Mem. Fac. Sci. Eng. Shimane Univ.
Series A
35, pp. 99-106 (2001)

文字列照合アルゴリズムの楽譜への適用

家山　智史・濱田　賢作
島根大学大学院総合理工学研究科数理・情報システム学専攻

Similarity of Music Scores using String Pattern Matching Techniques

Satoshi IEYAMA and Kensaku HAMADA

This paper studies similarity between music scores using string pattern matching algorithms. The music score was converted into string of alphanumerical character. Of two strings (P and T), P was sequentially divided into fragment (P'). Matching between P' and T was calculated using naïve-pattern, Knuth-Moris-Pratt and Boyer-Moore algorithms, respectively. Matching score which presented a degree of the matching between strings P and T was defined in this paper. It was found that the similarity of music scores was detectable using the fragment which consisted of five characters.

1. はじめに

文字列照合アルゴリズムは，文書検索，文書編集など文書をコンピュータで扱う色々な場面で利用されている．また，バイオインフォマティクス（生命情報科学）の分野でDNAや蛋白質データベースから配列の検索，それらの相同性解析などに利用されており，適用範囲が非常に広い．音楽の世界では，楽譜により，音の高さ，長さを表現することでメロディとリズムを表現している．このことは，楽譜も1つの文字列の集合であるとみなすことができ，文字列照合アルゴリズムを適用することで，楽譜データベースからの検索，曲内及び曲間の類似性の検出などが可能となる．

本研究では，楽譜を文字列照合に適するようにフラグメントに分割し，新たに記号化し，単純な文字列照合アルゴリズム（単純法），Boyer-Mooreアルゴリズム（BM法）それにKnuth-Moris-Prattアルゴリズム（KMP法）を適用した楽譜の比較を行い，有効性について検討した．

2. 音符の記号化

楽譜では五線譜上に音符により各音の高さと長さの2つの要素を表現しており，2次元配列とみなすことができる．本研究に適用するアルゴリズムは1次元配列を対象としている．したがって，楽譜をそれらアルゴリズムに対応できる形式に変換する操作，すなわち記号化をすることが必要である．記号化は，Den3ro作成のフリーソフトであるABCオルガ

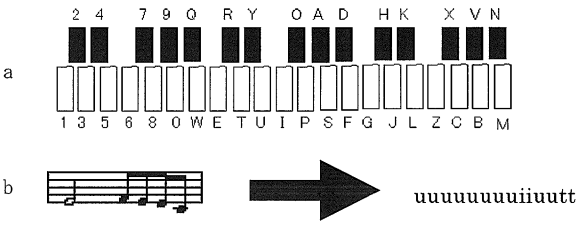

図 1-b ストリング化の例
例として一小節のストリング化を行う．文字の種類は音の高さを表し，文字の数は音の長さを表す．

表 1 採用した曲名とストリング数
成美堂出版の『フォーク＆フォーク』とドレミ楽譜出版社の『ハ調で弾くピアノ名曲 TV ドラマ＆CM ソング50選』から15曲選んで採用した．

曲番	曲名	文字数
A	今はもう誰も	576
B	バラが咲いた	512
C	「いちご白書」をもう一度	512
D	ある日突然	512
E	春雷	512
F	22才のわかれ	480
G	時には母のない子のように	432
H	ほっとけないよ	408
I	裸の王様	368
J	Peace Of My Wish	304
K	線香花火	258
L	フランシーヌの場合	256
M	ぼくたちの失敗	240
N	夏休み	192
O	受験生ブルース	128

ンにおける音符の記号化を参考にし，修正を加えて行った．記号化する音の高さの範囲は，多くの曲が３オクターブ以内で作曲されていたことから，３オクターブに限定した．そして，ドの音をE，レをTのように記号化し，半音（♯とb）も考慮して，図1-aに示すようにキーボードにアルファベットと数字からなる記号を対応させた．

音の長さの記号化において，全音符から16分音符までを記号化した．記号化の方式としては16分音符を１として考え，音符の長さに見合った文字の数で音の長さをあらわした．例えば，ドの４分音符はEEEEであり，レの８分音符はTTである．こう表すことにより簡潔な文字の集合が構成でき，照合を容易に行うことが出来るようになる．図1-bに楽譜の記号化の例を示す．

本研究で記号化した曲は，成美堂出版の『フォーク＆フォーク』とドレミ楽譜出版社の『ハ調で弾くピアノ名曲 TV ドラマ＆CM ソング50選』より抜粋した15曲であった．それぞれの曲名と曲番とストリング中の文字数を表１に示す．

3. 楽譜のマッチング

マッチングの方法：記号化された楽譜は音の高さと長さを1次元配列したものであり，知られている文字列照合アルゴリズムを単に適応して，比べられる曲（テキスト T）と比べる曲（パターン P）でマッチングを行うことはできない．本研究では音の高さと長さという

例
テキストT(m)：EEEETSUSSSTT・・・・・・EESTU
パターンP(n)：EETSUSSTT・・・・・・TSSTU

5文字でなるフラグメントの場合
P'_1：EETSU
P'_2：　ETSUS
P'_3：　　TSUSS
P'_4：　　　SUSST
P'_5：　　　　USSTT
　　・
　　・
　　・
P'_{n-f-1}：　　　　TSSTU

P'_1の場合
T(m)：EE[EETSU]SSSTT・・・・・・[EESTU]
P'_1：EETSU
2箇所で一致するので
$M_{P'_1}=2$

P'_2の場合
T(m)：EEE[ETSUS]SSTT・・・・・・EESTU
P'_2：ETSUS
1箇所で一致するので
$M_{P'_2}=1$
・
・
・
それぞれにおいてマッチングさせる

図2　パターンPのフラグメント法
　パターンPのフラグメントを取る際の例をあげる．5でフラグメントを取り，それぞれをTとマッチさせる．

表2 Pの作成における適切な文字の区切り方

表2-a フラグメントを5でとった場合のマッチ指数

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
A	100	0.9	3.8	16.0	0	1.0	5.5	0	0	0	0	1.1	0	1.3	0
B		100	0	0.5	0	0.5	0.3	0.2	0	0	0	0	0	0	0
C			100	0	0	0	0	0	0	0	0	0	0	0	0
D				100	0	0.1	7.5	0	0	0	0	0	0	0	0
E					100	18.7	0.3	0.1	20.9	2.1	4.3	4.9	0	1.4	0
F						100	0.1	1.5	31.5	0	0.3	7.5	0.1	0.9	0
G							100	0	0	0	0	1.0	0	1.6	0
H								100	0	0	0	0	0	0	0
I									100	0	0	16.3	0	2.2	0
J										100	0.5	0	0	0	0
K											100	0	0	0	0
L												100	0	4.4	0
M													100	0	0
N														100	0
O															100

表2-b フラグメントを10でとった場合のマッチ指数

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
A	100	14.0	6.5	27.6	0	5.0	25.0	1.2	0	0.1	0	5.4	0	5.4	0.1
B		100	6.5	58.7	1.2	6.4	31.1	7.1	0.5	1.1	0.7	4.1	0	5.8	1.6
C			100	0.2	0.1	0.2	0.2	0	0.3	0	0	0.1	0	0.2	0
D				100	0	1.0	33.8	2.1	0	1.2	0.1	2.9	0	1.0	1.2
E					100	55.0	2.1	6.8	37.9	20.6	25.6	10.3	1.3	11.6	0.4
F						100	3.7	8.4	31.1	12.0	5.9	9.7	1.2	10.4	1.2
G							100	2.0	0	0	1.0	10.1	0	6.9	0
H								100	0.8	17.3	4.8	0	0.6	0.5	2.0
I									100	1.5	6.0	22.5	0.2	24.4	0
J										100	25.4	2.4	1.3	2.6	2.1
K											100	11.3	4.0	13.5	0.4
L												100	0	50.4	0
M													100	0	0.9
N														100	0
O															100

二つの要素を1次元であらわしていることから，これらを包含する最適な長さ（文字数）からなるフラグメントにパターンPを分割して，文字列照合アルゴリズムを適用する方法でマッチングを行った．図2で示すように，パターンPをf文字からなるフラグメントに分割した新たなパターン$P'_1, P'_2, P'_3, \cdots, P'_{n-f-1}$を求めた．次に$P'_1, P'_2, P'_3, \cdots, P'_{n-f-1}$のそれぞれについて，$T$と単純なアルゴリズム，BM, KMPで照合させ，そして両者で一致した文字数$M_{P'_1}, M_{P'_2}, \cdots, M_{P'_{n-f-1}}$を記録した．

また，TをPとして，同様にして一致した文字数$M_{T'_1}, M_{T'_2}, M_{T'_3}, \cdots, M_{T'_{n-f-1}}$を求めた．$T$と$P$の楽譜の一致度を1式で表すマッチ指数で評価した．

表2-c フラグメントを16でとった場合のマッチ指数

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
A	100	34.7	13.5	54.3	2.8	6.0	40.7	6.5	0	0.8	2.1	13.0	0	13.0	2.7
B		100	53.1	91.5	15.0	17.4	38.3	14.1	21.9	10.8	15.9	16.8	0.1	22.6	13.8
C			100	2.3	2.2	2.3	2.2	0.6	3.1	0.6	1.4	1.3	0.02	2.1	0.4
D				100	2.2	4.5	51.0	6.7	0	4.2	4.0	10.0	0	7.7	8.1
E					100	83.7	6.0	21.9	50.3	35.6	41.5	10.7	20.7	19.7	7.0
F						100	5.0	15.4	33.4	36.5	23.8	8.4	19.1	14.4	8.4
G							100	5.9	0	0.3	3.2	26.3	1.1	16.8	2.0
H								100	43.3	82.5	59.3	15.9	32.0	21.1	21.0
I									100	10.2	28.8	19.5	3.1	34.1	0.1
J										100	46.4	6.5	41.1	14.6	17.3
K											100	28.9	28.7	53.3	14.4
L												100	0	88.2	2.2
M													100	3.4	14.0
N														100	2.0
O															100

単位は%で，列がPで，行がTである．

$$\text{マッチ指数}=\frac{\sum_{k=1}^{n-f-1} M_{P_k}}{\sum_{k=1}^{n-f-1} M_{T_k}}$$

実装：プログラムはGnuのC言語で開発し，CPUがSunUltraSPARC-IIi 400 MHzでメモリが128 Mbyte, OSはSolaris8の計算機で以下の研究を行った．

最適フラグメント：計算量は，フラグメントの数（$n-f-1$）だけマッチングを繰り返すために，KMP法では$\mathbf{O}(n)*(n-f-1)$，そしてBM法では$\mathbf{O}(n)*(n-f-1)$となる．このことからfの値が大きければ大きいほど計算効率がいいということは明確である．しかしながら，音の高さと長さを1次元配列にしたことから，フラグメントの長さによる影響が2つの楽譜の一致度に大きく影響する．フラグメントの文字数を16と10，そして5としたときのマッチ指数を求めた．表2に示すように120通りの組み合わせで，フラグメントの文字数が16のときマッチ指数が0%のものが71通りあり，最もよく一致したのはF（22才の別れ）とI（裸の王様）の31.5%であった．フラグメントの文字数が10のときは23通りで，B（バラが咲いた）とD（ある日突然）の58.6%であった．フラグメントの文字数が5のときは4通りで，最高はB（バラが咲いた）とD（ある日突然）の91.5%あった．表2にはKMP法についてのみを示したが，この傾向は他の方法についても同じであった．フラグメントの文字数が少ないと一致自体は良くなるが，音の高さと長さを表した1次元のストリングにおいて，本当に楽譜が一致しているかどうかを判別することが困難となるので，フレーズを聞き取れる程度のフラグメントの文字数として5が最適であるとした．

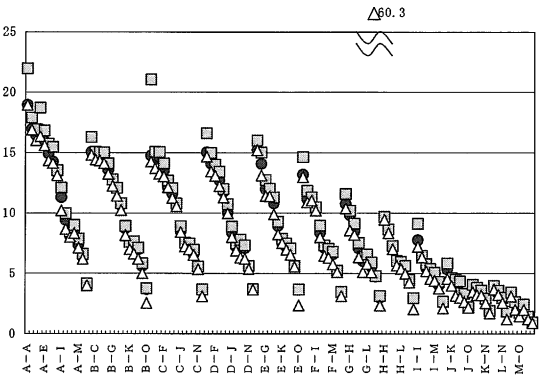

図3 マッチングアルゴリズムの速度比較
縦軸を時間（S）でとり，横軸を組み合わせで取る．三角がBMで，丸がKMP，四角が単純なアルゴリズムでマッチングを行った際の時間を示している．

4. マッチングアルゴリズムの速度比較

単純なアルゴリズムとKMP法それにBM法のそれぞれについて，マッチングに要する速度の検討を次のようにして行った．AからOの曲はストリング数の多い順に並べられており，元となる曲が比べる曲よりも大きくなる場合のみマッチングさせた．マッチングに要した時間は，マッチングを200回連続して繰り返し，関数timeで計測した．結果を図3に示す．

BMは相対的に実行速度が速かった．KMPも少しではあるが単純なアルゴリズムよりも速い時間でマッチを終わらせていた．全体的にストリング数が増えると実行速度も遅くなっていた．また，G（時には母のない子のように）とN（夏休み）のマッチングの際にBMの時間が他の二つよりも莫大にかかったのは，そのストリングがBMに適したストリングでなかったことが考えられる．

5. 楽譜の類似性

実際のマッチ指数の結果は図4に示す．

B（バラが咲いた）とD（ある日突然）が91.5%で最もマッチし，L（フランシーヌの場合）とN（夏休み），H（ほっとけないよ）とJ (Piece Of My Wish) も80%を超えるマッチ指数を示した．これらの曲のマッチ指数が高い原因として，繰り返しのメロディが多いということと，全音符や2分音符などの音の長さが長い音符が2つの曲で多く一致していることが考えられる．また，マッチ指数が0の曲がいくつか出ているのは低音の曲と高音の曲を比べたためであると考えられる．また，I曲の『裸の王様』は同じメロディの繰り返しが多く，その音符が他の曲で現れないためか，結果がA（今はもうだれも）とD（ある日突

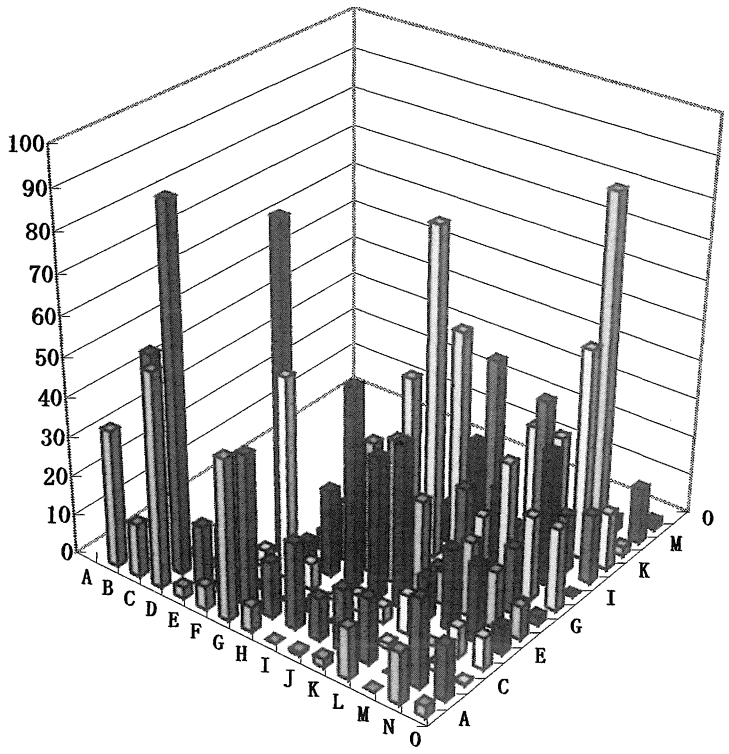

図4　15曲の組み合わせによるマッチ指数
　表2-aをグラフにしたもので，縦軸をマッチ指数，横軸を組み合わせによってとっている．単位は%である．

然）とG（時には母のない子のように）において0となっている．フォークであるA, B, C, D, E, F, G, K, L, N, OとTVテーマ&CMであるH, I, J, Mを比較すると全体的にマッチ指数が低くなるという興味深い結果が得られた．

ま　と　め

フラグメントを5で取ることによって，フレーズをあまり崩すことなく照合を行うことが出来た．マッチ指数を用いることによって効果的に類似した曲を見つけることができた．また，フォークというジャンルの曲に共通するパターンが存在することがわかった．実際には同じようなフレーズでも音の高さの違う曲では照合を行うことが出来ないことがわかった．今後は高低さの差分で一致を探すなど，アルゴリズム自体の見直しも必要である．

参 考 文 献

[1] 成美堂出版『フォーク＆フォーク』堀野羽津子編 1995. 4. 20発行
[2] 株式会社ドレミ楽譜出版社『ハ調で弾くピアノ名曲 TVドラマ＆CMソング50選』日名子紀代/丹羽あさ子編著 1994. 3. 30発行
[3] BOYER R. S., MOORE J. S., 1977, A fast string searching algorithm. *Communications of the ACM*. 20: 762–772.
[4] KNUTH D. E., MORRIS (Jr) J. H., PRATT V. R., 1977, Fast pattern matching in strings, *SIAM Journal on Computing* 6(1): 323–350.