OrpheusBB：Human-in-the-loop 型の自動作曲システム

北原鉄朗[†] 深山覚[††] 片寄晴弘[†††]
嵯峨山茂樹[††] 長田典子[†††]

これまで自動作曲に関する研究は数多く行われてきたが，旋律生成のための新たなアルゴリズムやモデルの探求に主眼が行われており，生成結果をユーザがどう修正するかに関しては十分な検討は行われてこなかった．本稿では，システムが生成した楽曲をシステムの支援を受けながら編集できる自動作曲システム OrpheusBB を提案する．本システムの特徴は，ユーザが生成結果を編集すると音楽的に不自然にならないように残りの箇所を確率推論によって自動的に再生成する点である．この機能により，ユーザは音楽的に破綻することを気にせずに試行錯誤を繰り返すことができる．この機能を取り入れることで，楽曲制作における試行錯誤をより気軽に行えるようになることが，明らかになった．

OrpheusBB: A Human-in-the-loop Automatic Music Composition System

Tetsuro Kitahara,[†] Satoru Fukayama,[††]
Haruhiro Katayose,[†††] Shigeki Sagayama[††]
and Noriko Nagata[†††]

Various attempts at automatic music composition systems have been made but they have not dealt with the issue of how to edit the composed piece by the user. In this paper, we propose an automatic music composition system, called *OrpheusBB*, where the user edits the composed piece after the automatic composition. The feature of this system is to regenerate, right after the user edits some part of the piece, the remaining part so that it musically matches the editted part. We confirmed that this function facilitated the user's trial-and-error in creating musical pieces.

1. はじめに

自動作曲は，作曲スキルのない人に作曲スキルを提供するという応用上の意義，人間の創作活動の数理的な定式化という科学的な興味の両方において重要な研究テーマである．そのため，古くから様々な研究者によって自動作曲研究が行われてきた[1)~6)]．

自動作曲エンジンは単に何らかの楽曲を生成するのではなく，ユーザとインタラクションを取りながら楽曲生成を繰り返すことで，ユーザが所望する楽曲に徐々に近づけていくことが肝要である．これは，ユーザの楽曲に対するイメージを事前に自動作曲エンジンに伝えることは難しく，1 回の処理でユーザのイメージに十分に近い楽曲を生成することはまれだからである．本稿では，このようにユーザとインタラクションを取りながら楽曲生成処理を繰り返すアプローチを「human-in-the-loop 型の自動作曲」と呼ぶ．

従来の自動作曲システムは，大きく 2 つに分けることができる．1 つは全自動型[1)~4)] である．これは，ユーザが歌詞や曲調などを入力すると，システム内のアルゴリズムや確率モデルなどに基づいて楽曲を生成するものである．これらの研究では，音楽的に優れたあるいは新奇性の高い楽曲を生成するアルゴリズムやモデルの探求に主眼が置かれている．そのため，生成された楽曲がユーザのイメージに合わない場合，どのようにユーザのイメージに近づけていくかといったことは，扱われていない．楽曲データを編集するためのソフトウェアは数多く市販されているので，生成された楽曲データをそのようなソフトウェアで編集することは可能である．しかし，楽曲データの一部を編集し，それに合うように残りの部分をもう一度システムに作曲させるというような，手作業による編集と自動作曲エンジンによる生成とをシームレスに組み合わせた作

† 日本大学文理学部
College of Humanities and Sciences, Nihon University
†† 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, the University of Tokyo
††† 関西学院大学理工学部
School of Science and Technology, Kwansei Gakuin Univeristy

業は不可能である．

もう 1 つは対話型進化計算を用いた半自動型の作曲システム[5),6)] である．これらのシステムでは，作曲システムが生成した楽曲をユーザが評価するというプロセスを繰り返すことで，ユーザの所望する楽曲に近づけていく．これは，上述の「human-in-the-loop 型の自動作曲」の一形態ととらえることができる．このアプローチは，ユーザは生成された楽曲に対して良い悪いの判断だけをすればよいので，専門知識が要求されない点がメリットである．しかし，一般にユーザが所望する楽曲にたどり着くには，聴取・評価を何度も繰り返さなければならず，ユーザに対する負担は小さくない．また，ユーザは良い悪いの判断しかできないので，上述したような，手作業による編集と自動作曲エンジンによる生成とをシームレスに組み合わせた作業は，こちらのアプローチでも不可能である．

本研究では，手作業による編集と自動作曲エンジンによる生成とをシームレスに組み合わせて作業が可能な，新たな自動作曲システム OrpehusBB を提案する．本システムでは，ユーザが歌詞を入力すると，その歌詞の韻律に沿ったメロディを生成する．その後，ユーザはメロディやコード進行を自由に編集することができる．このとき，ユーザがメロディを編集するとコード進行を，ユーザがコード進行を編集するとメロディを，音楽的に不自然（たとえば不協和）にならないように自動的に再生成する．これにより，生成された楽曲のある部分の印象を変えたいので，そこのコードを変更してメロディを自動で生成し直す，あるいは，生成されたメロディの一部を少し変えたいので，コード進行もそれに合うように自動で付け直す，といった手作業による編集と自動生成とを組み合わせて楽曲を仕上げていくことができる．

以下，2 章では，human-in-the-loop 型の自動作曲システムが満たすべき要件について議論する．3 章では，2 章で議論した要件を満たすために達成すべき技術的課題およびその実現方法について議論する．特に，本システムで中心的なコンセプトとなる「音楽的一貫性の自律的維持」について述べ，それを確率モデルで実現することを述べる．4 章では，3 章での議論に基づいて実現した自動作曲システム OrpheusBB の実装について述べる．5 章では，OrpheusBB を用いた作曲の実例を紹介し，6 章で考察する．最後に 7 章でまとめをする．

2. Human-in-the-loop 型の自動作曲

「Human-in-the-loop 型の自動作曲」とは，1 章で述べたように，ユーザの介入をプロセスの一部に含んだ自動作曲のアプローチである．ユーザは，自動作曲エンジンが生成した楽曲に対して，修正したい場所やそこをどう直すべきかを指定して，楽曲生成をもう一度行わせる．このような楽曲の修正過程は，音楽制作現場におけるプロデューサと作曲・編曲家による共同作業に例えて考えることができる．プロデューサは，商業上の戦略に基づいて楽曲の仕様を策定し，作曲・編曲家に楽曲制作を依頼する．楽曲が完成するとプロデューサは楽曲を試聴し，当初の仕様と合わない部分があれば，作曲・編曲家に修正を依頼する．このような作曲・編曲家による修正とプロデューサによるチェックを繰り返すことで楽曲を完成させる．

現状の音楽制作環境においても，自動作曲システムが生成した楽曲を市販の MIDI シーケンサなどを使って修正することは可能である．しかし，データを修正する過程では，楽曲データのすべてを自分の手で修正しないといけない．そのため，楽曲のある部分を修正したときの副作用（たとえば，メロディを修正するとコード進行と不協和を生むなど）にすべて自分で気づき，自らが対処しないといけない．ここが上とは決定的に異なる点である．音楽プロデューサは，メロディやコード進行の修正方法を具体的に指示するかもしれない．しかし，そのような場合でも，作曲・編曲家は単にプロデューサが指示した箇所を修正するわけではない．仮にプロデューサの指示通りに修正すると音楽的に不適切になるような場面でも，音楽的に不適切なまま放置したり，修正指示を拒否したりせず，楽曲中の他の部分を変更するなどして，その修正要求を満たした上で音楽的に妥当な解決策を自らの専門知識や経験を元に見出す．

Human-in-the-loop 型の自動作曲システムの最終的な目標は，上述のプロデューサと作曲・編曲家の共同作業による音楽制作のように，ユーザとシステムが共同作業を行いながら楽曲を仕上げていく環境の提供にある．そこで，human-in-the-loop 型の自動作曲システムは，次の 3 つの要件を満たすべきであると考える．

(1) ユーザが，メロディやコード進行の特定の箇所に対して修正したいと考えているときは，その修正したい箇所を直接修正できるべきである．
(2) ユーザによる編集は最大限尊重されるべきである．つまり，ユーザが編集した箇所を後に自動作曲エンジンが書き換えなくする手段が提供されるべきである．
(3) ユーザが編集することで楽曲全体が音楽的に妥当でなくなる場合（たとえば，メロディのある

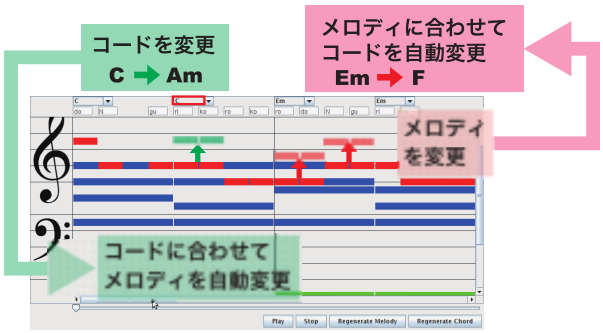

図 1　OrpheusBB の基本コンセプト

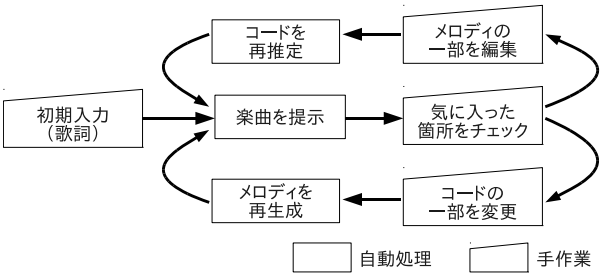

図 2　OrpheusBB におけるユーザ操作と自動処理の流れ

箇所を編集するとコード進行と不協和になる場合），音楽的に妥当なものになるよう，ユーザが編集した箇所以外を自動的に補正すべきである．

以下では，これらの要件を満たす自動作曲システムを提案していく．

3.　自動作曲システム OrpheusBB

本稿では，この 3 つの要件を満たす自動作曲システム OrpheusBB を提案する（**図 1**）．OrpheusBB では，最初に楽曲生成の種となる情報（歌詞）を入力して楽曲の生成を実行する．生成された楽曲は編集画面に表示され，楽曲のメロディやコード進行などをユーザが編集することができる（要件 (1) への対応）．ユーザがメロディの一部を書き換えると，書き換え後のメロディに対して最もふさわしいコード進行をただちに推論する．その結果，現在のものとは異なるコード進行が最もふさわしいと判断されたら，コード進行がリアルタイムに更新される（要件 (3) への対応）．同様に，ユーザがコード進行の一部を書き換えると，書き換え後のコード進行に最もふさわしいメロディをただちに推論し，更新する．メロディやヴォイシングの特定の箇所をユーザが気に入った場合，ユーザがその箇所にチェックを入れることができる．上述の推論を行う際にチェックの入っている箇所の値は更新対象から外すので，他の箇所の編集によって意図せずに更新されてしまうことを防ぐことができる（要件 (2) への対応）．操作および処理の流れを**図 2** に示す．

以下，OrpheusBB を実現する際の技術的課題について論ずる．

3.1　作曲の種となる情報としてユーザに何を入力させるか

音楽的に妥当な旋律は無限にあるので，旋律の制約となりうる情報をユーザに入力させる必要がある．ここで重要なことは，旋律をうまく制約しうる情報でありながらも，その入力に音楽の専門知識は必要とされないということである．作曲対象を日本語の歌唱曲に限定すれば，この条件を満たすものとして「歌詞」が考えられる．日本語はピッチアクセントの言語であり，日本語文が与えられたら自然な韻律（ピッチの高低）は，いわゆる標準語に限定すれば，ほぼ一意に求めることができる．日本語歌唱曲の作曲法として，歌詞の韻律と一致するように旋律のピッチの高低を決める方法が広く知られている．実際，ユーザが入力した日本語歌詞の韻律にピッチの高低を一致させるという制約の元で旋律を生成する自動作曲システムが開発されている[4]．そこで，本研究においても，ユーザに日本語歌詞を入力させ，その韻律のピッチの高低を制約条件に旋律を生成する方針を取る．

3.2　ユーザによる編集にともなう楽曲再生成をどのように実現するか

前章で述べた 3 つの要件を実現する上で，中心的な課題となるのは，ユーザによる楽曲編集の後に実行される楽曲再生成の実現方法である．システムは，メロディやコード進行などが編集されたときに，それによって全体の音楽的妥当性（たとえばメロディとコード進行の調和性）が失われるのかを即座に判定し，失われるなら何をどう改変すれば音楽的妥当性を維持できるのかを推論し，データを更新しなければならない．本稿では，これを**音楽的一貫性の自律的維持**（Autonomous Maintenance of Musical Consistency，AMMC）と呼ぶ．

AMMC を実現するには，メロディやコード進行の相互の依存関係を何らかの知識表現体系で記述し，その一部が書き換えられた（新たな値が観測された）ら，それと矛盾しないように残りの箇所を更新する仕組みが必要である．このような，ある知識表現の元で，新たな事実が与えられたときにそれと矛盾しないように既存の信念などを書き換える問題は，人工知能の分野で研究が行われている．Truth Maintenance System (TMS)[7] がその代表例である．しかし，TMS は論理による推論を基としているため，音楽のように論理的に真偽を決定するのが困難なドメインには適さない．

新たな事実の観測による信念の更新を実現するも

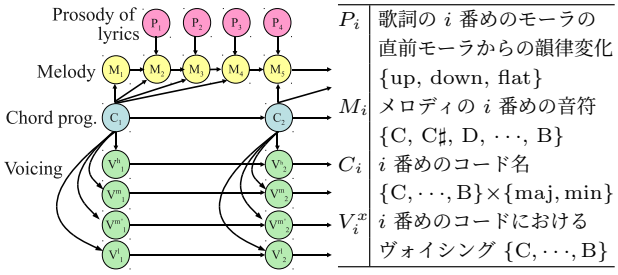

図 3 本システムで使用するダイナミックベイジアンネットワーク

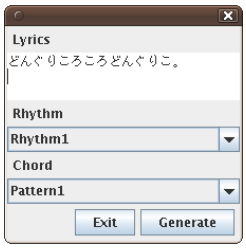

図 4 歌詞入力フェーズ用 GUI

う 1 つの枠組みにベイジアンネットワークがある．ベイジアンネットワークは，事象間の依存関係を有向グラフで表し，その有向グラフ上で確率推論を行う．メロディやコード進行の依存関係を確率的に表現できる（たとえば「メロディの音符が C♯ なら C のコードは滅多に来ない」など）ので，音楽のように決定論的に真偽を判定することが難しいドメインにも適用できる．

本稿では，歌詞の韻律，メロディ，コード進行，コードヴォイシング（各々をレイヤーと呼ぶ）の依存関係をダイナミックベイジアンネットワーク（DBN）で記述し，その下で最ももっともらしいメロディやコード進行を求めるという方針をとる．本研究で使用する DBN を**図 3** に示す．レイヤー内の経時的な依存関係と同時刻におけるレイヤー間の依存関係が，ノード間のアークとして表されている．メロディはその背後の和声から生まれるという考え[8] の下，コードからメロディへの向きにアークを設定した．同様にヴォイシングはコード名に依存して決まるので，コードからヴォイシングへの向きにアークを設定した．本来，メロディとヴォイシングとは依存しており，これらの間にもアークを設定するのが望ましい[9] が，計算量削減のため省略した．なお，この方法を用いるには，メロディの各ノードとコード進行の各ノードとで時間的な対応が取れている必要がある．そこで，楽曲に次の制約を置いた：

- メロディのリズムは最初の作曲時（4.2 節参照）に決められたものを維持し，事後の編集はできない．
- コードは 2 拍ごとに変化する．

AMMC は，図 3 の DBN 中のユーザが編集した箇所に相当するノードに値（エビデンス）をセットし，全体の確率が最も高くなるように残りのノードの値を推論する問題に帰着する．また，特定の音符にチェックを入れて更新対象から外す行為は，DBN 中の対応するノードの現在の値の確率を 1.0 にする（エビデンスとみなす）ことに相当する．そこで，次の処理により AMMC を実現する．

- ユーザが音符にチェックを入れると，それに対応する DBN 中のノードに対して，現在の値をエビデンスとしてセットする．
- ユーザが音符またはコードを編集すると，それに対応する DBN 中のノードに編集後の値をエビデンスとしてセットし，直ちに確率推論を行う．
- 確率推論後は，データを各ノードの最尤値に更新し，画面を更新して次の編集に備える．

4. OrpheusBB の実装

前章で述べた方針に基づき，ユーザとシステムがインタラクションを取りながら楽曲制作を行う自動作曲システム OrpheusBB を実装した．以下，実装した OrpheusBB の概要について述べる．

4.1 GUI

OrpheusBB は，歌詞入力フェーズと楽曲編集フェーズに分けることができる．ここでは，それぞれのフェーズに対応した GUI について述べる．

4.1.1 歌詞入力フェーズ（図 4）

OrpheusBB を起動すると，歌詞を入力する画面になる．ここでは，日本語のかな漢字混じり文を入力することができる．また，必要に応じてコード進行やメロディのリズムを変更することができる．入力後に「Generate」ボタンを押すと，入力した歌詞に韻律が最も合う旋律を生成する．生成が終わると，楽曲編集フェーズに移る．

4.1.2 楽曲編集フェーズ（図 5）

楽曲編集フェーズでは，前フェーズで生成された楽曲の内容（メロディ，歌詞，コード進行，コードヴォイシング）が，楽譜を模した形式で表示される．表示されている内容のうち，メロディ，コード進行，コードヴォイシングは編集することができる．これらを編集すると，後述の方法により，音楽的に不協和にならないようにその他の要素が即座に更新される．

楽曲データを編集する際によく用いられるデータの表示方法に，楽譜（五線譜）とピアノロールがある．ピアノロール表示は，音符の長さや相対的な音高の変化が直感的に分かりやすい半面，絶対的な音高（例えば中央のドがどこにあるか）を瞬時に把握するのは，

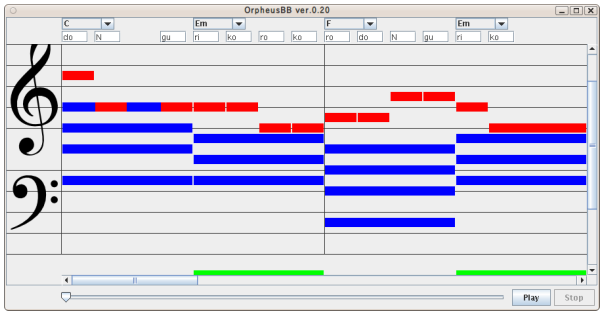

図 5　楽曲編集フェーズ用 GUI

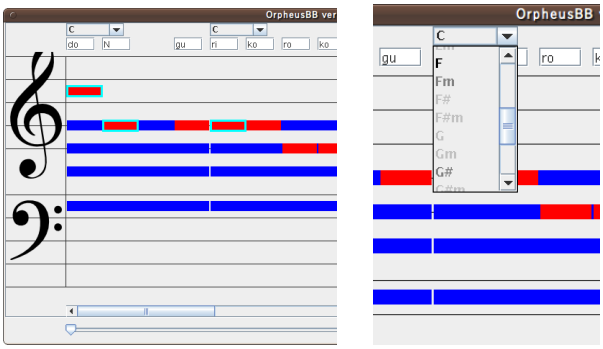

図 6　**左**：いくつかの音符にチェック（上書きされなくするための印）を入れたところ．シフトキーを押しながら音符をクリックすると，チェックが入れられたことを示す水色の枠が表示される．**右**：コードを選択するためのドロップダウンリストを開いたところ．現在のメロディや前後のコードとマッチする可能性の高いコード名は濃い色で，マッチする可能性の低いコード名は薄い色で表示されている．

五線譜に比べて難しい．そこで，本システムでは，ピアノロール表示で用いられる横長のバーを五線譜の上に配置するという表示方法を採用する．シャープ付きの音符は横長のバーに上矢印を，フラット付きの音符は横長のバーに下矢印を付けて区別する．

前述したように，メロディ，コード進行，コードヴォイシングは編集が可能である．メロディを編集するとコード進行が，コード進行を編集するとメロディが即座に再生成され更新される．コードヴォイシングも，コード進行が編集または更新されると更新される．メロディ，コード進行の各音符は，チェックを入れることができる（**図 6** 左）．チェックが入れられた音符は，メロディなどを再生成する際に所与の音符とみなされ，再生成後に他の音符に書き換えられることを防ぐことができる．この機能は，メロディなどの編集や再生成を繰り返す過程で気に入った箇所にチェックを入れていき，徐々にチェック箇所が増えていくことで，段階的にユーザのイメージ通りの楽曲に近づけていくことを想定している．

なお，コード進行はコード名で表示されるため，コード表記に慣れていない人には分かりにくい．そこで，コード名が表示されるドロップダウンリストにおいて，現状のメロディや前後のコードにマッチする可能性が高いコードを濃い色で，マッチする可能性が低いコードを薄い色で表示する（図 6 右）．「現状のメロディにマッチするコードを色の濃いコード名から選ぶ」といった使い方の他，「印象を大きく変えるためにあえて色の薄いコード名から選び，メロディをそのコードに合うように再生成させる」という使い方もできる．

4.2　歌詞からの楽曲生成

歌詞からの旋律生成は，Orpheus[4] で用いられている手法を用いる．まず，入力された歌詞は音声合成エンジンのテキスト解析フロントエンドによって，その発音と韻律が解析される．次に，デフォルトまたはユーザが選んだ基本リズムにしたがってメロディのリズムが決められる．メロディは，歌詞のモーラ数と同じ数の音符で構成されるという制約があるので，基本リズムの音符を必要に応じて分割（たとえば 4 分音符 1 個を 8 分音符 2 個に）または統合（たとえば 4 分音符 2 個を 2 分音符 1 個に）する．基本リズムのどこをどのように分割・統合するかに関する情報はリズム木構造パターンという木構造として保持している．韻律（ピッチの高低）とリズムが決まったら，音楽的な妥当性を失わないように，韻律とピッチの高低が一致するように各音符の高さを決定する．具体的には，各音高の出現確率，各音高から各音高への遷移確率をあらかじめ設定しておき，旋律全体の確率を最大化する音高系列の探索を行う．この問題は動的計画法を用いて効率的に解くことができる．

4.3　音楽的一貫性の自律的維持（AMMC）

AMMC は，前述のように DBN による確率推論として実現する．しかし，図 3 の DBN をそのまま利用するのは，計算量および学習データ量の問題から現実的ではない．そこで，次のようにして推論を行う．

(1)　メロディが編集された場合は，メロディレイヤーの全ノードを既知として，コードレイヤーのノードのみ推論する．この場合，隠れマルコフモデルと等価になり，Viterbi 探索を利用できる．

(2)　コードが編集された場合は，コードレイヤー，歌詞レイヤーの全ノードを既知としてメロディレイヤーのノードを推論する．この場合，元の Orpheus で用いていた最尤経路探索と等価になる．

(3)　(1), (2) の後，コードが編集または更新された場合，そのコードに対応するヴォイシングノードのみ未知にしてヴォイシングを推論する．

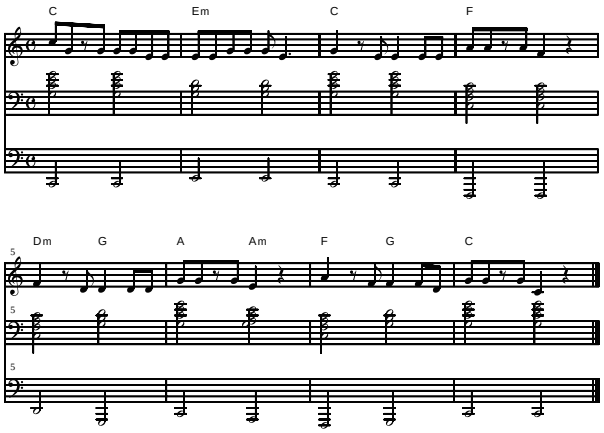

図 7 歌詞として「どんぐりころころどんぐりこ」を入力して生成された楽曲．入力された歌詞だけでは 8 小節分の長さにならないので,「ころころどんぐりこ」の部分を 3 回くりかえして 8 小節分の長さを確保している．

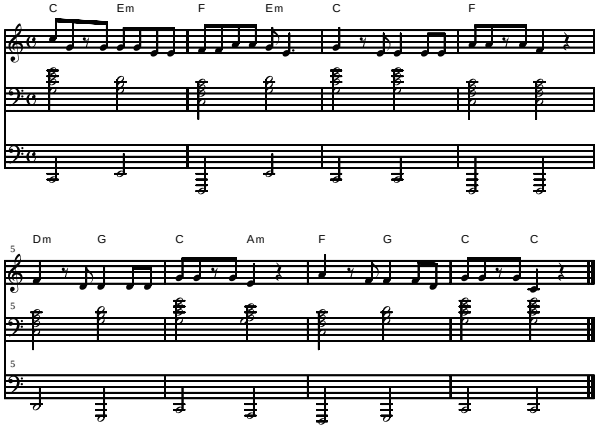

図 8 図 7 の 2 小節めの「ミミソソ」の部分を「ファファララ」に変更した結果．当該箇所のコードが「F」に変更されている．

5. OrpheusBB の試用

実際に OrpheusBB で行った作曲の一例を述べる．

歌詞として「どんぐりころころどんぐりこ」を入力して自動作曲を実行すると，**図 7** に示す楽曲が得られた．これに対して，2 小節めの「ミミソソ」の部分を「ファファララ」に変更するのを試みた．この場合，現状の当該箇所のコード「Em」のままだと短 2 度や長 2 度といった不協和な音程が頻出し，望ましくない．この操作をしたところ，当該箇所のコードが「F」に自動的に変更された（**図 8**）．また，それに合わせて前後のコードも変更になった．試聴したところ，変更前よりも明るくなった印象を受けたため，以後の作業で書き換えられないよう，これらの音符にチェックを入れた．

次に，この楽曲の終わり方の印象を変えるため，最後の 2 小節のコード進行を「F–G–C」から「F–A♭–G–C」に変更するのを試みた.「A♭」を採用したのは

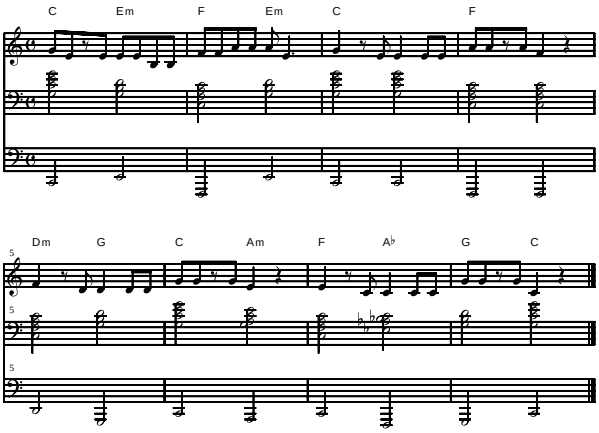

図 9 図 8 の最後の 2 小節のコード進行を「F–A♭–G–C」に変更したところ．これに合わせてメロディも自動的に変更された．

当該箇所のドロップダウンリスト中で非常に薄い色で表示されていたため，現在の文脈ではあまり出てこない，つまり意外性の高いコードであると判断したためである．実際にこのコード変更を行ったところ，**図 9** のように当該箇所のメロディが変化した．試聴したところ，コード進行に適度な意外感がもたらされ，それでいてメロディとコード進行との間に不協和が生まれていないことが確認できた．そこで，最後の 2 小節のメロディ，ヴォイシングの全音符にチェックを入れた．

しかし，メロディの最初の 2 小節のうちチェックを入れていない音符が変更されてしまった．これは，現在の実装では，どこかのコードを変更すると，メロディ中のすべての非チェック音を再生成してしまうからである．変更されてしまった箇所のうち，1 小節めの 7 音め，8 音め（ともに「シ」）と 2 小節めの 5 音め（「ラ」）が，前後の音とスムーズにつながっていない印象を受けたので，それぞれ「ド」と「ソ」に変更した．その結果，1 小節めの 2 つめのコードが「Em」から「Am」に変更になった．

最後に，楽曲の終わり方をもう少し印象的にするため，最後の小節の最後の「ソ」の音を「ラ」に変更した．ここの箇所はヴォイシングにチェックが入っているため，この変更によるヴォイシングパートの変更はなかった．最終的に完成した楽曲を**図 10** に示す．

このように，システムによる楽曲生成の後，(1) 手作業による修正とそれにともなう AMMC，(2) 試聴と気に入った箇所のチェックを繰り返すことで，徐々に気に入った箇所を増やしていき，楽曲を完成させることができた．

上の例では，コード進行を編集することで楽曲の終わり方に元とは異なる印象を与えた．一般に，コード進行は楽曲の印象を特徴づける重要な要素であり，上

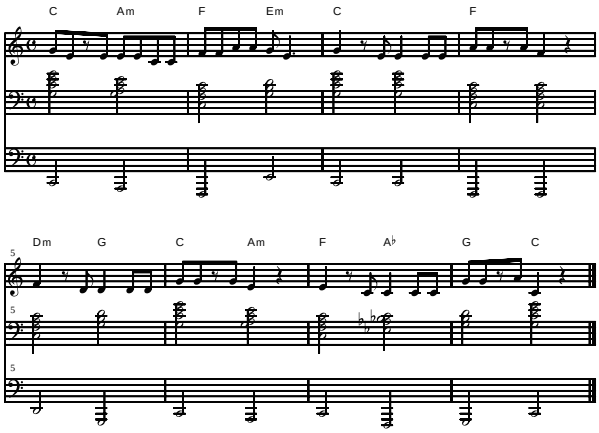

図 10 最終的に完成した楽曲

の例のように，当該の調では頻出されないコード（ノンダイアトニックコード）をあえて用いることで，意外感や新鮮さを演出することができる．しかし，通常の MIDI シーケンサでは，このような演出は和声理論に関する知識がないと難しい．一方，本システムでは，意外感や新鮮さを演出したい箇所でドロップダウンリスト中の色の薄いコード名を適当に選ぶことで，このような演出を試すことができる．その際，メロディも即座に再生成されるので，メロディを含めた楽曲全体を試聴しながら試行錯誤を進めることができる．

ただし，現在の実装ではアンドゥー機能がないため，試行錯誤の過程で以前のものに戻したいのにそれができないという場面があった．アンドゥー機能の実装は原理的には難しくないため，これを実装することでより試行錯誤がしやすいシステムになると思われる．

また，AMMC 機能は全般的には適切に機能したが，必ずしも書き換える必要がない場面でも書き換えられることがあった．この現象を解消するには，DBN 中の各ノードにおいて現在選択されている値の尤度がある程度高ければ，最尤でなくても書き換えずに維持するようにすればよい．

6. 考　　察

6.1 Directability との関連

片寄らは，音楽情報処理の生成系タスクにおいて，生成処理の自動化に研究の重点が置かれており，システムの生成結果をどのように修正するか，そのためのインタフェースはどうあるべきかという課題は後回しにされてきたと指摘している[10]．生成の自動化技術は，音楽生成研究においてその基盤となる重要な技術であることは間違いない．しかし，その目的が人間を支援することにあるのであれば，人間がそれをどのように使うのかを見据えた設計にするべきである．このような考えの下，片寄らは「directability」という概念を提唱した[10]．Directability とは，「個々の具体的なデザイン対象を直接操作するのではなく，人の知覚・認知に適合した，より抽象度の高い操作子を用いて対象の操作が可能であること」を指す．音楽で言えば，音符を 1 つ 1 つ直接操作するのではなく，その背後にある認知構造を操作することで，音楽を操作可能にするということである．

OrpheusBB では，直接の編集対象は音符やコードそのものではあるものの，その編集結果はその背後にある DBN に即座にフィードバックされ，確率推論によって楽曲が再生成される．その意味で，本システムは directability を実現した自動作曲システムであると考えることができる．片寄らも「計算過程の段階でユーザが関与できる形態」が望ましいと述べており，これは本システムのコンセプトと一致する．

6.2 AMMC 機能付楽曲編集インターフェースの発展可能性

本稿では，AMMC をメロディ編集にともなうコードの再推定およびコード編集にともなうメロディの再生成に用いたが，AMMC はより複雑な編曲を扱うときに一層効果を発揮すると考えられる．たとえば，ベースアレンジとしてウォーキングベースを用いる場合，コードとコードのつなぎ目をなめらかにするために適切な経過音を挿入することが重要となる．そのため，ある箇所のコードを変更したら，その前後（特に直前）のベースラインを適切に変更する必要がある．このような場合，本稿で用いた DBN にベースラインを表すノードを追加し，条件付確率テーブルを適切に設定または学習すれば，コードを変更するだけでベースラインの経過音などが自動的に変更されるようになる．実際の演奏現場では，コード進行のような大局的な事柄は作曲・編曲家が決めるが，ベースライン中の経過音のような細部については各楽器のプレイヤーが自己の判断で決める場合も少なくない．しかし，通常の MIDI シーケンサを使って計算機上で楽曲を仕上げる場合，すべてを自らが決めて入力しなければならない．このような場面で AMMC を利用すれば，編曲の細部は計算機に任せることができ，ユーザは，より大局的な部分に注力することが可能になる．

また，今後，DBN を詳細化して学習データを増やすなどして確率推論の精度を向上させることができれば，ユーザが行った編集に対してよりよい編集方法が見つかればそちらを提案するといったことも可能になる．たとえば，ユーザがメロディのある箇所を変更したが，それではメロディ全体の尤度が大幅に下がって

しまうというとき，別の編集方法を提案するという使い方である．このようなことが実現されれば，確率推論に基づく AMMC は，ユーザにとってもはや単なる編集の補助ではなく，楽曲をよりよく仕上げるための良きパートナーということができる．このような方法に本研究を発展させるのも，今後の課題の 1 つである．

6.3 今後の課題

前節であげた以外の今後の課題について議論する．

- **条件付確率テーブル（CPT）の学習**

今回の実装では，CPT はヴォイシング用のものを除き，システム実装者が手動で実験的に設定した．これを実コーパスから学習するというのは，主要な今後の課題の 1 つである．特に，ジャンルや作曲家ごとに CPT を学習すれば，CPT を切り替えることで生成されるメロディのスタイル（「ジャズ風」など）を変えられる可能性がある．

- **CPT を操作する GUI の提供**

CPT を切り替えることで生成されるメロディなどのスタイルを制御できるのであれば，CPT の確率値そのものを GUI 上で操作してしまう方法も考えられる．たとえば，CPT の全項目からなるベクトルを何らかの次元圧縮法によって 3 次元程度に圧縮し，その 3 次元空間上で操作することで CPT を直接編集するなどの方法が考えられる．

- **より複雑なモデル**

本システムでは，円滑な試行錯誤を促進するためにターンアラウンドタイムを極力短くするという観点から，簡略化された DBN を用いた．たとえば，コードヴォイシングは本来メロディにも依存するため，ヴォイシングノードとメロディノードの間にもアークがあるはずだが，省略した．今後，このような簡略化の悪影響を評価し，よりよい DBN を検討する必要がある．

- **音楽的素養のない人に対する楽曲編集 I/F の提供**

本システムでは，楽曲中の各音符そのものを操作対象としたので，楽譜を模写したユーザインタフェースを採用したが，これは，音楽的素養のない人にとって敷居の低いものではない．音楽を音符列に還元しないで聴取するような一般聴取者にとっても試行錯誤のしやすいものにするには，一般聴取者の音楽聴取の特性に合わせた楽曲編集インタフェースを提供する必要がある．たとえば，旋律の大まかなピッチの変化をフリーハンドかベジェ曲線で描き，その軌跡に沿った旋律を計算機が生成するといった方法が考えられる．

7. おわりに

自動作曲システムは，単にユーザからの入力に従って楽曲を生成するだけでなく，生成された楽曲に対して修正と再生成を繰り返しながら楽曲を仕上げていく環境を提供すべきである．このような考えの下，本稿では human-in-the-loop 型の新しい自動作曲システム OrpheusBB を提案した．本システムのキーとなるのは，ユーザがメロディやコード進行などを編集したらそれと不協和が生じないように残りの箇所を自動的に再生成する点である．この機能が存在することで，ユーザは音楽的に破綻することを気にせずに，様々なメロディやコード進行を試行錯誤することができる．本研究では，この機能を「音楽的一貫性の自律的維持」と呼び，ダイナミックベイジアンネットワークを用いた確率推論によって実現した．

今後は，本システムの有効性を被験者による評価実験を通じて明らかにしていく他，6 章で議論した方向性に研究を発展させていく予定である．

謝辞 本研究は，JST CrestMuse プロジェクトの支援により行われた．また，本システムの実装には，戸谷 直之氏，徳網 亮輔氏（関西学院大学）にご協力いただいた．

参 考 文 献

1) Hiller, L. and Isaacson, L.: Musical Composition with a High-speed Digital Computer, *Journal of Audio Engineering Society* (1958).
2) Ames, C. and Domino, M.: Cybernetic Composer: An Ovewview, *Understanding Music with AI* (Balaban, M., Ebcioglu, K. and Laske, O., eds.), AAAI Press, pp.186–205 (1992).
3) Cope, D.: *Computers and Musical Style*, Oxford University Press (1991).
4) 深山 覚，中妻 啓，米林裕一郎，酒向慎司，西本卓也，小野順貴，嵯峨山茂樹：Orpheus : 歌詞の韻律に基づいた自動作曲システム，情処研報，2008-MUS-76, pp.179–184 (2008).
5) 安藤大地，Dahlstedt, P.，Nordah, M.，伊庭斉志：対話型 GP を用いたクラシック音楽のための作曲支援システム，芸術科学会論文誌， Vol.4, No.2, pp.77–86 (2005).
6) Biles, J.A.: Genjam: A Genetic Algorithm for Generating Jazz Solos, *Proc. ICMC* (1994).
7) Doyle, J.: A Truth Maintenance System, *Artificial Intelligence*, Vol.12, No.3, pp.251–272 (1979).
8) 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，情処研報，99-MUS-34, pp.79–82 (2000).
9) 北原鉄朗，勝占真規子，片寄晴弘，長田典子：ベイジアンネットワークを用いた自動コードヴォイシングシステム，情処学論， Vol.50, No.3, pp.1067–1078 (2009).
10) 片寄晴弘，橋田光代：生成系音楽支援システムの Directability 視点からの考察，情処研報，2007-MUS-71, pp.99–104 (2007).