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

2008－HCI－128 (23)
2008－MUS－ 75 (23)
2008／5／29

N-Best探索アルゴリズムを利用した k-measure HMMによる作曲法

蓮井洋志*

*室蘭工業大学情報工学科

対話型作曲システム i-Sonneteer は k-measure HMM を個体とする対話型進化的計算手法を用いている。k-measure HMM は突然変異においてはランダムウォーク、それ以外においては最尤法を用いてメロディーを生成する。しかし、ランダムウォークでは生成するメロディーは乱数の加減によって一意に定まらない。そのため、メロディーが k-measure HMM の適応度とすると、k-measure HMM の適応度は乱数によって決定づけることになる。それでは、k-measure HMM の適応度評価として適切でない。そこで、我々はグラフの最短 N 経路アルゴリズムを用いて、n 番目に確率の高いメロディーを突然変異で用いる。このメロディーの生成方法を n-best 法という。これは、メロディーの多様性を保持したまま、探索を行なう突然変異としては最適であると考える。本研究では、k-measure HMM に最短 N 経路アルゴリズムを導入して、n-best 法を実現する。n-best 法でメロディーを生成する実験を行ない、対話型作曲システムでメロディーを生成する実験を行なった。ランダムウォークより n-best 法のほうが評価値の高いメロディーを生成する。主観的に見ると、n-best 法によってメロディーの多様性が保持されることが分かった。しかし、対話型作曲システムは n-best 法とランダムウォークはほぼ同じ評価のメロディーを生成した。n-best 法はあまり効果がないことが分かる。

Composing Method of k-measure HMM with N-Best Search Algorithm

Hiroshi Hasui*

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

We implement the interactive composing system, i-Sonneteer, with the interative evolutionary computation which used k-measure HMM as an individual. In mutation the system used randomwalk as the generating method of the melody, and in non-mutation the highest likelihood method used as the generating method of the melody. Because of using the random, randomwalk generates different melody with same k-measure HMM. It is not undesirable to use the randomwalk as the generating method in the fitness evaluation of k-measure HMM. We adopts the n-best likelyhood method with the N-Best path search algorithm for the generating method in the mutation. We assume that it is the most proper operation as the mutation in order to keep the variation of the melody and to generate the melody of the high estimation. In this study, the N-Best path search k-measure HMM is installed into k-measure HMM, and experiments for generating the melody with n-best likelyhood method and for composing the interative composing system. The result of the experiment shows that the n-best likelihood method is higher than the randomwalk. Although n-best method keeps the variation of the melody with respect of the subjective, the system of the n-best likelihood method composes the melody of the same estimation as the randomwalk. This results shows this study is not effective very much.

1 はじめに

対話型作曲システムは対話型進化的計算手法で実現する例が多い。対話型進化的計算手法とは、ユーザとシステムが対話しながら、ユーザの好みを進化的計算手法によって最適化する方法のことを言う。ユーザが譜面を自由に修正できる対話型作曲システムは、システムがユーザの好みを学習するだけでなく、ユーザの好みを明確化していく効果もある。この効果のために、ユーザの深い感性の中にある好みを絞りこむことができる。

我々は、対話型作曲システム i-Sonneteer[1, 2, 3]

を開発している。対話型進化的計算手法の個体として k-measure HMM を用いた。k-measure HMM は逐次的リズム音程生成モデル [4] の学習方法をもとに、音高の情報を記号として加え、生成方法を改良したモデルである。この個体の遺伝子はメロディーで、k-measure HMM はメロディーを学習する。

k-measure HMM のメロディー生成方法として、ランダムウォークと最尤法を考えた。ランダムウォークは突然変異のときに使われる生成手法で乱数によって決定する。しかし、ランダムウォークの生成するメロディーは生成する乱数に依存し、一定しない。また、生成したメロディーはよいメロディーである例は少ない。そこで、我々は突然変異時の生成方法として、n-best 法を提案する。n 番目に確率の高いメロディーを生成することで、解の多様性を保持する。HMM の状態遷移図をグラフとして、最短 N 経路探索を行なうことで n-best 法を実現する。

我々は、ランダムウォークを使わず n-best 法のみを使った。n-best 法の生成するメロディーは n を決定すれば一意に決定でき、確率が低すぎる解を拾ってくる可能性もない。そのため、対話型進化的手法の突然変異体の生成方法として適切と言える。

本論文では、2 章では我々が過去に提案した N-Best 探索アルゴリズム Dijkstra-Hasui 法について、3 章では対話型作曲システムの好みのモデルである k-measure HMM について、4 章では i-Sonneteer の実現方法について、5 章では n-best 法を用いて生成したメロディーの平均評価値について述べる。6 章で考察し、7 章でまとめる。

2 N-Best 探索アルゴリズム

2.1 Dijkstra-Hasui アルゴリズム

Dijkstra 法を使った最短 N 経路アルゴリズムを開発した。Dijkstra-Hasui 法と呼ぶ。このアルゴリズムを実行するために必要な変数を以下に示す。

```
// 目的解:
RouteTable  routes;        // 最短 N 経路候補
// ワーキング変数:
RouteTable  routetable;
          // 他頂点から目標頂点までの最短路
Vertexs     nextpoint, dp;// 頂点番号変数
int         k;            // 整数変数
Route       *r1, *r2, *r; // 経路変数
Route       *route;       // 最短経路変数
```

Dijkstra-Hasui アルゴリズムを以下に示す。

```
BestNSearch(Vertexs start, Vertexs end)
(1) end から他の全ての頂点までの最短路を routetable に保存する
(2) routetable の中の end から start までの最短路を検索し、それを routes に登録する
(3) routes が空ならば return グラフはつながらない;
    それ以外ならば (4) にすすむ
(4) routes.route[0]->dp に start をいれる
(5) k = 0, ..., N まで以下をループ
 (5-1) routes の中にある k 番目に短い経路 route の最初の頂点から route の最後の頂点まで以下をループ
  (5-1-1) dp に route->dp を代入し、dp まで頂点を進める
  (5-1-2) nextpoint に dp を中心として展開する頂点番号をいれる。nextpoint は route の経路と違う頂点だとする。dp が end ならば探索を (5) に戻す
   (5-1-2-1) r1 に route の dp までの頂点番号列をいれる
   (5-1-2-2) r1 に nextpoint をいれる
   (5-1-2-3) routetable から nextpoint から end までの最短路を検索し、それを r2 に入れる。
   (5-1-2-4) r に r1 と r2 をつなげていれる
   (5-1-2-5) r の経路の長さを求める
   (5-1-2-6) routes に r を登録する
  (5-1-3) route->dp に同経路の dp の次の頂点番号をいれる
(6) routes の中の短い方から N 個を解として返す
```

このアルゴリズムはまず、最短路を Dijkstra 法で求める。その最短路の中の分岐点で 1 辺外れた頂点からの最短路とそれまでの経路を加えたものを最短 N 経路候補とする。i 番目に短い経路が見付かったら、その経路から 1 辺それた頂点からの最短路とそれまでの経路を加えたものが次の最短 N 経路候補である。それらの候補の中で i+1 番目に短い候補を i+1 番目に短い経路とする。これを N 回繰り返す。

このアルゴリズムを実現するにあたって、routes は短い経路から順番に並べておき、r は N 番目の候補以上に長ければ登録せずに、メモリを解放することとした。これは、メモリ節約と高速化両方のために効果がある。

2.2 Dijkstra-Hasui 法の証明

グラフを $g(v_i, a_j, S, E)$ と定義する。v_i とは頂点を表し、a_j とはその頂点間の長さの属性である。S は開始頂点、E は終了頂点を表す。ここで、S から E の最短路を $v_{11}, v_{21}, ..., v_{n1}$ とする。このグラフの最短路は r_1、k 番目に短い経路は r_k とする。

S から展開された頂点 $v_{11}, v_{12}, ..., v_{1k}$ を通る経路の集合が S から E までの経路の集合である。

これを v_{11} に応用すると、v_{11} を通る経路の集合は S から v_{11} を通り、$v_{21}, v_{22}, ..., v_{2l}$ を通る経路の集合である。つまり S から E までの全経路の集合は、S から $v_{12}, ..., v_{1k}$ を通る経路と S, v_{11} を通ってしかも $v_{21}, ..., v_{2l}$ を通る経路の集合と一致する。

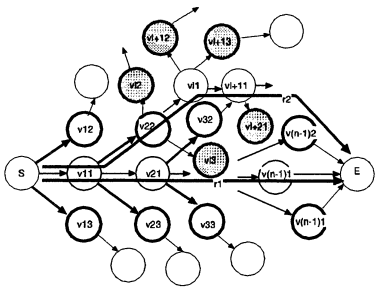


図 1: 最短 2 経路とそれ以外の経路に分割したグラフ

これを再帰的に繰り返すと最短路 r_1 とそれから 1 辺それた頂点 V_1 を通る経路の集合 R_1 の和集合が S から E までの全経路の集合であることがわかると思う。

図 1 に最短路とそれ以外の経路への分割したグラフを示す。真ん中の経路が最短路である。太い丸で囲まれた白抜きの頂点を通る経路の集合が最短路より長い経路の集合である。

R_1 の中の一番短い経路が 2 番目に短い経路である。つまり、V_1 の各頂点から E までの最短路にその頂点までの最短路の経路を加えあわせた経路集合の中に必ず 2 番目に短い経路が含まれていると言える。

次に、R_1 に含まれない経路の中で、2 番目に短い経路から、1 辺それた頂点、図 1 では太丸で囲まれた灰色の頂点であるが、それ を通る経路の集合 R_2 を考えると、$\{r_1+r_2+R_1-r_2+R_2\}$ が S から E までの全経路である。図 1 でいうと、$S, v_{11}, v_{22}, v_{l2}, ..., E$ などが R_2 に含まれる。

そうすると、そこから最短路と 2 番目に短い経路を取り除いた $\{R_1-r_2+R_2\}$ が、2 番目に短い経路より長い経路の集合である。

つまり、最短路かあるいは 2 番目に短い経路から 1 辺それた経路の中で 2 番目に短い経路が 3 番目に短い経路である。

これを繰り返し行うことで N-Best 解を探索することができる。$\{R_1+R_2+...+R_{N-1}-r_2-...-r_{N-1}\}$ が N 番目に短い経路より長い経路の集合である。つまり、その中で一番短い経路が N 番目の最短路であることがわかると思う。$N-1$ 個の最短路から 1 辺それた頂点から、E までの最短路に、それまでの経路を加えた経路の中で $N-1$ 番短い経路が N 番目に短い経路である。これは、アルゴリズムどおりの探索である。

3 k-measure HMM の学習、生成

3.1 k-measure HMM による学習

HMM には、逐次的リズム音程生成モデル [4] 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。

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

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

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

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。

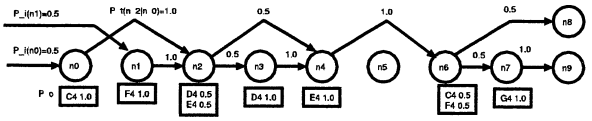
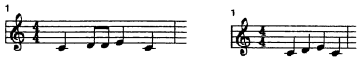


図 2: 学習した 1-measure HMM



(a) 最尤法　　(b) n-best 法

図 3: k-measure HMM で作曲したメロディー

1-measure HMM の例を図 2 に示す。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。

3.2 k-measure HMM による生成

3.2.1 最尤法

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

3.2.2 n-best 法

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 N 番目に高いリズムを探索する。探索には Dijkstra-Hasui アルゴリズムを用いた。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。

前節で学習した 1-measure HMM では、n_0, n_2, n_4, n_6, n_8 が最小距離のノード列である。これは、4 分音符、4 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(b) に前節で学習した 1-measure HMM が生成したメロディーを示す。この例では、$N = 2$ である。

4 対話型作曲システムの実現

4.1 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 8-measure 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

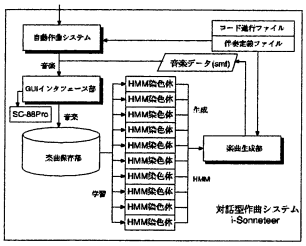


図 4: 対話型作曲システムの構成

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 8-measure HMM の適応度となる。

8-measure 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 8-measure 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 0.2 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。

また、突然変異の個体は n-best 法でメロディーを生成し、それ以外は最尤法を用いて生成する。n-best 法は $N = 7$ とした。7 番目に確率の高いメロディーを生成する。メロディーが 7 種類未満しか生成できない場合は一番確率の低い解を生成したメロディーとする。

4.2 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。

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。

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、評価時に満足すると終わる。

表 1: n-best 法とランダムウォークの適応度の比較

曲番号	最尤法	n-best 法 (n=7)	ランダムウォーク
1	0.55	0.53	0.27
2	0.50	0.47	0.42
3	0.50	0.50	0.47
4	0.64	0.63	0.42
5	0.53	0.52	0.36
6	0.52	0.48	0.32
7	0.55	0.51	0.34
8	0.60	0.55	0.40
9	0.62	0.50	0.52
10	0.57	0.52	0.39

5 実験

[実験 1] ランダムウォークと n-best 法で生成したメロディーの比較評価を行なった。HMM はユーザの修正した 20 曲のメロディーから 7 曲学習し、生成する。選択した 7 曲のメロディーは両方法とも同じものである。評価は 0 から 1 の小数である。$N = 7$ の例である。

ランダムウォークは全般的に低い評価である。評価値の平均は 0.381 である。n-best 法は最尤法よりわずか小さい値となっている。4 曲目が 0.60 を越えていて、9 曲目だけがランダムウォークより小さい評価値となっている。評価値の平均は 0.571 である。最尤法は全般的に 0.50 以上の高い評価値となっている。

ランダムウォークは乱数によって大きく評価値が変わる。n-best 法は比較的に安定した値がでている。曲想が類似しているものが 3 曲、大きく異なるものがものが 7 曲あった。

[実験 2-1] ランダムウォークと n-best 法を突然変異とした方法で i-Sonneteer を用いて作曲した。そのときの各世代ごとの平均適応度を図 5 に示す。被験者は大学 4 年生で音楽を聞くことを趣味として 8 年になる学生である。

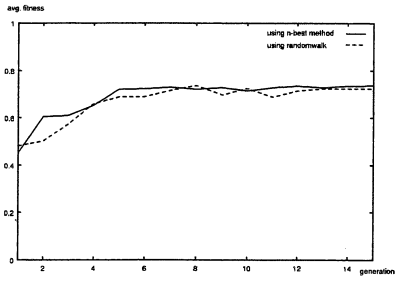


図 5: 平均適応度の推移

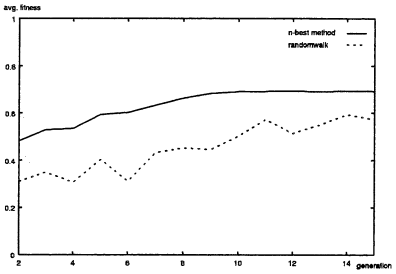


図 6: n-best 法とランダムウォークの平均適応度の推移

最尤法による平均適応度は 7 世代くらいで頭打ちとなった。n-best 法の平均適応度はわずかに上下しながら 15 世代まで少しずつ上がっていることが分かる。12 世代くらいでメロディーが収束した。メロディーが収束しはじめると n-best 法の伸びがにぶることが見て取れる。収束すると n-best 法も大きくほかと変わらないメロディーになった。

[実験 2-2] [実験 2-1] の実行結果における n-best 法とランダムウォークの各世代ごとの平均適応度を図 5 に示す。

n-best 法によって作曲したメロディーはランダムウォークによるものより評価が高く、世代によって平均評価値の上下動が少ない。しかし、n-best 法もランダムウォークも適応度が少しずつ上がって行く。これは、ユーザの好みが固まれば、n-best 法でもランダムウォークでも類似した音楽だけになることを表している。

6 考察

ランダムウォークの生成するメロディーの中には、最尤法とほぼ同じものあれば、全体的にまとまりがなくメロディーとは言えないようなものもある。それに対して、n-best 法のほうが安定したメロディーを生成する場合が多い。

楽曲保存部のメロディーを聞いてみると、n-best 法を用いた場合の突然変異によって生成したメロディーは、5 世代くらいまではユーザの好みを変化させる効果が見られたが、それ以外の世代では効果がなかったようである。好みは固まってくると変化しずらいものであることが窺える。

N-Best 探索は、形態素解析システムの言語モデルである HMM の復号化問題の解法として作成した。本研究では HMM の学習した状態遷移図の探索に用いた。このアルゴリズムは十分高速である。

Dijkstra-Hasui アルゴリズム以外にも我々は動的計画法を用いた N-Best 探索アルゴリズムを開発した。しかし、動的計画法によるものと Dijkstra-Hasui アルゴリズムはほぼ同じくらいの速度で動作する。本研究の n-best 法においては速度の差は見られなかった。

7 まとめ

N-Best 探索を行なうアルゴリズムを実現した。また、そのアルゴリズムが正しいことを証明した。N-Best 探索アルゴリズムを用いて、n-best 法という k-measure HMM のメロディー生成方法を提案した。n-best 法を用いたシステムでメロディーを作曲した。

n-best 法の N のパラメータを変化させることで、よりユーザの好みをうまく明確化する対話型作曲システムを実現することが今後の課題である。

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] 蓮井 洋志 小倉久和：対話型作曲システム i-Sonneteer の他楽曲形式への応用 (2008).

[3] 蓮井洋志 小倉久和：対話型進化的計算手法による作曲システムにおける HMM の作曲法，情報処理学会第 70 回全国大会講演論文集 (2008).

[4] 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).