伴奏定義ファイル (ニューミュージック)

3.2 伴奏パターンの設定

楽曲形式は基本的に音色、テンポ、拍子数、伴奏パターンとコード進行で決まる。音色はどの楽器でその形式の音楽を演奏するかが問題となる。最近のニューミュージックであれば、演奏はギター、ベース、シンセサイザー、ドラム、ボーカルで行なう。本システムでは、それぞれの演奏を MIDI 音源の音色で代用する。ボーカルはトロンボーンなどのブラス系の音色で代用する。それ以外の楽器は同じ名称の音色が MIDI 規格の中に存在する。

図 5 にニューミュージックの伴奏定義ファイルのヘッダの部分を書く。トラックは全部で 6 つで、1 トラックがメロディーパートでトロンボーン、2 トラックがオルガン、3,4 トラックがギター、5 トラックがベース、9 トラックがドラムである。ドラムパターンは 16 ビートを使う。テンポ 100 で、4 分の 4 拍子、小節数は 8 である。ニューミュージックの伴奏定義ファイルを作るにあたって文献 [3] を参考にした。

```
alltrack: 2
channel: 1
banknumber: 0
tone: 0
channel: 2
banknumber: 0
tone: 0
tempo: 140.0
measure: 16
beat: 3
drum: -1
# base pattern
1 -2 2 0.0 100 1.8
2 -1 2 2.0 98 1.9
3 -1 2 2.0 98 1.9
4 -1 2 2.0 98 1.9
2 -1 2 4.0 98 1.9
3 -1 2 4.0 98 1.9
4 -1 2 4.0 98 1.9
# pattern-1
3 -3 2 0.0 100 1.8
2 -1 2 2.0 98 1.9
3 -1 2 2.0 98 1.9
4 -1 2 2.0 98 1.9
2 -1 2 4.0 98 1.9
3 -1 2 4.0 98 1.9
4 -1 2 4.0 98 1.9
```

図 6: 伴奏定義ファイル (ショパン風ワルツ)

図 6 にショパン風ワルツの伴奏定義ファイルを示す。トラック数は全部で 2 で、両方とも音色番号 0、つまりグランドピアノで、ドラムは使わない。1 トラックはメロディーに、1 トラックは伴奏に使う。一小節の拍数は 3 で 3/4 拍子である。16 小節の音楽を生成する。テンポは 140 である。

11 行目以降は伴奏パターンである。1 番目の数字がコード構成音の番号で、2 番目の数字がコード構成音から何オクターブの音か、3 番目の数字がチャネル番号、4 番目が開始半拍数、5 番目がベロシティ、6 番目が音符の長さである。

パターンは、2 つ登録してあり、第 1 パターンは 1 拍目に 1 オクターブ下のコードの基音、2 拍目と 3 拍目はコードの第 2 構成音、第 3 構成音、第 4 構成音を押すことを表している。第 1 パターンは奇数番号の小節に使い、第 2 パターンは偶数番号に使う。最後の小節は偶数小節であっても第 1 パターンを使う。

伴奏定義ファイルは、ショパン風ワルツ、J-Rock、ニューミュージックを二種類の 4 つを用意した。また、コード進行はコード進行ファイルに登録する。楽曲形式とコード進行はユーザが起動前に選択する。

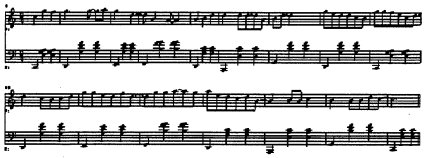

図 7: 作曲したショパン風ワルツ (自動作曲システム)

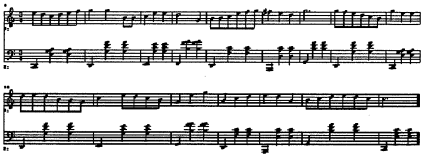

図 8: 作曲したショパン風ワルツ (対話型作曲システム)

4 実験

4.1 ショパン風ワルツの作品

自動作曲システム Sonneteer でショパン風のワルツを作曲した例を図 7 に示す。それに対して、対話型作曲システム i-Sonneteer で作曲した例を図 8 に示す。

対話型作曲システムのほうがリズムが整然として、音符の並びもスケールに沿ったものが多いことが分かるだろう。自動作曲システム Sonneteer では、図 8 にあるような 5 小節目のような意味のある 1 音符を抜かしたスケールに沿ってメロディーが進ませることは難しかった。

今回、作曲した曲は Sonneteer 設定次第では同じ曲を作曲できる可能性がある。しかし、設定を変更しても、自動作曲システム Sonneteer は、突然変異を使ってランダムに音符をならべているために、ユーザの好みにあった曲を生成できる可能性は低い。

4.2 ニューミュージックの作品

対話型作曲システムでニューミュージックを作曲した例を図 9 に示す。第 3 世代でできあがった。この曲は作者は満足行くできであったと言う感想を得た。被験者はニューミュージックが好きであるためであろう。

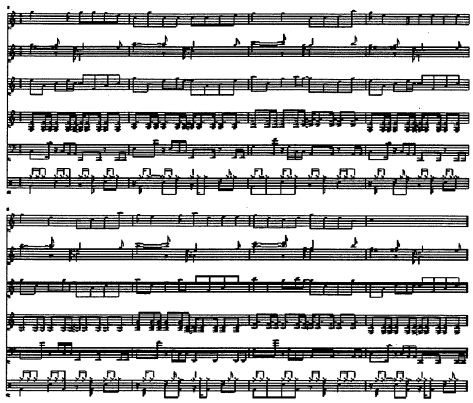

図 9: 作曲したニューミュージック

5 考察

伴奏パターンは 1 小節単位で登録する。2 小節にまたがったパターンは登録できない。普通、J-POP では 2 小節、4 小節単位でパターンが組まれることが多く、1 小節のパターンに直すことは楽曲スタイルを不完全なものにする。本研究では、1 小節のパターンに直しても不自然でないスタイルだけを選んで登録した。

伴奏定義ファイルでは、コード構成音だけしか伴奏として定義できない。コード構成音以外の音が使われている伴奏は多い。本研究では、コード外音は一番近いコード構成音に直して定義した。

ワルツはピアノで演奏することが基本であるために MIDI の演奏では使用者の評判が悪かったが、ニューミュージックは元来 MIDI を使うためか評判は良かった。

本システムは作曲したいメロディーがどういうものか明確化されていない場合に用いる。もし、明確なイメージがあるのであれば、自分で作曲したほうが速い。

ユーザの好みを明確化するためには、システムが好みと推測される多くの音楽を聞かせる必要がある。本システムを利用して作曲する場合には、30 曲から 50 曲のメロディーを評価したり、修正したりしなければならないが、これはシステムがユーザの好みを学習するためだけではなく、ユーザの好みを明確にするためにも必要な作業である。

本システムを使わずとも多分、作曲はできる。しかし、好みを明確化してから作曲するプロセスをとらない普通の作曲方法では、本当に作曲者の納得の行くメロディーができないと考える。

HMM は学習したメロディーを融合したものを生成しているために、ユーザが修正しないと新しい情報を獲得できない。最初に自動作曲システムが作曲したメロディーを融合するだけで終わってしまう。メロディーを評価するだけでなく、修正することも大切である。

対話型作曲システムはユーザもシステムの中の構成要素である。システムが学習することよりも、ユーザがどのくらいよい好みを見つけられるかのほうが大切である。もし、システムが生成する音楽が類似してきたら、一度異なった調子のメロディーをユーザに評価、修正させることが重要かも知れない。ユーザの好みをより明確化しなければ、本当に良いメロディーを作れないと考えるからである。

6 まとめ

我々は、対話型作曲システム i-Sonneteer を多様な楽曲形式に応用するために、伴奏定義ファイルを作成し、ニューミュージック、ショパン風ワルツなどを作曲するシステムを構築した。

実験では、ニューミュージックで 3 世代、ショパン風ワルツで 6 世代のときに被験者の満足の行くメロディーを作曲できた。20 曲から 50 曲くらい評価、修正することでユーザの好みの音楽を作曲できる。

人の好みは明確化する過程において、揺らぐものである。どのくらい揺らいでも好みと判断できるかを計測して、好みを明確化するモデルを考え、それに沿ってメロディーを生成すればよりよいシステムになる。

今後の課題は、2 や 4 小節の単位の伴奏パターンを登録すること、コード構成音以外の音を伴奏に取り入れること、好みを明確化するモデルを考え、それにあった進化モデルを考えることなどがある。

参考文献

[1] 蓮井 洋志 小倉久和：対話型進化的手法による作曲システム i-Sonneteer の作成，進化的計算シンポジウム 2007 講演論文集 (2007).

[2] 川村 修 大囿 忠親 伊藤孝行 新谷虎松：逐次的リズム音程生成モデルに基づく自動作曲，情報処理学会音楽情報科学研究会，pp. 19–24 (2005).

[3] 篠田元一：ザ・ロック&ポップス・プログラミングス，Rittor Music (1999).