Vol.2017-MUS-114 No.19
2017/2/27

秩序と混沌の度合いを考慮したメロディー生成に対する数理計画法の適用

村上智之[†1]　森口聡子[†1]

概要：これまで多くの作曲家により，アルゴリズム作曲法と呼ばれる数学的な手続きによる作曲法が数多く提案されてきた．本研究では，数理計画法の数理的な手続きからアルゴリズム作曲にアプローチする．ここで，良い音楽とは聴き手に次の展開を予測させておきその予測を裏切るような音楽である，との見方がある．そこで，人間にとっての「予測可能性と予測不可能性」や「秩序と混沌」といった相反する 2 つの性質を適度に含む音楽の生成を目指す．秩序と混沌の度合いを考慮しつつ数理計画法の定式化とアルゴリズムを用いて作曲を行う枠組みを提案する．

Application of Mathematical Programming to Melody Generation Considering the Degree of Order and Chaos

TOMOYUKI MURAKAMI[†1]　SATOKO MORIGUCHI[†1]

1. はじめに

私たちの身の回りにあふれている音楽の中には，意外性がありながらも心地良く感じられるものがある．私たちが音楽を楽しむのは，音楽から秩序と驚きが感じられるためであると言われている[3,6]．本研究では，数理的な手続きやアルゴリズムにより，秩序と混沌の両方が感じられるようなメロディーの生成を目指す．そのメロディー生成の過程に数理計画法を用いている．数理計画法とは，一定のルールの下でより良い方法を数理モデルと計算機によって求める方法である[2].「音高」や「音階」に関する数理モデルを作り，数理計画法の計算により得られる最適解や暫定解をメロディーの生成に利用する.提案する 2 つのテーマ「0-1 整数計画問題による音階の生成」,「音高の分散が異なる複数の和音列を元にしたメロディーの生成」について述べる．

2. 0-1 整数計画問題による音階の生成

2.1 音階の生成方法

ある曲に複数の音階が用いられることを想定し，0-1 整数計画問題により複数の音階を生成する．0-1 整数計画問題の中のあるパラメーターを変化させると，生成される音階の構成音が変化する．その音階の構成音の変化は，コンピューターは容易に予測できるが，人間は容易には予測できない．しかし，変化する音階の構成音同士に共通な音が多ければ,人間でも次の音階や音を予測できる部分がある．本研究では，生成される複数の音階の構成音の変化について，人間にとっての予測可能性と予測不可能性，秩序と混沌のバランスを検討する．

2.2 定式化と計算方法

音階を生成するための 0-1 整数計画問題による定式化について述べる．$x_i\ (i = 1, 2, \dots, 12)$は 0-1 変数とし，12 平均律における 1 オクターブ内の 12 個の音について，音階の構成音に使用するか否かを表すこととする.すなわち,C の音を 1 番目とし，$x_i = 0$はi番目の音を音階に使用しないことを表し，$x_i = 1$はi番目の音を音階に使用することを表す.目的関数は$x_i\ (i = 1, 2, \dots, 12)$の和であり，生成する音階に使用される音の数を表す．制約式は，jを自然数として，$x_i\ (i = 1,2,\dots,12)$ を $((i + j - 2) \bmod 12) + 1$ で重み付けした和が整数b以下であるという式について，$j = 1,2,\dots,12$としたものである．この 0-1 整数計画問題【SG0-1IP(b)】は次のように表される．

【SG0-1IP(b)】

$$\text{Max. } Z = \sum_{i=1}^{12} x_i$$

$$\text{s.t. } \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 1 \\ 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 1 & 2 \\ 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 1 & 2 & 3 \\ 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 1 & 2 & 3 & 4 \\ 6 & 7 & 8 & 9 & 10 & 11 & 12 & 1 & 2 & 3 & 4 & 5 \\ 7 & 8 & 9 & 10 & 11 & 12 & 1 & 2 & 3 & 4 & 5 & 6 \\ 8 & 9 & 10 & 11 & 12 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 9 & 10 & 11 & 12 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 10 & 11 & 12 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 11 & 12 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 12 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{pmatrix} \leq \begin{pmatrix} b \\ b \\ b \\ b \\ b \\ b \\ b \\ b \\ b \\ b \\ b \\ b \end{pmatrix}$$

$$x_i \in \{0,1\}\ (i = 1,2,\dots,12)$$

†1 首都大学東京
Tokyo Metropolitan University

ここで，bは入力パラメーターである．この 0-1 整数計画問題の最適解が，生成される音階の構成音となる．定式化のポイントは，全ての変数x_iを一様に扱うということである．つまり数学的な性質は全ての変数x_iについて等価である．これにより，bというパラメーターを変化させた場合，生成される音階の構成音に共通な音が多くなると考えられる．このことは，人間にとっての予測可能性や秩序の部分を担うと考えられる．一方で，bというパラメーターを変化させた場合に生成される音階がどのように変化するかは，最適化の計算に用いる数理計画ソルバーの計算結果に依存する．ソルバーによる計算の結果を人間が容易に予測できないという点は，人間にとっての予測不可能性や混沌の部分を担うと考えられる．

本節での最適化の計算には GLPK(GNU Linear Programming Kit)を用いる．GLPK は線形計画問題，混合整数線形計画問題を解くための最適化ソルバーであり，単体法，主双対内点法，分枝カット法を実装している[1]．GNU により無償で配布されている．本節の計算では，GLPK のパッケージ含まれている GLPSOL を用い，問題の記述には CPLEX LP 形式を用いる．計算環境は，1.7GHz デュアルコア Intel Core i5, 4GB 1600MHz メモリ, 64bit OS X El Capitan である．

2.3 計算結果

$b \leq 11$の場合は$x_i = 0\ (i = 1, 2, \ldots, 12)$，$b \geq 78$の場合は$x_i = 1\ (i = 1, 2, \ldots, 12)$であることは自明であるため，$12 \leq b \leq 77$の範囲について GLPK を用いて求解を行う．計算結果を表 1 に示す．表 1 の見方について説明する．1 行目を除く 1 列目の数字は，今回変化させるパラメーターbの値である．それぞれのbの値に対応した 0-1 整数計画問題の最適値を 2 列目に，最適解を 3 列目以降に示している．今回設定した目的関数から，最適値は生成する音階に使用される音の数を表す．今回重要なのは音階の構成音を表す最適解の方であり，この表では$x_i = 1$，つまり音階の構成音として使用することを表すセルの背景色を緑色としている．最も下の行には，x_iに対応する音名を示している．表 1 からは例えば，$b = 38$の場合に生成される音階の構成音は{D, E, G, A, B}である，といったことが読み取れる．

2.4 考察

表 1 には，bを変化させたときの音階の構成音の変化について，共通な音を持ちつつ変化していく部分が見られる．全体的な音階の変化について，中程度の相関関係が見受けられる．すなわち，音階の構成音の変化について，規則的な部分とランダムな部分の両面が見受けられる．規則的な部分は全ての変数x_iを一様に扱ったことに起因すると考えられ，ランダムな部分は GLPK の分枝カット法のアルゴリズムに起因すると考えられる．以上のことから，構成音の変化が秩序と混沌の両方の性質を持ち合わせている複数の音階を生成することができたと言える．生成した複数の音階の使用例を次節で示す．また定式化の際，全ての変数x_i

表 1　$12 \leq b \leq 77$ の場合の最適解
Table 1　Optimal solutions for $12 \leq b \leq 77$.

b	Z	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12
12	1	0	0	0	0	0	0	0	0	0	0	0	1
13	1	0	0	0	0	0	0	0	1	0	0	0	0
14	1	0	0	0	0	0	1	0	0	0	0	0	0
15	1	0	0	0	1	0	0	0	0	0	0	0	0
16	1	0	0	0	0	0	0	0	0	1	0	0	0
17	1	0	0	0	0	0	0	1	0	0	0	0	0
18	2	0	0	0	0	0	1	0	0	0	0	0	1
19	2	0	0	0	0	0	1	0	0	0	0	1	0
20	2	0	1	0	0	0	0	0	0	0	1	0	0
21	2	0	0	0	0	0	0	0	1	0	0	1	0
22	2	0	0	0	0	1	0	1	0	0	0	0	0
23	2	0	0	0	0	0	1	0	0	1	0	0	0
24	3	0	0	1	0	0	0	1	0	0	0	1	0
25	3	1	0	0	0	1	0	0	0	0	1	0	0
26	3	1	0	0	0	1	0	0	0	0	1	0	0
27	3	1	0	0	0	0	1	0	0	0	1	0	0
28	3	0	0	1	0	0	0	1	0	0	0	0	1
29	3	0	0	0	1	0	1	0	0	1	0	0	0
30	4	0	0	1	0	0	1	0	0	1	0	0	1
31	4	1	0	0	1	0	0	1	0	0	1	0	0
32	4	0	1	0	1	0	0	1	0	0	1	0	0
33	4	1	0	0	1	0	0	1	0	1	0	0	0
34	4	0	0	1	0	0	1	0	1	0	0	1	0
35	4	0	1	0	1	0	0	1	0	1	0	0	0
36	4	0	1	0	1	0	1	0	0	0	0	0	1
37	4	0	1	0	1	0	0	0	0	0	1	0	1
38	5	0	0	1	0	1	0	0	1	0	1	0	1
39	5	1	0	1	0	1	0	0	1	0	0	1	0
40	5	1	0	0	0	1	0	1	0	1	0	1	0
41	5	0	1	0	1	0	0	0	1	0	1	0	1
42	6	0	1	0	1	0	1	0	1	0	1	0	1
43	6	1	0	1	0	1	0	1	0	1	0	1	0
44	6	0	1	0	1	0	1	0	1	0	1	0	1
45	6	1	0	1	0	1	0	1	0	1	0	1	0
46	6	1	0	1	0	1	0	1	0	1	0	1	0
47	6	1	0	1	0	1	0	1	0	1	0	1	0
48	6	1	0	0	0	1	1	0	1	1	0	1	0
49	6	1	1	0	1	0	0	0	1	0	1	1	0
50	6	1	1	0	0	0	1	0	1	0	1	1	0
51	7	1	0	1	0	1	0	1	1	0	1	0	1
52	7	1	1	0	1	1	0	1	0	1	1	0	0
53	7	0	1	1	0	1	1	0	1	0	1	0	1
54	7	1	0	0	1	1	0	1	1	0	1	0	1
55	7	0	0	1	1	0	1	1	0	0	1	1	1
56	8	1	1	0	1	1	0	1	1	0	1	1	0
57	8	0	1	1	0	1	1	0	1	1	0	1	1
58	8	0	1	1	0	1	1	0	1	1	0	1	1
59	8	0	1	1	0	1	1	0	1	1	0	1	1
60	8	1	1	1	0	1	1	0	1	1	0	1	0
61	8	1	1	1	0	1	1	0	0	1	1	1	0
62	8	1	0	0	1	1	0	1	1	1	0	1	1
63	9	1	1	0	1	1	1	0	1	1	1	0	1
64	9	1	1	1	0	1	1	1	0	1	1	1	0
65	9	1	0	1	1	1	1	0	1	1	1	0	1
66	9	1	1	1	0	1	1	1	0	1	1	0	1
67	9	1	1	1	0	1	1	1	1	0	1	0	1
68	9	0	0	1	1	1	1	1	1	0	1	1	1
69	9	0	1	1	1	1	1	1	0	1	0	1	1
70	10	1	1	1	1	0	1	1	1	1	1	0	1
71	10	1	1	0	1	1	1	1	0	1	1	1	1
72	10	1	0	1	1	1	1	1	1	1	0	1	1
73	10	1	1	1	0	1	1	0	1	1	1	1	1
74	10	1	1	1	1	1	1	1	0	1	0	1	1
75	10	1	1	1	0	1	0	1	1	1	1	1	1
76	10	1	1	1	1	1	1	1	1	0	1	0	1
77	11	1	1	1	1	1	1	1	1	1	0	1	1
		C	C#	D	D#	E	F	F#	G	G#	A	A#	B

を一様に扱うポイントを押さえれば，また違った複数の音階を生成することが可能であると思われる．例えば，制約式の係数行列の全ての要素に同じ整数を加えるなどの操作が考えられる．この操作を行なっても，全ての変数x_iが数学的に一様に扱われることに変わりはない．

3. 音高の分散が異なる複数の和音列を元にしたメロディーの生成

3.1 音高の分散による曲のモデル

私たちが耳にする音楽は通常，低い音から高い音まで様々な音高の音が用いられて 1 つの曲として成立している．また，高音域や低音域に比べ，中音域の音の方がよく出現すると考えることができる．そこで，ある曲に用いられている数々の音の音高が，ある音の音高を中心として上下に広がっている，というモデルを考える．ある音高を中央値としたときの音高の分散が小さい音列から大きい音列までを重ね合わせることで，低い音から高い音まで様々な音高の音が存在する状況が生じる．そのイメージを図 1 に示す．図 1 は，音高の分散の大きさが，分散(1) < 分散(2) < 分散(3) < 分散(4)であるような 4 つの異なる音列を重ねた音列を表す．図 1 から，音高の分散が小さい音列から大きい音列までを重ねることで，低い音から高い音まで様々な音高の音が存在し，かつ中音域の音の方がよく出現する状況が現れることが分かる．本節では，複数の音高の分散を持つ和音列を元にしてメロディーを生成することを試みる．最初に生成するメロディーを「和音列」，和音列を元にして最終的に生成するメロディーを「メロディー」として区別している．

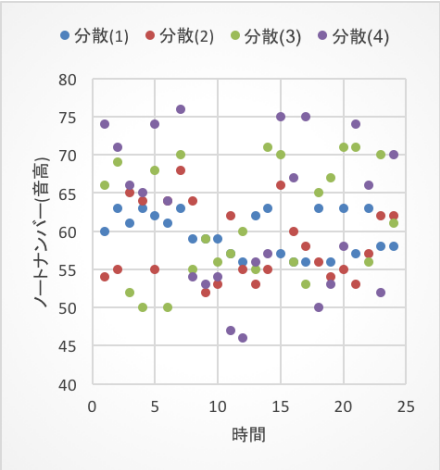

図 1　音高の分散が異なる 4 つの音列を重ねた音列

Figure 1　A tone sequence made of 4 tone sequences in different pitch variance.

3.2 方法

まず，メロディー生成の全体の流れについて述べる．最初に和音列を生成するために，分散が0である初期値から始め，分散を目的関数とした最大化問題を内点法により解く．この最大化問題を解く途中で得られる暫定解をノートナンバーにマッピングすることにより，音高の分散が異なる複数の和音列が得られる．そのようにして得られた和音列からある音階を抽出して時間的に少しずつずらして配置することで，メロディーを生成する．

次に，和音列を生成する方法について述べる．3 声体の和音列を 2 小節分生成する．具体的には，音価が 1 拍の 3 声体の和音を 8 個生成して並べることを考える．3 声体の和音の構成音のうち，最も音高が高い音から順に，第一声部，第二声部，第三声部という．生成する音符の数は$3\times8=24$個であり，そのノートナンバー（音高）をx_{ij}と表す．ここで$i=1,2,3$であり，それぞれ第一声部，第二声部，第三声部に対応している．また$j=1,2,\ldots,8$であり，何番目の和音の音かを表す．例えば，x_{24}は 4 番目の和音の第二声部の音のノートナンバー（音高）を表す．

3.3 定式化と計算方法

和音列生成のための定式化について述べる．本節において分散は標本分散としている．音高の分散を目的関数とし，次のように表される二次計画問題【PVQP】に対して，内点法により目的関数の最大化を目指す．

【PVQP】

$$\text{Max.}\quad \frac{1}{ij}\sum_{i=1}^{3}\sum_{j=1}^{8}\left(x_{ij}-\bar{x}\right)^2$$

$$\text{s.t.}\quad \bar{x}=\frac{1}{ij}\sum_{i=1}^{3}\sum_{j=1}^{8}x_{ij}$$

$$x_{ij}\in\mathbb{Z}$$

$$x_{1j}-x_{2j}\geq 2\ \ (j=1,2,\ldots,8)\quad\cdots\cdots(1)$$
$$x_{2j}-x_{3j}\geq 2\ \ (j=1,2,\ldots,8)\quad\cdots\cdots(2)$$

$$54\leq x_{1j}\leq 83\ \ (j=1,2,\ldots,8)\quad\cdots\cdots(3)$$
$$45\leq x_{2j}\leq 74\ \ (j=1,2,\ldots,8)\quad\cdots\cdots(4)$$
$$36\leq x_{3j}\leq 65\ \ (j=1,2,\ldots,8)\quad\cdots\cdots(5)$$

$$x_{31}=x_{22}\quad\cdots\cdots(6)$$
$$x_{12}=x_{33}\quad\cdots\cdots(7)$$
$$x_{23}=x_{14}\quad\cdots\cdots(8)$$
$$x_{34}=x_{25}\quad\cdots\cdots(9)$$
$$x_{15}=x_{36}\quad\cdots\cdots(10)$$
$$x_{26}=x_{17}\quad\cdots\cdots(11)$$
$$x_{37}=x_{28}\quad\cdots\cdots(12)$$

ここで，制約式(1)は，第一声部は第二声部よりも音高が高いことを表し，式(2)は，第二声部は第三声部よりも音高が高いことを表す．ただし今回，第一声部と第二声部の間，第二声部と第三声部の間に短 2 度の不協和音程が生じることを避けるために，半音 2 個分以上の差をつけた．式(3)，式(4)，式(5)については，それぞれ第一声部，第二声部，第三声部の音域の制約を表す. 式(5)までが基本的な制約となる．

基本的な制約のみで最適化の計算を行うと，音高が声部ごとに同じになる傾向がある．このことは，全く同じ和音が 8 回連続し，秩序の度合いのみが大きいメロディーが生成されることを意味する．ゆえに，混沌の度合いをある程度含ませるためには，発展的な制約が必要となる．発展的な制約には様々なものが考えられるが，今回は 2 音間の音程関係についての式(6)から式(12)を付加した．たとえば式(6)は，1 番目の和音の第三声部の音の音高と 2 番目の和音の第二声部の音の音高が同じ，ということを表す．式(7)から式(12)についても同様である．

最大化問題【PVQP】の内点法による解の探索には，MATLAB[4] と Optimization Toolbox[5] を用いる．Optimization Toolbox には，線形計画法，混合整数線形計画法，二次計画法，非線形最適化などのソルバーが含まれている．計算環境は，1.7GHz デュアルコア Intel Core i5, 4GB 1600MHz メモリ, 64bit OS X El Capitan である．

分散の最大化を目指す途中で内点法の反復(解の更新)を終了することで得られる暫定解をノートナンバーにマッピングし，和音列を生成する．この方法は内点法の反復回数に上限を設けることにより実現する．この上限は，1 回から最適解が得られるまでの回数とする．最適解が得られる回数は制約条件により異なる．制約と変数x_iの初期値によっては，内点法の初期の反復では暫定解が制約を満たさない場合がある. 今回は暫定解が制約を満たさない場合でも，その暫定解を受け入れて使用する．

初期値は$x_{ij} = 60\ (i = 1, 2, 3, j = 1, 2, \ldots, 8)$とする．目的関数については，MATLAB で予め用意されている var 関数を用い，第二引数に 1 を指定し，標本分散とした．また，Optimization Toolbox で用いたソルバーは制約付き非線形最小化であるため，実際に解く問題の目的関数は分散に-1を乗じたものである．Optimization Toolbox で連続緩和問題を解き，その解を整数に丸めた．また目的関数の値も重要度が低いことから整数に丸め，-1を乗じ，本来の目的関数である分散の値とした．

3.4 計算結果とメロディーの生成

計算結果を表 2, 3 に示す．本実験では，35 回目の反復で得られた暫定解が最適解であった．この計算結果を可視化したものを図 2 に示す．図 2 の横軸は変数のインデックスを表している．内点法の反復回数が少ないほど点の色を薄く，反復回数が多いほど点の色を濃く示しており，内点法の計算による各変数の推移が分かる．

表 2　1 回から 18 回の反復回数における暫定解
Table 2　Incumbent solutions for iterations 1 to 18.

反復回数	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
分散	0	1	6	11	101	163	165	190	202	217	220	223	227	237	251	255	257	259
x_{11}	60	61	63	64	76	81	81	83	83	83	83	83	83	83	83	83	83	83
x_{12}	60	60	60	60	60	60	60	59	59	59	59	59	59	59	59	59	59	59
x_{13}	60	62	64	66	77	81	81	83	83	83	83	83	83	83	83	83	83	83
x_{14}	60	61	62	63	68	70	70	71	72	73	74	74	74	74	74	74	74	74
x_{15}	60	60	60	60	58	58	58	57	57	56	56	56	55	54	54	54	54	54
x_{16}	60	62	65	66	78	83	83	83	83	83	83	83	83	83	83	83	83	83
x_{17}	60	61	62	63	70	73	73	74	74	74	74	74	74	74	74	74	74	74
x_{18}	60	60	62	62	66	68	68	69	70	70	71	71	72	75	77	78	79	80
x_{21}	60	60	61	61	63	63	63	64	64	65	65	65	66	68	72	73	74	74
x_{22}	60	59	58	57	52	49	49	48	47	46	46	45	45	45	45	45	45	45
x_{23}	60	61	62	63	68	70	70	71	72	73	74	74	74	74	74	74	74	74
x_{24}	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60
x_{25}	60	59	58	57	51	49	49	47	46	45	45	45	45	45	45	45	45	45
x_{26}	60	61	62	63	70	73	73	74	74	74	74	74	74	74	74	74	74	74
x_{27}	60	60	60	61	62	63	63	63	64	64	64	65	66	68	71	72	72	72
x_{28}	60	59	58	58	54	53	53	52	52	51	51	51	50	48	45	45	45	45
x_{31}	60	59	58	57	52	49	49	48	47	46	46	45	45	45	45	45	45	45
x_{32}	60	58	56	54	43	38	38	36	36	36	36	36	36	36	36	36	36	36
x_{33}	60	60	60	60	60	60	60	59	59	59	59	59	59	59	59	59	59	59
x_{34}	60	59	58	57	51	49	49	47	46	45	45	45	45	45	45	45	45	45
x_{35}	60	58	56	54	43	39	38	37	36	36	36	36	36	36	36	36	36	36
x_{36}	60	60	60	60	58	58	58	57	57	56	56	56	55	54	54	54	54	54
x_{37}	60	59	58	58	54	53	53	52	52	51	51	51	50	48	45	45	45	45
x_{38}	60	58	56	55	45	41	41	40	38	36	36	36	36	36	36	36	36	36

表 3　19 回から 35 回の反復回数における暫定解
Table 3　Incumbent solutions for iterations 19 to 35.

反復回数	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
分散	265	265	265	265	267	269	269	270	275	275	275	275	275	275	275	275	275
x_{11}	83	83	83	83	83	83	83	83	83	83	83	83	83	83	83	83	83
x_{12}	58	58	58	58	57	54	54	54	54	54	54	54	54	54	54	54	54
x_{13}	83	83	83	83	83	83	83	83	83	83	83	83	83	83	83	83	83
x_{14}	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74
x_{15}	54	54	54	54	54	54	54	54	54	54	54	54	54	54	54	54	54
x_{16}	83	83	83	83	83	83	83	83	83	83	83	83	83	83	83	83	83
x_{17}	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74
x_{18}	83	83	83	83	83	83	83	83	83	83	83	83	83	83	83	83	83
x_{21}	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74
x_{22}	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45
x_{23}	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74
x_{24}	61	61	61	61	61	63	64	66	72	72	72	72	72	72	72	72	72
x_{25}	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45
x_{26}	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74	74
x_{27}	72	72	72	72	72	72	72	72	72	72	72	72	72	72	72	72	72
x_{28}	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45
x_{31}	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45
x_{32}	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36
x_{33}	58	58	58	58	57	54	54	54	54	54	54	54	54	54	54	54	54
x_{34}	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45
x_{35}	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36
x_{36}	54	54	54	54	54	54	54	54	54	54	54	54	54	54	54	54	54
x_{37}	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45
x_{38}	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36

ソルバーによる計算で得られた内点法の各反復での暫定解をノートナンバーにマッピングすることにより，和音列を得る．その和音列を用い，次の 2 つの操作を行う．

・操作(1): ある 4 つの和音列を選び，16 分ずつずらして配置する．

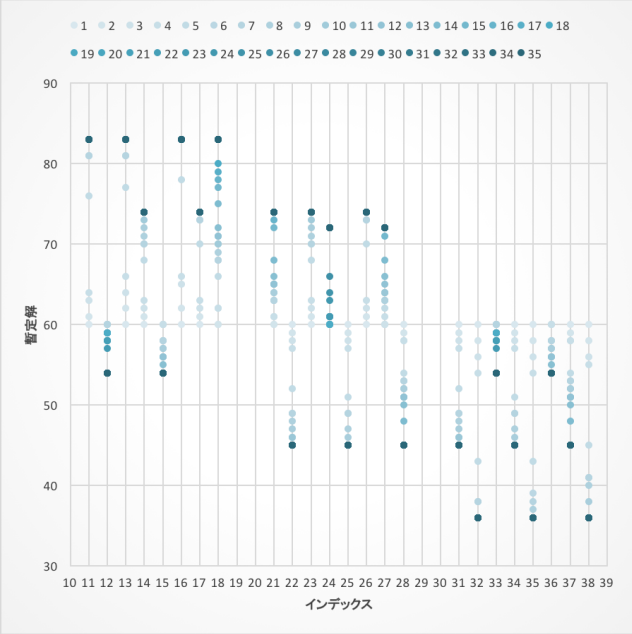

図 2　内点法の計算過程における各変数の値の推移
Figure 2　The transition of variable values in the iterations of interior point method.

・操作(2): 和音列からある音階を抽出して並べる．

操作(1)は，図 1 で示したように，音高の分散が小さい音列から大きい音列までを重ねることで，低い音から高い音まで様々な音高の音が存在し，かつ中音域の音の方がよく出現する状況を作るための操作である．ただし，和音列の発音のタイミングは 16 分ずつずらす．4 つの和音列の選び方には様々なものが考えられる．今回用いた選び方について説明する．まず，mを，整数に丸めた解が初めて内点法の反復回数の違いにより変化しなくなる反復回数未満の最大の 4 の倍数とする．次に，mと，mを 4 で除した商をmから 1 回，2 回，3 回と引いていった数から同じ整数nを引いた配列を作り，その要素を反復回数として選ぶ．その配列は次のように表される．

$$\left[m-n, \frac{3m}{4}-n, \frac{m}{2}-n, \frac{m}{4}-n\right] \quad (n=0,1,2,\ldots,\frac{m}{4}-1)$$

今回の場合$m=16$であるので，具体的な配列は

$$[16,12,8,4],[15,11,7,3],[14,10,6,2],[13,9,5,1]$$

となる．

操作(2)は，秩序と混沌の両方が感じられるメロディーにするための操作である．ある音階を抽出した場合にどのようなメロディーになるかということは，人間は容易には予測できない．しかし，抽出する複数の音階に何らかの予測可能性があれば，その複数の音階を抽出したものを並べることで，秩序を感じることができるだろうと考えた．今回は次の 2 通りの音階を抽出した．

・音階(1): ハ長調から属調をたどりヘ長調に至るまでの 12 個のダイアトニックスケール

・音階(2): 2 節で示した 0-1 整数計画問題により得られる音階で$b=12$から$b=77$としたもの

$[16,12,8,4],[15,11,7,3],[14,10,6,2],[13,9,5,1]$の和音列から音階(1)を抽出して並べ，この順に結合したものを“var_diatonic”と名付けた[*1].

また，$[16,12,8,4],[15,11,7,3],[14,10,6,2],[13,9,5,1]$の和音列から音階(2)を抽出して並べたものを，それぞれ“var_01IP_16”,“var_01IP_15”,“var_01IP_14”,“var_01IP_13”と名付けた[*2,*3,*4,*5].

3.5 考察

音階(1)は属調へ転調していくものであり，転調するたびに音階の構成音が 1 つずつ変化していく．その変化はゆるやかなものであるため，“var_diatonic” は繰り返しがよく感じられるものとなった．音階(2)についても，2 節で述べたように変化が緩やかな部分があり，音階(2)を抽出したメロディーにも時折繰り返しが感じられる．この繰り返しは秩序の部分を担っている．一方で，繰り返しを感じさせつつもメロディーは変化していき，どのように変化するかは予想がつかないものとなった．これは混沌の部分を担っている．

今回数理計画法を用いたメリットとして，まず，制約を容易に実現できることが挙げられる．確率分布を用いてメロディーを生成する方法で制約を実現するには，生成された音に対して条件判定をする必要がある．しかし，数理計画法を用いることにより，制約を記述しやすく，また実現が容易になっている．また図 2 から分かるように，内点法によって導出される暫定解は, 過去の文脈を保持している．すなわち，ある反復回数における和音列は，それまでの反復回数における和音列の影響を受けている．これは，実行可能領域の内部を通り最適解へ近づく内点法のアルゴリズム[7]に起因する．このことから，生成されるメロディーにある程度の自己相関が生じ，全く無相関で滅茶苦茶なメロ

*1 https://soundcloud.com/muraka-2/var_diatonic
*2 https://soundcloud.com/muraka-2/var_01ip_16
*3 https://soundcloud.com/muraka-2/var_01ip_15
*4 https://soundcloud.com/muraka-2/var_01ip_14
*5 https://soundcloud.com/muraka-2/var_01ip_13

ディーが生成されにくいというメリットもある．

本節で提案した手法は，メロディー生成の枠組みの 1 つである．制約条件を変えたり，操作(1)で選ぶ 4 つの和音列の選び方を変えたりすることで，また違うメロディーを生成することが可能であると思われる．

参考文献

[1] GLPK (GNU Linear Programming Kit) https://www.gnu.org/software/glpk/ (アクセス:2017-1-16)
[2] 株式会社 NTT データ数理システム, Numerical Optimizer 数理計画用語集 数理計画法 http://www.msi.co.jp/nuopt/glossary/term_095d42c3c6149d3c695df7fb95946600ab7152a5.html (アクセス:2017-1-14)
[3] マーチン・ガードナー『別冊日経サイエンス マーチン・ガードナーの数学ゲーム I 』 一松信 訳, 日経サイエンス社, 2015
[4] MathWorks® https://jp.mathworks.com (アクセス:2017-1-2)
[5] MathWorks® Optimization Toolbox https://jp.mathworks.com/products/optimization.html (アクセス:2017-1-2)
[6] 大村英史，柴山拓郎，高橋達二，澁谷智志，大原育夫「音の高さと音の長さの相対的な物理的関係性と情報理論に基づいた音楽生成モデルの提案」，情報処理学会研究報告, Vol.2015-MUS-109, No.8, 2015
[7] OR 辞典 Wiki, 内点法 http://www.orsj.or.jp/~wiki/wiki/index.php/内点法 (アクセス:2017-2-1)