Machine composition of Korean music via topological data analysis and artificial neural network

Mai Lan Tran\textsuperscript{a,b,†}, Dongjin Lee\textsuperscript{c,b,†}, and Jae-Hun Jung\textsuperscript{a,b,*}

\textsuperscript{a}Department of Mathematics, Pohang University of Science & Technology, Pohang, Korea \textsuperscript{b}POSTECH Mathematical Institute for Data Science (MINDS), Pohang University of Science & Technology, Pohang, Korea \textsuperscript{c}Graduate School of Artificial Intelligence, Pohang University of Science & Technology, Pohang, Korea

(Received 2 April 2022; accepted 26 March 2023)

Common AI music composition algorithms train a machine by feeding a set of music pieces. This approach is a blackbox optimization, i.e. the underlying composition algorithm is, in general, unknown to users. In this paper, we present a method of machine composition that teaches a machine the compositional principles embedded in the music using the concept of overlap matrix. In (Tran Mai Lan, Changbom Park & Jae-Hun Jung (2023) Topological data analysis of Korean music in Jeongganbo: a cycle structure, Journal of Mathematics and Music, DOI: 10.1080/17459737.2022.2164626), a type of Korean music called Dodeuri music has been analysed using topological data analysis (TDA). To apply TDA, the music data is first reconstructed as a graph. Through TDA on the constructed graph, a unique set of cycles is found. The overlap matrix lets us visualize how those cycles are interconnected in music. We explain how we use the overlap matrix for machine composition. The overlap matrix is suitable for algorithmic composition and also provides seed music to train an artificial neural network.

Keywords: Machine composition; Korean music; topological data analysis; persistent homology; cycles; overlap matrix; artificial neural network

AMS Classification Codes: 00A65; 55N31; 68T07

1. Introduction

Topological data analysis (TDA) has been used in music analysis recently by applying persistent homology (Bigo et al. 2013; Bergomi and Baratè 2020; Bergomi 2015). Persistent homology is effective for music analysis as it captures the cyclic structures of data (Zomorodian and Carlsson 2005; Cohen-Steiner, Harer, and Edelsbrunner 2007; Edelsbrunner and Harer 2009). In Tran, Park, and Jung (2021), TDA was used to analyse Korean Jung-Ak music for the first time. Jung-Ak music\textsuperscript{1} is a type of music that was played at Royal palaces or among noble communities in old Korea. It is a slow, solemn and intricately melodic music which was a representation of peaceful and calm surroundings, evoking feelings of serenity from anyone who would listen. The noblemen enjoyed and found this music graceful. Dodeuri music is one of the most popular Jung-Ak music pieces. As its name indicates, the main characteristics of Dodeuri (repeat-and-return) music are in its frequent repetition and variation patterns. To analyse such patterns, TDA,
particularly persistent homology, has been utilized in Tran, Park, and Jung (2021). Since the raw form of the music, i.e. the sequence of music notes, is not suitable for TDA, the given music is represented as a network (Bryan and Wang 2011; Liu, Tse, and Small 2010; Ren, Chazal, and Del Genio 2015), for which proper definitions of nodes and edges are provided in Tran, Park, and Jung (2021). A node is defined as a two-dimensional vector whose first component is the pitch and the second the duration of the music note. If two nodes are placed side-by-side in the music sequence, those two nodes are directly connected and the edge between those two nodes is defined as the connection. The weight of the edge is defined as the frequency of the side-by-side appearance of those two nodes. In such a way, the weight is a non-negative integer. In order to apply TDA, the notion of distance between two nodes is defined as the reciprocal of the edge weight of those two nodes if they are connected directly. If two nodes are only connected through paths containing at least two edges, the distance between those two nodes is defined by the sum of the reciprocals of the weights of edges involved between those two nodes. For uniqueness, such edges between two nodes are picked in the first path by lexicographic order among all possible paths, which has the smallest number of edges. Once the distance is defined, persistent homology is calculated and the corresponding one-dimensional barcodes are obtained. The one-dimensional barcode contains the one-dimensional hole information (Carlsson 2009), that is, one-dimensional cycles in the given graph. In Tran, Park, and Jung (2021), we analyse the cycle structure of three famous Korean old music pieces played by the haegeum, a traditional Korean string instrument, resembling a vertical fiddle with two silk strings. For Suyeonjangjigok (or Suyeonjang in short), one of the most popular pieces of Dodeuri music, a unique set of a total of 8 one-dimensional cycles was found. Figure 1 shows the first few lines of Suyeonjangjigok directly translated from Jeongganbo, the old Korean music notation. The version in the figure is a simple version of Suyeonjangjigok without ornaments. In this paper, we mainly use Suyeonjangjigok as an example for the development of the proposed method.

In Section 2, a new concept of the overlap matrix of $s$-scale will be introduced. The overlap matrix of $s$-scale provides a visualization, in matrix form, that shows how homology cycles are interconnected over the music at $s$-scale. It was shown in Tran, Park, and Jung (2021) that the interconnection between cycles is useful in understanding how the given music is composed and to classify music. In fact, the overlap matrix explains surprisingly well, and quantitatively, why Dodeuri music is different from the Taryong music, a music known as a non-Dodeuri music. In this way, the overlap matrix can be interpreted as the composition algorithm or composition principle of the considered music. The current paper is based on our assumption that the overlap matrix reveals the composition algorithm of the music considered. Upon such assumption, we propose a way of machine composition using the overlap matrix.

Machine composition with artificial intelligence (AI) techniques based on artificial neural networks is well known even to non-experts these days (Lopez-Rincon, Starostenko, and Martín 2018). There are various AI composition software packages available as well. The purpose of AI music composition algorithms based on deep neural networks is to train a machine by feeding music pieces and create artificial neural networks that can produce music similar to the input music data. These approaches are considered as blackbox optimizations. That is, how machine composes with the constructed network is not known to users and the underlying composition algorithm of the generated music pieces is, in general, not explainable.

In this paper, we present a way of machine composition that trains a machine the composition principle embedded in the given music data. Our proposed method is based on the overlap matrix explained above. As explained in detail in Section 2, the overlap matrix is a visualization method that shows how the key cycles of the music found via TDA are distributed and interconnected.

---

2 Readers can also listen to Suyeonjangjigok played by the haegeum instrument at https://www.youtube.com/watch?v=_DKo8FjL7Mg&t=461s from 0:24 to 5:24.
throughout the music flow. The main idea of the current paper is to teach a machine the compositional principles represented by the overlap matrix, expecting that the resulting algorithmically composed music mimics the input. One can simply create or design the overlap matrix and the machine generates the music with such overlap matrix as seed data. First, we will explain that the overlap matrix can be used directly to compose music algorithmically. Second we will explain how we use overlap matrices to train artificial neural networks that generate music.

The paper comprises the following sections. In Section 2, we will explain the key elements of the current paper. We first explain the music network and TDA over the constructed network. Then we give detailed mathematical properties of the overlap matrix. In Section 3, we will explain the concept of node pool, which provides the nodes used for the composition. In Section 4, we explain how we use the overlap matrix to generate music algorithmically – Algorithm A. In Section 5, we explain how we use the overlap matrix in the context of artificial neural network. First we propose three different methods that can generate the seed music using the overlap matrix in order to use towards the construction of the artificial neural networks. Then, we provide a way to train a machine with the overlap matrix and construct the corresponding artificial neural networks. In Section 6, we provide a brief concluding remark and future research questions.

2. Cycles and overlap matrix

In this section we first explain how to construct a network from the given music. The raw form of music is not suitable for TDA. We represent the given music as a graph so that TDA can be applied. As in Tran, Park, and Jung (2021), we consider Suyeonjangjigok, a Dodeuri-type monophonic music written in Jeongganbo. Jeongganbo is a unique Korean music notation similar to a
matrix. Figure 1 shows first few lines of Suyeonjangjigok translated directly from Jeongganbo. For the explanation of reading Jeongganbo, see Tran, Park, and Jung (2021). Once the music network is constructed, we apply TDA and obtain the so-called persistent barcode, together with the cycles corresponding to the bars in dimension one. Based on the barcode and cycles, we build the overlap matrix which will be used together with the cycle information and node frequency distribution for generating new music. We refer readers to Tran, Park, and Jung (2021) and Carlsson (2009) for more details of TDA through persistent homology.

2.1. Construction of music network

Consider a monophonic music piece composed of \(d\) notes and let \(\mathcal{L} = \{n_1, n_2, \ldots, n_d\}\) be the ordered sequence of notes. Each note has the information of height (pitch) and length (duration) of the sound to be played, i.e.

\[
n_i = (p_i, l_i),
\]

where \(p_i\) is the pitch of note \(n_i\) and \(l_i\) is its length. Note that by the definition of \(n_i\), it is possible that \((p_i, l_i) = (p_j, l_j)\) for \(i \neq j\). We then construct the music network \(G = (\mathcal{N}, \mathcal{E})\), where \(\mathcal{N}\) is the set of nodes and \(\mathcal{E}\) is the set of edges in \(G\). Here, \(\mathcal{N}\) is the set of distinct notes in \(\mathcal{L}\), sorted in ascending order in terms of pitch first then length. That is, \(\mathcal{N} = \{v_1, v_2, \ldots, v_q\}\), where \(q \leq d\) is the number of distinct notes in \(\mathcal{L}\) and \(v_j\) has a higher pitch than \(v_i\) or both have the same pitch but \(v_j\) has a longer length than \(v_i\) if \(j > i\). We draw an edge between two nodes \(v_i\) and \(v_j\), \(i \neq j\), if they occur adjacent in time. Let \(e_{ij} \in \mathcal{E}\) be the edge whose end points are \(v_i\) and \(v_j\). The weight of the edge \(e_{ij}, w_{ij}\), between \(v_i\) and \(v_j\) is the number of occurrences of those two nodes being adjacent in time. For two nodes \(v_i\) and \(v_j\) with \(i < j\), let \(p_{ij}\) be the path with the minimum number of edges between \(v_i\) and \(v_j\) found by Dijkstra algorithm. The distance between nodes \(v_i\) and \(v_j\), \(i < j\) is defined to be:

\[
\delta(v_i, v_j) = \sum_{e_{kl} \in p_{ij}} w_{kl}^{-1}
\]

where \(w_{kl}\) represents the weight of the edge \(e_{kl}\) and \(p_{ij} = \bigcup e_{kl}\). For the music we consider in this paper, since there is no empty Jeonggan where the music is not played, i.e. there is no isolated node, there always exists at least one path between any two distinct nodes \(v_i\) and \(v_j\) even if they are not adjacent anywhere in the music. Also, it is obvious that \(w_{kl} \geq 1\) for any edge \(e_{kl}\). Thus, the distance in (1) is well-defined. Then we form the distance matrix \(D = \{\delta_{ij}\}\) as follows:

\[
\delta_{ij} = \begin{cases} 
\delta(v_i, v_j), & i < j \\
0, & i = j \\
\delta_{ji}, & i > j 
\end{cases}
\]

2.2. TDA: barcode and cycles

We do not attempt to explain TDA here but refer readers to Carlsson (2009). The graph introduced above is defined using the distance. Consider a point cloud composed of all nodes in \(\mathcal{N}\). As all the pair-wise distances between \(v_i\) and \(v_j\) are defined, we first build a simplicial complex out of the point cloud as a Vietoris-Rips complex to compute persistent homology (see Carlsson 2009; Edelsbrunner and Harer 2009). We note that there are other approaches rather than Vietoris-Rips complex for persistent homology on a graph. Since the main purpose of the current research is to propose a machine composition algorithm, the choice of algorithm for the complexes and filtration method is not critical. Using the distance matrix \(D\), we build the corresponding Vietoris-Rips complex and barcode, for which we use the software package Javaplex (Adams and Tausz 2019).
Suyeonjang has total $d = 440$ notes composed of $q = 33$ distinct notes. Figure 2 shows the zero-dimensional (top) and one-dimensional (bottom) barcodes generated by Javaplex applied to $G$ of Suyeonjang.

In Figure 2 the horizontal axis is the filtration value $\tau$. Vertically we have multiple intervals that correspond to generators of the homology groups. In the zeroth dimension we have 33 generators that correspond to 33 components when $\tau$ is zero or small, which eventually are connected into a single component when $\tau = 1$. The 33 components are those 33 distinct nodes defined in Suyeonjang. All these components constitute a single component because of the fact that any node in the network connects at least one time with another node, which means that at most when distance $\tau = 1$ all nodes in the network are connected. On the other hand, the fact that one component is formed exactly when $\tau = 1$ implies that there exists at least a pair of nodes $\nu_i, \nu_j$ that has distance $\delta(\nu_i, \nu_j) = 1$, i.e. $\nu_i$ and $\nu_j$ are adjacent only once. In the first dimension we see 8 generators which topologically correspond to eight cycles. If we consider each of the eight cycles as a set of nodes and analyse the relation between them as a whole, it turns out that the interconnection between these cycles is related to the repetition of music melodies known as Dodeuri (Tran, Park, and Jung 2021).

For each persistence interval we use the persistence algorithm computing intervals to find a representative cycle. The method `computeAnnotatedIntervals` in Javaplex is used to find the nodes in the intervals of persistence. In the one dimensional case, the annotated intervals consist of the components in the loops generated in the process of filtration.

Figure 3 shows 8 cycles identified by TDA corresponding to 8 persistence intervals in the one-dimensional barcode of Suyeonjang. We enumerate the cycles by the order of appearance of their corresponding persistence intervals in the barcode. That is, the earlier the 1D barcode dies, the lower number is assigned to the corresponding cycle. For example, the death of cycle $i$ is earlier
Figure 3. The 8 cycles identified by TDA in Suyeonjang (Figure 7 in Tran, Park, and Jung 2021).

than the death of cycle \( j \) if \( i < j \). Note that this order is different from the order of their appearance in the actual music and can be done arbitrarily without affecting the proposed composition algorithms. In the figure, each cycle is shown with persistence interval, node information including node number, pitch and length, the latter two of which are encrypted in the circles filled with different colours and centred by Chinese letters. The Chinese letter in the centre of each filled circle corresponds to a specific pitch, and the colour of each circle illustrates the node length (see Table 1). The figure also shows edge weight (in normal size in blue), distance between nodes (in small size in blue in parentheses) and the average weight (in red in centre) which is the simple mean of all edge weights. As shown in the figure, the minimum number of nodes that constitute a cycle is 4 and the maximum number is 6. The information of corresponding notes found in all cycles is given in Table 1. For the purpose of this paper, we will use only the node
information of the cycles \( C_i, i = 1, \ldots, 8 \). More precisely, we will use the information of which nodes each cycle consists of. For example, in the case of Suyeonjang, \( C_1 = \{v_{18}, v_{20}, v_{22}, v_{27}\} \), \( C_2 = \{v_3, v_6, v_{12}, v_{18}\} \) and so on. It should be noted that we do not need detailed information about the names, pitches, or lengths of the notes for building the composition algorithms. That actual note information will be used at the finishing stage where we generate music. In the next section, we explain in detail how the cycles will be used to construct the overlap matrix, one of the key ingredients in generating music.

### 2.3. **Overlap matrices**

In this section, we provide a formal definition of the overlap matrix and its mathematical properties. From now on, let \( s \) be a positive integer.
Definition 2.1 A binary matrix is a matrix whose entries are either 0 or 1.

Definition 2.2 A binary matrix \( M_{k \times d}^s = \{m_{ij}^s\} \) is said to belong to \( s \)-scale if for all \( i = 1, \ldots, k \) we have that \( m_{ij}^s = 1 \) if and only if there exist nonnegative integers \( t, l \) satisfying \( t + l \geq s - 1 \) such that

\[
m_{ij}^s = m_{i,j-l}^s = \cdots = m_{i,j+t-1}^s = m_{i,j+t}^s = 1.
\]

Notice that there are \( t + l + 1 \) entries from \( m_{i,j-l}^s \) to \( m_{i,j+t}^s \):

Thus, a binary matrix \( M_{k \times d}^s = \{m_{ij}^s\} \) belongs to \( s \)-scale if and only if on each row of \( M_{k \times d}^s \) any entry equal to 1 is in a consecutive sequence of at least \( s \) columns that equal 1.

Unless otherwise mentioned, let \( O \) be a musical piece composed of \( d \) notes in the order \( L = \{n_1, \ldots, n_d\} \) and assume that the barcode for it in the first dimension consists of \( k \) generators which topologically correspond to \( k \) cycles, \( C_1, \ldots, C_k \). We define the \( s \)-scale binary and integer overlap matrices for \( O \) as follows.

Definition 2.3 Matrix \( M_{k \times d}^s = \{m_{ij}^s\} \) is called the \( s \)-scale binary overlap matrix for \( O \) if it satisfies the following conditions

\[
m_{ij}^s = \begin{cases} 
1, & \text{if } \exists t, l \geq 0 \text{ satisfying } t + l \geq s - 1 \text{ such that } n_{j-l}, n_{j-l-1}, \ldots, n_j, \ldots, n_{j+t-1}, n_{j+t} \in C_i, \\
0, & \text{otherwise}, 
\end{cases}
\]

for all \( i = 1, \ldots, k; j = 1, \ldots, d \).

Definition 2.4 Matrix \( M_{k \times d}^s = \{m_{ij}^s\} \) is called the \( s \)-scale integer overlap matrix for \( O \) if it satisfies the following conditions

\[
m_{ij}^s = \begin{cases} 
n_j, & \text{if } \exists t, l \geq 0 \text{ satisfying } t + l \geq s - 1 \text{ such that } n_{j-l}, n_{j-l-1}, \ldots, n_j, \ldots, n_{j+t-1}, n_{j+t} \in C_i, \\
0, & \text{otherwise}, 
\end{cases}
\]

for all \( i = 1, \ldots, k; j = 1, \ldots, d \).

Remark 2.5 Given the \( s \)-scale integer overlap matrix for a music \( O \), its corresponding \( s \)-scale binary overlap matrix is uniquely determined and easily obtained by replacing nonzero entries in the integer overlap matrix with 1. The converse is not true.

Proposition 2.6 If \( M_{k \times d}^s = \{m_{ij}^s\} \) is the \( s \)-scale binary overlap matrix for \( O \), then the followings hold:

(i) \( M_{k \times d}^s \) is a binary matrix belonging to \( s \)-scale.
(ii) \( m_{ij}^s = 1 \) implies that \( n_j \in C_i \).
(iii) If \( n_j \notin C_i \) then \( m_{ij}^s = 0 \).

Remark 2.7 The converse of (ii) and (iii) is not necessarily true.
Proof of Proposition 2.6  Let $M^s_{k \times d} = \{m^s_{ij}\}$ be the $s$-scale binary overlap matrix for $O$. It is easy to see that (ii) and (iii) are straightforward from Definition 2.3. To prove (i), since $m^s_{ij}$ is either 0 or 1 for all $i, j$, hence $M^s_{k \times d}$ is a binary matrix. It remains to show that on each row of $M^s_{k \times d}$ any entry equal to 1 stays in a consecutive sequence of at least $s$ columns that equal to 1.

Let $m^s_{ij} = 1$. By Definition 2.3, there exist $t, l \geq 0$ satisfying $t + l \geq s - 1$ such that $n_j - l, n_j - l - 1, \ldots, n_j, \ldots, n_j + t - 1, n_j + t \in C_i$. In other words, there exists a consecutive sequence of at least $s$ notes including $n_j$ in $O$ that belong to $C_i$. Now, in turn, $m^s_{ij} = 1$ since there exist $u = 0, r = t + l \geq 0$ satisfying $u + r = t + l \geq s - 1$ such that $n_j - l - u (= n_{j-l}), n_j - l - u - 1 (= n_{j-l-1}), \ldots, n_j - l + r - 1 (= n_{j-l+r-1}), n_j - l + r (= n_{j+l}) \in C_i$. Analogously, we can show that $m^s_{ij} = \cdots = m^s_{i,j+t} = 1$. Thus, $M^s_{k \times d} = \{m^s_{ij}\}$ belongs to $s$-scale. ■

Remark 2.8  Given a music piece $O$, the $s$-scale integer overlap matrix (and thus the $s$-scale binary overlap matrix as well, by Remark 2.5) $M^s_{k \times d}$ for $O$ is uniquely determined.

In Algorithm 1 we give an algorithm to compute the $s$-scale integer overlap matrix $M^s_{k \times d}$ for a given music $O$.

Algorithm 1  Algorithm to compute the $s$-scale integer overlap matrix $M^s_{k \times d}$

Given $O$ and cycle information $C_1, \ldots, C_k$.
Set $M^s_{k \times d} = \emptyset_{k \times d}$ (zero matrix).
Let $j = 1$.
For each row $i = 1, \ldots, k$, repeat the following until $j = d$.

Step 1: Find
\[ q = \arg \min_\beta \{ j \leq \beta \leq d : n_\beta \in C_i \}. \] (2)
if (2) has no solution then
break
else
Go to Step 2.

Step 2: Find
\[ r = \arg \min_\gamma \{ q < \gamma \leq d : n_\gamma \notin C_i \}. \] (3)
if (3) has no solution then
if $d - q \geq s - 1$ then
\[ m^s_{ij} = n_j, \quad j = q, \ldots, d. \]
break
else if (3) has a solution and $r - q \geq s$ then
\[ m^s_{ij} = n_j, \quad j = q, \ldots, r - 1. \]

Set $j = r$ and come back to Step 1.

In the case of Suyeonjang which is composed of $d = 440$ notes and has in total eight cycles, the 4-scale binary overlap matrix $M^4_{8 \times 440}$ is displayed in Figure 4. In Figure 4 the horizontal axis represents the time sequence of the music and the vertical axis represents the cycle number, from $C_1$ to $C_8$. The zero entries are left blank and the entries that equal 1 are coloured. Notice that at $s$-scale, each coloured block is of length at least $s$ entries.
According to the definition of the binary overlap matrix $M_{k \times n}^s$, the entries $m_{ij}^s$ can be either 0 or 1 depending on whether there exists a consecutive sequence of at least $s$ notes containing the note $n_j$ of the music that belongs to the cycle $C_i$ or not. A zero entry $m_{ij}^s = 0$ does not necessarily mean that the note $n_j$ does not belong to the cycle $C_i$. It can be the case that the note $n_j$ belongs to the cycle $C_i$ but the consecutive sequence of notes containing note $n_j$ belonging to cycle $C_i$ is not long enough on the scale being considered. On the other hand, if $m_{ij}^s = 1$ we can say for sure that the note $n_j$ of the music belongs to cycle $C_i$. Thus, the $j$th column of the matrix $M_{k \times n}^s$ provides the information of how many cycles, as well as which ones, are overlapping “at $s$-scale” at this point. This is indeed the reason why we call it the overlap matrix. For example, let us take a close look at the 4-scale binary overlap matrix $M_{8 \times 440}^4$ for Suyeonjang. The first column of $M_{8 \times 440}^4$, which is

\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}^T
\]

implies that the first node $n_1$ does not belong to any cycle, or in fact it does belong to some cycle but at least one of the notes $n_2, n_3, n_4$ does not belong to that cycle. On the other hand, the 25th column of $M_{8 \times 440}^4$ which is

\[
\begin{bmatrix}
0 & 1 & 0 & 0 & 1 & 1 & 1 & 0
\end{bmatrix}^T,
\]

implies that the 25th note belongs to at least 4 cycles: $C_2, C_5, C_6, C_7$. As with the first note $n_1$, it is unclear whether or not the 25th note belongs to $C_1, C_3, C_4, C_8$.

**Definition 2.9** A cycle $C_i$ is said to **survive** at note $n_j$ at $s$-scale if $m_{ij}^s = 1$.

Denote by $S_j$ the set of cycles which survive at note $n_j$ at $s$-scale. It is obvious that

\[
S_j = \begin{cases} 
\emptyset, & \text{if } m_{ij}^s = 0, \forall i = 1, \ldots, k, \\
\{C_i | t \in I_j\}, & \text{if } m_{ij}^s = \chi_I(i),
\end{cases}
\]
for \( j = 1, \ldots, d \). Here, \( I_{j} \) is an index set, \( I_{j} \subset \{1, 2, \ldots, k\} \) and \( \chi_{I_{j}} \) is the indicator function \( \chi_{I_{j}} : \{1, 2, \ldots, k\} \rightarrow \{0, 1\} \) such that

\[
\chi_{I_{j}}(x) = \begin{cases} 
1, & \text{if } x \in I_{j}, \\
0, & \text{if } x \notin I_{j}.
\end{cases}
\]

Indeed, \( I_{j} \) is the set of all the indices \( i \) where \( m_{ij} = 1 \) for a given \( j \). In the case of Suyeonjang we have, for example, that \( S_{1} = \emptyset \) and \( S_{25} = \{C_{2}, C_{5}, C_{6}, C_{7}\} \).

3. Node pool

First consider the algorithmic composition given by following the pattern of the \( s \)-scale binary overlap matrix of the music under consideration. Notice that at \( s \)-scale, cycles obtained from the music by TDA only overlap at certain notes (see the binary overlap matrix of Suyeonjang in Figure 4), and sometimes they do not overlap at all (see the binary overlap matrix of a music called Taryong (Tran, Park, and Jung 2021) in Figure 5). Also, there are many notes in the music where there is no cycle surviving at \( s \)-scale. At those notes, there is more freedom of node choice. We will build up a so-called node pool (denoted by \( \mathcal{P} \)), from which we choose a node for each of those places where there is no cycle surviving at the considered \( s \)-scale. The node pool is a collection of nodes that satisfy the node frequency distribution. Let us take Suyeonjang as an example again. According to our node definition, Suyeonjang is of length 440 notes that consist of 33 distinct nodes. The node frequency distribution of Suyeonjang is shown in Table 2, where the node frequency distributions of two additional music pieces, Songkuyeo and Taryong are also shown.

Imagine that we have a set of 440 nodes in which, for example, node \( n_{18} \) has 76 copies, \( n_{6} \) has 57 copies and so on. Then the chance of randomly picking up the node \( n_{j} \) is its probability:

\[
\text{Node probability of } n_{j} = \frac{\text{Node frequency of } n_{j}}{\text{Total number of node frequencies}}
\]

In general, consider music of length \( d \) notes, \( \mathcal{L} = \{n_{1}, \ldots, n_{d}\} \). Let \( \{v_{1}, \ldots, v_{q}\} \) be the set of its distinct nodes as before, with node frequencies \( f_{1}, \ldots, f_{q} \), respectively.
Figure 5. The 4-scale binary overlap matrix for Taryong. The zero entries are left blank and the entries that equal 1 are coloured.

Then $\mathcal{P}$ for this music is a multiset made of all the nodes $v_1, \ldots, v_q$, where node $v_1$ appears in the set $f_1$ times, $v_2$ appears $f_2$ times and so on.

$$\mathcal{P} = \{v_1, \ldots, v_1, v_2, \ldots, v_2, \ldots, v_q, \ldots, v_q\}$$

Notice that

$$f_1 + \cdots + f_q = d.$$ 

Thus, $\mathcal{P}$ contains exactly $d$ nodes made up from the $q$ distinct nodes from the music. In other words, $\mathcal{P}$ is a permutation of the set $\mathcal{L} = \{n_1, \ldots, n_d\}$. The chance of picking a node from $\mathcal{P}$ is equal to its probability, $p(v_j) = \frac{f_j}{d}$.

4. Algorithmic composition – Algorithm A

The overlap matrices show how the cycles are distributed and interconnected across the music. For Suyeonjang, the interconnection between cycles – visualized by the overlap matrix in Figure 4 – is found to be related to the unique structure of the Dodeuri pattern (Tran, Park, and Jung 2021). The method for creating new music algorithmically is shown in the flowchart in Figure 6.

For the preparation we need the node pool, cycles and binary overlap matrix. Given the music that we consider (Suyeonjang for example), it is straightforward to get the node frequency distribution and then build up the node pool. On the other hand, from the seed music we can construct the music network and then find the distance matrix. Next, the distance matrix is plugged into Javaplex and by using TDA tools we find the barcode and corresponding cycle information. The cycles are then used to obtain the binary overlap matrix.

Given music of length $d$ notes in Jeongganbo, $\mathcal{L} = \{n_1, \ldots, n_d\}$, our goal is to algorithmically create new music of the same length $d$, $\mathcal{L}' = \{n'_1, \ldots, n'_d\}$, such that the pattern of the binary overlap matrix of a given $s$-scale of the seed music is strictly followed, and the new music sounds
Table 2. Frequency by node for Suyeonjang, Songkuyeo, and Taryong (Tran, Park, and Jung 2021).

| Rank | Suyeonjang | Songkuyeo | Taryong |
|------|------------|-----------|---------|
| 1    | n_{18}     | n_{20}    | n_{16}  |
| 2    | n_{6}      | n_{31}    | n_{11}  |
| 3    | n_{11}     | n_{13}    | n_{13}  |
| 4    | n_{22}     | n_{26}    | n_{26}  |
| 5    | n_{1}      | n_{8}     | n_{29}  |
| 6    | n_{20}     | n_{18}    | n_{31}  |
| 7    | n_{27}     | n_{4}     | n_{28}  |
| 8    | n_{3}      | n_{33}    | n_{3}   |
| 9    | n_{28}     | n_{6}     | n_{18}  |
| 10   | n_{12}     | n_{16}    | n_{15}  |
| 11   | n_{16}     | n_{25}    | n_{12}  |
| 12   | n_{26}     | n_{19}    | n_{22}  |
| 13   | n_{31}     | n_{24}    | n_{10}  |
| 14   | n_{2}      | n_{27}    | n_{32}  |
| 15   | n_{4}      | n_{28}    | n_{17}  |
| 16   | n_{23}     | n_{32}    | n_{20}  |
| 17   | n_{9}      | n_{2}     | n_{4}   |
| 18   | n_{10}     | n_{15}    | n_{9}   |
| 19   | n_{5}      | n_{7}     | n_{4}   |
| 20   | n_{8}      | n_{11}    | n_{14}  |
| 21   | n_{13}     | n_{12}    | n_{2}   |
| 22   | n_{0}      | n_{14}    | n_{5}   |
| 23   | n_{7}      | n_{17}    | n_{7}   |
| 24   | n_{17}     | n_{21}    | n_{8}   |
| 25   | n_{19}     | n_{23}    | n_{19}  |
| 26   | n_{21}     | n_{3}     | n_{21}  |
| 27   | n_{25}     | n_{9}     | n_{27}  |
| 28   | n_{29}     | n_{3}     | n_{33}  |
| 29   | n_{30}     | n_{0}     | n_{34}  |
| 30   | n_{32}     | n_{10}    | n_{35}  |
| 31   | n_{14}     | n_{36}    | n_{1}   |
| 32   | n_{15}     | n_{34}    | n_{10}  |
| 33   | n_{24}     | n_{1}     | n_{23}  |
| 34   | n_{5}      | n_{24}    | n_{1}   |
| 35   | n_{22}     | n_{25}    | n_{1}   |
| 36   | n_{29}     | n_{1}     | n_{30}  |
| 37   | n_{30}     | n_{38}    | 1       |
| 38   | n_{36}     | 1         | 1       |
| 39   | n_{37}     | 1         | 1       |
| 40   | n_{39}     | 1         | 1       |

similar to the seed music, containing particular patterns. Below we explain how to choose each note $n'_j, j = 1, \ldots, d$ of the new music.

Assume that following the process described above we found $k$ cycles $C_1, C_2, \ldots, C_k$. Let $S_j$ be the set of cycles which survive at note $n_j$ at $s$-scale, as defined in (4). Across the music, at each note $n_j$ either there are some cycles surviving at $s$-scale ($S_j \neq \emptyset$) or none of the cycles survives ($S_j = \emptyset$). Denote by $I_j$ the set of nodes belonging to the intersection of those cycles surviving at $s$-scale that overlap at note $n_j$:

$$I_j = \left\{ n_i | n_i \in \bigcap_{C_i \in S_j} C_i \right\}.$$  

If some of the cycles survive at note $n_j$ at $s$-scale, i.e. $S_j \neq \emptyset$, then the new note $n'_j$ is randomly chosen from the intersection of those cycles:

$$n'_j = \text{random choice from } I_j \quad \Leftrightarrow \quad S_j \neq \emptyset, j = 1, \ldots, d.$$
Otherwise, if none of the \( k \) cycles survives at note \( n_j \) at \( s \)-scale, i.e. \( S_j = \emptyset \), we randomly pick up a node from node pool \( \mathcal{P} \), with or without a constraint depending on whether or not there exist cycles surviving at node \( n_{j-1} \) and node \( n_{j+1} \) at \( s \)-scale as follows:

\[
n_j' = \begin{cases} 
    \text{random choice from } \mathcal{P} & \iff S_{j-1} = S_j = S_{j+1} = \emptyset, \\
    \text{random choice from } \mathcal{P} \setminus I_{j-1} & \iff S_{j-1} \neq \emptyset, S_j = S_{j+1} = \emptyset, \\
    \text{random choice from } \mathcal{P} \setminus I_{j+1} & \iff S_{j-1} = S_j = \emptyset, S_{j+1} \neq \emptyset, \\
    \text{random choice from } \mathcal{P} \setminus (I_{j-1} \cup I_{j+1}) & \iff S_{j-1} \neq \emptyset, S_j = \emptyset, S_{j+1} \neq \emptyset,
\end{cases}
\]

for \( j = 1, \ldots, d \). It is easy to see that in this way, we strictly follow the pattern of the binary overlap matrix.

**Remark 4.1** After getting the new music \( \mathcal{O}' \), we can apply the process of constructing the music network to it, followed by using TDA tools, to find its corresponding binary and integer overlap matrices. We observe that the new music \( \mathcal{O}' \) generated by the procedure in Figure 6, although it sounds nice, neither necessarily has the same number of cycles nor does it necessarily reflect the overlap pattern of the original seed music \( \mathcal{O} \). In other words, both the binary and integer overlap matrices of \( \mathcal{O}' \) can be very different from those of \( \mathcal{O} \). This is illustrated in the following examples where we use Suyeonjang as the seed music. We provide here only the binary overlap matrices since it is obvious that if two musics have different binary overlap matrices then their integer overlap matrices are also different.

**Example 4.2** Figure 7 shows an example of music generated from Suyeonjang which has only four cycles.

**Example 4.3** Figure 8 shows another piece of music generated from Suyeonjang which also has only four cycles. The overlap pattern of this music is quite different from that of the music shown in Example 4.2.
Example 4.4 Figure 9 shows music generated from Suyeonjang which has six cycles. Although this music has more cycles than the music shown in Examples 4.2 and 4.3, it obviously has less cycles than Suyeonjang. It is not exactly the same as the binary overlap matrix of Suyeonjang but shows some similarity in the overlapping sense.
5. Creating new music with an artificial neural network – Algorithm B

An alternative approach to generating new music is to use the artificial neural network described in the following section.

5.1. Generating seed overlap matrix

Given the $s$-scale integer overlap matrix for Suyeonjang $M_{k\times d}^s$, our first goal is to generate an integer overlap matrix $\tilde{M}_{k\times d}$ that has the same size and similar pattern as $M_{k\times d}^s$, which will be used as a seed overlap matrix towards the artificial neural network.

For the given music, there could be various ways of generating a seed overlap matrix of the same size as the overlap matrix of the given music, having similar patterns as those generated from the given music. Below we introduce three algorithms for generating a seed integer overlap matrix $\tilde{M}_{k\times d}$ from $M_{k\times d}^s$.

The common strategy of the following three algorithms is to generate a binary overlap matrix that has the same size and mimics the overlapping pattern of the given integer overlap matrix first, then convert it to an integer overlap matrix. The specificities of each algorithm are as follows.

Row-by-row method: Overlap Matrix Algorithm #1 is the row-by-row approach. That is to say, the first binary row is determined based on the number of blocks of consecutive nonzero entries of the first row of the given overlap matrix. Next, using the overlapping pattern and node information, the second binary row is generated so that it overlaps or does not overlap the first row depending on whether the first and the second rows of the given overlap matrix overlap or not. This process is continued for all rows. As a result we get a binary overlap matrix that has the same size, same frequency of blocks of consecutive nonzero entries and preserves the overlapping pattern of the given overlap matrix. To convert the generated binary overlap matrix to an integer overlap matrix, we convert column by column using the node information.

Element-by-element method: Overlap Matrix Algorithm #2 is the element-by-element approach. That is, nonzero entries in given overlap matrix are first replaced with 1 to generate a binary overlap matrix that has exactly the same overlapping pattern as the given overlap matrix. Then, from the node information, entries equal to 1 in the generated binary overlap matrix are converted back to integer numbers. This algorithm is the simplest one if we just want to get a new but very similar overlap matrix.

Column-by-column method: Overlap Matrix Algorithm #3 is the column-by-column approach. After converting the given integer overlap matrix to a binary overlap matrix, we collect all kinds of columns in it and generate a new overlap matrix according to the frequencies of the columns. In this way we automatically preserve the overlapping pattern of the given overlap matrix, while still have some flexibility in the number of blocks as well as in the length of each block of consecutive nonzero entries. The difficulty in this algorithm is that a new column has to be carefully chosen so that the number of consecutive nonzero entries is not less than $s$, in order to satisfy the definition of an overlap matrix of $s$-scale.

5.2. Training data set

In order to input the seed overlap matrix into the artificial neural network for generating new music, we need to construct an optimized artificial neural network corresponding to the given music. Such a network can be obtained by training a machine with the given music. Note that the number of samples of the given music is not large. For example, if we consider a single music piece, e.g. Suyeonjang, the number of the input music samples with which we train the machine is simply one. That is, the amount of data required for training is actually very small. In this
Algorithm 2 Overlap Matrix Algorithm #1

**Step 1:** Find the frequency $f_i$ of blocks of consecutive nonzero entries in row $i$ of $M_{k \times d}^s$, $i = 1, \ldots, k$.

**Step 2:** For each cycle $C_i$, identify the set of cycles that do not overlap $C_i$. Let $S_i$ be the set of all indices $j$ of nodes $n_j$ that constitute those cycles.

**Step 3:** Let $B_{k \times d}^s$ be a $k \times d$ zero matrix. Then for each $i$ randomly pick up $f_i$ indices $j_1, \ldots, j_{f_i}$ that are not in $S_i$ and set $\sim b_{i,d,p} = 1$, $p = 1, \ldots, f_i$; $q = 0, 1, \ldots, s - 1$. Repeat this for all $i = 1, \ldots, k$. This step generates $B_{k \times d}^s$ as a binary overlap matrix.

**Step 4:** For each non-zero column $b_j$ of $B_{k \times d}^s$ let $U_j$ be the set of all nodes that constitute those cycles that correspond to nonzero entries in $b_j$. Then replace all entries equal to 1 in $b_j$ by a random node index $n_j$ chosen from $U_j$. This step converts $B_{k \times d}^s$ to an integer overlap matrix $\tilde{M}_{k \times d}^s$.

Algorithm 3 Overlap Matrix Algorithm #2

**Step 1:** Replace nonzero entries in $M_{k \times d}^s$ with 1 to convert $M_{k \times d}^s$ to a binary overlap matrix $B_{k \times d}^s$.

**Step 2:** Assume that $C_i = \{n_1^i, n_2^i, \ldots, n_{n_i}^i\}, i = 1, \ldots, k$. Replace entries equal to 1 in row $i$ of $B_{k \times d}^s$ by a random node $n_j^i$ belonging to $C_i$ by the method `random.choice(C_i)`.

**Step 3:** Repeat step 2 for all rows in $B_{k \times d}^s$.

Algorithm 4 Overlap Matrix Algorithm #3

**Procedure 1:** Generate a binary overlap matrix $\tilde{B}_{k \times d}^s$.

**Step 1:** Convert $M_{k \times d}^s$ to a binary overlap matrix $B_{k \times d}^s$.

**Step 2:** Choose a column for $\tilde{B}_{k \times d}^s$ by a random selection from the set of all distinct columns in $B_{k \times d}^s$ according to the frequencies of the columns by the method `random.choices(C,F)`, where $C = [c_1, \ldots, c_l]$ denotes the list of all distinct columns in $B_{k \times d}^s$ and $F = [f_1, \ldots, f_l]$ is the list of their corresponding frequencies.

**Step 3:** Select a number for how many times the column selected in step 2 will be repeated in $\tilde{B}_{k \times d}^s$. This number is chosen by `random.choices` from the set of repeated columns according to the repeating frequencies.

**Step 4:** The next column is also chosen by the method `random.choices` but from the set of all distinct columns in $B_{k \times d}^s$ that are adjacent to the column selected in step 2. The resulting column in step 4 after placing in $\tilde{B}_{k \times d}^s$ should satisfy the continuous condition, meaning that any entry equal to 1 in $\tilde{B}_{k \times d}^s$ should stay in a consecutive sequence of at least $s$ columns that equal to 1.

**Step 5:** Repeat step 2-4 if necessary until the frequencies of blocks of consecutive entries equal to 1 on each row of $B_{k \times d}^s$ and $\tilde{B}_{k \times d}^s$ are similar.

**Procedure 2:** Convert $\tilde{B}_{k \times d}^s$ to an integer overlap matrix $\tilde{M}_{k \times d}^s$.

**Step 1:** Find all kinds of columns that $M_{k \times d}^s$ has and save them in a variable named `col_choice`.

**Step 2:** For each binary column $b_j$ in $\tilde{B}_{k \times d}^s$, find all columns in `col_choice` that have the same indices of nonzero entries as $b_j$, then randomly choose one of them, say $\tilde{m}_j$, to add to $\tilde{M}_{k \times d}^s$. 

section we use a simple method of \textit{periodic extension} of the given music to generate more music for training. In our future research, however, we need to investigate more general approaches to address the issue of the number of the input music samples. Periodic extension is a reasonable approach reflecting the Dodeuri music patterns.

Let $M_{k \times d}$, which from now on will be denoted as $M$, be the integer overlap matrix of the given music and $\mathcal{L}$ be the ordered sequence of notes in the music. We augment seed music by shifting the original music one space at a time:

$$
\begin{align*}
M^{(i)} &= \begin{bmatrix} M & i \end{bmatrix}_{i:i+d-1}, \\
\mathcal{L}^{(i)} &= \begin{bmatrix} \mathcal{L} \end{bmatrix}_{i:i+d-1},
\end{align*}
$$

where $\bar{M} = [M \ M]$, $\bar{\mathcal{L}} = \begin{bmatrix} \mathcal{L} \end{bmatrix}$, $d$ is the length of a seed music sample and $i = 1, \ldots, d$. For illustration purposes, let us use Suyeonjang as the seed music. As explained above, for this case we have $d = 440$ and $k = 8$ for Suyeonjang. Here, we think of $\mathcal{L}$ as a vector whose elements are indices of notes. Let $\mathcal{X} = \{\begin{bmatrix} \bar{M} & i \end{bmatrix}_{i=1}^{440}\}$ and $\mathcal{Y} = \{\begin{bmatrix} \bar{\mathcal{L}} & i \end{bmatrix}_{i=1}^{440}\}$. Figure 10 illustrates the construction of our dataset.

We design a neural network $f_\theta$ such that $f_\theta(M') = y'$ where $M' \in \mathcal{X}$ and $y' \in \mathcal{Y}$. Note that $f_\theta$ only satisfies $f_\theta(M') = y'$ in the optimization sense.

5.3. \textit{Construction of a music generation network $f_\theta$}

For the construction of $f_\theta$ introduced above, we seek a set of parameters $\theta^*$ that maximizes the probability of the real music flow $\mathcal{L}$ given the integer overlap matrix $M_{k \times d}^i$:

$$
\theta^* = \arg \max_{\theta} \sum_{(\mathcal{L}^{(i)}, M^{(i)})} \log p \left( \mathcal{L}^{(i)} \mid M^{(i)} \right),
$$

where $\theta$ is a set of parameters of our model, $p$ is the conditional probability distribution and $(\mathcal{L}^{(i)}, M^{(i)})$ is the $i$th pair of the real music flow and its corresponding integer overlap matrix induced by $\mathcal{L}$ and $M_{k \times d}^i$. We model the conditional probability distribution $p$ with a multi-layer perceptron (MLP), but one can use any nonlinear function. MLPs are a sequence of affine transformations followed by element-wise nonlinearity. Let $f^{(i)}$ be the $i$th hidden layer of a MLP and
let \( a^{(l-1)} \in \mathbb{R}^{d^{(l-1)}} \) be an input vector of \( f^{(l)} \) where \( d^{(l-1)} \) is the dimensionality of \((l-1)\)th hidden layer. The output of \( f^{(l)} \) is:

\[
f^{(l)}(a^{(l-1)}) = \sigma \left( W^{(l)}a^{(l-1)} + b^{(l)} \right),
\]

where \( \sigma \) is an activation function, \( W^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}} \) and \( b^{(l)} \in \mathbb{R}^{d^{(l)}} \) are a learnable weight matrix and bias vector, respectively. A nonlinear function \( \sigma \) is applied to each element of a vector. Each hidden layer can use a different activation function. Typical choices for \( \sigma \) are sigmoid function, hyperbolic tangent function and rectified linear unit (ReLU). Then, a MLP \( f_\theta \) is the composition of hidden layers \( f^{(l)} \):

\[
f_\theta = f^{(L)} \circ f^{(L-1)} \circ \ldots \circ f^{(1)},
\]

\[
\theta = \{ W^{(l)}, b^{(l)} : l = 1, \ldots, L \},
\]

where \( L \) is the number of hidden layers and \( a^{(0)} = x \) is an input vector of \( f_\theta \). For MLPs, the number of hidden layers, \( L \), and the dimensionality of each hidden layer, \( d^{(l)} \), are hyper-parameters to be determined.

In general, MLPs take a vector as an input, while our input is a matrix. The simplest way to feed a matrix into a MLP is to flatten it to one dimensional vector. Our model \( f_\theta \) takes the flattened vector of \( M^{(i)} \) and outputs the \( d \) probability distributions over \( q \) distinct notes. We note that each element of \( L^{(i)} \) is the node index so that we apply one-hot encoding to it. Hence, we generate the corresponding \( d \times q \) matrix \( L^{(i)} \) such that \( L^{(i)}_{jk} = 1 \) if \( j \)th note is equal to \( v_k \), and \( L^{(i)}_{jk} = 0 \) otherwise. For our network, the output of the last hidden layer \( a^{(L)} \) is a \( dq \)-dimensional vector. Then, we reshape \( a^{(L)} \) into the \( d \times q \) matrix \( \hat{L}^{(i)} \), and takes the softmax function over each row. The output of our model for \( M^{(i)} \) is as follows.

\[
\hat{L}^{(i)}_{jk} = \frac{\exp \left( a^{(L)}_{q(j-1)+k} \right)}{\sum_{l=1}^{q} \exp \left( a^{(L)}_{q(j-1)+l} \right)}.
\]

\( \hat{L}^{(i)} \) can be interpreted as the probability that the \( j \)th note in the generated music is \( v_k \). Then, a set of parameters of our model is updated towards minimizing the cross entropy loss between output probability distributions and the real music:

\[
\text{CrossEntropyLoss} \left( \hat{L}^{(i)}, L^{(i)} \right) = -\frac{1}{d} \sum_{j=1}^{d} \sum_{k=1}^{q} L^{(i)}_{jk} \log \hat{L}^{(i)}_{jk}.
\]

Figure 11 shows the architecture of our model.

To evaluate our model, we generated Suyeonjang-style music samples using our model. The length \( d \) of Suyeonjang is 440 and it has \( q = 33 \) nodes and \( k = 8 \) cycles. We used the 4-scale binary overlap matrix. We obtained 440 data from the augmentation in (5) and used 70% of them for training and the rest for evaluation. The detailed architecture of a MLP is shown in Table 3. We optimized (7) with respect to the model’s parameters using an Adam optimizer with learning rate 0.001 over 500 epochs.
Figure 11. The architecture of our model for Suyeonjang. We use a MLP with two hidden layers each of which has 440 dimensionality. The output is the 440 probability distributions over 33 distinct nodes. The learnable parameters are updated towards minimizing the cross entropy loss.

| $l$ | $d^{(l)}$ | $\sigma$ |
|-----|-----------|----------|
| 1   | 440       | ReLU     |
| 2   | 440       | ReLU     |
| 3   | 440 × 33  | Softmax in (6) |

5.4. Examples

We generated musical pieces with Algorithm A and Algorithm B. For Algorithm B, we used Overlap Matrix Algorithm #1. Figure 12 shows the original Suyeonjang for Haegeum (top), one of the generated musical pieces with Algorithm A (middle), and one of the generated musical pieces with Algorithm B (bottom). The musical pieces in the middle and bottom are randomly selected from the automatically generated music pool.

6. Conclusion

In this paper we used topological data analysis, overlap matrices and artificial neural network approaches for machine composition trained with Korean music samples, particularly from Dodeuri music. Using the concept of training the composition principle, we could generate similar music pieces to Dodeuri music. Although no comparative study between the original and generated music pieces is provided in this paper, professional players who played the generated music pieces in the YouTube links provided above reported that the generated music pieces are well structured with many similarities to the original music. Although the proposed method provides a framework of machine composition of Korean music, there are several issues that need further rigorous investigations. First of all, we will need to analyse the overall structures of the generated music through Algorithm A and Algorithm B, compare them with the original pieces, and study its musical implications. More specifically, we need to analyse how the music pieces generated by the proposed algorithms are different from the original samples. For example, we may want to investigate if the generated scores also belong to the same category of the original music and if they sound similar, etc. Second, the current research considered only limited aspects of Korean music reflected in the overlap matrix, but for a full consideration, we will need

3 Some of the generated music pieces were played in June and July, 2021. Readers can listen to those using the following YouTube links: https://www.youtube.com/watch?v=_DKo8FjL7Mg&t=461s (June 5, 2021) and https://www.youtube.com/watch?v=AxXXKoFRIqIQ&t=751s (July 29, 2021). The original music for Haegeum is played from 0:24 to 5:24 in the first link and 0:10 to 4:52 in the second link.

4 Private communications
to consider other unique characteristics of Korean music such as metre, ornamenting symbols, Sikimse, \(^5\) etc. Third, it would be interesting to extend the application of overlap matrices to harmonic and polyphonic cases. Also, the current research used a rather simple periodic extension method for generating the training data set from the given seed music. Our future work will conduct a study on how to provide training data when the number of considered music pieces is small. These should be fully considered for the construction of more generalized machine composition of Korean music.

**Disclosure statement**

No potential conflict of interest was reported by the author(s).

**Funding**

This work was supported by National Research Foundation of Korea [grant number 2018R1D1A1A02086097 and [grant number 2021R1A2C3009648].

---

\(^5\) Sikimse is a unique technique of Korean music that variates the given note by vibrating, sliding, breaking, etc.
References

Adams, Henry, and Andrew Tausz. 2019. “JavaPlex TUTORIAL.” Github appliedtopology/javaplex.

Bergomi, Mattia. 2015. “Dynamical and Topological Tools for (Modern) Music Analysis.” PhD Thesis. Universite Pierre et Marie Curie, Paris.

Bergomi, Mattia G., and Adriano Baratè. 2020. “Homological Persistence in Time Series: An Application to Music Classification.” Journal of Mathematics and Music 14 (2): 204–221. doi:10.1080/17459737.2020.1786745.

Bigo, Louis, Moreno Andreatta, Jean-Louis Giavitto, Olivier Michel, and Antoine Spicher. 2013. “Computation and Visualization of Musical Structures in Chord-Based Simplicial Complexes.” Vol. 7937, 06.

Bryan, Nicholas, and Ge Wang. 2011. “Musical Influence Network Analysis and Rank of Sample-Based Music.” 01, 329–334.

Carlsson, Gunnar. 2009. “Topology and Data.” Bulletin of the American Mathematical Society 46: 255–308. doi:10.1090/S0273-0979-09-01249-X.

Cohen-Steiner, D., J. Harer, and H. Edelsbrunner. 2007. “Stability of Persistence Diagrams.” Discrete and Computational Geometry, 37103–37120. https://doi.org/10.1007/s00454-006-1276-5.

Edelsbrunner, Herbert, and John L. Harer. 2009. Computational Topology: An Introduction. Duke University, Durham, North Carolina: American Mathematical Society.

Liu, Xiao Fan, Chi K. Tse, and Michael Small. 2010. “Complex Network Structure of Musical Compositions: Algorithmic Generation of Appealing Music.” Physica A: Statistical Mechanics and Its Applications 389 (1): 126–132. https://www.sciencedirect.com/science/article/pii/S0378437110006827.

Lopez-Rincon, Omar, Oleg Starostenko, and Gerardo Ayala-San Martin. 2018. “Artificial Music Composition Based on Artificial Intelligence: A Survey.” In 2018 International Conference on Electronics, Communications and Computers (CONIELECOMP), 187–193.

Ren, Iris Yuping, Frederic Chazal, and Charo I. Del Genio. 2015. “Topological Data Analysis on Music Networks.” M2 Project Report, University of Warwick.

Tran Mai Lan, Changbom Park & Jae-Hun Jung. 2023. “Topological Data Analysis of Korean music in Jeongganbo: A Cycle Structure.” Journal of Mathematics and Music. doi:10.1080/17459737.2022.2164626.

Zomorodian, Afra, and Gunnar Carlsson. 2005. “Computing Persistent Homology.” Discrete and Computational Geometry 33: 249–274. doi:10.1007/s00454-004-1146-y.