2005－MUS－59　(10)
2005／2／18

対話型進化論的計算による作曲支援システム:CACIE

安藤 大地 †1,2　Palle Dahlstedt†3　Mats Nordahl†3　伊庭 斉志 †3

†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

対話型進化論的計算 (IEC) を用いた作曲支援システムに関する研究は，近年発展を続けている．しかしながら，既存の研究の多くは実際の作曲家に積極的に使用されることはなかった．その大きな理由は，対話型進化論的システムを用いた作曲と伝統的な作曲手法との間に大きな差異が存在することであると考えらえる．そこで筆者らは，第一線の作曲家が実際にシステムを利用することを目的として，それに適した遺伝子表現や作曲プロセスを考案した．また，それらを用いた作曲支援システムを構築し，有効性を確認した．

CACIE: Computer Aided Composition System by Interactive Evolutionary Computation

Daichi Ando†1,2　Palle Dahlstedt†3　Mats Nordahl†3　Hitoshi Iba†1

†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 characters 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 deference 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.

1　はじめに

乱数などの不確定要素を楽曲生成のパラメータとして用いる確率論的な作曲手法は，古来より作曲の一手法として利用されてきた．また，計算機が登場してからより高度な次元で，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できることを証明した [1]．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 [2, 3]．

確率論的な作曲手法を用いる利点は，作曲者自身も予想ができ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．

そこで Dawkins によって提案された対話型進化論的計算 [4] 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ものへ収束していき，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．また対話型進化論的計算の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ため，ユーザが予想できなかったより良い結果を得る余地も多分に残されている．

進化論定計算の探索効率は，どのように問題を遺伝子として表現するか，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は，作曲プロセスや楽曲をどのような問題としてとらえ，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なく，ユーザインタフェ

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．これは，ユーザが直接評価を与える対話型進化論的計算では，扱える個体数や世代数が制限されるという理由による．対話型進化論的計算のこれらの問題を解決する方法の研究例は，[6, 7, 9, 8] などである．

過去にも様々な音楽の遺伝子表現やユーザインタ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 Algorithm, GA) と遺伝的プログラミング (Genetic Programming, GP) を用いた研究は [10] にまとめられている．

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 [11, 12] などがある．また，音楽の遺伝子表現も様々なものが提案されてきた．楽譜表現や音響を数式によって記述する方法 [13, 14]，GP のツリー構造によって音符やコードの接続の関係性を表現する [15, 8] などがある．特に後者のツリー構造によって楽譜表現を記述する方法は，古典的な音楽学の分析作業でも用いられる．そのため理解が容易であり，手動による遺伝子の修正などの IEC の諸テクニックが用いやすいという利点がある．さらに，ツリー構造による楽譜表現をさらに発展させた，クラシック音楽に見られる典型的な繰り返し構造を表現する方法も提案されている．[16]

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．そのためユーザが提示された染色体を見た時に，生成される楽曲の構造の直感的な把握が難しくなる．したがって，前記のユーザによる染色体の手動修正などが適用しづらいという問題が生じる．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終息の効率が悪化する．これらの理由により，既存のシステムは，フレーズかメロディのような比較的短い音列の生成への適用が中心で，ユーザの負担を考えると，長くても小品程度の長さの曲しか合成できなかった．

このような過去の研究の成果と問題を踏まえ，筆者らは新たなシステム CACIE(Computer Aided Composition using Interactive Evolution) を構築した．CACIE は無町の現代音楽作品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. 終息効率を悪化させずに充分な長さを持った曲を生成することが可能．

2　システムの構築

2.1　システムの概要

CACIE のシステムは，Java1.4 を用いて開発した．現在の Java は，音響や MIDI(Musical Instruments Digital Interface) 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．

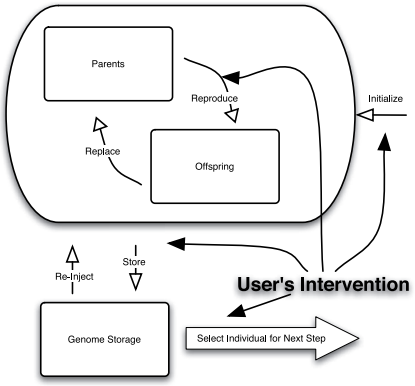

図 1: Overview of CACIE

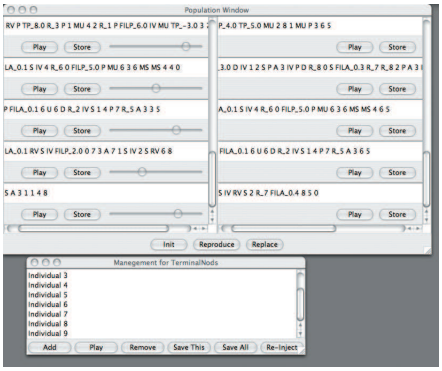

図 2: “CACIE” interface

図 1 は CACIE のシステムの概要を示している．通常の対話型進化論的計算ではユーザは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．各プロセスの詳細は後述する．

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 [20, 21]．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．

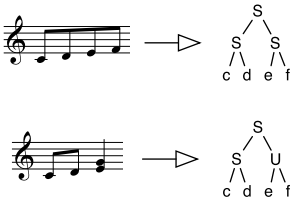

図 3: Tree representation of musical phrase

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．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．つまり一つの音符には Note Number，Amplitude，Duration，OnsetTime 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．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．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．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．表 1 に実装された関数の一部を提示する．

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，ユーザは，システムで提供された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である．

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

MS	Return a sequence as follows: `(MS (a b c) (d e)) = (a d b c e)`
MU	Return a sequence as follows: `(MU (a b) (c d)) = ((U a c) (U b d))`
CAR	Return a sequence as follows: `(CAR50\% (a b c d)) = (a b)`
CDR	Return a sequence 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 contains repetition with pitch transposing as follows:
	`(FILP+2 (60 63) 65)`
	`= (60 63 62 65 64 67 65)`

2.2.2 再帰終端ノード

筆者らは，通常の終端，非終端ノードの他に，RecursiveTerminal Node という特殊な終端ノードを用意した．これにより染色体の肥大化を防ぎながら，効率的にクラシック音楽的な繰り返し構造を実現することができる．Recusive Terminal Node は自らの直上の非終端ノードを参照し，その部

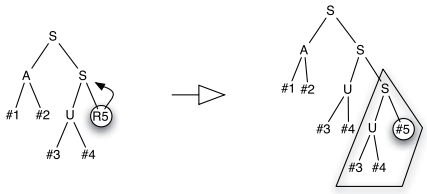

図 4: Developmental process of a Recursive Terminal Node

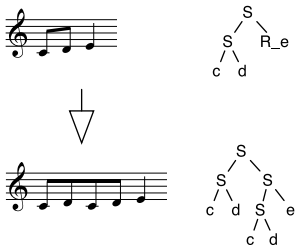

図 5: Example of a Recursive Terminal Node

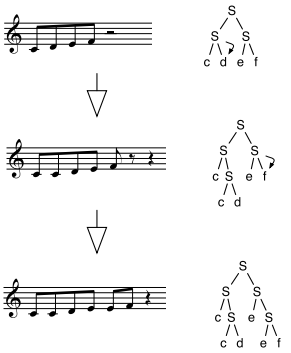

図 6: Increase mutation

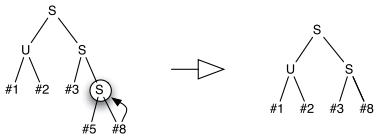

図 7: Decrease mutation

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 に展開例を示す．

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．

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，過去の研究で有効性が確認されている，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 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．

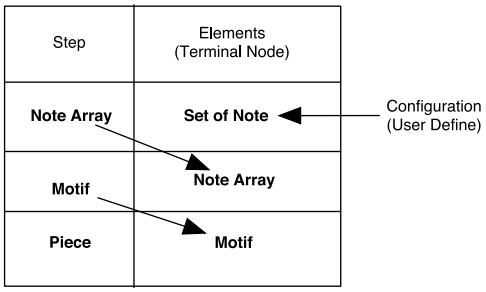

図 8: Multiplex Structure: Phenotypes of a previous step are used for terminal nodes of the next step

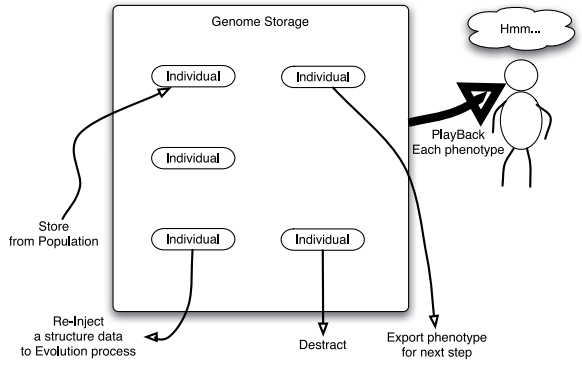

図 9: Genome Storage - Storing the user's ideas

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．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．

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．

3 実験結果

遺伝子表現とシステムの妥当性を検証するために，実際にモチーフと楽曲の合成を行った．結果は SMF の形で World

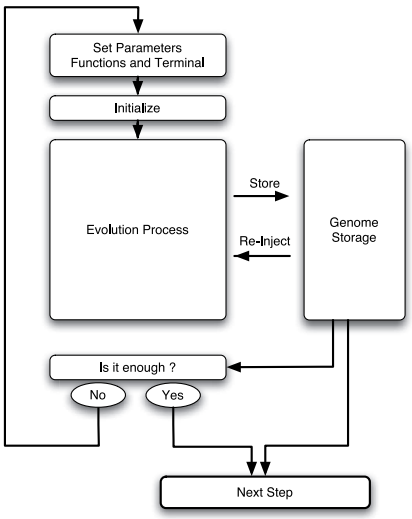

図 10: Flow chart of evolutionary process: User should collect materials for the next step, trying to apply various combinations of parameters, functions and terminal nodes.

Wide Web 上に提示した．

`http://cad.lolipop.jp/public/projects/ecmusic/master_thesis/master_thesis.html`

検証は，三段階にわけて行った．まず始めに，ツリー構造と Recursive Temrinal Node，特殊な遺伝子操作を確かめるために，Genome Storage を使わずに極短い音列の生成を行った．ツリー構造が変化を伴った繰り返し構造を生成するように成長していることを確認した．進化の過程の結果を図 11 に示す．

次に，対話型進化論的計算のユーザインタフェイスの妥当性を確かめるために Genome Storage を利用したモチーフ

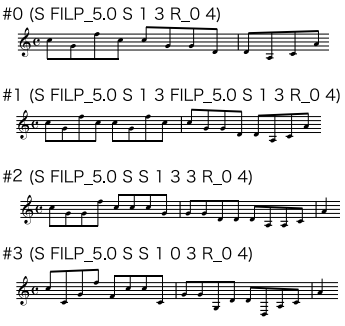

図 11: A family of score fragments: #0 is parent. #1, #2 and #3 are offsprings.

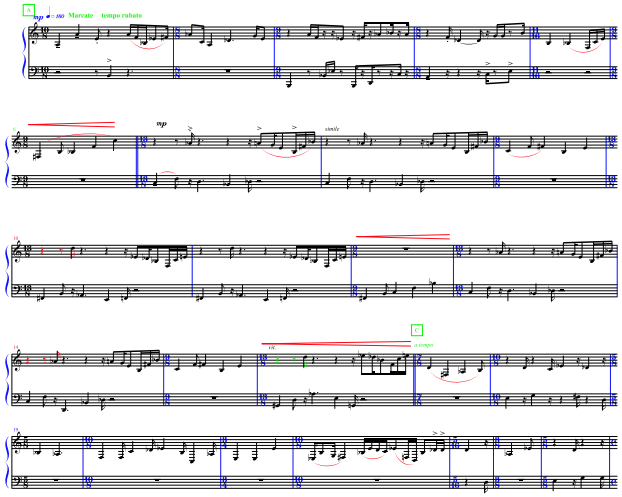

図 12: A evolved piano miniature

のと変奏の生成を行った．Genome Storage を使用し，結果の進化への再挿入などを行うと，少ない集団サイズにもかかわらず，音楽的に意味のある変奏を効率よく生成できることが確認できた．

最後に，複合構造を検証するために，比較的長い曲の合成を行った．ここでは，前記の再挿入などの手法の他に，ユーザの手動による染色体の修正も行い，音楽的な構造を容易に生成できることを確認した．

それらの結果を踏まえ，最終的にピアノの現代曲的な小品の合成を行った．得られた結果には，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．生成された曲の冒頭の部分を図 12 に示す．

また合成した曲を職業演奏家 (Torgny Stintzing 氏．元 Gothenburg 大学音楽学部ピアノ演奏教師) に演奏してもらった所，比較的高い評価をえることが出来ている．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．

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[17] や IR[18] などの楽曲分析手法を，対話型進化論的計算システムにマッピングすることを考えている．

参考文献

[1] Hiller, L., and L., Isaacson. 1959. “Experimental Music” New York, McGraw-Hill.

[2] Xenakis, I. 1960. “Elements of stochastic music” in Gravesaner Blatter 18, pp84-105.

[3] Xenakis, I. 1971. “Formalized Music” Bloomington, Indiana University Press.

[4] Dawkins, R. 1986. “The Blind Watchmaker” Longman, Essex.

[5] Roads, C. 1996. “The Computer Music Tutorial” MIT Press.

[6] Takagi, H. 1996. “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.

[7] Biles, J. A., Anderson, P. G. and Loggi, L. W. 1996. “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.

[8] Tokui, N. and Iba, H. 2000. “Music Composition with Interactive Evolutionary Computation” in Proceedings of the 3rd International Conference on Generative Art:GA2000.

[9] Unemi, T. 1998. “A Design of Multi-Field User Interface for Simulated Breeding” in Proceedings of the 3rd Asian Fuzzy System Symposium:The Korea Fuzzy Logic and Intelligent Systems

[10] Burton, A. R. and Vladimirova, T. 1999. “Generation of Musical Sequences with Genetic Techniques.” in Computer Music Journal, 23:4, pp.59-73, Winter 1999.

[11] Biles, J. 1994. ”GenJam : A genetic algorithm for generating jazz solos.” in Proceedings of the 1994 International Computer Music Conference, Arhus:ICMA, September 1994.

[12] Jacob, B. L. 1995. “Composing with genetic algorithms” in Proceedings of the International Computer Music Conference, Banff Alberta:ICMA, September 1995.

[13] Larine, P., and Kuuskankare M. 1994. “Genetic Algorithms in Musical Style Oriented Generation.” in Proceedings of the First IEEE Conference on Evolutionary Computation, Washington, DC:IEEE, pp.858-861.

[14] Putnam, J. B. 1994. ”Genetic Programming of Music.” in Technical Report, New Mexico Institute of Mining and Technology.

[15] Johanson, B. E., Poli, R. 1998. “GP-Music: An Interactive Genetic Programming System for Music Generation with Automated Fitness Raters” in Technical Report CSRP-98-13, School of Computer Science, The University of Birmingham.

[16] Dalhstedt, P., and Nordahl, M. 2004. “Augumented Creativity: Evolution of musical score material” appendix of Palle Dahlstedt, “Sounds Unhead of”, Ph.D. thesis, Chalmers University of Technology.

[17] Lerdahl, F., and Jackendoff, R. 1983. “Generative Theory of Tonal Music” MIT Press.

[18] Narmour, E. 1990. “The Analysis and Cognition of Basic Melodic Structures” University of Chicago Press.

[19] Miranda, E. R. 2001. “Composing Music with Computers” Focal press.

[20] Takagi, H., Ohya, K. 1996 “Discrete Fitness Values for Improving the Human Interface in an Interactive GA” in Proceedings of the IEEE 3rd International Conference on Evolutionary Computation (ICEC'96), Nagoya, pp.109-112.

[21] Takagi, H., Ohya, K., Ohsaki, M. 1996. “Improvement of Input Interface for Interactive GA and its Evaluation” in Proceedings of the International Conference on Soft Computing (IIZUKA'96), Iizuka, pp.490-493, World Scientific.