Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 225 (2023) 2624–2633

Procedia Computer Science

www.elsevier.com/locate/procedia

27th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2023)

ACO and generative art – artificial music

Urszula Boryczka[a,*], Mariusz Boryczka[a], Paweł Chmielarski[a]

[a]*Institute of Computer Science, University of Silesia in Katowice, Bedzińska 39, 41-200 Sosnowiec, Poland*

Abstract

The paper presents the adaptation process of an Ant Colony Optimization system to the task of composing music. The main aims of the study include designing and implementing the above-mentioned system, conducting research, measuring the obtained results, and drawing conclusions. The application was implemented in Python and it allows you, based on the loaded MIDI file, to generate new melodies played on various instruments. The way of composing a melody is configurable by modifying the parameters of the ACO algorithm and different composition modes. To measure the quality of the obtained results, an evaluation function was created, which is used in research to measure the correlation between the parameters of the system. Based on the conducted research, it was determined how individual parameters affect the results obtained by the system. The paper covers the presentation of theoretical knowledge, system design, implementation description, research, and conclusion.

Keywords: ant colony optimization; ant system; generative art; artificial music

1. Introduction

Artificial Intelligence (AI) can be used to compose new music, create unique mashups, and also help create robotic actors. The area in which AI can be used to create new things is simply limitless, and the possibilities are endless – always with the aforementioned limits imposed by programming or human provision of information. AI is also capable of creating lyrics with predetermined emotionality, creating previously unknown musical genres, and exploring the boundaries of music.

When we think about AI's manifestation of activity, we often look at those areas where humans can easily be replaced: high-level computation, manual labor, or data-driven optimization. Yet, there is now a new wave of emerging potential for AI in creative industries – one of which happens to be musical composition.

E-mail address: {urszula.boryczka,mariusz.boryczka}@us.edu.pl,
pawel.chmielarski@gmail.com

* Corresponding author. Tel.: +48-32-3689713

1877-0509
10.1016/j.procs.2023.10.254

The application of AI in creative spaces, like music and art, is not new, but recent years have seen a dramatic expanse in machine learning capabilities in music composition. AI is being used by researchers and startups to compose soundtracks and soundscapes and to create original songs within the style of specific genres and artists.

The main contribution of the paper is to present how the use of the ant colony system affects the quality of the created new pieces of music. This work aims to investigate what effect the values of individual parameters have on the new use of the metaheuristic system. Population systems have been used in music composition, however, the biggest problem is their adaptation to the problem under analysis and thus the selection of correct parameter values. This is the most frequently discussed problem in scientific papers, and there are no known methods for determining the order of parameter selection: fine-tuning or superior selection. Inappropriate parameter sets can result in a completely unusable metaheuristic being analyzed and at the same time, scalability cannot be further experimented with.

This article is organized into five sections. Section 2 provides a brief overview of the subject of the paper. Section 3 describes the Ant Colony Optimization (ACO) technique. Section 4 delves into the problem of adaptation of the ACO to the process of music composition. Section 5 describes and discusses the experimental studies. Finally, in section 6, conclusions and future work are supplied.

2. Artificial intelligence and music composition

The history of the use of computers in the process of composing music dates back to the 1950s. In 1958, Hiller and Isacson [12] created a pioneering project on the ILLIAC computer, which resulted in the composition of a piece for a string quartet. The process followed a ”generate and test” convention, while sounds were selected pseudo-randomly using Markov chains and then tested using the heuristic composition rules of classical harmony and counterpoint. However, they concluded that the method of Markov chains did not produce satisfactory results in music composition. The probabilistic method used by Hiller and Isacson was not an artificial intelligence method, unfortunately. The first paper based on a rule-based approach to the task of composing music was published by Rader in 1974 [13]. With the development of artificial intelligence algorithms, the MUSACT system was developed in 1993, which was one of the first systems using neural networks to learn a musical harmony model [3]. Since then, many music composition systems have been developed, and most of the currently available commercial solutions are based on deep learning neural networks.

The most popular such system is AIVA [1] – developed in 2016. The aforementioned system was initially able to compose 2-3 minute pieces of classical music, and recently its capabilities have been extended to include other musical genres. By accessing a huge database of songs (30,000 for classical music alone) and using reinforcement learning, the network can generate high-quality musical pieces. Other commercially available systems of this type include Amper Music, Ecrett Music, Humtap, and Amadeus Code.

Few systems using algorithms other than neural networks are available. However, there have been emerging research projects using genetic algorithms or ant systems. The first attempt to apply ant colony algorithms to the task of composing music was made by Christophe Guéret and Nicolas Monmarché in 2004 [11], and the publication that emerged at that time inspired this thesis. Of note is the AntsOMG project [4], composing melodies using the ”cantus firmus” technique, or the Geis and Middendorf project [10], in which the ant algorithm was adapted to compose Baroque harmonies.

Recently L.B. Rosello et al. [14] have proposed a methodology for music generation that makes use of Ant Colony Optimization algorithms on multilayer graphs. In the presented methodology they first defined a new multilayer graph model of music that contains several voices of musical works. Then, they adapted ACO algorithms to allow multiple ant colonies to generate solutions on each layer while interacting with each other. This methodology was illustrated with some example configurations that show how music emerged as a result of the interaction of different simultaneous ant colony optimization instances.

3. Ant Colony Optimization

The Ant System [7] is the first ACO algorithm proposed by Dorigo in 1991. It was developed from a mathematical model, constructed by J.-L. Deneubourg, of how an ant colony functions and was applied to solve the Travelling

Salesman Problem (TSP). TSP consists of finding the minimum Hamiltonian cycle in a complete weighted graph. The transition probability $P_{ij}(t)$ at a given time is described by the formula:

$$P_{ij}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}(t)]^{\beta}}{\sum_{j \in \Omega} [\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}(t)]^{\beta}}, & \text{for } j \in \Omega \\ 0, & \text{otherwise} \end{cases}$$

where:

- τ_{ij} – the intensity of the pheromone trail,
- η_{ij} – the value of the heuristically estimated transition quality from node i to node j,
- α – parameter controlling the importance of the intensity of the pheromone trail τ_{ij},
- β – parameter controlling the validity of the value η_{ij},
- Ω – the set of decisions an ant can make while at node i.

The pheromone trail update occurs after a full cycle of the algorithm when each ant has built its solution. First, the pheromone trail is evaporated on each edge according to the formula:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t)$$

where ρ is the evaporation coefficient of the pheromone trail.

Each ant then applies a pheromone to the edges it has visited according to the formula:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \Delta\tau_{ij}, \quad \forall e_{ij} \in S_k,$$

where $\Delta\tau_{ij} = f(C(S_k))$ is the amount of pheromone applied, depending on the quality of the solution S_k.

Using the TSP as an example of the combinational optimization problem, an analysis was performed [8], the result of which showed that the optimal number of ants is $m = s_1 \cdot n$, where s_1 is a small constant of the order of unity and n is the number of vertices of the graph. The computational time complexity of the ant algorithm was shown to be of the order of:

$$T(c,n) = O(c \cdot n^3),$$

where:

- c – number of iterations of the algorithm,
- n – number of vertices of the graph.

The Ant Colony System [6] is an improved version of the Ant System that was presented by Dorigo and Gambardella in 1996. The transition rule between states has been modified by introducing ways to select the next state by exploitation or exploration mechanisms. In addition, the method of depositing the pheromone trail was changed to a global and local method of updating the pheromone trails.

Over the years, many modifications and interpretations of the original ant system have appeared [2]. These algorithms can be modified quite freely as required and various techniques such as parallel computing can be applied to them (tab. 1).

4. Adapting ACO to the music composition process

Agents-ants will not compose music from scratch, but rather improvise on the basis of a set of sounds provided to them. As ant algorithms are designed to solve problems that can be represented as a graph, they are well suited to the concept mentioned above. The algorithm described further is based on inspiration taken by the authors from the publication [11]. However, it is a different approach that has been modified and extended here.

Table 1. Overview of early modifications to the ACO [2]

Name of the algorithm	Year of origin
Ant System	1991
Elitist AS	1992
Ant-Q	1995
Ant Colony System	1996
Max-Min Ant System	1996
Ranked-based AS	1997
ANTS	1998
Best-worst AS	2000
Population-based ACO	2002
Beam-ACO	2004
HyperCube ACO	2004

In the AntsBand project, melodies are composed based on the probabilities of transitions in the graph. The process starts by reading MIDI events from the input file. From each event, which signifies the playing of a sound, the pitch value of that sound is read. For example, the following sequence of sounds is read from a certain track: {41, 44, 36, 39, 41, 36}, which means the sounds {F, G#, C, D#, F, C} played in the 3rd octave. In MIDI, pitches are specified in the range from 0 to 127, which makes it possible to operate in 11 incomplete octaves, and it is on these values that the described algorithm works.

After reading the sounds from the input file, the next step is to place these sounds on the vertices of the graph. During the initialization of the graph, a pheromone is put on the edges of the graph. Edges connecting sounds that were adjacent to each other in the input file are given a different pheromone value to give the ants the opportunity to suggest the original melody from the input file. The initialization of a pheromone in a graph with n vertices is defined by the following rule:

$$\tau_{ij} = \begin{cases} \sigma, & j = i + 1 \\ \frac{1}{n^2}, & j \neq i + 1 \end{cases}$$

where σ is the pheromone initiation factor for neighboring sounds in the input sound sequence. At the same time, a transition cost matrix is created between the sounds, based on the difference in distance between these sounds. Thus, the greater the difference in pitch of the sounds, the greater the transition cost between them. In addition, when creating the matrix, a rule was introduced stating that if the sounds are of the same pitch, the value of the distance is randomly drawn from a predefined set of numbers. With this rule, the value of zero transition cost between nodes can be avoided, as well as determining how willing the ants are to place the same notes next to each other in the output melody. Tab. 2 shows an example of a transition cost matrix between graph nodes, where the zero distances have been replaced by values of 10 or 100.

In the graph for the example sequence of sounds, transitions initiated by a pheromone with a value of s are marked with a bold line (fig. 1).

Once the graph is complete, it is necessary to find a transition path between its nodes using the AS or ACS algorithms, which were described in Section 3. Each node of the graph should be visited exactly once, so the task is

Table 2. Example matrix of transition costs between graph nodes

	F (41)	G# (44)	C (36)	D# (39)	F (41)	C (44)
F (41)	10	3	5	2	100	5
G# (44)	3	10	8	5	3	8
C (36)	5	8	100	3	5	100
D# (39)	2	5	3	100	2	3
F (41)	10	3	5	2	100	5
C (44)	5	8	10	3	5	10

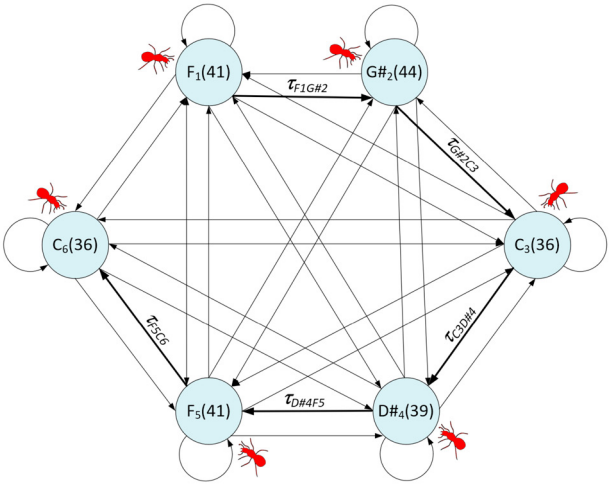

Fig. 1. Example graph for 6 sounds. The nodes contain the sound, the index, and the MIDI value in brackets

similar to the TSP problem, except that without return to the starting node. The AntsBand system assumes that the length of the output melody must be the same as the input melody. This provides more possibilities for processing the finished song and synchronizing the sounds. The AntsBand algorithm returns the found path in the graph in the form of consecutive node numbers. The path thus obtained must then be processed into a sequence of MIDI events corresponding to the graph nodes. As the ant algorithm only operates on pitch values, other values, such as the duration of a sound, can be taken from the original sequence of notes in the input file or determined from new events resulting from a transition in the graph. In the case of the first option, the melody in the output piece will vary in pitch, while the rhythmic values remain the same. In the second case, a completely new melody is obtained (see Algorithm 1).

Algorithm 1: AntBand System

Input: System parameters: n (number of ants), m (number of vertices in the graph of pits), $c_{i,j}$ (transition cost from vertex i to j), q_0 (randomness coefficient), α (pheromone trail influence weight), β (heuristics influence weight)

Output: The best solution found by the ant colony system

```
Initialization of pheromones at all edges;
for t ← 1 to T do
    for i ← 1 to n do
        Selection of starting vertex s_i for ant i;
        for j ← 1 to m − 1 do
            Selection of the next vertex v_j by ant i based on the probabilities determined by pheromones and
              matrix of costs; Updating the pheromone trail on the edges (s_i, v_j);
            if j = m − 1 then
                Calculate the path quality L_i created by ant i;
    Updating the global best solution; Update the pheromone level on the edges;
return Global best solution
```

Due to the polynomial computational complexity [7] of ant algorithms, it can happen that for long sequences of sounds, a graph large enough is created that the processing time is unacceptably long. In this case, a modification is introduced in the AntsBand algorithm, allowing a given sequence of sounds to be decomposed into several smaller parts.

An important step here is the creation of a transition cost matrix between the sounds. For a given n-element list of sounds, a matrix of dimension $n \times n$ is created. The transition cost between sounds is calculated as the absolute value of the difference in distance between the sounds. For example, if there is a semitone difference between two sounds, the transition cost between them will be 1. Thus, the agents-ants prefer to choose smaller intervals between sounds. If two tones have the same pitch, the transition cost between them is drawn at random from among the numbers: {0.1, 0.5, 1, 5, 10, 100}. Then, depending on the type of ant colony algorithm algorithm chosen, an instance of the algorithm, the corresponding graph, and the resulting solution are created. The solution returned by the anthill algorithm is in the form of a list containing the indices of the following notes.

5. Evaluation function

The evaluation function for the AntsBand system was decided to be implemented based on a multi-criteria optimization problem [9]. This problem consists of the need to simultaneously consider the values of several objective functions when calculating a decision task. For the evaluation function described here, three criteria were created to evaluate the generated music, based on which the main criterion is calculated using the weighted criteria method [15]. This method involves reducing a multidimensional task to a one-dimensional task, as described by the following formula:

$$g(x) = \sum_{i=1}^{n} w_i \cdot f_i(x)$$

where:

- $g(x)$ – final evaluation function, substitute criterion,
- n – the number of criteria, $n \in N$,
- w_i – weight for the i-th criterion,
- $f_i(x)$ – single i-th criterion,
- $w_i \in \langle 0, 1 \rangle$ and $\sum_{i=1}^{n} w_i = 1$.

The values of the subsequent criteria are multiplied by the corresponding weights and then summed to obtain the value of the final evaluation function.

The first component criterion is a function that checks the quality of note placement over time in a given sequence of sounds. This function iterates over successive MIDI events of the sound sequence, counting the time elapsed since the beginning of the song. Based on the song's meter and the number of MIDI clock ticks per beat, it calculates through modulo division operations whether a given note hits the bar value corresponding to a sixteenth or eighth note. Each note can be assigned 0, 1 or 2 points here. Finally, the average value of points obtained by a single note is calculated. This function returns a value of $\langle 0, 1 \rangle$, where low values mean that the notes are incorrectly shifted in time, and high values let you know that they are accurately placed in the rhythm of the piece.

Another component function for the evaluation criterion examines the repetition of the sequence of sounds found in a melody. In the analyzed melody, phrases with a length of 4 notes or more are searched for, and the longest phrase can be half the length of the considered sequence of sounds. Then all possible positions of the occurrence of phrases in the sequence of sounds are checked and their repeated occurrences are counted. This function, due to its 3 nested loops, has a rather high computational complexity of the order of $O(n^3)$. The returned value is the quotient of the number of phrase repetitions found to the maximum number of repeated phrases that can occur in a given sequence of sounds.

The third and final criterion for the evaluation function is the search for melodic consonances in a sequence of sounds. This function calculates the values of the intervals between successive sounds in a melody and then checks whether this value occurs in the consonance table [5]. If so, the consonance counter is incremented. The function

Table 3. Paramaters of the examination of the AntsBand

Input parameters	Output parameters
ant_count – number of ants	evaluation_result – result of evaluation function
generations – number of iterations	notes_in_time_factor – criterion for the arrangement of sounds in time
alpha – α	repeated_sequences_factor – criterion for repetition of sequences
beta – β	cosonance_factor – criterion of consonances
rho – ρ	similar_notes_factor – similarity of notes
q – pheromone intensity	similar_times_factor – similarity of times
sigma – σ	execution_time – execution time
phi – φ (ACS only)	cost – graph transition cost
q_0 (ACS only)	midi_filename – MIDI file name

returns a value that is the ratio of the number of consonances found to the maximum number of consonances that can occur in a sequence of sounds.

The biggest problem in adapting ACO to music composition is the issue of fine-tuning values of the individual parameters that manage the work of ACO (tab. 3). The first to be calculated is the correlation, which involves examining the effect of input parameters on the values of output parameters. Spearman's rank correlation coefficient is used, since it is not known whether the relationships between variables are always linear and the variables under study are qualitative. The method in question involves assigning ranks to the values of the variables. Ranks are the numbers of places in a non-decreasing series for the value of a given variable. Spearman's rank correlation coefficient is calculated from the formula:

$$R_{xy} = 1 - \frac{6 \cdot \sum_{j=1}^{n} (X_j - Y_j)^2}{n \cdot (n^2 - 1)}$$

The value of the calculated correlation coefficient is a number in the range of $\langle -1, 1 \rangle$.

As part of the research, 5 tests were prepared covering different configurations of the AntsBand system and generating results for multiple sets of output parameters. The tests used 3 different system configurations:

- K1 – A sequence of sounds processed as a whole by a single anthropoid. The duration of the sounds is retained from the input track. Default track length.
- K2 – A sequence of sounds divided into 4 parts and processed by separate ant colonies. The order of the processed parts is randomized in the output track. The duration of the sounds is new and quantized. Default track length.
- K3 – A sequence of sounds divided into 4 parts and processed by separate ant colonies, and the default order of the parts is preserved in the output track. The duration of the sounds is retained from the input track. Default track length.

Various combinations of the following parameter values were tested:

- number of ants = {1; 5; 10; 20; 30; 60},
- number of iterations = {1; 10; 20; 30; 50},
- $\alpha = \{0,1; 0,5; 2; 5\}$,
- $\beta = \{0,1; 1; 2; 5; 10\}$,
- $\rho = \{0,2; 0,5; 0,9\}$,
- $q = \{1; 5\}$,
- $\varphi = \{0,2; 0,5; 0,9\}$,
- $q_0 = \{0,2; 0,5; 0,9\}$,
- $\sigma = \{1; 5; 10; 20\}$.

The main focus of this paper is to investigate the sensitivity of the ACO method to the selection of parameters during music composition. The parameters associated with ACO are based on pheromone deposition, pheromone evaporation, and visibility (which depends on the cost of possible paths in the analyzed graph). Results of a com-

Table 4. Cases of the longest execution time of the algorithm

	execution_time	Ants	generations	α	β	ρ	q	φ	q_0	σ
T1	9.715	60	50	5.0	0.1	0.2	1	-	-	10
T2	3.403	60	50	0.5	5.0	0.5	1	-	-	5
T3	7.717	60	50	5.0	0.1	0.2	1	0.2	0.2	10
T4	2.772	60	50	5.0	10.0	0.2	5	0.2	0.2	5
T5	5.841	60	50	0.1	2.0	0.2	5	-	-	10

prehensive parametric study for the ACO method are presented to assess the sensitivity of the ACO method to the selection of these parameters.

There are 6 parameters that need to be selected (divided into two groups: quantitative and qualitative): (1) the number of ants in a population, and the number of iterations, (2) the parameter for controlling the relative importance of pheromone for the node choice probability – α, (3) the parameter for controlling the relative importance of the local heuristic factor for the node choice probability – β, (4) the pheromone persistence coefficient (5) the pheromone reward factor – the amount of an added pheromone value and (6) the pheromone penalty factor – evaporation of the pheromone. The ranges of values for these parameters have been defined in detail by the developers of the ant colony system.

Reward and penalty are two mechanisms that alter the pheromone concentrations for each of the options at each of the decision nodes. Positive reinforcement of the pheromone concentration results in solutions of high quality and negative reinforcement in solutions which results in longer running times for the algorithm and a longer wait for satisfactory compositions.

T2 and T4 test matrices contain much less significant correlations between parameters. This is due to the K2 system configuration used, which involves splitting a sequence of sounds into parts, randomizing them, and assigning new durations to the sounds. For these tests, the results obtained are more unpredictable and less dependent on the system's input parameters' settings. For the T2 test, a slight dependence of the cost of the obtained melody on the parameter α can be observed. This means that for higher values of the a parameter, which favors the pheromone trace, the obtained melodies have larger intervals between sounds. This can also mean in some configurations a higher degree of similarity to the input tune. For the T4 test, the dependence of cost on α is even lower.

The T1 test was performed in a basic system configuration, and in this case, more of the dependence between parameters can be seen. Significant here is the parameter β, controlling the validity of the heuristic information, which, when increased, tends to increase the number of repeated sequences of sounds in the generated melody. As expected, the cost decreases when β increases, as ants then prefer to choose sounds that are less distant from each other. On the other hand, higher values of βdo not positively affect the content of consonances in the output melody. The described dependencies of the β parameter can also be seen for the T3 and T5 tests. Although the T5 test was performed in the K3 configuration, which involves splitting the tracks, the β parameter here has a similar effect on the aspects discussed. This may be because in T5, the cut fragments are retained finally in their original order, and the note durations do not change concerning the input track.

As for the execution time of the algorithm, as expected, it is strongly correlated with the number of virtual ants and the number of iterations. This correlation can be seen for all tests. Computation time is also often negatively correlated with the cost of the solution obtained, i.e. the longer time the calculation takes, the greater the chance that the intervals between sounds will be smaller. Table 4 shows that with a large number of ants and iterations, the AS algorithm in the K1 configuration is by far the slowest. Dividing the same problem into 4 parts and processing them separately usually results in almost 3 times faster computation. It was also observed that the ACS algorithm is faster than the AS algorithm. This is because exploitation has lower computational complexity, i.e. it is faster than exploration. There is some slight correlation between the parameter q_0 and the execution time of the ACS algorithm. The fastest results were obtained for the T4 test, which operates the ACS algorithm in sound sequence division mode.

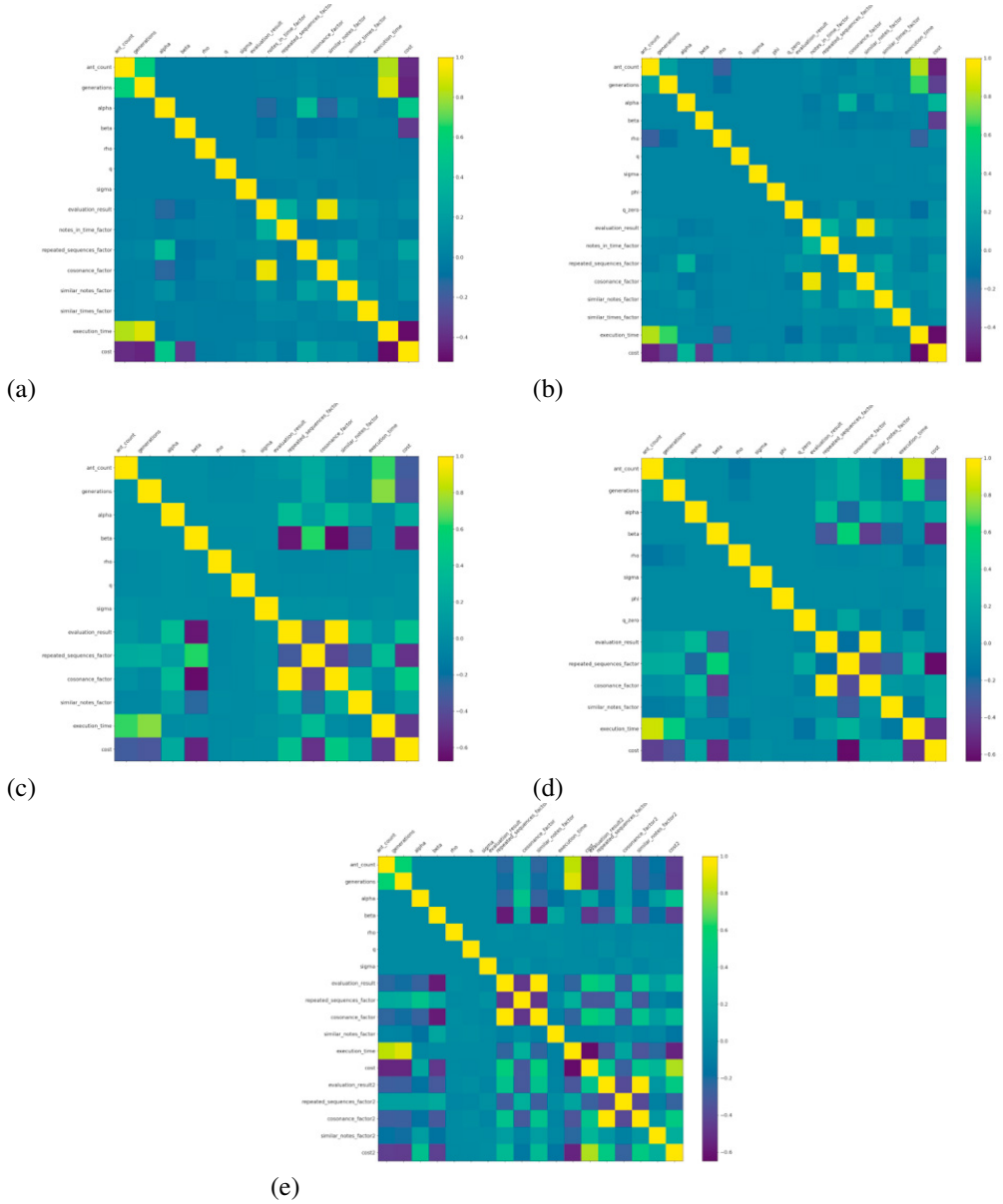

Fig. 2. Spearman's rank correlation coefficients for tests: (a) T2 – AS with K2 configuration, (b) T4 – ACS with K2 configuration, (c) T1 – AS with K1 configuration, (d) T3 – ACS with K1 configuration, (e) T5 – AS with K3 configuration

6. Conclusions and future work

As a part of the tests on the implemented system, the algorithm was executed 291780 times, with multiple sets of input parameters and in different system configurations. Based on the resulting data, it was possible to find out the meaning and relationships between the various parameters of the system through the calculation of correlations. The tests for which the best or worst results were obtained in the context of a given attribute were also identified. Through the analysis of the results obtained, it was possible to determine that an increased value of the parameter a positively affects the size of intervals in the output melody and the similarity of the output melody to the input melody. Very important turned out to be the parameter β, which, when enlarged, causes an increase in the number of repeated sequences of sounds, smaller values of intervals between successive sounds, but for a lower content of consonances in

the output melody. The research also shows that the mode of dividing the melody into parts speeds up the execution of the algorithm, while the value of the correlation between the parameters decreases if the parts are randomized. It is shown that the ACS algorithm is faster than the AS algorithm and that the number of ants and iterations significantly slows down the execution of the algorithm. Given the purpose of adapting the ant system to the task of composing music, in which it is not important to obtain the shortest path in the graph, it makes little sense to set the number of ants and iterations to values greater than 10. The most important thing is to achieve an appropriate balance between suggesting the input melody and heuristic selection of transitions, which can be achieved by setting the parameters α and β accordingly.

Music seems to be the most important of all man-made fine arts. It is a difficult task to describe it using mathematical functions and even more so to make a mathematical evaluation. However, this does not mean that it is an impossible task, as evidenced by the achievements of other artists in this field. Music these days is increasingly being composed in digital environments. This is a potential application of the AntsBand system, which could be developed and combined with a digital MIDI synthesizer to create an advanced tool for creating electronic music, supporting the artist's creativity in the process. Also of interest seems to be the use of the system to generate long, non-repeating tracks for use in computer games, for example.

References

[1] AIVA Technologies (accessed 2023-04). AIVA. https://www.aiva.ai/.

[2] Akhtar, A. (2019). *Evolution of Ant Colony Optimization Algorithm – A Brief Literature Review*. SEECS-NUST, Islamabad.

[3] Bharucha, J. (1993). Musact: A connectionist model of musical harmony. *Music Perception: An Interdisciplinary Journal*, 11(1):43–80.

[4] Chang, C.-Y. and Chen, Y.-P. (2020). Antsomg: A framework aiming to automate creativity and intelligent behavior with a showcase on cantus firmus composition and style development. *Journal of New Music Research*, 49(3):237–256.

[5] Chmielarski, P. (2022). *Composing musical pieces with the use of ant colony systems (in Polish)*. University of Silesia, Sosnowiec, Poland.

[6] Dorigo, M. and Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66.

[7] Dorigo, M., Maniezzo, V., and Colorni, A. (1991). The ant system: An autocatalytic optimizing process. Technical Report 91-016 Revised, Dipartimento di Elettronica, Politecnico di Milano, Italy.

[8] Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 26(1):29–41.

[9] Eschenauer, H., Koski, J., and Osyczka, A. (1990). Multicriteria design optimization. *Structural optimization*, 2(4):202–211.

[10] Geis, M. and Middendorf, M. (2007). An ant colony optimizer for melody creation with baroque harmony. In *International Work-Conference on Artificial Neural Networks*, pages 123–130. Springer.

[11] Guéret, C., Monmarché, N., and Slimane, M. (2004). Ants can play music. *BioSystems*, 78(1-3):109–113.

[12] Hiller, L. and Isaacson, L. (1958). Musical composition with a high-speed digital computer. *Science*, 128(3327):1102–1105.

[13] Rader, G. M. (1974). A method for composing simple traditional music by computer. *Computer Music Journal*, 1(4):34–38.

[14] Rosselló, L. B. and Bersini, H. (2023). Music generation withÂ multiple ant colonies interacting onÂ multilayer graphs. In Johnson, C., Rodríguez-Fernández, N., and Rebelo, S. M., editors, *Artificial Intelligence in Music, Sound, Art and Design*, pages 34–49, Cham. Springer Nature Switzerland.

[15] Wściubiak, M. (2000). *Evolutional multicriteria optimization (in Polish)*. Wydawnictwa Naukowo-Techniczne.