


## Article

# Generating Fingerings for Piano Music with Model-Based Reinforcement Learning

Wanxiang Gao <sup>1</sup>, Sheng Zhang <sup>1,\*</sup>, Nanxi Zhang <sup>2</sup>, Xiaowu Xiong <sup>1</sup>, Zhaojun Shi <sup>1</sup> and Ka Sun <sup>1</sup>

<sup>1</sup> School of Information Engineering, Nanchang Hangkong University, No. 696, Fenghe South Avenue, Nanchang 330063, China; gf7600gs@gmail.com (W.G.); 2104085400005@stu.nchu.edu.cn (X.X.); 36040@nchu.edu.cn (Z.S.); sunka1982@163.cn (K.S.)

<sup>2</sup> Institution of Education, University College London, Gower Street, London WC1E 6BT, UK; dtvnz2@ucl.ac.uk

\* Correspondence: zhangsheng168@nchu.edu.cn

**Abstract:** The piano fingering annotation task refers to assigning finger labels to notes in piano sheet music. Good fingering helps improve the smoothness and musicality of piano performance. In this paper, we propose a method for automatically generating piano fingering using a model-based reinforcement learning algorithm. We treat fingering annotation as a partial constraint combinatorial optimization problem and establish an environment model for the piano performance process based on prior knowledge. We design a reward function based on the principle of minimal motion and use reinforcement learning algorithms to decide the optimal fingering combinations. Our innovation lies in establishing a more realistic environment model and adopting a model-based reinforcement learning approach, compared to model-free methods, to enhance the utilization of samples. We also propose a music score segmentation method to parallelize the fingering annotation task. The experimental section shows that our method achieves good results in eliminating physically impossible fingerings and reducing the amount of finger motion required in piano performance.

**Keywords:** piano fingering model; reinforcement learning; symbolic music processing; combinatorial optimization



**Citation:** Gao, W.; Zhang, S.; Zhang, N.; Xiong, X.; Shi, Z.; Sun, K. Generating Fingerings for Piano Music with Model-Based Reinforcement Learning. *Appl. Sci.* **2023**, *13*, 11321. <https://doi.org/10.3390/app132011321>

Academic Editor: Lamberto Tronchin

Received: 21 August 2023

Revised: 19 September 2023

Accepted: 11 October 2023

Published: 15 October 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

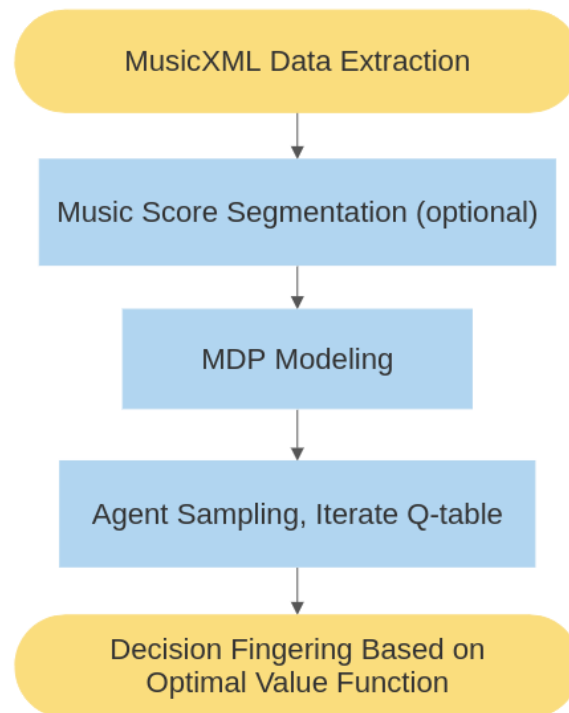
Piano composers and arrangers add fingering to sheet music to indicate which finger should be used to play each note. However, most piano sheet music only provides detailed notation of the musical notes themselves, without specific fingering instructions. Human performers rely on their professional knowledge, personal physical conditions, the actual sequence of notes, and multiple rehearsals to determine the fingering. This process involves a significant amount of decision-making. A scientifically and logically arranged fingering not only reduces the difficulty of performance and relieves mental burden for the pianist but also enhances the musicality and fluidity of the performance. Generally, when novice performers receive a piece of music they are about to practice, their first task is to annotate their own fingering on the sheet music. In theory, any key on the piano can be played with any finger. However, the number of possible fingering paths increases exponentially with the length of the note sequence, and the majority of these fingerings are difficult to execute or even physically impossible.

The fingering annotation task is mainly approached through two categories: rule-based and data-driven methods. Rule-based methods [1–7] aim to summarize the anatomical and kinematic principles of piano playing, model the piano keyboard and human hand as realistically as possible, and scientifically quantify the motion cost of fingering transitions. They make use of the state information in simulated performances to decide on fingerings. Data-driven methods [8–13], on the other hand, utilize piano scores annotated with human

fingerings to extract implicit patterns of how humans arrange fingerings. These patterns are then used to predict fingerings for new music scores.

In earlier works, due to the lack of human expert annotated dataset, rule-based methods were often used. However, in recent years, the availability of the PIG dataset [10] has shifted the research focus towards data-driven methods. Nevertheless, in recent research [12], we have found that the accuracy of the PIG dataset needs improvement, as it contains fingerings like “crossed chords” or “thumbless cross” that are not realistically playable by humans. This directly affects the effectiveness of data-driven methods, often resulting in the generation of fingerings that include such unplayable fingerings.

In this research, we employed rule-based approaches. We model the generation of piano fingerings as a Markov decision process (MDP). The agent interacts with its environment and evaluates the fingerings using a reward function. We introduce the prioritized sweeping algorithms [14] to implement model-based reinforcement learning. In order to execute the algorithm without enumerating the state space and action space of the piano fingering problem, we replaced the traditional Q-table used in tabular reinforcement learning with a hash table based on key-value storage. Additionally, we propose a method for segmenting the music score to enable the parallel execution of the algorithm. The completed processing steps are shown in Figure 1.



**Figure 1.** Processing steps of automatic piano fingering annotation.

The main contributions are summarized as follows:

- We constructed a more realistic environment model for piano performance, and invalid action masking is used to maximally constrain physically impossible fingerings.
- We used model-based reinforcement learning to address the fingering annotation task, which has improved the sampling efficiency compared to the previous model-free approach.
- We introduce a Q-table based on key-value storage to achieve tabular reinforcement learning without enumerating action and state spaces.
- We propose a piano sheet music segmentation method to parallelize problem-solving.

This paper is divided into seven sections. Section 1 provides a brief overview of the problem. Section 2 presents related work in the field. Section 3 describes the approach used to model the environment for the problem. Section 4 describes learning algorithm we used.

Section 5 discusses the experimental details and result evaluation. In Section 6, a method for segmenting the music score is presented. Finally, Section 7 concludes the paper and provides suggestions for future research.

## 2. Related Work

Piano fingering automatic annotation can be broadly categorized into two methods: rule-based and data-driven.

The earliest rule-based method for piano fingering annotation was proposed by R. Parncutt and others [1]. Hart [2] introduced a method that utilizes dynamic programming to compute optimal fingerings for single-note melodies. Balliauw [3,4] applied two different search algorithms to tackle this problem. Ramoneda [5] and Koornstra [6] explored model-free reinforcement learning algorithms to find fingerings with minimal hand movement; however, since it is based on a very simple environment model, it can only handle single-note melodies. Xu [7] conducted research on the application of reinforcement learning to the piano performance of humanoid robotic hands with only four fingers. Applying reinforcement learning to other combinatorial optimization problems has also been the subject of extensive research [15–17]. Huang et al. [18] conducted research on invalid action masking in reinforcement learning.

E. Nakamura and Y. Yonebayashi [8,9] proposed hidden Markov models (HMMs) for piano fingering, respectively, for single-handed and double-handed outputs, laying the foundation for subsequent data-driven approaches using statistical learning to predict fingerings. In their subsequent work [10], E. Nakamura formalized the task as a statistical learning problem and achieved good fingering consistency by modeling fingerings with a third-order HMM and a long short-term memory (LSTM) network, which is similar to the part-of-speech tagging task in natural language processing. They also introduced the chord hidden Markov model (CHMM) for chord fingering modeling and published the PIG dataset. Guan et al. [11] proposed the pitch difference model, which efficiently utilized the PIG dataset by using relative pitch instead of absolute pitch. Srivatsan et al. [12] proposed checklist models, which introduced soft constraints and used the REINFORCE algorithm to optimize evaluation metrics, improving the overall fluency of fingerings, but the experimental results show that the soft constraint rules used in this work do not fundamentally eliminate unplayable fingerings such as crossed chord, and the optimization objectives of the reinforcement learning part also need to be improved. Randolph et al. [13] treated the fingering annotation as an information retrieval problem and used the Czerny corpus as the dataset.

The quality of data determines the upper limit of data-driven approaches. Existing data-driven methods often struggle to directly eliminate physically impossible fingering predictions. Additionally, during data preprocessing, they often create unrealistic fingering, for example, due to limited data, data-driven methods adopt a technique of flipping left-hand finger labels and pitch to create a second “right-hand portion” in order to augment the training data. However, although the human left and right hands are mirror images of each other, the arrangement of black and white keys on the keyboard is not a mirror-symmetric structure. Simply flipping the left-hand pitch and finger labels cannot accurately represent a right-hand performance. Rule-based methods, on the other hand, have difficulties describing the performance process with fixed paradigms, and their modeling accuracy needs improvement, as many studies are limited to fingering for single-note melodies.

As a continuation of rule-based method research, we aim to formalize the piano performance process through MDP and propose a quantifiable method for calculating motion to enhance the accuracy of the environmental model. Our research aims to achieve a certain level of matching with human fingerings while maximizing the limitation of physically impossible fingerings and minimizing the motion, even seeking fingerings superior to human annotations. Our action constraints and modeling methods can also be applied to future research combining data-driven methods.

### 3. Environmental Model

#### 3.1. Overview of Model Definitions

In reinforcement learning, an environment model is a simulation model that interacts with an agent. It allows the agent to observe the state, receive the agent's action signals, return rewards to the agent, and transition to the next state according to probability. The environment model is an MDP, which can be composed of a 4-tuple  $(S, A, P_a(s, s'), R_a(s, s'))$ .  $S$  is the state space,  $A$  is the action space,  $P_a$  is the state transition probability,  $R_a$  is the reward function for state transition,  $s$  and  $s'$  represent the current state and the new state to which it transitions after executing action  $a$ , respectively.

In this problem, the state  $s$  is composed of a triplet  $(i, f_g, n_{next})$ , where  $i$  represents the position of a note or chord in the musical piece in terms of its chronological occurrence;  $f_g$  represents the current fingering action, which is a sequence composed of note–finger pairs, denoted by  $p_{nf} = (n, f)$ ; therefore,  $f_g$  can be represented as  $f_g = (p_{nf1}, p_{nf2}, \dots, p_{nf5})$ ;  $f$  is a finger number (1 = thumb, 2 = index finger,  $\dots$ , 5 = little finger) on both hands;  $n_{next}$  represents the sequence of notes to be played in the next time step.

Due to the difficulty of fully enumerating the state space, we employ the tabular reinforcement learning method with key-value storage described in Section 4.1 to avoid enumerating the state space  $S$  and action space  $A$ . Since the state transitions are deterministic,  $P_a$  is always 1. The reward function  $R_a$  quantifies the difficulty of fingering transitions for more challenging fingerings, and the reward function  $R_a$  returns lower rewards, or conversely, returns higher rewards. The objective of the reinforcement learning algorithm is to maximize the accumulated reward, thereby determining the optimal fingering decision path.

#### 3.2. Simulation of Keyboard and Hand Interaction

This study utilizes the “white key distance” metric to describe the distance between two keys on a keyboard. It converts the distance between any two keys on the keyboard into a count of white keys, with black keys counting as 0.5 white keys.

$$k_w = [21, 23 \dots 107, 108]$$

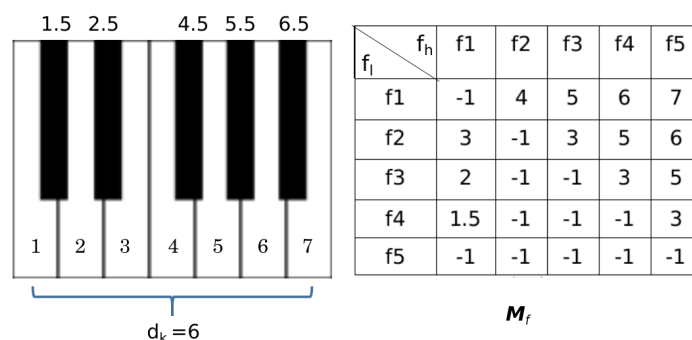
$$d_{A0}(\text{note}) = \text{note} \in k_w ? \text{index}(\text{note}, k_w) : \text{index}(\text{note} + 1, k_w) - 0.5$$

$$d_k(\text{note1}, \text{note2}) = |d_{A0}(\text{note1}) - d_{A0}(\text{note2})| \quad (1)$$

where  $k_w$  represents the enumeration of MIDI note numbers for all white keys.  $d_{A0}$  represents the distance from the leftmost A0 key on the piano to a specific note.  $d_k$  refers to the white key distance between two piano keys,  $\text{note1}$  and  $\text{note2}$ .

Regarding hand features, we can sample the performer's hand and represent it with a matrix  $\mathbf{M}_f$ .

White key distance  $d_k$  and hand feature matrix  $\mathbf{M}_f$  are illustrated in Figure 2. And, Figure 3 illustrates the correspondence between finger numbers and specific fingers.



**Figure 2.** White key distance and hand feature matrix. The distance  $d_k$  between keys can be obtained by subtracting the numbers on the keyboard. The numbers in the matrix  $\mathbf{M}_f$  represent the maximum stretchable distance for the hands.

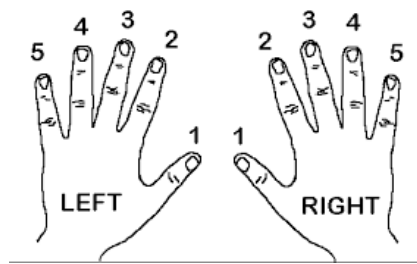


Figure 3. Finger numbers.

$f_1$ – $f_5$  represent the thumb to little finger. The values in the matrix are measured in white key distance and record the maximum expansion distance or maximum crossing distance of each finger. When the value is  $-1$ , this means that the corresponding action cannot be performed. Taking the right hand as an example, let  $f_l$  represent the finger which has a lower label,  $f_h$  represent the fingers has a higher label, and  $f_c$  represent the fingers used for cross-fingering. The maximum expansion distance  $d_{smax}$  and maximum crossing distance  $d_{cmax}$  between two fingers can be represented using the following method:

$$\begin{aligned} d_{smax} &= \mathbf{M}_f[f_l, f_h] \\ d_{cmax} &= \mathbf{M}_f[f_c, 1] \end{aligned} \quad (2)$$

### 3.3. Invalid Action Masking

In an MDP, we want the agent to always perform anatomically possible actions through invalid action masking. This way, we do not need to penalize physically impossible fingerings in the design of the reward function (invalid action penalty). Instead, we can quantitatively evaluate the difficulty of all possible finger technique transfers. Therefore, we need to define a mask function (Algorithm 1) that returns the set of physically possible actions for different states  $s$ . On the other hand, invalid action masking has increased the sampling efficiency of the agent by reducing the search space.

---

#### Algorithm 1 Invalid action masking for fingering

---

```

procedure MASK(notes)
  all_fingerings  $\leftarrow$  enumerating all potential fingerings by note counts
  if length(notes) == 1 then ▷ Only single note at next time
    valid_fingerings  $\leftarrow$  remove thumbless crossing, hop, invalid cross fingering,
    invalid expansion and contraction
  else ▷ More than one notes at next time
    for fingering  $\in$  all_fingerings do
      note_finger_pairs  $\leftarrow$  pair(fingering, notes)
      for finger_combine  $\in$  combination(note_finger_pairs, 2) do
        if invalid expansion and contraction in this finger_combine then
          break
        end if
      end for
      push(valid_fingerings, fingering) if all combinations valid
    end for
  end if
  return valid_fingerings
end procedure

```

---

For the next moment when the number of notes requiring finger allocation is  $n$ , enumerating all potential fingerings can be seen as a simple combination problem: selecting  $n$  fingers from the five available fingers. The number of combinations is calculated as  $C_5^n$ . However, not all potential fingerings are playable. Some fingerings may have a distance between two fingers that exceeds the maximum expansion distance  $d_{smax}$ , or violate finger

contraction rules. To determine whether a fingering exceeds the maximum expansion distance, we can compare the white key distance  $d_k$  between two notes in the fingering with the maximum expansion distance  $d_{smax}$ . If  $d_k > d_{smax}$ , it is considered physically impossible for the two fingers to stretch that far. To identify violations of finger contraction, we can compare the number of piano keys between two notes with the number of fingers involved. If  $|note1 - note2| < |finger1 - finger2|$ , it is considered a violation of finger contraction, where neither  $finger1$  nor  $finger2$  can be the thumb. Furthermore, transfer constraints should also take into account physically impossible scenarios in 1-to-1 fingerings, such as the thumb not being able to transition from a white key to a black key during cross fingering.

### 3.4. Quantification of Fingering Transition Difficulty

Below are some metrics for quantifying the fingering transition difficulty, which will be involved in the computation of the reward function.

#### 3.4.1. Stretching Rate

During single-note playing, finger stretching and contracting frequently occur, and the stretching abilities of different fingers are not equal. This study proposes a quantification method called a “finger stretching rate” to measure finger extension, as shown in Figure 4a.

We first define the finger extension between two fingers: when two fingers are placed on the keyboard, the white key distance between their finger labels  $f_a$  and  $f_b$  is defined as the natural distance  $d_{nature} = |f_a - f_b|$ . The maximum extension distance can be obtained from the matrix  $\mathbf{M}_f$  mentioned earlier, denoted by  $d_{max}$ , which represents the maximum extension ability between the two fingers. The actual distance between the two fingers on the keyboard is the white key distance  $d_k$  between two pressed notes.

The stretching rate can be defined as  $r_s = (d_k - d_{nature}) / (d_{max} - d_{nature})$ . A value close to 1 indicates that the fingers are approaching their maximum stretching ability, resulting in greater discomfort and a lower reward. Similarly, for finger contraction, we can define the contraction rate  $r_s = (d_{nature} - d_k) / d_{nature}$ . A value close to 1 indicates that the fingers are approaching their maximum contraction ability, resulting in stronger discomfort and a lower reward.

During chord playing, we allocate  $n$  fingers on the keyboard. We can calculate the combinations of fingers used, with the number of combinations being  $C_n^2$ . The overall stretching rate of the chord fingering can be obtained by averaging the stretching rates  $r_s$  of each finger combination. Specifically, the formula is  $r_{all} = (\sum_{k=1}^c (r_{sk})^a) / c$ . Since lower stretching rates result in less discomfort, the discomfort is usually more noticeable when approaching maximum stretching. Therefore, a parameter  $a > 1$  is introduced to slow down the growth of the stretching rate at lower values of stretching.

#### 3.4.2. Hand Movement Distance

Hand position refers to the position of the hand on the keyboard. Since the hand covers an area of multiple piano keys, it is not convenient to be precise about specific keys. However, by observing the anatomical diagrams of the human hand, we can approximate the hand position with the positions of middle finger fingers when the hand is naturally relaxed on the keys. In cases where multiple fingers are involved in playing chords and other techniques, the hand position can be approximated by averaging the positions of the thumb and little fingers. The hand displacement distance can be calculated by considering the changes in hand position before and after fingerings, as shown in Figure 4b.

For the left hand,  $h = -1$ , and for the right hand,  $h = 1$ . In the case of single-note fingerings, where  $f_n$  represents the finger pressing the note  $n$ , the hand position can be calculated as  $p_s(f_n, n) = d_{A0}(n) + h \cdot (3 - f_n)$ . For chord fingerings with multiple notes, let us consider the finger  $f_l$  and note  $n_l$  of the lowest pitch in the fingering, as well as the finger  $f_h$  and note  $n_h$  of the highest pitch. The hand position for the chord fingering can be determined as  $p_c(f_l, n_l, f_h, n_h) = (p_s(f_l, n_l) + p_s(f_h, n_h)) / 2$ .



If the positions before and after the fingering transition are denoted by  $p_{pre}$  and  $p_{next}$ , respectively, the hand position movement distance can be calculated as  $d_h = p_{pre} - p_{next}$ . A larger hand position movement indicates a stronger discomfort during the fingering transition, resulting in a lower reward.

### 3.4.3. Cross Fingering Distance

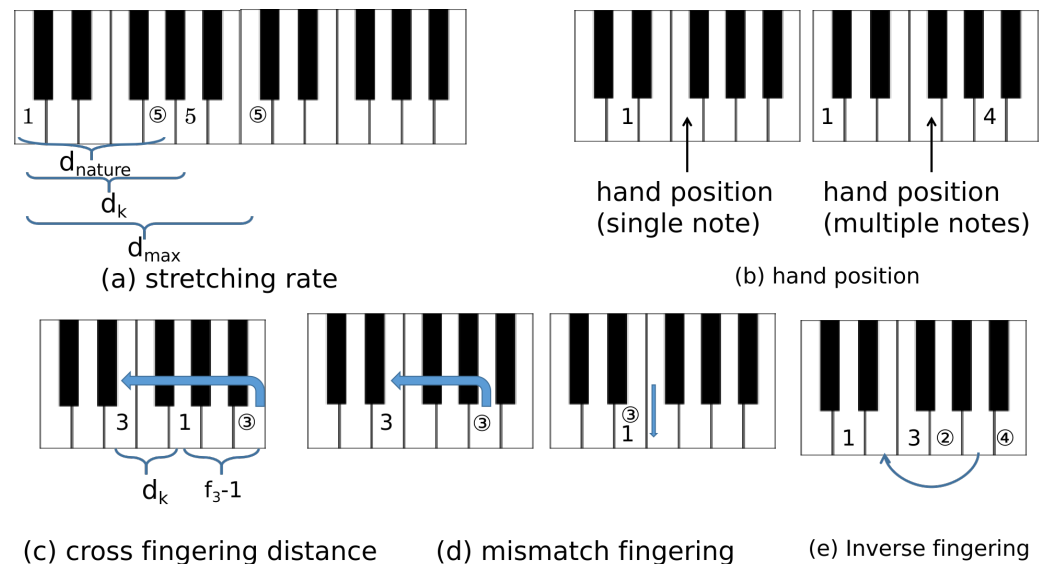
Cross fingering is a special finger technique where one finger crosses over the other fingers, as shown in Figure 4c, it commonly used in fast-paced performances that cover a wide range of musical notes. Cross fingering refers to fingers 2, 3, and 4 crossing over finger 1, or finger 1 passing under fingers 2, 3, and 4. When calculating the cross fingering distance, let us assume the note played by finger 1 (thumb finger) is  $n_a$ , and the note played by finger  $f_c$  participating in the finger crossing is  $n_c$ . The cross fingering distance is calculated as  $d_c = d_k(n_a, n_c) + f_c - 1$ . The greater the cross fingering distance, the stronger the discomfort in finger movement and the lower the reward.

### 3.4.4. Fingering Mismatches Count

The term “fingering mismatch count”  $c_m$  refers to the number of occurrences in an adjacent fingering transitions where the same note is played using different fingers or where a single finger plays two different notes, as shown in Figure 4d. In such cases, finger transitions can also pose certain difficulties.

### 3.4.5. Inverse Fingering Count

The “inverse fingering count”  $c_i$  involves rearranging all the notes within a pair of adjacent fingerings from low to high and calculating the inversion number of finger labels corresponding to each note, as shown in Figure 4e. If an inverse fingering exists, it makes fingering transitions more challenging, and thus lower rewards should be assigned.



**Figure 4.** Quantification of motion in fingering transfer, the blue arrows represent the transitions of fingerings. The numbers on the keys represent fingering labels, while the numbers enclosed in circles represent the fingering in the previous moment.

## 3.5. Reward Function

The reward function represents the evaluation of fingering difficulty. We quantify the motion information of fingering transitions mentioned above and assign higher rewards to fingering transitions that are more comfortable and easier to play. The maximum reward for each fingering transition is set to around 50. The parameters of the reward function have been carefully balanced through iterative experiments.

For single-note to single-note fingering transitions (i.e., 1-to-1 transitions as shown in Table 1), there are three possibilities: finger expansion/contraction, cross fingering, and hand position movement. We prioritize finger expansion and contraction as the first choice, followed by cross fingering, and finally consider hand position movement.

$$reward = \begin{cases} 40 + 10(1 - r_s^2) & \text{if finger stretches} \\ 20 + 2.5(4 - d_c) & \text{if cross fingering} \\ -10 - 0.5d_h & \text{if hand movement} \end{cases} \quad (3)$$

For single-note to multiple-note or multiple-note to multiple-note fingering transitions (i.e., 1-to-n and n-to-n transitions, as shown in Table 1), when the hand position movement distance is significant, we prioritize fingering options with lower expansion rates

$$reward = \begin{cases} 20(1 - r_{all}) + (5 - d_h) & \text{if } d_h > 5 \\ (10(1 - r_{all}^2) + 8(5 - d_h)) \cdot (1 - (c_i + c_m)/n_t) & \text{if } d_h \leq 5 \end{cases} \quad (4)$$

For multiple-note to single-note fingering transitions (i.e., n-to-1 transitions as shown in Table 1), the main consideration is to minimize the hand movement distance.

$$reward = (50 - d_h) \cdot (1 - (c_i + c_m)/n_t) \quad (5)$$

**Table 1.** The motion metrics required for different types of fingering transitions.

Fingerings Transitions Types	Metrics
1-to-1	stretching rate $r_s$ cross fingering distance $d_c$ hand movement distance $d_h$
1-to-n n-to-n	hand movement distance $d_h$ stretching rate $r_{all}$ fingering mismatches count $c_m$ inverse fingering count $c_i$ total number of notes before and after fingering transition $n_t$
n-to-1	hand movement distance $d_h$ fingering mismatches count $c_m$ total number of notes before and after fingering transition $n_t$ inverse fingering count $c_i$

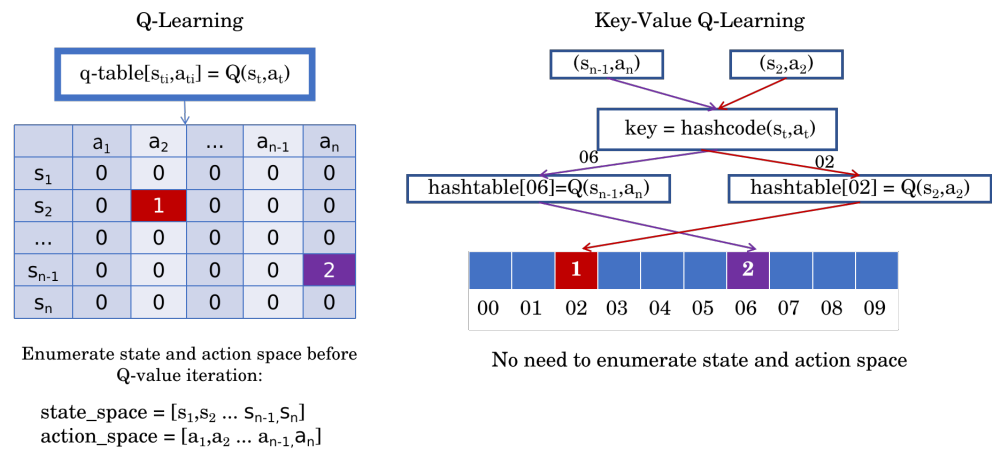
## 4. Learning Algorithm

### 4.1. Tabular Reinforcement Learning with Key-Value Storage

In this task, attempting to directly enumerate the discrete state space and action space to construct a  $|S| \times |A|$  table is very cumbersome, especially since the combination of 88 piano keys and 5 finger states is very large. Moreover, for any state  $s$ , it is impossible to access the complete action space.

For this, we designed a key-value storage tabular reinforcement learning method, as shown in the Figure 5, replacing the underlying access of the Q-table with an arbitrary key-value storage system, such as a hash table. This method avoids the complete enumeration of the state space and action space. Like classical tabular reinforcement learning, this method is only applicable to MDPs with discrete state spaces and discrete action spaces, and requires that the state-action pairs in the MDP be finite. After the agent performs action  $a$  in state  $s$ , it will use a hash function to calculate the hash value of the state-action pair  $(s, a)$  as the address of the hash table. This address stores the Q-value of the corresponding state-action pair  $(s, a)$ . After performing the action and obtaining the reward  $r$ , the iterated new Q-value is written back to the corresponding address in the hash table, and then the agent enters the next state.





**Figure 5.** Traditional tabular Q-learning and key-value storage Q-learning.

#### 4.2. Model-Based Reinforcement Learning in Fingerings Annotation

In earlier studies on piano fingering annotation using reinforcement learning, model-free algorithms such as Q-learning [19] or deep Q network (DQN) [20] were commonly employed. Model-free algorithms do not estimate the probability distribution of state transitions and the reward function of the MDP. They can only learn through interaction with the environment.

Instead, model-based algorithms can learn by predicting the rewards and state transitions after executing actions. This process is referred to as model learning [17]. In this task, since the environmental model is deterministic, meaning the state transition probability  $P_a(s, s')$  is always 1, utilizing model-based reinforcement learning algorithms can reduce the interaction between the agent and the environmental model compared to model-free algorithms, thereby improving the sampling efficiency. Therefore, a model-based reinforcement learning approach like prioritized sweeping is more suitable for this task.

Prioritized sweeping is essentially an improvement to the Dyna architecture [21]. It introduces a priority queue, during the planning step of Dyna, which utilizes the experience in model learning to prioritize learning state–action pairs with a larger TD-error. In Algorithm 2, we replaced the Q-table and  $Model(s, a)$  in the original algorithm with implementations using hash tables instead of 2D matrices. This is intended to avoid the enumeration of the state space and action space.

---

#### Algorithm 2 Prioritized sweeping (key-value storage)/

---

**procedure** PRIORITIZED SWEEPING

Initialize empty hash table  $Q(s, a)$ ,  $Model(s, a)$  and empty priority queue  $PQueue$ .

**while** true **do**

$s \leftarrow \text{current}(\text{nonterminal})\text{state}$

$a \leftarrow \text{policy}(s, Q)$

Execute action  $a$ , observe resultant state,  $s'$ , and reward,  $r$

$Model(s, a) \leftarrow s', r$

$p \leftarrow |r + \gamma \max_{a'} Q(s', a') - Q(s, a)|$

$\text{push}(PQueue, (s, a), p)$

**while** !isEmpty( $PQueue$ ) **do**

$s, a \leftarrow \text{pop}(PQueue)$

$s', a \leftarrow Model(s, a)$

$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$

**for**  $\bar{s}, \bar{a}$  predicted to lead to  $s$  **do**

$\bar{r} \leftarrow \text{predicted reward}$

$p \leftarrow |r + \gamma \max_{a'} Q(s', a') - Q(s, a)|$

**if**  $p > \theta$  **then**

$\text{push}(PQueue, (\bar{s}, \bar{a}), p)$

**end if**

**end for**

**end while**

**end while**

**end procedure**

---

## 5. Experiment and Evaluation

### 5.1. Implementation Details

This experiment utilizes music scores in MusicXML format as experimental materials and incorporates the PIG dataset for algorithm comparison. The process begins with parsing the MusicXML into a music21 [22] stream. Then, all chord symbols are removed, and the resulting treble clef and bass clef parts are separately exported into two MIDI files. The MIDI.jl [23] library is employed to parse the notes from the MIDI files, sorting them in chronological order, and combining them into a two-dimensional array based on their play time. The MDP modeling is implemented using POMDPs.jl [24], and a corresponding prioritized sweeping solver is developed.

The agent's exploration strategy utilizes the epsilon-greedy algorithm with an exploration rate of  $\epsilon = 0.8$ . For the reinforcement learning algorithm in a fully deterministic environment, a learning rate of 0.99 is employed, emphasizing the learning of new Q-values to facilitate faster convergence. Since the constructed MDP for this task does not possess cyclic structures, the agent can reach the terminal state within a finite number of steps. Hence, a discount factor of  $\gamma = 1$  is utilized, maximizing the prediction of absorbing future rewards, i.e.,  $\max_a Q(s_{t+1}, a)$ . In the planning process of the prioritized sweeping, a threshold of  $\theta = 3$  is chosen for the priority queue. Any state-action pair with a TD-error greater than  $\theta$  is added to the priority queue. During the reinforcement learning algorithm's iterations, the total reward is calculated based on the optimal value function every 10 iterations. If the change in total reward is within the range of  $\pm 0.5$ , we consider that the algorithm has converged, and the iteration process is stopped.

Finally, the experiment was conducted on a regular computer with an i5-9500 CPU and 8GB RAM. The algorithm was deployed on a Linux 6.1 LTS operating system and executed using Python 3.11 and Julia 1.9 interpreters to run the experimental code.

### 5.2. Influence of Model-Based Method on Sampling Efficiency

We used model-free Q-learning and model-based prioritized sweeping to annotate the same music score, and recorded the total rewards obtained based on the optimal actions after each iteration, as shown in Figure 6.

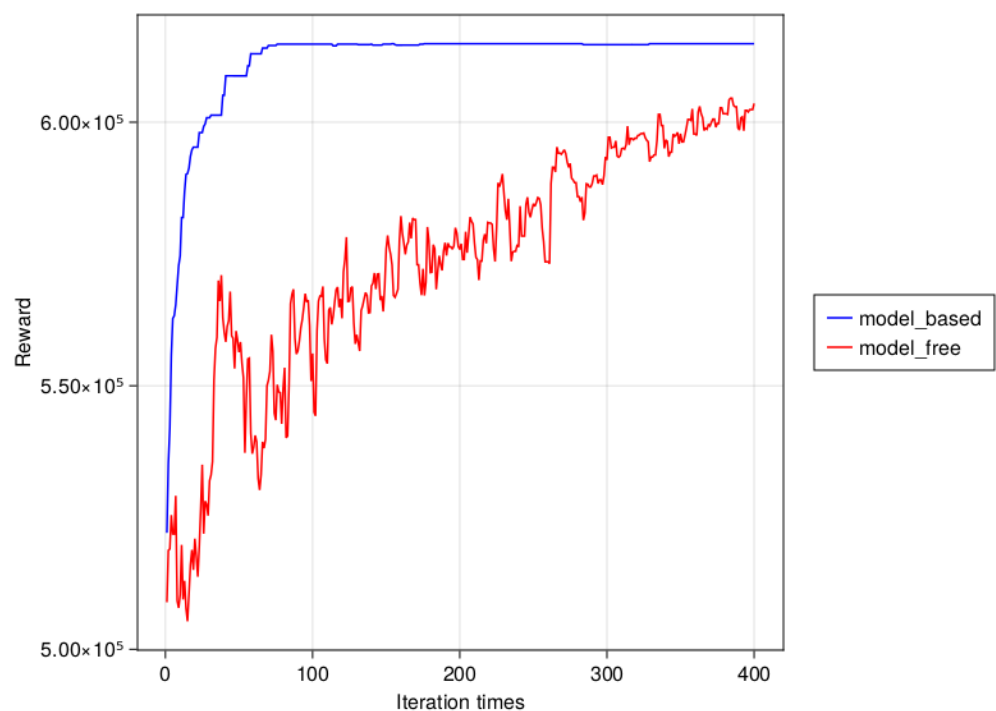


Figure 6. Comparison between the model-free and model-based RL methods.

It can be observed that model-based methods have a higher sampling efficiency compared to model-free methods. This is because the prioritized sweeping agent utilizes the experience from the learned model during the planning step to iteratively update the state–action pairs where the TD-error exceeds a threshold, and propagates this process forward.

### 5.3. Analysis of Fingering Results

To facilitate comparison, we conducted experiments using the PIG dataset. Due to differences in input representations, we performed some preprocessing steps. For instance, we treated the space between longer rests as separate sequences and considered arpeggio as chords rather than continuous single notes. We compared our algorithm with the statistical learning algorithm [10] and the DQN reinforcement learning algorithm [5]. Since the DQN implementation only supports monophonic melodies and is based on a simpler environmental model, we removed most of the melodies containing chords and used less data for comparison.

We evaluated the results using the match rate  $M_{gen}$  [10] for annotation accuracy and the metrics crossed chord, thumbless cross, hop, and step spread [12] for fingering quality. The dataset ground truth is presented in Table 2. The experimental results are presented in Table 3.

We made an improvement in evaluating the overall hand stretching using the step spread metric. Previously, the method used the semitone difference  $\Delta p$  divided by the difference in finger labels  $\Delta f$ . In our approach, we replaced the semitone difference with the difference in white key distances  $\Delta d_k$ , resulting in  $StepSpread = \Delta d_k / \Delta f$ . This modification provides a better reflection of the physical distances on the keyboard.

**Table 2.** Dataset ground truth.

	Crossed Chord	Thumbless Cross	Hop	Step Spread
Full dataset	92	224	561	1.18
Test set	72	119	323	1.18
Test set (De-dupe) <sup>1</sup>	10	18	47	1.20

<sup>1</sup> Keeping only one sample for each different piece.

**Table 3.** Experimental result. HMM represents a previous data-driven statistical learning method. DQN is a model-free reinforcement learning algorithm, implemented based on a simpler environmental model. Prioritized sweeping is the model-based reinforcement learning algorithm utilized in this research.

Method	$M_{gen}$	Crossed Chord	Thumbless Cross	Hop	Step Spread
1rd HMM	61.7	25	6	14	1.18
2rd HMM	64.3	17	8	44	1.17
3rd HMM	64.5	16	9	48	1.18
DQN	41.7	— <sup>1</sup>	21	28	1.16
Prioritized sweeping	58.4	0	5	4	1.14

<sup>1</sup> This implementation cannot label chords.

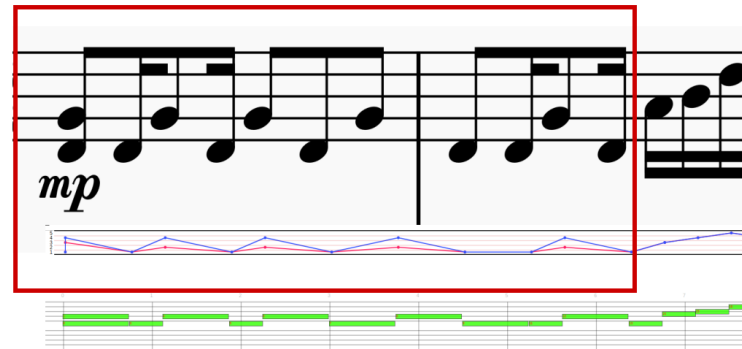
Our method exhibited a lower match rate  $M_{gen}$  compared to the statistical approach, which was expected considering that reinforcement learning algorithms do not acquire any experience from the PIG dataset. However, our method achieved significant improvement in several metrics that measure poor fingering, such as crossed chord, thumbless cross, and hop. We were able to eliminate nearly all instances of physically impossible fingerings and difficult-to-play fingerings.

Our method achieved the best results in the evaluation of the overall hand stretching metric, step spread. This indicates that our reinforcement learning algorithm has been optimized for motion efficiency, leading to superior performance in terms of hand stretching.

#### 5.4. Example Results

##### 5.4.1. More Comfortable Fingering

Due to our method's optimization of motion cost, it exhibits smaller finger movement compared to statistical learning approaches. For instance, Figure 7 shows the fingering results of Masaru Yokoyama's "Syotengaru":

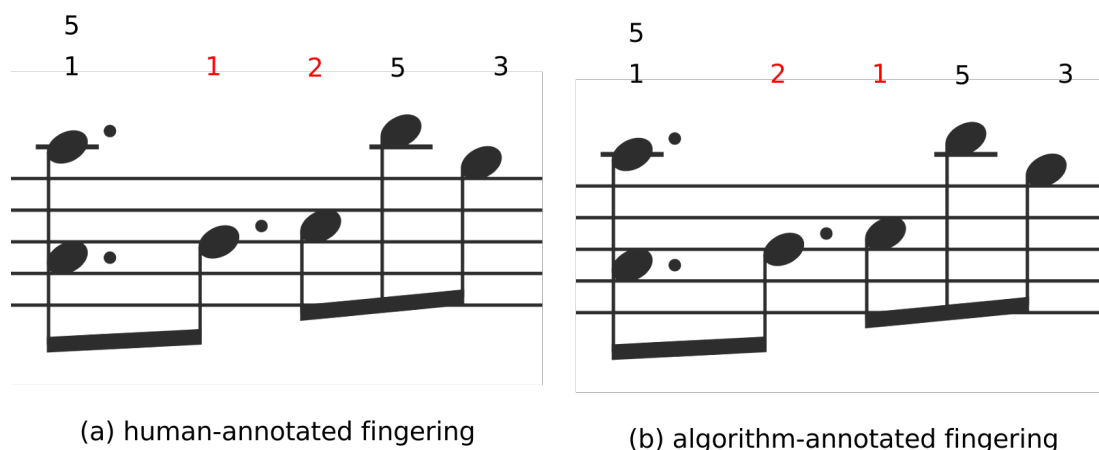


**Figure 7.** Thered line represents reinforcement learning fingering result, while the blue line represents the statistical learning result. Lower fluctuations in the line imply greater finger stretches. The green blocks below represent the MIDI visualization of the melody.

The reinforcement learning approach selects finger 4 to play G4, while the statistical approach selects finger 3 and 2. Clearly, the statistical learning method involves greater hand stretching, whereas our method produces fingerings with reduced stretching, offering enhanced comfort.

##### 5.4.2. Explore Better Fingering

Our method has the potential to discover fingerings that are better than human annotated fingerings, as shown in Figure 8.



**Figure 8.** Comparison of human-annotated fingering and algorithm-annotated fingering. The numbers on the musical notes indicate fingering labels.

Human-annotated fingerings are in line with first intuition, as there are two octaves, A4-A5 and B4-B5, choosing to directly move hand positions is intuitive. However, the fingering choice derived from the algorithm does not involve hand position change. It selects the index finger to play B4 and then cross thumb finger to play C5. This approach not only avoids hand movement but also reduces the discomfort caused by the significant stretch of using the finger transition 2–5 to play C5-B5, as indicated in the human-annotated fingering.

### 5.5. Human Evaluation

We conducted a small-scale human evaluation of the fingering generated by the algorithm. We recruited a music master's degree holder with several years of experience playing piano and provided them with ten excerpts of sheet music annotated with fingering generated by the algorithm.

The pianist provided feedback on the fingering segments we provided. He noted that our method ensured fundamental playability, consistent with our expectations. Our method was beneficial in optimizing finger stretches in most cases, especially in piano arrangements for some popular music pieces, where these musical techniques typically require less demand on finger agility. However, the pianist also observed that in a very few instances, our method did not perform well on melodic segments with specific sequential patterns, where each sequential repetition requires the same fingerings for consistent fingering memorization and smooth performance rather than simply minimizing motion. Clearly, our method cannot account for musical semantics. This suggests that our future work must involve a combination of data-driven and rule-based methods.

## 6. Music Score Segmentation

### 6.1. Segmentation Method

Due to the exponential growth of the number of note combinations with sequence length, heuristic search methods or reinforcement learning methods often require more time when annotating long sequences of music scores. In previous studies, there have been no attempts to parallelize the task of automatic piano fingering annotation. Due to the presence of Markovian properties in the fingering allocation process, it is often believed that the notes to be annotated must be modeled as a complete HMM or a MDP. However, based on E. Nakamura's statistical analysis [10] of human-annotated piano fingering scores, they discovered that an apex note in a monophonic passage has an almost unique choice of finger, i.e., the little finger. We found that this property is not only global but also holds true for the apex single notes in local melodic fragments.

A local region is defined as a gradually expanding range centered around a single note. The notes within this range must be played in quick succession, with no significant gap in the time between two notes (usually not exceeding the length of a half note), and the pitch range should exceed a perfect fifth. If within this range, there are at least five different pitches of notes or, if the pitch range exceeds an octave, and the pitch of the center single note is the highest within the range, then that single note is considered a local apex note within the local region, as shown in Figure 9. The same applies to the lowest single note for the left hand.



**Figure 9.** Local apex note. The red arrow represents the current local apex note. The blue arrow indicates a gradually expanding local range. The numbers represent the count of expansions.

We analyzed the fingering data in the PIG dataset that were annotated by at least four individuals to verify the probability of local apex notes being played with the little finger.

The results indicate an accuracy of 98.41% for right-hand notes and 99.34% for left-hand notes. This indicates that the fingering selection for local apex notes is also nearly unique.

The uniqueness of the fingering for local apex notes allows us to partition the entire music score into smaller sequential segments, enabling the parallel decomposition of the task and reducing the depth of the search. By scanning the unannotated complete music score sequence, we can identify the local apex notes in the right hand and the local lowest notes in the left hand. These points can serve as segmentation positions to divide the music score into smaller segments, treating each segmented music score sequence as an independent annotation task, as shown in Figure 10.



**Figure 10.** The fingering annotation task between two local apex notes (red square) can be considered as independent.

## 6.2. Influence of Music Score Segmentation on Processing Speed

In order to compare the influence of music score segmentation on the speed of fingering annotation, we conducted a set of comparative experiments. We segmented the right-hand part of Bach Invention No. 1 C into six melody segments of roughly equal length. Then, we constructed these melody segments into an MDP and parallelized the execution of the reinforcement learning algorithm using different six CPU cores. In another group of experiments, we did not segment the music score and directly constructed the complete right-hand part into an MDP, using the same CPU to execute the algorithm. Tables 4 and 5 are the performance records of the experiments averaged over multiple runs.

**Table 4.** Model-free method (Q-learning) execution time and memory usage.

	Time	Memory Usage
segmentation	5.16 s	2.80 GB
no segmentation	13.21 s	6.85 GB

**Table 5.** Model-based method (prioritized sweeping) execute time and memory.

	Time	Memory Usage
segmentation	7.15 s	1.95 GB
no segmentation	7.61 s	2.52 GB

It can be observed that the performance is the poorest when using a model-free reinforcement learning algorithm without any processing of the music score. Both using model-based algorithms and segmenting the music score for parallel processing result in performance improvements. However, during the experiments, we also discovered that when parallelizing the execution of the reinforcement learning algorithm, it is not advisable to divide the sequence into very small segments. This is because the overhead of thread scheduling may exceed the algorithm's execution cost.

## 7. Conclusions

In this study, we continue the exploration of rule-based fingering automatic annotation by utilizing model-based reinforcement learning algorithms to find fingerings with minimal



motion. We constructed a more realistic environmental model, improved the sampling efficiency through model-based reinforcement learning, and propose a method for segmenting long sequential music scores. Our method achieves good results in eliminating physically impossible fingerings and reducing the amount of finger motion required in piano performance.

Since our algorithm is built upon a simulation model of the performance process and generates physically playable fingerings, it is particularly suitable for application in piano performances by humanoid robots in the future. The constraint methods we propose can also be applied in statistical learning approaches.

However, we must acknowledge that a purely rule-based fingering annotation method is not the ultimate solution. Firstly, the parameters in the reward function are difficult to determine directly and require repeated trial and adjustment. Secondly, the intricate patterns of hand movements are difficult to describe comprehensively with detailed rules, making it challenging to fully characterize the problem. Thirdly, due to our reinforcement learning algorithm needing to simultaneously optimize multiple objectives (such as reducing finger stretches and movements at the same time), it is difficult to assign clear priorities for each optimization objective to all states, leading to conflicts in rules.

We believe that a combination of rule-based and data-driven approaches will be the direction of future work. We believe that a true hard constraint machine learning system is needed to effectively improve the quality of fingerings. In addition, modern end-to-end constrained optimization learning is also a solution worth exploring. However, how to balance runtime and accuracy will be a challenge.

Moreover, considering that classical music in PIG has entered the public domain, we hope that subsequent research can establish an open source fingering dataset to facilitate collaboration among researchers to correct errors in the dataset and improve the quality of fingering annotations for data-driven methods. We also anticipate that future datasets can adopt more common formats, and include metadata such as tempo and time signature. Additionally, the playing techniques of notes such as ornaments and staccato can also be marked.

**Author Contributions:** W.G. designed and conceived the experiments; N.Z. provided professional piano fingering suggestions; X.X. parsed the musical score data; W.G. and X.X. performed the experiments; W.G. wrote the paper; S.Z., Z.S. and K.S. reviewed the paper and gave some suggestions for improvement. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the National Natural Science Foundation of China, grant number 62262043, and the Science and Technology Project of Jiangxi Province Education Department, grant number GJJ170575.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Informed consent was obtained from all subjects involved in the study.

**Data Availability Statement:** PIG dataset are available at <https://beam.kisarazu.ac.jp/~saito/research/PianoFingeringDataset/> (accessed on 20 August 2023). As part of our research, we transcribed the music score of commercially copyrighted piano music for analysis and investigation. Due to the proprietary nature of the transcribed copyrighted music, we are unable to provide public access to these materials.

**Acknowledgments:** The numerical calculations in this paper were performed on the computing server of the Information Engineering College of Nanchang Hangkong University. Software: MIDI.jl v2.5, POMDPs.jl v0.9.5, Music21 v9.0, Python 3.11 and Julia 1.9.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

## References

1. Parncutt, R.; Sloboda, J.A.; Clarke, E.F.; Raekallio, M.; Desain, P. An ergonomic model of keyboard fingering for melodic fragments. *Music Percept.* **1997**, *14*, 341–382. [\[CrossRef\]](#)
2. Hart, M.; Bosch, R.; Tsai, E. Finding optimal piano fingerings. *UMAP J.* **2000**, *21*, 167–177.
3. Balliauw, M.; Herremans, D.; Cuervo, D.P.; Sörensen, K. Generating fingerings for polyphonic piano music with a tabu search algorithm. In Proceedings of the International Conference on Mathematics and Computation in Music, London, UK, 22–25 June 2015; Springer: Cham, Switzerland, 2015; pp. 149–160.
4. Balliauw, M.; Herremans, D.; Palhazi Cuervo, D.; Sörensen, K. A variable neighborhood search algorithm to generate piano fingerings for polyphonic sheet music. *Int. Trans. Oper. Res.* **2017**, *24*, 509–535. [\[CrossRef\]](#)
5. Ramoneda, P.; Miron, M.; Serra, X. Piano fingering with reinforcement learning. *arXiv* **2021**, arXiv:2111.08009.
6. Koornstra, T. Comparing a Q-Learning Agent’s and Human-Generated Piano Fingerings. Bachelor Thesis, Utrecht University, Utrecht, The Netherlands, 2021.
7. Xu, H.; Luo, Y.; Wang, S.; Darrell, T.; Calandra, R. Towards Learning to Play Piano with Dexterous Hands and Touch. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; pp. 10410–10416.
8. Nakamura, E.; Ono, N.; Sagayama, S. Merged-Output HMM for Piano Fingering of Both Hands. In Proceedings of the ISMIR, Taipei, Taiwan, 27–31 October 2014; pp. 531–536.
9. Yonebayashi, Y.; Kameoka, H.; Sagayama, S. Automatic Decision of Piano Fingering Based on a Hidden Markov Models. In Proceedings of the IJCAI, Hyderabad, India, 6–12 January 2007; Citeseer: State College, PA, USA, 2007; Volume 7, pp. 2915–2921.
10. Nakamura, E.; Saito, Y.; Yoshii, K. Statistical learning and estimation of piano fingering. *Inf. Sci.* **2020**, *517*, 68–85. [\[CrossRef\]](#)
11. Guan, X.; Zhao, H.; Li, Q. Estimation of playable piano fingering by pitch-difference fingering match model. *EURASIP J. Audio Speech Music Process.* **2022**, *2022*, 7. [\[CrossRef\]](#)
12. Srivatsan, N.; Berg-Kirkpatrick, T. Checklist Models for Improved Output Fluency in Piano Fingering Prediction. In Proceedings of the ISMIR, Bengaluru, India, 4–8 December 2022.
13. Randolph, D.A.; Di Eugenio, B.; Badgerow, J. Expected reciprocal rank for evaluating musical fingering advice. In Proceedings of the Sound and Music Computing Conferences, Virtual, 29 June–1 July 2021.
14. Moore, A.W.; Atkeson, C.G. Prioritized sweeping: Reinforcement learning with less data and less time. *Mach. Learn.* **1993**, *13*, 103–130. [\[CrossRef\]](#)
15. Mazyavkina, N.; Sviridov, S.; Ivanov, S.; Burnaev, E. Reinforcement learning for combinatorial optimization: A survey. *Comput. Oper. Res.* **2021**, *134*, 105400. [\[CrossRef\]](#)
16. Wang, D. Reinforcement Learning for Combinatorial Optimization. In *Encyclopedia of Data Science and Machine Learning*; IGI Global: Hershey, PA, USA, 2023; pp. 2857–2871.
17. Moerland, T.M.; Broekens, J.; Plaat, A.; Jonker, C.M. Model-based reinforcement learning: A survey. *Found. Trends® Mach. Learn.* **2023**, *16*, 1–118. [\[CrossRef\]](#)
18. Huang, S.; Ontañón, S. A Closer Look at Invalid Action Masking in Policy Gradient Algorithms. In Proceedings of the Thirty-Fifth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2022, Hutchinson Island, Jensen Beach, FL, USA, 15–18 May 2022. [\[CrossRef\]](#)
19. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [\[CrossRef\]](#)
20. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [\[CrossRef\]](#)
21. Sutton, R.S. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bull.* **1991**, *2*, 160–163. [\[CrossRef\]](#)
22. Cuthbert, M.; Ariza, C. Music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data. In Proceedings of the International Society for Music Information Retrieval Conference, Utrecht, The Netherlands, 9–13 August 2010.
23. Datseris, G.; Hobson, J. MIDI.jl: Simple and intuitive handling of MIDI data. *J. Open Source Softw.* **2019**, *4*, 1166. [\[CrossRef\]](#)
24. Egorov, M.; Sunberg, Z.N.; Balaban, E.; Wheeler, T.A.; Gupta, J.K.; Kochenderfer, M.J. POMDPs.jl: A Framework for Sequential Decision Making under Uncertainty. *J. Mach. Learn. Res.* **2017**, *18*, 831–835.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.