芸術科学会論文誌 Vol. 4 No. 2 pp. 77 – 87

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

安藤 大地 †1,2　Palle Dahlstedt†3　Mats G. Nordahl†3　伊庭 斉志 †1
dando@iba.k.u-tokyo.ac.jp

†1 東京大学大学院新領域創成科学研究科基盤情報学専攻
†2 Art & Technology, IT University of Gothenburg, Sweden
†3 Innovate Design, Chalmers University of Technology, Sweden

近年，音楽作曲の分野への対話型進化論的計算 (IEC) の応用に関する研究は非常に発展してきている．この発展の背景には，人間の感性をコンピュータシステムに取り込むことは，コンピュータシステムの発展にとって必要不可欠という認識がある．しかしながら，IEC を作曲に応用した従来のシステムは，実際の作曲家には積極的に使われてこなかった．この理由は，主にシステムを用いた作曲過程や扱うデータ形式が伝統的な作曲技法のそれとは大幅に異なるためである，と考えられる．そこで筆者らは，実際の作曲に IEC システムを活用することを目的として，新しい作曲支援システムを構築した．新しいシステムの主な特徴は，クラシック音楽の作曲家が馴染みやすい遺伝子表現や作曲過程である．また，実際にシステムを利用してピアノの小品を作曲し，その有効性を確認した．

キーワード: 作曲支援，対話型進化論的計算，遺伝的プログラミング

Computer Aided Composition for Contemporary Classical Music by means of Interactive GP

Daichi Ando†1,2　Palle Dahlstedt†3　Mats G. Nordahl†3　Hitoshi Iba†1
dando@iba.k.u-tokyo.ac.jp

†1Dept. of Frontier Informatics, Graduate School of Frontier Sciences, The University of Tokyo.
†2Art & Technology, IT University of Gothenburg, Sweden
†3Innovate Design, Chalmers University of Technology, Sweden

Research on the application of Interactive Evolutionary Computation (IEC) to the field of musical composition has been improved in recent years, marking an interesting parallel to the current trend of applying human characteristics or sensitivities to computers systems. However, past techniques developed for the IEC-based composition have not necessarily proven very effective for the sake of professional use. This is due to the large difference between data representation used by IEC and authorized classic music composition. To solve these difficulties, we purpose a new IEC approach to music composition based on the classical music theory. In this paper, we describe an established system according to the above idea, and how successfully we can compose a piece using the system.

Keywords: Computer Aided Composition, Interactive EC, Genetic Programming

1　はじめに

1.1　研究の背景

計算機支援による作曲手法は，大別すると決定論的手法と確率論的手法の二つがある [13].

決定論的手法とは厳密に定義された形式的な作曲手法であり，計算機が作曲支援に用いられる以前から，音楽の旋律を生成するために長く使われて来た．通常の場合，曲全体を通して使用されることはないが，対位法や機能和声に代表される厳密なメソッドを用いて楽譜の断片 (スコアマテリアル) を生成することは，クラシック音楽の作曲では一般的である．なお，本稿で” クラシック音楽” と表記した場合，古典的なクラシックの作曲で用いられてきた，音と音との関係性や音楽的な構造を重要視して作曲された音楽を表す．このような作曲技法は，非常に厳密に定義さ

れてきたために計算機への実装が容易であり，計算機の登場とともにプログラムとして実装され，決定論的手法による論理作曲と呼ばれている．

一方，確率論的手法とは，乱数などの不確定要素を音楽生成のパラメータとして用いる方法である．例えば，モーツァルトは賽子による乱数と彼自身が定義したテーブルを使い作曲を行うという手法を考案した．また，計算機が登場してからはより高度な次元で，このような確率論的な計算手法は，旋律や和音進行の生成に積極的に用いられるようになった．Hiller は最初の計算機支援作曲の作品であるイリアック組曲で，基本的な確率論的手法であるモンテカルロ法を用いて全ての古典的楽譜表現が記述出来る事を証明した [6]．楽譜表現とは実際に鳴っている音響ではなく，作曲家が楽譜に記譜する演奏情報を持たない抽象化された音楽の表現のことである．Xenakis は SMP(Stochastic Music Program) という作曲支援プログラムを開発し，積極的に彼自身の作曲に応用した [19, 20]．

決定論的手法に対して，確率論的手法を作曲に用いる利点は，作曲者自身も予想が出来ない結果を得ることにある．しかしながら従来の確率論的な作曲手法では，作曲者が計算機の出力を評価選択し，得られた結果をさらに修正する必要があった．このような作曲手法では，旋律の生成過程は数学的な確率論的手法を応用して実装されている．乱数を直接音符のパラメータの一つとして用いるわけではなく，作曲者によって用意されたメソッドのパラメータとして用いる．したがって出力に対してある程度の方向性が与えられていて，音楽として成り立たない旋律が生成されることはない．しかしながら，乱数を用いるため，出力の段階で作曲者が完全に満足出来る結果を得ることは難しい．

そこで Dawkins によって提案された対話型進化論的計算 [5] を，作曲支援に用いるシステムが提案された．対話型進化論的計算 (Interactive Evolutionary Computation, IEC) では，初期集団は既存の確率論的手法と同じく，ユーザが定義した範囲内でランダムに生成されるが，ユーザとシステムの対話を通して徐々にユーザが求める一点へ収束していき，ユーザは出力結果として完全に満足できるものを得られる．対話型進化論的計算では，初期集団の生成以外でも，交配や突然変異などの遺伝子操作などは基本的にランダムに行われる．そのため，確率論的な手法の一番の特徴である，ユーザが予想できなかったより良い結果をシステムが提示する可能性も充分に残されている．なおかつユーザの思うように収束していくため，確率論的手法の，システムの最終的な出力をユーザが大幅に手直ししなければならないという弱点をカバーすることができる．Roads は著書の中で，計算機支援作曲の二つの大きな流れである確率論的な作曲手法と決定論的な手法をもちいたインタラクティブな作曲に関して，それぞれ述べている [13]．彼の論法に従えば，対話型進化論的計算を用いた作曲システムは，確率論的手法の有利な点である，予想外の結果を得られる可能性と，決定論的手法を用いたインタラクティブな方法の利点である，手直しをしなくてもよい結果を手に入れられるという，両者の利点を兼ね備えている作曲支援の手法であるといえる．

進化論的計算の探索効率は，どのように問題を遺伝子表現としてエンコーディングするか，どのような遺伝子操作を用いるかに大きく影響を受ける．つまり進化論的計算を作曲に応用する時には，作曲プロセスや楽曲をどのような問題として捉え，進化論的計算のメソッドにどのようにマッピングするかが重要になる．

また，遺伝子表現や遺伝子操作だけではなく，ユーザインターフェイスや進化の手順なども探索効率に大きな影響を及ぼす．ユーザは予想できない結果を得ることを期待して，作品の創作に対話型進化論的計算を用いる．収束ポイントや終了条件は最初に決められるわけではなく，ユーザとシステムの対話の中で常に変化していく．したがって通常の問題の探索よりもさらに集団内の多様性が求められる．また対話型進化論的計算はユーザがそれぞれの個体を直接評価するシステムであるため，評価時のユーザの負担を考えると取り扱う事ができる集団サイズを大きい値に設定することができない．この問題を解決するためにルールベースやニューラルネットなどの手法を用いて，巨大な集団の中からユーザの好みに合致する個体のみを提示するフィルタリング手法や提示個体のソートを行う手法も提案されている [2, 14, 17]．この方法はいわゆる「素人」がそれなりの結果を得るには好都合であるが，第一線の作家が実際に創作に使用する場合，予想できない出力を妨害してしまうため，そもそも確率論的な手法を用いる意味が薄れてしまう．したがって，少ない集団サイズを用いて多くのバリエーションを生成する方法が必要である．

また，生成された子集団が親集団を直接置き換えずに，別のウィンドウに表示されるマルチユーザインターフェイスという手法も提案されている [18]．この手法では，ユーザは生成された子集団の成績の良い個体のみ親集団に挿入し，再び子集団の生成を試みることができる．ユーザが進化プロセスに関与するという対話型システムの利点を活かしながら，収束を早める一つの方法である．

1.2 過去の研究事例と問題点

過去にも様々な音楽の遺伝子表現やユーザインターフェイスが試みられてきた．進化論的計算，特に遺伝的アルゴリズム (Genetic Algorighm, GA) と遺伝的プログラミング (Genetic Programming, GP) を作曲に応用した諸研究は Burton によってまとめられている [3]．

Burton が紹介している通り，既存の進化論的計算の作曲への応用の研究におけるエンコーディングやシステムの実装には様々なものがある．

システムの実装では，ユーザの評価における負担軽減を狙い，個体の提示や評価をエージェントとの対話として実装した Biles の GenJam システム [1] や，その仕組みを応用した，エージェントが評価を行う Jacob の COMPOSER-EAR モデル [7] などがある．

音楽の遺伝子表現も様々なものが提案されてきた．まず，楽譜表現や音響を数式によって表現する方法がある．この方法としては Laine らによる旋律の数式表現 [9]，Putnam らによるノイズを合成する関数を合成する方法 [12] などがある．この方法の問題点は，実際にシステムを使う作曲家にとって分かりづらい遺伝子表現であるために，収束の高速化を狙ったユーザの手動による染色体の部分マスキングなどの IEC の諸手法を用いづらいということにある．

また，GP の木構造 (ツリー構造) を用いてクラシック音楽に表れる階層構造を表現するという方法も提案されてい

る．クラシック音楽の世界では，古来からアナリーゼと呼ばれる既存楽曲の分析は重要だとされてきた．また古典的な楽曲の分析結果は，多くの場合，個々のモチーフを示す終端ノードと，モチーフ同士の関係性を示した非終端ノードとの木構造で抽象的に表現されることが可能である．そのため，このツリー構造を用いた表現はクラシックの作曲家にとって非常に馴染みやすく，手動でのマスキングや修正が行いやすいと考えられる．

Johanson らは音符やコードを終端ノード，終端ノードの接続の関係性を非終端ノードとし，これらのノードの組合せによるツリー構造で楽譜表現を表現するシステムを構築した [8]．

Tokui らは，GA と GP のコンビネーションにより，リズムフレーズを表現するシステムを構築した [17]．GA の個体はごく短いリズムパターンを表す．GP の個体は GA によって生成されたリズムパターンを終端ノードとして取り，それらをどのように結合するかを定めた関数を非終端ノードに取り，ツリー構造で抽象的に表す．このような階層構造により，比較的長いフレーズを簡潔で分かりやすい形で表現することが可能になった．

Dahlstedt らは，クラシック音楽における楽曲の生成は全て繰り返し構造として表現できるとして，繰り返し構造をツリー構造で表現することを可能にする Recursive Representation of Music という遺伝子表現を提案した [4]．彼らの方法では，終端ノードは一つの音符かいくつかの音符の配列である．また，二つの引数を連結すると基本的非終端ノードとそれらの非終端ノードに作用して変化を伴った繰り返しを生成する三つのオペレータが提供される．これらのノードの組合せにより，クラシック音楽におけるほぼ全ての楽譜表現を表現することが可能となった．

しかしながら，これらのツリー構造による遺伝子表現や遺伝子操作では，巨大な曲を生成しようとした時にツリー構造が非常に複雑になる．そのため，ユーザが染色体を見ても生成される楽曲の構造を想像できなくなるという問題点が生じる．また小さい集団サイズで複雑な染色体を扱おうとすると収束の効率が悪化する．従って，これらのシステムはフレーズかメロディのような比較的短い音列の生成が中心で，長くても小品程度の長さの曲しか合成出来なかった．

1.3 提案するシステムの特徴

このような過去の研究の成果と問題をふまえ，筆者らは新たなシステム CACIE(Computer Aided Composition using Interactive Evolution) を構築した．CACIE は主に，クラシック音楽の中の無調無拍子の現代曲の作曲を支援する対話型 GP システムである．このシステムの主な特徴を以下に示す．

1. ユーザ (作曲家) が積極的に進化と作曲のプロセスに関与する．
2. スコアマテリアルの生成に特化した，伝統的な作曲家が理解しやすい遺伝子表現と遺伝子操作．
3. 収束効率を悪化させずに充分な長さを持った曲を生成する事が可能．

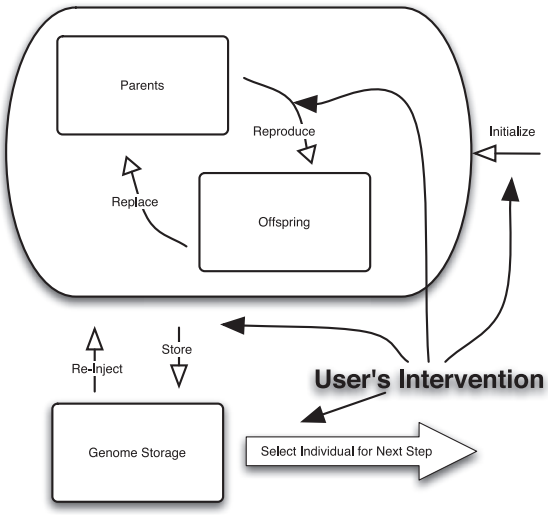

図 1: Overview of CACIE

これらの特徴は，Genome Storage，Multi-Field User Interface，Store と Re-inject，Multiplex Structure，ユーザ定義の染色体の挿入などの機能の実装によって実現されている．

2 システムの構築

2.1 システムの概要

CACIE は，演奏情報や音響を含まない，五線譜上に記述される抽象化された情報 (楽譜表現) を扱う．楽譜表現の実装においては，電子楽器や音楽作成ツールで多く用いられている MIDI(Musical Instruments Digital Interface) を用いた．MIDI は半音単位で音の高さの情報を持っているため，五線譜上の音を全て表現することが可能である．

システムは，Java1.4 を用いて開発した．現在の Java は，音響や MIDI を容易に扱う事を可能とする，Java Sound API ライブラリを提供している．また Java Sound は MIDI ソフトウェアシンセサイザを含んでいるため．CACIE はサウンドボードのみがインストールされていれば，ほぼ全てのコンピュータシステムで繁雑な設定無しに実行可能である (例えば Microsoft Windows，Linux，FreeBSD，Apple MacOSX などで動作する)．

図 1 は CACIE のシステムの概要を表す．通常の対話型 GP システムではユーザは生殖 (Reproduce) 時に適合値を与えるだけであるが，CACIE ではユーザは全てのプロセスに対して積極的に関与する．図 1 では，初期化 (Initialize) 時における各パラメータの設定，Genome Storage への保管 (Store) と集団への再挿入 (Re-inject)，次の段階で使用する個体の選択 (Select Individual for Next Step) などを示した．図中に示した他に，遺伝子のユーザよる直接編集や定義なども行われる．各プロセスの詳細は 2.4 で述べる．

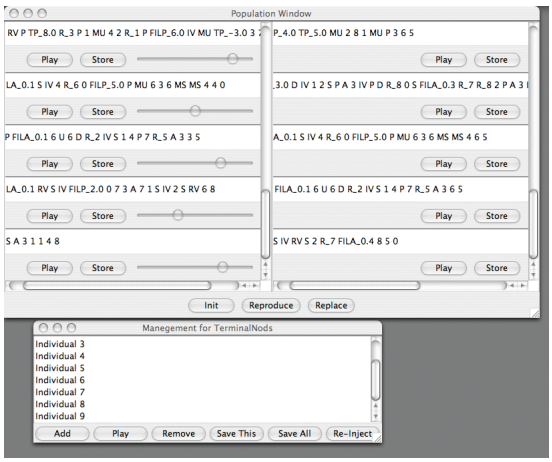

図 2: “CACIE” interface

また図 2 は CACIE のユーザインターフェイスである．上部のウィンドウは二つに分けられ，左が親集団，右が子集団を表している．親集団へ適合値を与えるためにはスライダーを使用する．スライダーで適合度を与えれば，状況に応じてユーザが大まかにも細やかにも適合度を与えられるという利便性がでる．これは，Takagi らが示した適合値のレンジが小さいことによるユーザの負担軽減を狙ったものである [15, 16]．内部的な適合値のレンジは 100 段階としたので，細かな差異が必要になった場合には別のフィールドに直接数値を入力することが可能である．

2.2 遺伝子表現

2.2.1 木構造による音楽構造の表現

CACIE では，楽譜表現の遺伝子表現としてツリー構造を採用した．クラシック音楽の学習における楽曲の分析において，分析結果をツリー構造で表すことは多い．そのため，ツリー構造による音楽構造の表現は音楽家にとって理解しやすい表現手法であると考えられる．ユーザに取って理解しやすい表現であるため，手動での修正など IEC の諸手法が用いやすいという利点がある．

ツリー構造による音楽表現では，非終端ノードの工夫により，様々な音楽的構造を簡易に表現することが可能である．その他にも非終端ノードを適当に用いることで，同一形態の表現で小規模なフレーズから大きな楽曲構造まで表現することができる．

図 3 にスコアとツリー構造の相互変換を示す．一つの音符には，音高を表す Note Number，音の強さを表す Amplitude，音の長さを表す Duration，発音タイミングを表す Onset Time の各要素が含まれている．また Amplitude が 0 の時は，その音符は休符として扱われる．ここで非終端ノードの S は Sequence の略であり二つの引数を順番に演奏する関数である．同様に U は Unite の略であり二つの引数を同時に演奏する関数である．終端ノードとして

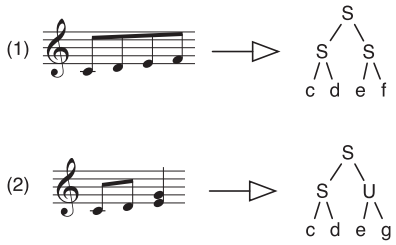

図 3: Tree representation of musical phrase

は，一つ又は複数の音符からなるリストを取る．一つの音符は MIDI をベースにした形で表現される．

図 3，(1) において，木構造の終端ノードは左からそれぞれ，c，d，e，f の音高を持っている．また，各音の長さは八分音符とする．この時，この木構造は左に示した旋律となる．また，(2) においては，e の音を持ったノードと g の音を持ったノードが U 関数で接続されているため，同時に発音される．また，OnsetTime は関数が音列を出力する時に自動で調節されるように実装を行った．なお，このツリー構造は線形 GP を使って実装されている．

また，Sequence と Unite 以外にも様々な関数が提供される．表 1 に実装された関数の一部を提示する．

用意されている全ての関数は典型的なクラシック音楽で使用されている音楽構造を作曲家が理解しやすい形で実装したものである．例えば，D は二つ目のノードに取った音列のリズムパターンを一つめの音列に適用する関数である．表中においては，一つめのノードとして a という Note Number が 60，Amplitude が 100，Duration が 20 の音符，二つ目のノードとして b という Note Number が 62，Amplitude が 120，Duration が 10 の音符を取った例を提示した．D 関数により，二つ目のノードの Duration 要素 10 が一つめのノードの Duration を置き換え，結果として，a’ という Note Number が 60，Amplitude が 100，Duration が 10 の音符を返す．また，RV，IV は一つの子ノードを取り，それぞれ逆行形，反行形を作り出す関数である．TP+n は音列全体の音高 n 半音分変更する．MS は二つのノードを取り，それぞれのノードが表す音列から一音づつ交互に演奏していく関数である．SR+n は二つの引数を交互に合計 n 回繰り返し演奏する関数である．FILP+n は，一つめのノードの音列に対して n 半音幅の音高変更を行いながら，二つ目のノードの音列の始めの音に達するまで繰り返しを行う関数である．このように複数の音列の要素を変更し，それらを組み合わせて新しい音列を作り出す作業は，変奏の生成として実際の作曲の過程において頻繁に行われるために，有効であると判断した．また，関数として実装する音楽構造の選択にあたっては，二つの終端ノードをどのように繋げるかという関係性を重視した．

これらの関数は再利用可能なライブラリとしても提供した．したがって，ユーザである作曲家が，新しい作品のアイデアとして音楽構造を考案した場合，システムに用意されている関数を組合せて新しい関数をプログラミングし，新たにシステムに取り込むことが可能になっている．

表 1: List of part of functions

S	Connect two arguments continuously `(S a b) = (a b)`
U	Connect two arguments simultaneously `(U a b) = (ab)`
SR	Make a repetition of two arguments `(SR+5 a b) = (a b a b a)`

D	Apply rhythm pattern of 2nd argument to 1st argument
	`(D a(60,100,10) b(62,120,20))`
	`= a'(60,100,20)`

P	Apply pitch pattern of 2nd argument to 1st argument
	`(P a(60,100,10) b(62,120,20))`
	`= a'(62,100,10)`

A	Apply articulation pattern of 2nd argument to 1st argument
	`(A a(60,100,10) b(62,120,20)`
	`= a'(60,120,10)`

RV	Reverse ordering `(RV (a b)) = (b a)`

IV	Pitch Inversion
	`(IV a((60,100,10)(62,120,20))`
	`= a'((62,100,10)(60,120,20))`

TP	Pitch Transposing
	`(TP+5 a((60,100,10)(62,120,20))`
	`= a'((65,100,10)(67,120,20))`

MS	Return a sequence of arranged notes taken from two nodes alternately as follows: `(MS (a b c) (d e)) = (a d b c e)`
MU	Return a sequence of arranged notes taken from two nodes simultaneously as follows: `(MU (a b) (c d)) = ((U a c) (U b d))`
CAR	Return a sequence that containts the front X% of number of notes as follows: `(CAR50% (a b c d)) = (a b)`
CDR	Return a sequence that contains the rear X% of number of notes as follows: `(CDR50% (a b c d)) = (c d)`
ACML	Return an accumulated sequence `(ACML (a b c)) = (a a b a b c)`

FILP	Return a sequence that contains a repetition with pitch transposing until second node pitch as follows:
	`(FILP+2 (a(60,amp,dur) b(63,amp,dur))`
	`c(65,amp,dur))`
	`= (a(60,..) b(63,..)`
	`a'(62,..) b'(65,..)`
	`a''(64,..) b''(67,..) c(65,..))`

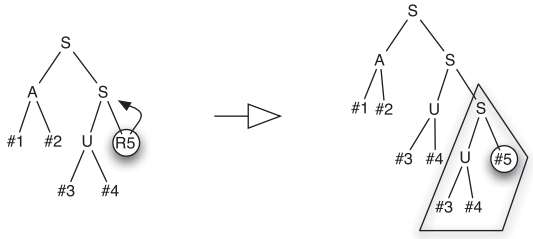

図 4: Developmental process of a Recursive Terminal Node

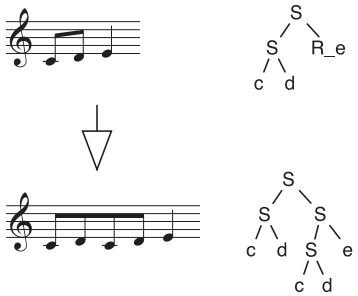

図 5: Example of a Recursive Terminal Node

2.2.2 再帰終端ノード

CACIE は Recursive Terminal Node という特殊な終端ノードを持っている．これは前述の Dahlstedt らの方法を改良し GP への適用を容易にしたもので，染色体の肥大化を防ぎながら，クラシック音楽的な繰り返し構造の効率的な生成を実現する．この終端ノードは自分が Recursive Terminal Node である事を示すラベルと終端ノード一つを表すラベルを保持する．Recursive Terminal Node は Ontogeny フェイズの時点で，自らの直上の非終端ノードを参照し，その部分木で自分を置き換える．同時にコピーされた部分木内の Recursive Terminal Node は，自身が保持する通常の終端ノードに置き換えられる．図 4，5 は Recursive Terminal Node の展開例である．

2.3 遺伝子操作

交配は通常の部分木の交換と線形 GP の一点交差を採用した．また突然変異として，通常の個体内での部分木の交換の他に Increase と Decrease という特殊な操作を採用した．

Increase 操作は前述の Recursive Terminal Node の働きを突然変異として実装したものである．図 6 は Increase 操作の例を示している．この操作を適用されるノードはランダムに選択される．

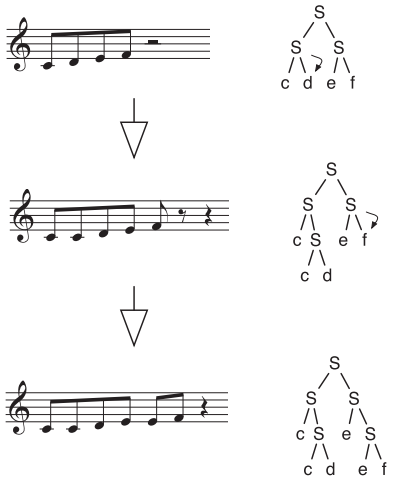

図 6: Increase mutation

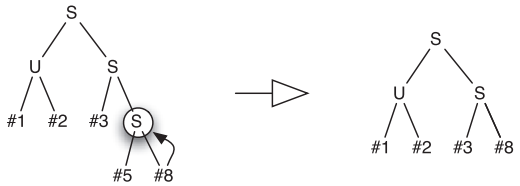

図 7: Decrease mutation

Decrease は Increase の反対の操作である．ランダムに選択された非終端ノードをその直下の終端ノードで置き換える．例を図 7 に示す．また Decrease 操作は，個体の染色体の長さがユーザによって決められた限界を越えた時に優先的に適用される．

また，遺伝操作を適用する個体は，Fitness-proportional Selection によって決められる．

2.4 インターフェイスと作曲プロセス

2.4.1 マルチフィールド・ユーザインターフェイス

CACIE では，前述した Multi-Field User Interface の考えかたを採用した．図 2 が示す通り，インターフェイスのウィンドウは二つに分けられ，それぞれ親集団と子集団が提示される．子集団は生成された直後に親集団を置き換えずに別のウィンドウに表示される．もし結果が気に入らなければ，ユーザは再度子集団を生成することが可能である．また廃棄する子集団の中に気に入った固体があれば，後述する Genome Storage を用いて親集団の中にその個体を挿入することもできる．

また，Unemi らによって提案された，ユーザが染色体の修正や定義する手法も採用した．この手法は対話型進化

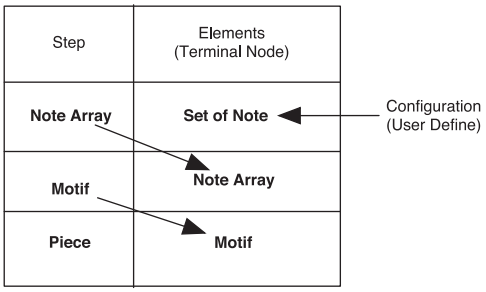

図 8: Multiplex Structure: Phenotypes of a previous step are used for terminal nodes of the next step

論的計算において収束の高速化において有効である．CACIE では後述する Genome Storage を通じて，染色体や phenotype である MIDI イベントリストをテキストファイルとして出力したり，取り込みを行うことが可能である．ユーザは出力をテキストエディタで修正することにより，染色体の修正や定義を行う．

2.4.2 作曲プロセスの複合構造化

CACIE における楽曲の生成は，クラシック音楽の作曲の過程をモデル化した Multiplex(複合) 構造によって行われる．典型的な古典クラシック音楽の作曲の過程はいくつかに分けられる．最初はごく短い音列を作成する段階，次に音列を用いたモチーフとその変奏を作成する段階，最後は生成されたモチーフや変奏を変形しながら一つの曲として組み合わせる段階である．筆者らはこのクラシック音楽の複合構造的な作曲方法を取り込んだ．CACIE では，ユーザは一度の探索で曲を生成するのではなく，段階的に曲を生成していく．この複合構造は，それぞれの探索の結果を MIDI をベースとしたイベントリストとしてエクスポートし，次回の探索の終端ノードとして初期集団生成時にインクルードするという実装法で実現している．

図 8 に，複合構造を用いた CACIE での作曲の一例を示す．最初のステップでは，作曲家が，使用したい基準となる音を終端ノードとして与える．その他に，初期集団の木構造群を生成するために必要な，木の深さ，使用する関数とその出現頻度，一つの木に含まれるノード数の最小値と最大値を，設定 (Configuration) として与える．この段階でユーザはシステムを使い，気に入った音列 (Note Array) を複数生成する．これをデータファイルとして出力し，次の構造 (図中では Motif) を生成する段階で，終端ノードとして初期化に用いるという流れである．各段階ごと，もしくは初期化ごとに初期集団の生成のための設定は変更を行うこともできる．この繰り返しにより，最終的に長い曲の生成を行う．

2.4.3 遺伝子ストレージ

図 2 の下部に見えるウィンドウは Genome Storage と呼ばれる．この Genome Storage は進化の過程に対するユーザの積極的な関与を実現するために実装した．図 9 に Genome Storage の機能の概要を示す．ユーザはいつでも親集団や子集団から好きな個体を選択し，その染色体と phenotype を Genome Storage に保管 (Store) することが可能である．また，Genome Storage に蓄えられた個体をいつでも親集団へ挿入 (Re-inject) することが可能である．

その他にも Genome Storage は，前述の個体の染色体とイベントリストをテキストファイルとして出力し，取り込む機能を持つ．これによりユーザが染色体を手動で修正し集団に Re-inject したり，またユーザ本人が全く新たに定義した染色体を集団に挿入する操作が可能である．また，それらの操作の支援のために，各個体をプレイバックする機能も持っている．

Multiplex 構造はこの Genome Storage の機能を用いて実現されており，作曲の各ステップの流れは図 10 が示すような流れになる．

作曲家は，初期化時に，終端ノードとして一つの音符や音列のセット，非終端ノードとして使用する関数と登場頻度のリスト，木構造の深さや複雑さをパラメータとして与える (図中 i)．システムはそれをもとに初期集団を生成し，作曲家は生殖と置き換えを繰り返す (ii)．この進化プロセスの中で，作曲家は個体に対する評価値を与えるだけでなく，そのつど気に入った個体を Genome Storage に保管し (iii)，手動で修正などを加えながら (iv)，個体の変奏を作るために進化を行っている集団に再挿入を行う．Genome Storage に，次のステップで使える個体が充分に揃ったと作曲家が判断した場合，進化を打ちきり，Genome Storage の中の評価値が高い個体群とその変奏を出力し，次のステップへ移る (v)．充分な数が収集できていなければ，他の音列や関数のセット，木構造生成のパラメータなどを変更し，再び同じ過程を繰り返す．この状態でも一度 Genome Storage に保管した個体を再挿入することは可能である．通常の対話型進化論的計算の場合，ユーザの好みにあった場所へ集団が収束するまで進化プロセスは行われるが，CACIE の場合は，最終的に曲を合成するステップを除いて，気に入った個体を収集することが目的となる．

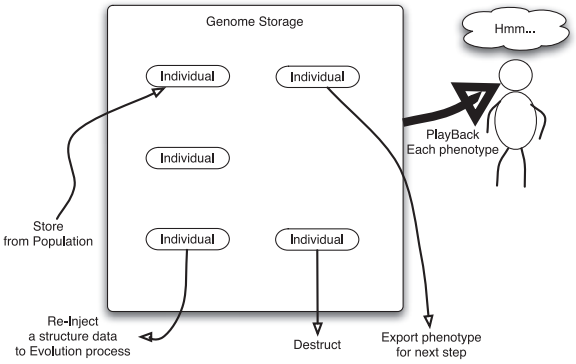

図 9: Genome Storage - Storing the user's ideas

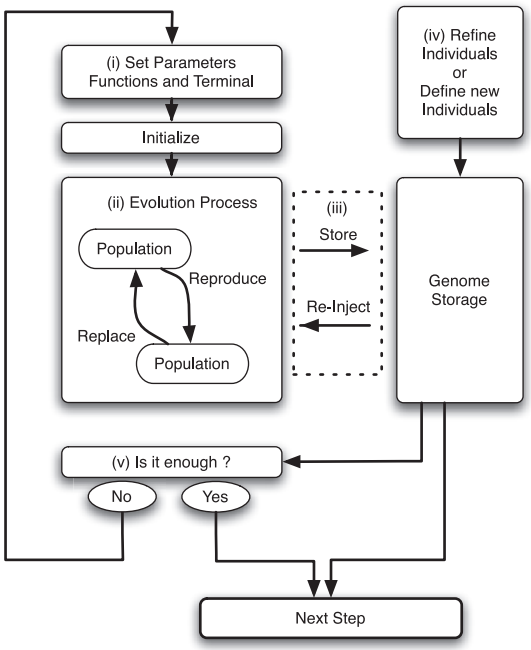

図 10: Composition Process in each step: User should collect materials for the next step, trying to apply various combinations of parameters, functions and terminal nodes.

3 実験結果と考察

3.1 基本的な進化

遺伝子，染色体表現と遺伝子操作の妥当性を確かめるために，基本的な非終端ノードのみをシステムに与え，簡単なフレーズの合成を行った．この実験では，Store や Re-inject などの操作を一切行っていない．図 11，12 は生成された結果である．

この実験では，終端ノードとして音価が全て 8 分音符の C，E，G，A と一オクターブ上の C，非終端ノードとしては S を二つ，MS，SR，FILP をシステムに与えた．各関数の初期集団への登場の割合は，ここで与えた個数に

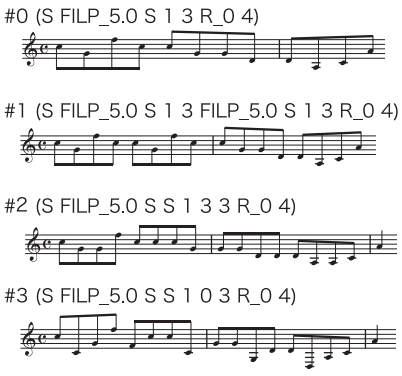

図 11: Evolved Code Example(1)

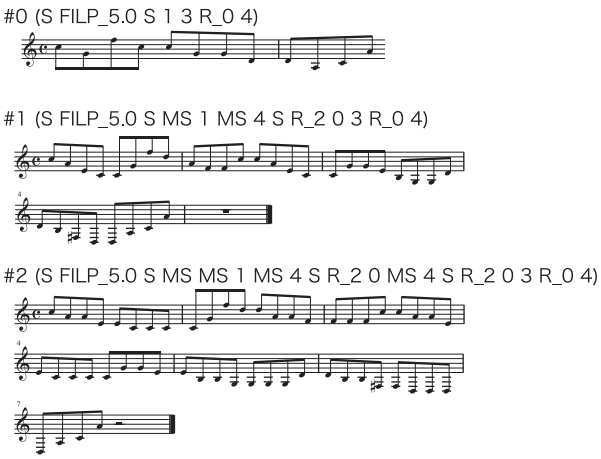

図 12: Evolved Code Example(2)

よる．また，集団数は 16，初期集団のツリーの深さは最大 3 に設定した．また，Recursive Teminal Node の登場の割合はすべてランダムである．

図 11 においては#0 が初期世代，#1 と#2 が第一世代，#3 が第二世代から，それぞれ抽出したものである．いずれの個体も典型的なクラシック音楽に見られるような変化を伴った繰り返し構造を持った変奏が生成されている．これらの繰り返しは主に RecursiveTerminal Node によるものである．

図 12 は Figure11 と同じ親から始まる別の進化である．#0 が初期世代，#1 が第一世代，#2 が第二世代である．この例の場合，世代を経るにつれて変化を伴った繰り返し構造が複雑化し，ツリー構造が成長していることが示されている．これは Increase 操作が有効に働いているためであると考えられる．

3.2 音列，モチーフの段階の合成

次に，実際の作曲で使われるいくつかの音符の配列やモチーフの生成，そして変奏の生成を行う実験を行った．この実験では，Genome Storage を用いた Store と Re-inject を行っている．図 13 は生成されたフレーズ，図 14 は生成過程を，それぞれ示している．

生成のプロセスは以下のようになる．

1. 初期集団を生成し，#0 のみを選択．自己交配と突然変異を適用し，子集団から#1 と#2 を取り出す．
2. #0，#1，#2 を Genome Storage に Store する．
3. 新たに初期集団を生成し，#0，#1，#2 をその集団に Re-inject し，全ての個体に対し適合値を与える．
4. 3 の集団に対し，交配と突然変異を適用．
5. #3，#4，#5，#6 が生成された．

この実験では，集団サイズを 16 個体に設定した．通常の場合，少ない集団サイズはでの進化は，生成された個体が偏ってしまい多様性が失われ，効率的な変奏の生成を阻害する．しかしこの生成例では，クラシック音楽の返送の生成過程においてよく行われる，旋律の全体や一部の繰り返しやピッチトランスポーズ，音型の概要を残したリズムパターンの拡張や圧縮などの音楽的に有効な変奏が，ひとつのパターンに偏ることなくバリエーション豊かに生成されていることがわかった．このことから，Store と Re-inject の操作を行うことで，少ない世代数でも効率的に音楽的に意味のある多くのパターンの変奏を生成できることが確認できた．

3.3 旋律，曲の段階の合成

最後に，曲の中のセクションや，小品として充分成り立つ長さを持った曲の生成の実験を行った．システムが出力した結果を，そのまま SMF として World Wide Web 上に提示した．

`http://www.iba.k.u-tokyo.ac.jp/~dando/public/projects/ecmusic/master_thesis/master_thesis.html`

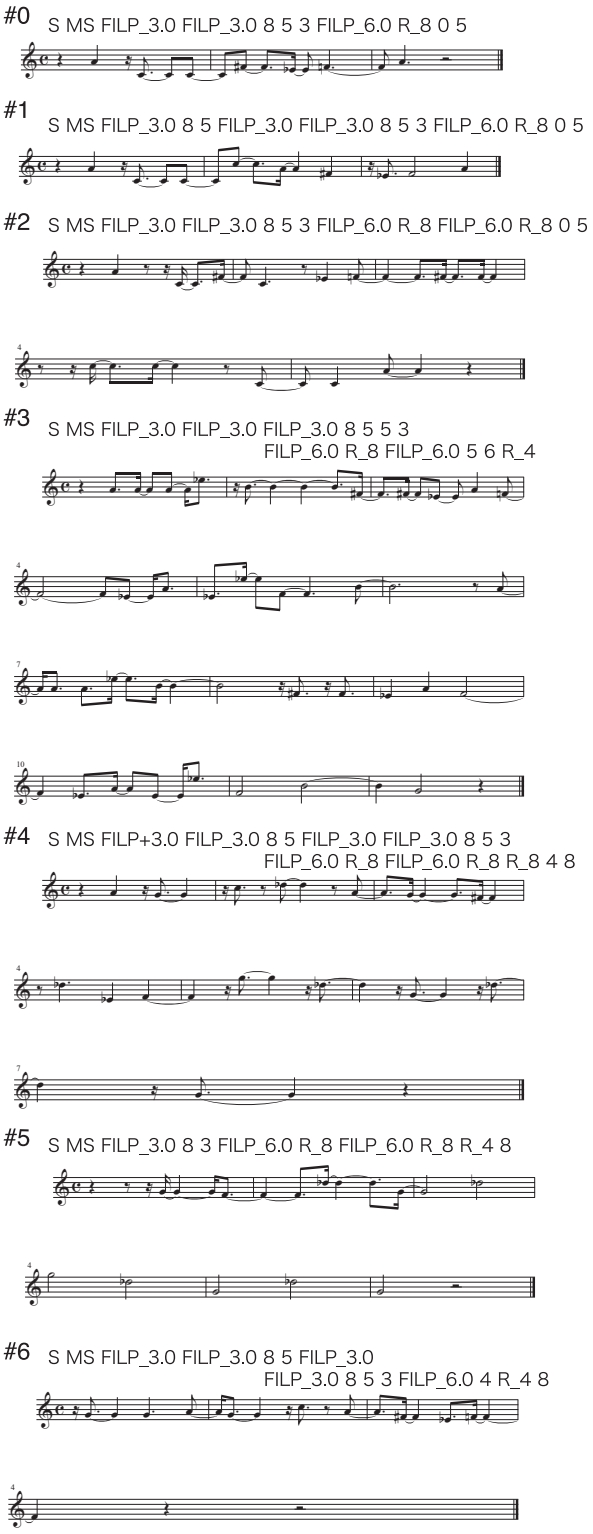

図 13: Phrase example

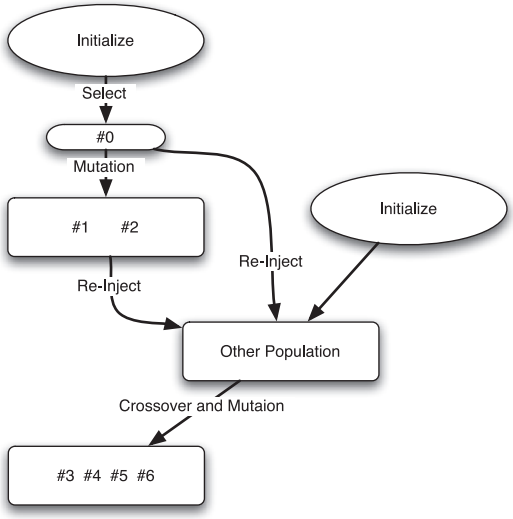

図 14: Transition of evolution for an individual shown in figure 13

`via3step_1.mid`, `via3step_2.mid`, `via3step_3.mid` は Multiplex 構造を 3 段階用いて生成された例である．ここでは，ユーザによる染色体の修正や定義は行っていない．最初のステップでは 3 から 5 音の短い音列の生成を行った．次に，その結果の phenotype を終端ノードとして，5 から 15 音からなるモチーフを生成した．最後に，一番目と二番目のステップで生成された結果を終端ノードとして，曲中のセクションを生成した．同様に，`via4step_1.mid` は 4 段階で生成した例である．

いずれの結果も多くの音符を含んでいるにもかかわらず，最終的なステップにおける木構造はそれほど肥大化していない．ユーザがツリー構造を見て，phenotype がどうなるかを予測できる程度である．また，phenotype において典型的な音楽構造がはっきりと表れていて，理解しやすいものになっている．

しかしながら，進化の効率という観点から見るとこの結果は問題を含んでいる．Developing Example 2 において，筆者は親の個体に対して曲の終止をより強く感じさせるような旋律を曲の最後に付け加えたいと感じた．しかしながら生成された子集団には筆者の希望通りの個体は含まれていなかった．これは，少ない集団サイズのためであると考えられる．

そこで，筆者らは，Genome Storage の染色体エクスポート機能によって得られたテキストファイルに対して手動による染色体の修正を行い，最終的にピアノの小品を生成した．出力結果に対して，MIDI 編集ソフトウェア上で，オクターブトランスポージングなどのごく基本的なアレンジを手動で付加した SMF を前述の Web ページに掲載した．また，楽譜編集ソフトウェアで楽譜の形にし，楽譜上にアーティキュレーションの指定を付加した楽譜も掲載した (前述の Web ページの `rattfylla.mid`,

rattfylla.pdf).

得られた曲には，クラシック音楽に見られる典型的な繰り返し構造が大きなレベルから小さなレベルまで見られた．曲の全体的な構造は A-B-C-D-A'-B'-E という形になっている．また各セクションの中にも，モチーフの変奏を生成しながら繰り返しを行う旋律が多く見られた．

この合成した曲を楽譜の形に変換し，アーティキュレーションの指定などを行った上で，職業演奏家 (Torgny Stintzing 氏．元スウェーデン Gothenburg 大学音楽学部ピアノ演奏教師) に演奏してもらった所，コンサートで演奏できるぐらいの曲にはなっているとの，比較的高い評価をえることが出来ている．この評価から，本システムがある程度の妥当性を持っていることが示された．また，作曲者はこの曲を生成する過程で，本システムの自由度を活かしスウェーデンの民族音楽リズムを取り込んでいるため，評価が高い曲を得られたと考えられる．この結果を元に，このような専門職とのコラボレーションを現在行っている．

3.4 システムの評価

筆者らは本システムのプレゼンテーションを行うとともに，数人のクラシック現代音楽の作曲家にシステムの使用をしてもらい，感想と改善の余地を自由形式で答えてもらった．好評だった点は，複合構造の採用により長い曲を生成できる点と，音楽構造を直接参照し修正できる点であった．その他に，気に入った個体のみを容易に選択し，その個体の変奏が容易に生成できることも利点としてあげられた．また，木構造による楽譜表現に関しては，配列などを用いた遺伝子構造に比べ理解しやすい，とのコメントが得られた．しかしながら，直接参照，修正などのインタフェースが GUI でないことが不満点として多くあげられた．作曲のためのプログラミングを行う作曲家は多いため，本システムにおけるプログラミングを利用した作曲家自身の定義による新しい音楽構造の導入などの自由度は評価するが，GUI でプログラミングが行えると利便性が増す，との指摘もあった．

以上の結果から，システムの概念自体は評価されるが，ユーザインタフェースの改善を望む意見が多く，インタフェースについての考察と改善は急務であると考えられる．

4 おわりに

本稿では，伝統的な作曲家が使うための対話型進化論的計算についての考察と新たなシステムの構築の報告を行った．筆者らが新たに構築したシステムは，クラシック音楽のアナリーゼ手法に基づいた遺伝子表現と作曲手法に基づいた生成プロセスを基本としている．結果として，伝統的な音楽表現を含んだ比較的長い曲の合成に成功した．

今回のシステムは，経験的に獲得されたアナリーゼ手法をマッピングしたものである．今後の展望としては，音楽認知の方面で提案されてきた GTTM[10] や IR[11] などの楽曲分析手法を応用し，作曲支援システムとして対話型進化論的計算にマッピングすることを考えている．

参考文献

[1] J. Biles. Genjam: A genetic algorithm for genrating jazz solos. In *Proceedings of 1994 International Computer Music Conference, Arhus*. ICMA, September 1994.

[2] J. Biles, P. G. Anderson, and L. W. Loggi. Neural network fitness functions for a musical iga. In *Proceedings of IIA'96/SOCO'96. Int'l ICSC Symposia on Intelligent Industrial Automation And Soft Computing, Reading, UK*, pp. B39–44, 1996.

[3] A. R Burton and T. Vladimirova. Generation of musical sequences with genetic techniques. *Computer Music Journal*, Vol. 24, No. 4, pp. 59–73, Winter 1999.

[4] P. Dahlstedt and M. G. Nordahl. Augumented creativity: Evolution of musical score material. *appendix of Palle Dahlstedt, Ph.D. thesis, Sounds Unhead of, Chalmers University of Technology*, 2004.

[5] R. Dawkins. *The Blind Watchmaker*. Longman, Essex, 1986.

[6] L. Hiller and L. Isaacson. *Experimental Music*. MacGraw-Hill, New York, 1959.

[7] B. L. Jacob. Composing with genetic algorithms. In *Proceedings of 1995 International Computer Music Conference, Alberta*. ICMA, September 1995.

[8] B. E Johanson and R. Poli. Gp-music: An interactive genetic programming system for music generation with automated fitness raters. *Technical Report, CSRP-98-13, School of Computer Science, The University of Birmingham*, 98.

[9] P. Laine and M Kuuskankare. Genetic algorithms in musical style oriented generation. In *Proceedings of First IEEE Conference on Evolutionary Computation, Washington D.C*, pp. 858–861. IEEE, 1994.

[10] F. Lerdahl and R Jackendoff. *Generative Theory of Tonal Music*. MIT Press, 1983.

[11] E. Narmour. *The Analysis and Cognition of Basic Melodic Structures*. University of Chicago Press, 1990.

[12] J. B. Putnam. Genetic programming of music. *Technical Report, New Mexico Institute of Mining and Technology*, 1994.

[13] C. Roads. *The Computer Music Tutorial*. MIT Press, 1996.

[14] H. Takagi. Interactive ga for system optimization: Problems and solution. In *Proceedings of 4th European Congress on Intelligent Techniques and Soft Computing (EUFIT'96), Aachen, Germany*, pp. 1440–1444, 1996.

[15] H. Takagi and K. Ohya. Discrete fitness values for improving the human interface in an interactive ga. In *Proceedings of IEEE 3rd International Conference on Evolutionary Computation (ICEC'96), Nagoya*, pp. 109–112. IEEE, 1996.

[16] H. Takagi, K. Ohya, and M. Ohsaki. Improvement of input interface for interactive ga and its evaluation. In *Proceedings of International Conference on Soft Computing (IIZIKA'96), Iizuka*, pp. 490–493. World Scientific, 1996.

[17] N. Tokui and K. Iba. Music composition with interactive evolutionary computation. In *Proceedings of 3rd International Conference on Generative Art (GA2000)*, 2000.

[18] T Unemi. A design of multi-field user interface for simulated breeding. In *Proceedings of 3rd Asian Fuzzy System Symposium: The Korea Fuzzy Logic and Intelligent Systems*, 1998.

[19] I. Xenakis. Elements of stochastic music. *Gravesaner*, Vol. 18, pp. 84–105, 1960.

[20] I. Xenakis. *Formalized Music*. Indiana University Press, Bloomington, 1971.