TAB譜と左手運指の自動生成アルゴリズムの開発

横道政裕[1)] ・ 岩野公保[2)]

Development of automatic generation algorithm of Tablature and left hand finger work

Masahiro Yokomichi, Kimiyasu Iwano

Abstract

When one pe rforms music with guitar, there exist a lot of finger positions. Therefore, it is difficult f or novice pla yers to discove r the best fi nger positi on. It is needed to relax th is difficulty by developing an algorithm which ge nerates the best finger position automatical ly. Previously, several approaches have been proposed, but they have shortcomings about the class of the applicable melodies or the computational costs.

In this paper, the authors propose a hy brid optimization based algorithm. This algorithm can be applied for wider class of melodies and th e co mputational cost is reduc ed by selectively using the normal s earch and GA. The ef fectiveness of the approach is exa mined by experiments.

Key Words :
guitar, tablature, fingering, tablature notation, finger position, genetic algorithm

1. 序論

ギター等の弦楽器は一つの音程に対して複数の押弦位置が存在する(図１)。それに加えて,左手のどの指で押さえるかという点も考慮すると,１つの楽曲における組み合わせの数は「{(音程の押弦位置の数)×(左手の指)}の(曲の音符数)乗」という膨大なものとなってしまう。そのため,初心者が五線譜から適切な押弦方法を導き出すのは困難と言える。よって,適切な押弦方法を導出するアルゴリズムが求められている。図 1 でミ音に対応する押弦位置を示している。この図における縦メモリをフレット,横メモリを弦という。

TAB 譜とはギター演奏において押弦位置が記述されている楽譜のことであり,TAB 譜と左手の運指を自動で導出するアルゴリズムを開発することにより初心者の楽曲演奏の際の負担を軽減させることができる。

過去に TAB 譜と左手運指を自動生成する研究として動的計画法(以下 DP)を用いたもの[1]と,遺伝的アルゴリズム(以下 GA)を用いたもの[2]がある。
これらの過去の研究を参考にし,本研究では

① 和音や特殊な押弦方法(セーハ等)に対応させる
② 実行不可能解を導出させない
③ 処理時間を短縮しつつ良い解を導出させる
④ 実際に演奏しやすい TAB 譜と運指を作る

以上の４つを目標にしていく。

なお,今回適用するギターのモデルは,六弦ギター

1) 情報システム工学科准教授
2) 情報システム工学科学部生(2009 年修士課程入学)

のスタンダードチューニングのものとする。

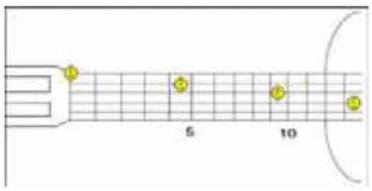

図1 ギターの押弦位置

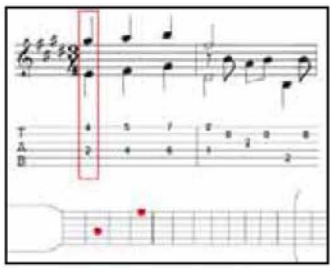

図2 TAB譜例

2. 最適な押弦位置・左手運指とは

ギターを演奏する際に評価すべき点として次のようなことがあげられる

- 左手のポジションと指の移動量(LM)

  左手のポジションとは左手の人差し指を基準に決定させ,そのポジションが何フレット移動したかをポジションの移動量とする。また,左手の指の移動量にはマンハッタン距離を用いる。ポジションと指の移動量が小さいほど評価を高くする。

- フレットの高低(FH)

ギターの構造上,低フレットであるほど引きいためフレットが低いほど評価を高くする,フレットが低いほど評価を高くする。

これらをもちいて次の式

$$Fitness[i] = LM + FH$$

のFitnessを評価量とする。

後述するが本研究では探索をフレーズ内と全体という2段階において探索を行うため,フレーズ内探索のFitnessの総和をFitPhrase,全体探索のものをFitAllとし最終的な評価は

$$FitnessF = FitPhrase + FitAll$$

によって行う。

3. 自動生成アルゴリズム

3.1 五線譜からのデータ取得

五線譜からデータを取得する際,まずその楽曲における最短の音符を1単位時間とする。そして単位時間ごとに,

- 単位時間内の音符数 N[t]
- 音程 Tone[t][n]
- 音長 Length[t][n] (n=N[t])

のデータを取得する。さらに,

- フレーズ数
- 各フレーズの開始時間

を取得する。この2つのデータと音長の利用法は後述する。

なお,各音程に対応する押弦位置はあらかじめ与えておく。

3.2 システムの流れ

全体のシステムの流れとして,まず五線譜をフレーズに分割しデータを取得する。そして各フレーズ内において探索を行い,評価が高いものをそのフレーズのエリートとして保存する。そして,そのエリートの組み合わせを探索し,最も評価が高いものを最適なTAB譜と左手運指として導出する。各段階(フレーズ内と全体)において探索を行う際に,要素数の量によって探索法を変える。フレーズ内探索においては,フレーズ内の音符の数が基準値未満の場合は全探索を行い,基準値以上の場合にはGAを用いた探索を行う。またフレーズ内探索を行う際に2段階において実行不可能解の除去を行う。全体探索においても,フレーズ数が基準値未満だった場合は全探索を行い,基準値以上だった場合はGAを用いた探索を行う。本研究では,フレーズ内探索では基準値を8,全体探索においては基準値を4としたが,より良い基準値を決定することも今後の課題として考えられる。

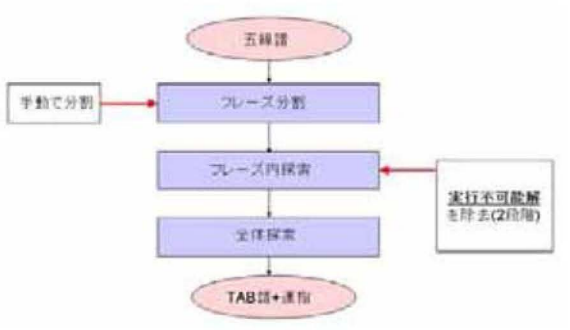

図 3 全体のシステムの流れ

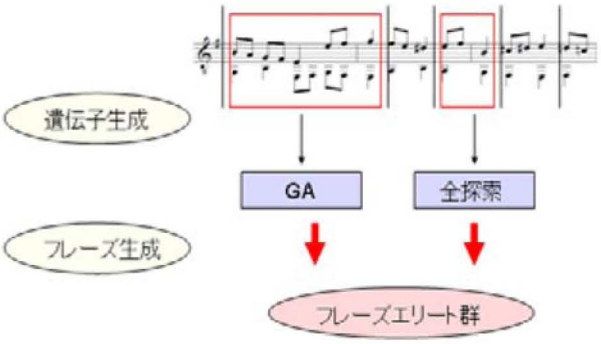

図 4 フレーズ分割およびフレーズ内探索

３．３　探索方法

本研究では探索方法として GA と全探索を併用する。GA と全探索を比較した時,GA は全探索に比べ処理時間は短くなるが近似値を求めるという性質上,全探索よりも探索精度は落ちてしまう。また全探索は探索精度が高いが,処理時間が長くなってしまう。そこで,探索すべき範囲が広いときには GA を用い,狭い時には全探索を用いることにより処理速度と探索精度の向上を両立させることができると考え,この探索方法を使用した。

３．４　フレーズ分割とフレーズ内探索

フレーズ分割をし,フレーズ内探索を行う理由として,まずフレーズ内の音の繋がりは重要であることが挙げられる。そのため全体で平均的に繋がりをよくすることよりも,フレーズ内での繋がりを重視するためにフレーズ分割を行う。また,各フレーズで解を一つしか導出しなかった場合,全体での組み合わせが一通りに決定してしまい,それがとんでもない TAB 譜となってしまう可能性がある。そこで,各フレーズの最初と最後の音に関する全ての押弦位置と左手の指の組み合わせについてそれぞれ探索を行い,評価が高いものをそのフレーズのエリート群として保存していく。

フレーズの分割方法については,本研究では音符の繋がりが薄い箇所(８分音符が続く時に出てくる４部音符の箇所など)を手動で分割していった。

３．４．１　GA について

本研究で用いる GA について説明する。まず,フレーズ内探索を行う際は単位時間毎に音符に対する押さえ方,押弦位置の全ての組み合わせを生成させ,ランダムに組み合わせたものを親個体として保存する。さらにここで実行不可能解の除去を行う。この実行不可能解を除いて,個体数を満たすまでこの生成を繰り返す。生成した個体を評価関数にかけ,評価値の高い順にソートにかけランキングを行う。

次に親個体からランダムで２つの個体を選択し,交叉を行う。交叉によって生み出された子個体が親個体と重複していないかを検査し,重複していない場合は評価関数にかけ親個体群と比較を行う。ここで,あるランクの親個体より評価が高ければそこに割り込ませる。

これらの操作を指定した世代数分繰り返し行い,最終的に最も評価が高いものを解として導出する。

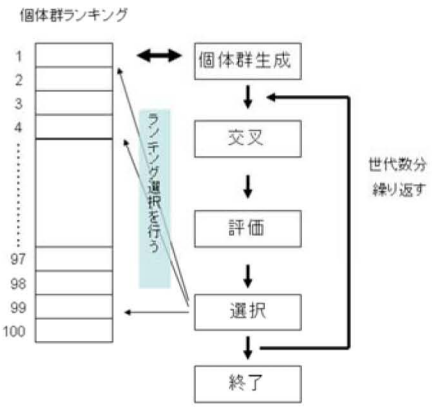

図 5　GA の流れ

３．４．２ GAのパラメータ

本研究で使用したGAのパラメータを以下に示す。

世代数 100

個体数 1000

交叉点の数 2

突然変異確率 0

この値は実験を何度か繰り返しある程度良い解を示し短い時間で終わるように調節をおこなったものである。

３．５ 実行不可能解

実際に演奏不可能なパターンを実行不可能解として除去する。実行不可能解の例として，まず図６のように、フレット間が離れすぎて押さえきれない場合。図７のように，左手の構造上の問題で演奏できない，または演奏が非常にしにくい場合。図８のパターンでは単位時間は八分音符で区切ってあり，時刻 t の音符は四分音符であるため時刻 t+1 まで音を保持しなければならない。しかし，次の時刻 t+1 において同一弦を別の音符で使用すると時刻 t の音が消されてしまうためこの場合も実行不可能解とする。

この実行不可能解を制限するために，五線譜から音長のデータを取得する。

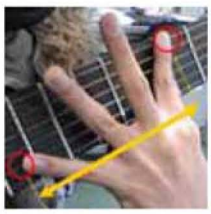

図６ 実行不可能例１

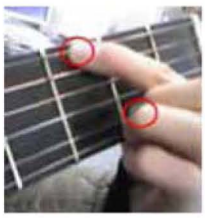

図７ 実行不可能例２

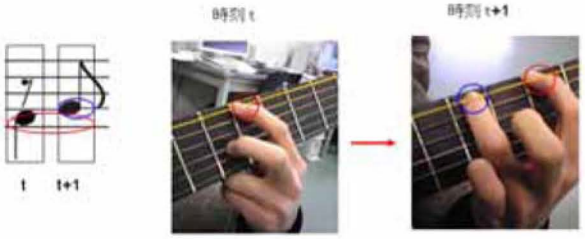

図８ 実行不可能例

４． 実験

４．１ 実験方法

本研究では次の２つの実験を行った。

I. 処理時間及び処理結果の比較

II. 演奏しやすさ

実験Ｉについては目的③を達成できたかどうかを判断するために，本研究のアルゴリズムにおいて探索方法を，

- 全探索
- GA
- 本システム

で行った場合の処理時間を比較する。なお実験は以下の環境で行った。

OS：Microsoft Windows XP

CPU：Intel Pentium Dual CPU E2180 @ 2.00GHz

メモリ：1.99 GB

実験 II について目標④を達成できたかどうかを判断するために，

- 本研究のアルゴリズムで生成された TAB 譜と運指
- 一般的に演奏されている TAB 譜と運指

以上の二つを演奏しやすさについて比較する。その際、ギター初心者～中級者に実際に演奏してもらい，評価はそのTAB譜と運指での弾きやすさを10点満点で得点付けしてもらう。

４．２ 実験結果Ｉ

それぞれの探索による処理時間を図９に，フレーズ数が５の時のそれぞれの探索の譜面を図 10 に示す。グラフを見ていくと全探索はフレーズ数が 3 の時は実行速度が最も早いが，フレーズ数が 4 になると極端に遅くなり、グラフには描画されていないが，フレーズが 5 の時は約８０秒，６の時には 10 分以上と比較にならないほどの処理時間となってしまっている。比べて GA はフレーズ数が３の時は，全探索より処理時間が遅くなっている。しかし、フレーズ数が増加していっても，それほど全探索ほどの処理時間増加は見られない。

GA&全探索においては,フレーズ数が基準値である4までは全探索とほぼ同じ処理時間だが,基準値をこえるとGAの処理時間に近づき,全探索に比べ大幅に時間短縮できていることが分かる。
また図9を見た時,GAに比べ全探索の譜面に近いものとなっているため,ある程度の探索精度も確保されていると言える。
これらにより,目標③は達成できたと言える。

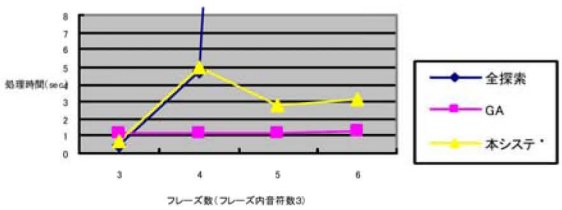

図9 処理時間

図10 全探索 GA 本システム

4．3 実験結果 II

図10及び図11に演奏してもらったTAB譜と運指を、表1にそれぞれの楽譜の評価を示す。
図10,図11を見た際、和音やセーハが出てきていることから目標①が,実際に演奏不可能な押さえ方は導出されていないことから目標②が達成されたと言える。
しかし,表1をみるとどの演奏者も一般的なTAB譜と運指を演奏しやすいという結果になっており,また自身も本システムにより作成されたTAB譜と運指が演奏しやすいと思えないため,目標④は達成させることができなかったといえる。

	初級者	初級者	中級者	中級者
一般	8	7	9	8
GA+全探	7	5	3	4

表1 採点表

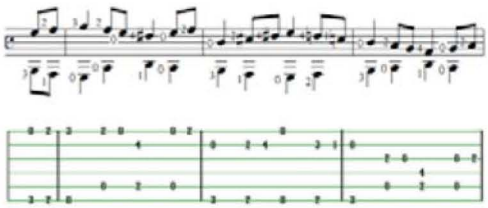

図10 一般的なTAB譜

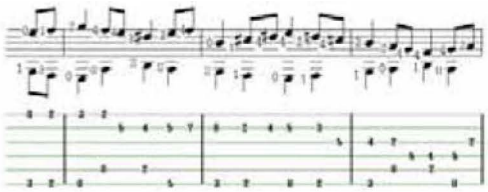

図11 本研究のシステムによるTAB譜

5． 結論

今回の研究では,目標①～③は達成させることができたが,目標④を達成させることができなかった。実験Iを見ていった時,全探索にかなり近い結果を出していることを考慮すると,評価関数の設定が原因であると推察される。
今後の課題として,

- 評価関数改善による目標④の達成
- GAのパラメータ、探索の基準値の改善による目標③のより良い地点での達成。
- 楽譜のフレーズ分割の自動化

があげられる。

参考文献

[1] 林田 巧,伊藤 雅 “単旋律におけるギター押弦運指の最適化” 電気学会論文誌 2004
[2] Daniel R. Tuohy, Walter D. Potter “CREATING TABLATURE AND ARRANGING MUSIC FOR GUITAR WITH GENETIC ALGORITHMS AND ARTIFICIAL NEURAL NETWORKS” B.S., The University of Georgia 2004
[3][譜面データ] Johann Sebastian Bach (1685-1750) Bourrée, en mi mineur BWV 996 http://www.delcamp.net/ja/framset/framset_baroque_ja.html