A Tabu Search Algorithm to Generate Piano Fingerings for Polyphonic Sheet Music

Matteo Balliauw, Dorien Herremans,
Daniel Palhazi Cuervo, and Kenneth Sörensen

University of Antwerp, Faculty of Applied Economics,
Prinsstraat 13, 2000 Antwerp, Belgium
{matteo.balliauw,dorien.herremans,daniel.palhazicuervo,
kenneth.sorensen}@uantwerpen.be
http://www.uantwerpen.be

Abstract. A piano fingering is an indication of which finger is to be used to play each note. Good piano fingerings enable pianists to study, remember and play pieces fluently. In this paper, we propose an algorithm to find a good piano fingering automatically. The tabu search algorithm is a metaheuristic that can find a good solution in a short amount of execution time. The algorithm implements an objective function that takes into account the characteristics of the pianist's hand in complex polyphonic music.

Keywords: Piano Fingering, Tabu Search, Metaheuristics, OR in Music, Combinatorial Optimisation

1 Introduction

Piano fingerings are often added by players to sheet music in order to indicate the appropriate finger that should be used to play each note. For a piano player, it is important to have good *piano fingerings* at hand as not every finger is equally suited to play each note. Some combinations of fingers are better suited to play certain note sequences than others [1]. In addition, having a clear piano fingering can help a pianist study and remember a piece. A well thought out piano fingering can also enhance the interpretation and the musicality of a performance [2].

The purpose of this research is to help pianists with little experience in deciding on a good fingering and pianists who quickly want to obtain an easy fingering. It requires a considerable amount of time and expertise to elaborate a suitable fingering for a piano piece. In this paper, we look at the generation of a good piano fingering as a combinatorial optimisation problem. In this problem, a finger has to be assigned to every note in the piece. The quality of a fingering is evaluated by means of an objective function that measures playability. This objective function also takes into account the characteristics of the player's hands. We develop an efficient optimisation algorithm that is able to find a good fingering solution for a complex polyphonic piano piece in a reasonable amount of execution time. To this end, we developed a tabu search (TS), as it has already been applied efficiently to many other combinatorial optimisation problems such as the travelling salesman problem [3] and the vehicle routing problem [4]. More recently, it has been also used in the field of music for computer aided composing [5].

This paper is divided into six sections. Section 2 gives a short overview of the literature concerning the generation of piano fingerings and algorithms developed to solve this problem. In Section 3, a mathematical formulation of the problem is introduced. Section 4 explains in detail the tabu search algorithm and its implementation. In Section 5, we show and discuss an example of a fingering generated by the TS for an existing piece of music. The final section gives some conclusions and advice for future research.

2 Literature Review

In the 18th century exercises to learn frequently used fingering combinations fluently were published for the first time. Carl Philipp Emanuel Bach has been amongst the first musicians to write down an entire set of rules for piano fingering. Previously, this was only transmitted orally through lessons. In the 19th century, many composors followed his example [6]. At the end of the 20th century, the first attempts to generate a mathematical formulation that can model a piano fingering and develop a suitable algorithm were published. Similar research has been carried out amongst others for string instruments [7].

2.1 Fingering Quality

In order to evaluate a piano fingering, it is necessary to quantify how easy and fluently it can be played. Three important dimensions of a piano fingering add to its quality. The main element is the ease of playing, which is considered in this research. Additional dimensions that add to the quality of a fingering are the ease of memorisation and the facilitation of the interpretation [8].

The main approach for evaluating the quality of a piano fingering is using an objective fucntion based on hand-made rules [8][9][10]. This approach is also followed in this research. Another possibility is using a machine learning approach such as Markov Models based on transition matrices [2][11][12][13] or Hidden Markov Models (HMM) [14] based on probabilities.

In order to evaluate the playability of a fingering the set of rules proposed by Parncutt et al. [8] and Parncutt [9] serve as the basis for this research. Every source of difficulty in a fingering, both monophonic and polyphonic, is assigned a cost. Every cost factor is attributed a weight. The objective function consists of the weighted sum of costs and is minimised. This approach has the advantage that pianists can express their trade-offs between the sources of difficulty. This can be done by assigning different weights to each source of difficulty in the objective function. Parncutt's original cost factors for piano fingering have been expanded by Jacobs [15] and Lin and Liu [16]. More recently, some improvements and additions were made by the authors [17]. It is that set that is used in this research. Issues such as personal preferences, the use of the left hand and the difference between playing a black and white key have been ignored previously as indicated by other authors [18], but are taken into account in this paper.

2.2 Algorithms Used in Literature

In many applications, dynamic programming is used to find optimal piano fingerings according to the selected objective function. Dijkstra's algorithm has been used by Parncutt et. al. [8] on monophonic music and by Al Kasimi et. al. [13] on short monophonic pieces with some simple polyphonic chords. Similar dynamic programming algorithms have been used in literature by Robine [2] and Hart et. al. [11] on monophonic music. A rule based expert system that optimises a fitness function for monophonic music was developed by Viana et. al. [10] and for the HMM-model in monophonic music, Yonebayashi et. al. used a Viterbi Algorithm [14]. Although some improvements have been made to reduce the computing time of these algorithms, it is often impossible to deal with complex polyphonic pieces (where simultaneous notes can have different lengths). For this reason, we develop a tabu search metaheuristic algorithm that can deal with complex polyphony in an effective and efficient way. This is explained in Section 4.

3 Problem Description

When generating a piano fingering, it is necessary to decide which finger should play each note. Each finger is represented using the traditional coding from 1 (thumb) to 5 (little finger).

It is also necessary to define an objective function that measures the quality of a solution, by looking at the playability of the piece. This objective function was described by Balliauw [17] as an adaptation of the work of Parncutt et al. [8] and Jacobs [15].

The objective function takes into account a distance matrix, displayed in Table 1. This contains information for each finger pair on the distances that are easy and difficult to play, respectively called `Rel` (relaxed range), `Comf` (comfortable range) and `Prac` (practically playable range). These allowed distances can be adapted by the user of the algorithm according to the biomechanics of his hand and are calculated by subtracting the corresponding values of the keys in Fig. 1.

The objective function consists out of three sets of rules Using this distance matrix, a first set of rules can compare the actual distance and the allowed distance between two simultaneous or consecutive notes for the proposed pair of fingers to play these two notes. A penalty score is applied when the actual distance is larger than the different types of allowed distances.

As Jacobs [15] argued, the distances in the previous research of Parncutt et. al. [8] were not accurate. To calculate these penalties more accurately, Balliauw [17] increased the distances between E and F and between B and C to two half notes, to equal the distances between other adjacent white keys on a piano keyboard.

The second set of rules is implemented to prevent unnecessary and unhandy hand changes. The third group of rules prevents difficult finger movements in monophonic music. In this third set, two additional rules proposed by Balliauw promote the choice of logical, commonly used finger patterns.

The weighted penalty scores of these rules, that are included in Table 2, are summed to obtain the objective function value. As this value indicates the difficulty of the fingering, it has to be minimised.

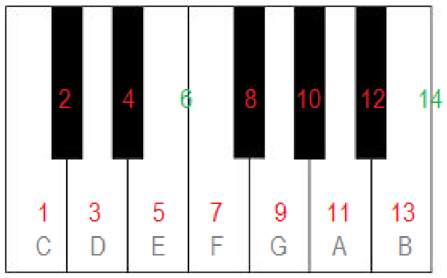


Fig. 1. Piano keyboard with additional imaginary black keys.

Table 1. Example distance matrix that describes the pianist's right hand.

Finger pair	`MinPrac`	`MinComf`	`MinRel`	`MaxRel`	`MaxComf`	`MaxPrac`
1-2	-10	-8	1	6	9	11
1-3	-8	-6	3	9	13	15
1-4	-6	-4	5	11	14	16
1-5	-2	0	7	12	16	18
2-3	1	1	1	2	5	7
2-4	1	1	3	4	6	8
2-5	2	2	5	6	10	12
3-4	1	1	1	2	2	4
3-5	1	1	3	4	6	8
4-5	1	1	1	2	4	6

The problem also takes into account one hard constraint. This constraint here is that one finger can not be used to play two different, simultaneous notes, as this is no feasible to execute on a real piano.

Table 2. Set of rules composing the objective function [17].

Rule	Application	Description	Score
1	All	For every unit the distance between two consecutive notes is below `MinComf` or exceeds `MaxComf`.	+2
2	All	For every unit the distance between two consecutive notes is below `MinRel` or exceeds `MaxRel`.	+1
3	Monophonic	If the distance between a first and third note is below `MinComf` or exceeds `MaxComf`: add one point. In addition, if the pitch of the second note is the middle one, is played by the thumb and the distance between the first and third note is below `MinPrac` or exceeds `MaxPrac`: add another point. Finally, if the first and third note have the same pitch, but are played by a different finger: add another point.	+1 +1 +1
4	Monophonic	For every unit the distance between a first and third note is below `MinComf` or exceeds `MaxComf`.	+1
5	Monophonic	For every use of the fourth finger.	+1
6	Monophonic	For the use of the third and the fourth finger (in any consecutive order).	+1
7	Monophonic	For the use of the third finger on a white key and the fourth finger on a black key (in any consecutive order).	+1
8	Monophonic	When the thumb plays a black key: add a half point. Add one more point for a different finger used on a white key just before and one extra for one just after the thumb.	+0.5 +1 +1
9	Monophonic	When the fifth finger plays a black key: add zero points. Add one more point for a different finger used on a white key just before and one extra for one just after the fifth finger.	+1 +1
10	Monophonic	For a thumb crossing on the same level (white-white or black-black).	+1
11	Monophonic	For a thumb on a black key crossed by a different finger on a white key.	+2
12	Monophonic	For a different first and third note, played by the same finger, and the second pitch being the middle one.	+1
13	All	For every unit the distance between two following notes is below `MinPrac` or exceeds `MaxPrac`.	+10
14	Polyphonic	Apply rules 1, 2 (both with doubled scores) and 13 within one chord.	
15	All	For consecutive slices containing exactly the same notes (with identical pitches), played by a different finger, for each different finger.	+1

4 Tabu Search Algorithm

A common approach to solve combinatorial optimisation problems are exact algorithms, that ensure that the optimal solution is found. The problem is that the execution time can grow exponentially with the size of the instance treated. For this reason, in the worst case they are often suited to deal with short or simple instances only. Metaheuristics form an alternative approach to deal with complex polyphonic music (large and complex instances) in a reasonable amount of calculation time, without giving up too much of the optimality. Metaheuristics offer guidelines to build algorithms that make use of rules of thumb, so called heuristics [19].

Different categories of metaheuristics exist. One such category are Local Search (LS) metaheuristics. LS starts from a *current solution*, to which a small, incremental change is made, called a *move*. All moves performing the same changes are part of the same *move type*. The set of solutions that can be reached from the current solution by a certain move type is called the *neighbourhood* of the solution [19]. A *local optimum* is reached when the neighbourhood contains no improvement for the current solution.

One can amongst others escape from such a local optimum or avoid getting trapped into circles by using a *tabu list*. This list contains a number (the actual value is called *tabu tenure*) of moves that were performed right before the current move and that are excluded from the neighbourhood of the current solution. These moves are *tabu active*. The move with the best objective function value from this neighbourhood is performed, even if it worsens the current solution. In this way, the algorithm avoids getting trapped into circles and can explore the solution space around the local optimum to eventually arrive at a better optimum. The tabu tenure and the amount of allowed iterations without improvement are the two parameters determining such a *tabu search* [20].

When no more improvements can be made to a local optimum within one neighbourhood, the algorithm can move to another neighbourhood, defined by a different move type which can contain a better optimum. This strategy is called *Variable Neighbourhood Search (VNS)* [21].

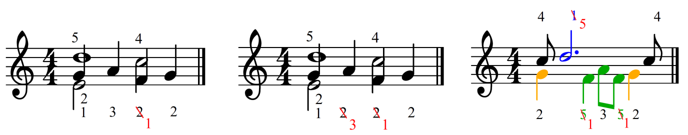
The TS algorithm, displayed schematically in Algorithm 1, starts from a random initial solution. The optimisation processes for both hands are identical. First, a preprocessing step Swap is applied to the initial solution. This step makes significant improvements in a very short amount of time, by swapping a finger in the entire piece. This preprocessing step has a positive impact on the solution quality and reduces the required execution time.

In the algorithm, there are three neighbourhood operators $o \in \mathcal{O}$, each defined by a move type. Each time, the best solution from the neighbourhood is selected with a steepest descent strategy. As a result, the move that leads to the best fingering is chosen in each iteration. A first neighbourhood, called Change1, is defined by a move type that changes the finger of a note to any other possible finger. The move type Change2 is similar to Change1, whereby it is expanded to two adjacent or simultaneous notes. Sometimes, changing two adjacent notes together might improve the solution, whereas this move cannot be identified by

Algorithm 1: Tabu Search Algorithm for each Hand

```
Input  : File $\mathcal{F}$
Output: File $\mathcal{F}'$
1  $\mathcal{P} \leftarrow$ Parse($\mathcal{F}$)
2  $\mathcal{S} \leftarrow$ Rand_Sol($\mathcal{P}$)
3  $\mathcal{S} \leftarrow$ Best_Sol(Swap)
4  $\mathcal{S}' \leftarrow \mathcal{S}$
5  Init_Tabu_List(tabutenure)
6  $\mathcal{T} \leftarrow$ True
7  while $\mathcal{T}$ = True do
8      for $o \in \mathcal{O}$ do
9          for $i <$ maxiters do
10             $\mathcal{N} \leftarrow \mathrm{Nbh}_o(\mathcal{S})$
11             Exclude from $\mathcal{N}$: $t_{ij} \in$ Tabu_List
12             $\mathcal{S}'' \leftarrow$ Best_Sol($\mathcal{N}$)
13             Update_Tabu_List
14             if $f(\mathcal{S}'') < f(\mathcal{S}')$ then
15                 $\mathcal{S}' \leftarrow \mathcal{S}''$
16                 $i = 0$
17             else
18                 $i++$
19         if $f(\mathcal{S}') < f(\mathcal{S})$ then
20             $\mathcal{S} \leftarrow \mathcal{S}'$
21             $\mathcal{T} \leftarrow$ True
22         else
23             $\mathcal{T} \leftarrow$ False
24         Clear_Tabu_List()
25 $\mathcal{P} \leftarrow \mathcal{S}$
26 $\mathcal{F}' \leftarrow$ Unparse($\mathcal{P}$)
```

Change1. This is because changing one note separately might have a detrimental effect on the fingering, and therefore might be discarded. To increase the interchangeability of two fingers in polyphonic music, moves of SwapPart will change the fingering of a note a from k to l, where the fingering from all notes b, played with finger l, starting before the end of note a and ending after the start of note a, are changed from l to k. An example of each move type can be found in Figure 2.



(a) Change1 (b) Change2 (c) SwapPart

Fig. 2. Examples of each move explored in every neighbourhood considered by the Tabu Search algorithm.

To perform Tabu Search in a neighbourhood $o \in \mathcal{O}$, a *tabu list* is initialized after the preprocessing step. The *tabu tenure* is defined as a percentage of the number of notes in that hand. The tabu list considers as a forbidden move changing or swapping the fingering of a specific note i to fingering j, if the couple t_{ij} is *tabu active*. A couple t_{ij} becomes tabu active after the fingering of note i has been moved to finger j and remains active for a number of moves, equal to the tabu tenure. As a result, it is possible that moving to the best neighbour reduces the quality of the current solution. The number of allowed iterations without improvement is defined as a parameter, `maxiters`. In this way, the algorithm can explore the solution space around the current solution and escape from a local optimum, given that enough iterations without improvement are allowed (here defined as a percentage of the tabu tenure). A different optimisation path can hence be pursued to arrive in a different area of the solution space and escape the local optimum. An example is shown in Fig. 3.

When the number of iterations in a neighbourhood reaches `maxiters` without improving the solution in one neighbourhood, the content of the tabu list is cleared and in order to escape from the local optimum, the algorithm starts to perform TS in a subsequent neighbourhood o. When a loop through all neighbourhoods $o \in \mathcal{O}$ improved the solution $\mathcal{S}$, all neighbourhoods o are explored again. Otherwise, the algorithm has reached its final solution and it is outputted to a MusicXML file. This output can be processed by open source music sheet software, like MuseScore[1].

[1] Available from `musescore.org`

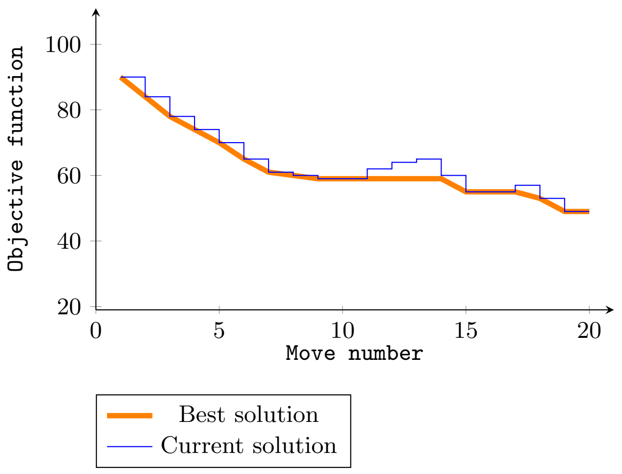


Fig. 3. Example of the execution of the TS algorithm.

5 Results

Fig. 4 shows an example output of the described TS for the first variation on the Saraband from G. F. Händels Suite in D minor (HWV 437). The algorithm was run with all neighbourhoods included, the tabu tenure set as 50 % of the number of notes in the hand and the allowed iterations without improvement set as 5 times the tabu tenure (i.e. 500 %).

The output of the algorithm in Fig. 4 show first eight bars of the piece. The execution time is very short (5 seconds) and the difficulty score was 712. Experts confirmed that the solution displayed in the output is easily playable and thus forms a good fingering.

6 Conclusion and Future Research

The problem of finding a good piano fingering has been described as a combinatorial optimisation problem. Different sources of difficulty in the implemented objective function allow to deal with complex polyphony; to analyse the left and right hand; and to have the option to adapt some parameters to personal preferences and biomechanics of the hand. A tabu search algorithm, a metaheuristic, was implemented. We showed that it can find in a relatively short amount of execution time a very good solution which has been illustrated by an output.

In the future, this research could be expanded by including rules in the objective function that specify the interpretation and memorisation aspects of a piano fingering. Examples of such rules could be the use of a strong finger on the first beat of a bar, or using the same fingering patterns for identical note sequences. The objective function could also be improved by verifying if machine learning techniques can be applied using a database of piano fingerings. The algorithm could benefit from the inclusion of extra, more sophisticated neighbourhoods.

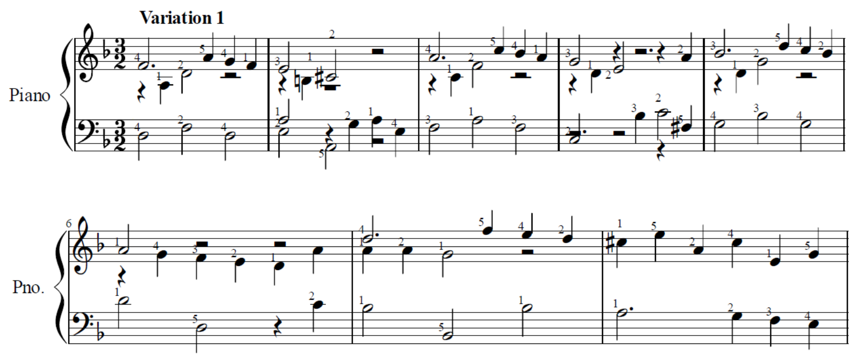


Fig. 4. Output for the first eight bars of the first variation on the Saraband from Suite in D minor (HWV 437) by G.F. Händel.

References

1. Sloboda, J.A., Clarke, E.F., Parncutt, R., Raekallio, M.: Determinants of finger choice in piano sight-reading. J. Exp. Psychol.-Hum. Percept. Perform. 24, 185–203 (1998)
2. Robine, M.: Analyse automatique du doigté au piano. In: Proceedings of the Journées d'Informatique Musicale, pp. 106–112 (2009)
3. Fiechter, C.-N.: A parallel tabu search algorithm for large travelling salesman problems. Discrete Appl. Math. 51, 243 – 267 (1994)
4. Gendreau, M., Hertz, A., Laporte, G.: A Tabu Search Heuristic for the Vehicle Routing Problem. Manag. Sci. 40, 1276 – 1290 (1994)
5. Herremans, D.: Tabu Search voor de Optimalisatievan Muzikale Fragmenten. Master's thesis at UAntwerp, Faculty of Applied Economics, Antwerp (2005)
6. Gellrich, M., Parncutt, R.: Piano Technique and Fingering in the Eighteenth and Nineteenth Centuries: Bringing a Forgotten Method Back to Life. B. J. Music Ed. 15, 5–23 (1998)
7. Sayegh, S. I.: Fingering for String Instruments with the Optimum Path Paragdigm. Comput. Music J. 13, 76 – 84 (1989)
8. Parncutt, R., Sloboda, J.A., Clarke, E.F., Raekallio, M., Desain, P.: An ergonomic model of keyboard fingering for melodic fragments. Music Percept. 14, 341–382 (1997)
9. Parncutt, R.: Modeling piano performance: Physics and cognition of a virtual pianist. In: Proceedings of Int. Computer Music Conference, pp. 15–18 (1997)
10. Viana, A.B., de Morais Júnior, A.C. Technological improvements in the siedp. In: IX Brazilian Symposium on Computer Music, Campinas, Brazil (2003)
11. Hart, M., Bosch, R., Tsai, E.: Finding optimal piano fingerings. The UMAP Journal, 2, 167–177 (2000)
12. Radicioni, D.P., Anselma, L., Lombardo, V.: An algorithm to compute fingering for string instruments. In: Proceedings of the National Congress of the Associazione Italiana di Scienze Cognitive, Ivrea, Italy (2004)

13. Al Kasimi, A., Nichols, E., Raphael, C.: A simple algorithm for automatic generation of polyphonic piano fingerings. In: 8th International Conference on Music Information Retrieval, Vienna (2007)
14. Yonebayashi, Y., Kameoka, H., Sagayama, S. Automatic decision of piano fingering based on a hidden markov models. In: IJCAI, pp. 2915–2921 (2007)
15. Jacobs, J.P.: Refinements to the ergonomic model for keyboard fingering of Parncutt, Sloboda, Clarke, Raekalliio and Desain. Music Percept., 18, 505–511 (2001)
16. Lin, C.-C., Liu, D.S.-M.: An intelligent virtual piano tutor. In: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications, pp. 353-356 (2006)
17. Balliauw, M.: A variable neighbourhood search algorithm to generate piano fingerings for polyphonic sheet music. Master's thesis at UAntwerp, Faculty of Applied Economics, Antwerp (2014)
18. Sébastien, V. Ralambondrainy, H., Sébastien, O. Conruyt, N.: Score analyzer: Automatically determining scores difficulty level for instrumental e-learning. In: ISMIR, pp. 571-576 (2012)
19. Sörensen, K., Glover, F.: Metaheuristics. In: Glass, S.I., Fu, M.C. (eds.) Encyclopedia of Operations Research and Management Science, pp. 960–970 (2013)
20. Glover, F., Laguna, M.: Tabu search. Kluwer Academic Publishers (1993)
21. Mladenović, N., Hansen, P.: Variable neighbourhood search. Computers and Operations Research, 24, 1097–1100 (1997)

7 Checklist of Items to be Sent to Volume Editors

Here is a checklist of everything the volume editor requires from you:

- ☐ The final LaTeX source files
- ☐ A final PDF file
- ☐ A copyright form, signed by one author on behalf of all of the authors of the paper.
- ☐ A readme giving the name and email address of the corresponding author.

8 info

8.1 Checking the PDF File

Kindly assure that the Contact Volume Editor is given the name and email address of the contact author for your paper.

8.2 Copyright Forms

The copyright form may be downloaded from the "For Authors" (Information for LNCS Authors) section of the LNCS Website: `www.springer.com/lncs`. Please send your signed copyright form to the Contact Volume Editor, either as a scanned pdf or by fax or by courier. One author may sign on behalf of all of the other authors of a particular paper. Digital signatures are acceptable.