Toward Human-level Tonal and Modal Melody Harmonizations

Jan Mycka[a,*], Adam Żychowski[a], Jacek Mańdziuk[a,b]

[a] *Warsaw University of Technology, Warsaw, Poland*
[b] *AGH University of Science and Technology, Krakow, Poland*

Abstract

This paper considers the problem of automatic music generation (melody harmonization) with the use of Evolutionary Algorithms (EA), based on the rules derived from music theory and practice. The rules, whose role is to ensure the fulfillment of both the formal requirements of harmonization and its less-formalized aesthetic requirements are encoded in the fitness function of EA. The fitness function is composed of several modules, each of which consists of smaller parts corresponding to the implemented music rules. The above modular design allows for flexible modification and extension of this function. The way the fitness function is constructed and tuned towards better quality harmonizations is discussed in the context of music theory and technical EA implementation. In particular, we show how could generated harmonizations be modeled by means of adjusting the relevance of particular fitness function components or extended by adding new components to the fitness function. The proposed algorithm is tested on two types of music: tonal and modal. Although tonal and modal harmonizations are significantly different, the achieved results (assessed by a human expert) indicate that the constructed harmonizations are both technically and aesthetically correct (i.e. they adhere to the theoretical rules and are nice to listen to). This study extends our previously published conference paper [1].

Keywords: evolutionary algorithm, harmonization, music generation

*Corresponding author

Email addresses: jan.mycka.dokt@pw.edu.pl (Jan Mycka), a.zychowski@mini.pw.edu.pl (Adam Żychowski), mandziuk@mini.pw.edu.pl (Jacek Mańdziuk)

1. Introduction

For centuries, the fine arts have been developed by people and, invariably, a major aspect of this process has been creativity of an artist. For this reason, Artificial Intelligence (AI) researchers attempt to implement *computational creativity* [2] to mimic the creative behaviors of AI agents. One of the fields in which AI creativity has been intensively developed is music [3]. AI methods are applied to create new compositions [4] or complement / expand existing pieces [5]. In both tasks, they managed to demonstrate human-level performance. Another line of research is the imitation of a particular composer's style, e.g. F. Chopin [6, 7]. Various AI methods are used for all these problems, and among the most popular ones are neural networks. However, the choice of the right method often depends on the detailed analysis of the considered problem.

One of the basic problems in music is the enrichment of a given melodic line by adding chords – the so-called harmonization of the melodic line. Adding new notes is based on the relationship between them and the existing notes, both vertically (simultaneous sound) and horizontally (time sequence).

In music, there are many types of harmony (and therefore many types of harmonization of the melodic line) and each of them has a different sonic character. The most common types of harmony are tonal harmony (the most popular), modal harmony (used for example for Gregorian chant), and, among others, jazz harmony (characteristic for jazz music).

Although harmonization is a creative process in which the main role is played by intuition, talent, and experience of a musician [8], the process is constrained by various rules resulting from music theory and centuries of musical practice. An important difference between the harmony types is the rules of using and combining chords that complement the main melodic line. Each such type is based on different, though intersecting, rules corresponding to the type of harmony used. The algorithm proposed in this work applies AI methods to create suitable harmonization and is based on harmonization rules defined in music theory. In general, creating music (or other forms of Art) is considered a challenge for AI agents, as mastery in this field requires special gifts that are rare even among humans.

1.1. Contribution

The main contribution of this paper is the following: (1) we propose a novel Evolutionary Algorithm (EA) approach capable of creating melodic

line harmonizations that are formally correct, i.e. fulfill music harmonization rules; (2) to this end we design a specific module-based fitness function that can be easily extended or tuned to reflect the desired aspects of the resulting harmonizations; (3) to demonstrate the efficacy and generality of the proposed algorithm, we create two sets of rules used in the fitness function, based on the theory of music harmonization: one for tonal harmony and the other one for modal harmony (4) both proposed fitness functions cover not only harmonic but also melodic aspects (voices leading) of created harmonizations; (5) through modification of the weights of individual components of the fitness functions, solutions can be tuned to meet the desired requirements; (6) the resulting harmonizations are evaluated by a (human) expert and assessed as both technically correct and **possessing human-like characteristics**.

1.2. Extension of the ICCS 2022 conference publication

This article extends our preliminary work on this topic, presented at ICCS 2022 [1], by considering the harmonization problem in the context of modal music (as opposed to more straightforward tonal music discussed in [1]. In effect, this work extends the conference paper in the following 3 major aspects:

- an adjustment of the fitness function for the modal harmonization problem (Section 5.2),
- analysis of the results generated by the algorithm for modal harmonization (Section 7) including a human expert evaluation of these results (Section 7.4) and an ad-hoc improvement of modal harmonizations by extending the set of rules encoded in the fitness function (Section 7.5),
- a comparison of the outcomes achieved for tonal and modal harmonization (Section 8).

Furthermore, several other parts of the paper, such as Section 4 or Section 9, have been extended and adjusted to the broader scope of the paper.

2. Problem definition

Harmonization is the process of creating an accompaniment for a given melody line. The created accompaniment consists of three new melodic lines

(voices). Usually, the highest voice (the soprano, the melody) is the base for harmonization and the other three voices (the alto, the tenor, and the bass) are to be created. Four notes, one from each voice, form a chord. Creating harmonization is mainly based on the musician's experience and intuition. However, throughout the years many theoretical rules that harmonization has to fulfill were developed [9, 10, 11]. These rules do not specify how harmonization is to be constructed, only whether or not the harmonization is correct.

The aim of the proposed algorithm is to create harmonization fulfilling selected theoretical rules for tonal or modal harmony, respectively. The harmonization is generated not for the raw melody, but for the melody extended with harmonic functions or chord labels assigned to specific notes. These labels determine which notes (pitches) should be used across all the voices. However, they do not specify the number of these notes or the voices in which they should be placed.

3. Related literature

The melodic line harmonization problem has been addressed in the literature using various AI methods. Popular approaches use neural networks [12] or hidden Markov models [13]. Both papers address the task of harmonization of chorales based on J.S. Bach's style and the resulting harmonizations occasionally do not follow theoretical musical rules.

The algorithm described in this paper relies on a different method, the Evolutionary Algorithm (EA), in which the required harmonization rules are directly imposed by means of a fitness function. Moreover, unlike neural networks, EA does not require training, which makes this approach independent of the composer's style, implicitly present in the training set.

Another approach, which uses Markov Decision Processes is presented in [14] and evaluates connections between two consecutive chords. The evaluation rules are based on music theory. Yet another work [15] hybridizes heuristic rules with dynamic programming method.

The use of EAs in the melody harmonization problem has been considered in a few recent works [16, 17, 18]. In [16] multiobjective genetic algorithm is proposed which for a given melody generates a set of suitable harmonic functions, however, without adding new melodic lines. Similarly, in [18], only chords are created, not entire melodic lines. In [17], the algorithm solves a broader problem, i.e. not only adds new melodic lines to a given melody but

also complements the melody with harmonic functions. Furthermore, the method uses a wider range of harmonic functions and fewer theoretical rules than in our approach. The fitness function consists of two parts, the first one evaluates the created harmonic functions and the other one evaluates the melodic lines added.

Each of the above-mentioned works considers a slightly different formulation of the harmonization problem which renders a direct comparison impossible. Hence, the assessment of the resulting melody lines proposed in the paper is two-fold: by means of a numerical fitness value assigned by the algorithm and by a human expert - a harmony teacher.

4. Proposed method

In order to solve the harmonization problem an EA, that maintains candidate solutions (population of individuals) is proposed. In each generation, the currently maintained individuals undergo mutation and crossover operations. Subsequent generations are composed of individuals with gradually higher fitness values, i.e. achieve higher evaluations, on average. The core element of the algorithm is the fitness function that evaluates candidate solutions, which is based on theoretical rules of music harmonization. These rules may differ depending on the type of harmony one wants to obtain

The algorithm is run for a predefined number of n generations. Afterward, the best individual in the last population is returned as the final result.

4.1. Problem search space - admissible *chords*

The input data is the highest voice melodic line with a particular harmonic function (tonal music) or chord label (modal music) assigned to the selected notes. This label indicates at least three and at most five notes, which have to be used in all the voices. If the label indicates only three notes, one must be doubled, and if five of them, one must be omitted.

In each created chord, the highest note is fixed and derived from the given melodic line (the soprano). In addition, an ambitus (the lowest and the highest possible pitch of a voice) is defined for each voice, derived from the theory. Thus, for each harmonic function/chord label it is possible to define a set of all admissible chords corresponding to this label. The evolutionary operators (mutation, crossover), as well as the construction of the initial population, are based on these pre-created sets of admissible chords associated with particular labels.

4.2. Stopping condition

The algorithm generates consecutive populations. For one run there is a predefined number of generations that the algorithm has to perform. The algorithm ends when the last population is generated and the returned result is the best individual from the last population.

4.3. Population generation process

Each initial candidate harmonization is in the form of a sequence of admissible chords. Each chord in a sequence must satisfy the two basic conditions:

(i) the chord corresponds to the harmonic function/chord label assigned to the completed note,

(ii) the note given in the input voice is located in the chord in the same voice (the soprano/melody).

It is worth noticing that the construction of a solution is performed by manipulating the whole chords, not single notes.

The first population is generated randomly. Each individual consists of randomly chosen chords from a pre-defined set of all admissible chords which fulfill conditions (i)-(ii). In each subsequent generation, first s_e *elite* (i.e. currently best) individuals are promoted from the previous generation without any adjustments, so as to ensure that the best solutions found in the entire run of the algorithm will not be lost. The rest of the population is generated by means of a selection procedure and genetic operators (mutation and crossover), according to Algorithm 1.

4.4. Selection method

The selection of individuals is performed in a tournament of size t_s with a roulette element added. In the first step, t_s individuals are drawn uniformly with replacement from the population and their fitness score is calculated. Let's define by x_i the i-th individual of the tournament according to the fitness score ranking. Its probability of winning the tournament $p(x_i)$ is calculated according to (1):

$$p(x_i) = \begin{cases} p_s & \text{if} \quad i = 1 \\ (1 - \sum_{j=1}^{i-1} p(x_j)) \cdot p_s & \text{if} \quad 1 < i < t_s \\ (1 - \sum_{j=1}^{i-1} p(x_j)) & \text{if} \quad i = t_s \end{cases} \tag{1}$$

where $p_s \geq 0.5$ is the so-called *selection pressure*.

```
GenerateNewPopulation (P)
    CalculateFitnessValues(P)  // calculates fitness of each
     individual
    P_new ← GetElite(P, s_e)    // population of new individuals
    while |P_new| < |P| do
        c_1 ← Selection(P)
        if rand([0,1]) < p_c then // crossover
            c_2 ← Selection(P)
            c_new = Crossover(c_1, c_2)
        else
            c_new ← c_1
        end
        c_new ← Mutation(c_new)
        P_new = P_new ∪ {c_new}
    end
    return P_new
```

Algorithm 1: Next generation population procedure.

4.5. Mutation and crossover

Each individual, before being added to a new generation, undergoes mutation. Each chord in a given harmonization is mutated with probability $\frac{p_m}{l}$, where l is the length of harmonization (number of chords in created harmonization) and p_m is the mutation coefficient (algorithm's parameter). Mutating a chord consists of its replacement by another chord uniformly sampled from the pre-defined set (all chords that meet requirements (i)-(ii) – see Algorithm 2. An example of mutation is presented in Figure 1a.

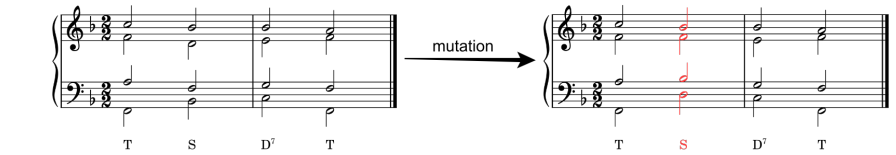
```
Mutation (c)
    for i ∈ [1, ..., l] do
        if rand([0,1]) < p_m / l
         then
            c[i] = mutate(c[i])
        end
    end
    return c
```

Algorithm 2: Mutation.

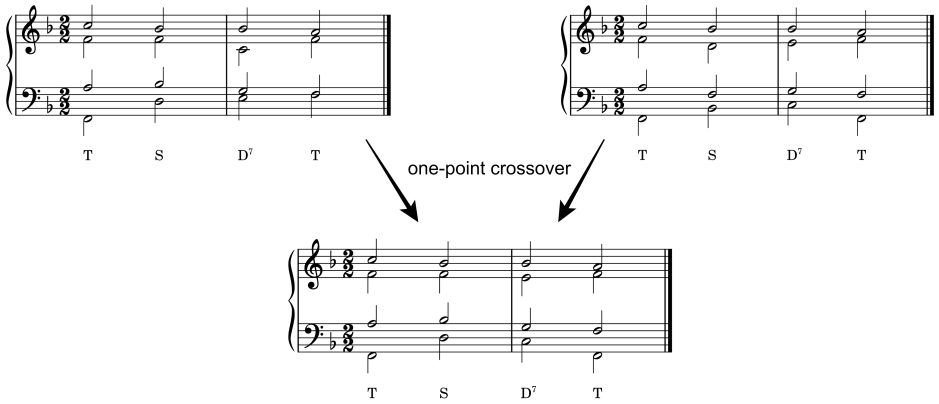
```
Crossover (c_1, c_2)
    k ← rand(1, ..., l)
    for i ∈ [1, ..., l] do
        if i < k then
            c[i] = c_1[i]
        else
            c[i] = c_2[i]
        end
    end
    return c
```

Algorithm 3: Crossover.

The algorithm uses a one-point crossover which happens with probability p_c. In the crossings of the two sampled individuals the whole chords are considered, not the single notes. Let us define by $c[i], i = 1, \ldots, l$ the chord located at the i-th position of harmonization c. One-point crossover combines the initial part of one harmonization with the subsequent part of the other harmonization – see Algorithm 3. The use of a one-point crossover offers a chance to preserve already correctly created (highly-evaluated) harmonization fragments. An example of crossover application is presented in Figure 1b.



(a) Example of mutation. The second chord has changed.



(b) Example of one-point crossover with $k = 2$.

Figure 1: Examples of mutation and crossover methods.

4.6. Fitness function – a generic construction

The fitness function is applied to evaluate individuals with respect to fulfilling harmonization rules (referring to chord building) and certain music theory rules (e.g. voice leading or fluidity of the melodic line). An important feature of the fitness function is its modular design, which allows its easy extension by means of adding new rules if required. It is worth noticing in this context, that the rules used in this paper for the construction of the fitness function do not cover the entire set of existing rules, but only a part

of it. A modular design of the fitness function allows for adding/deleting the rules with ease.

Each rule (sub-function) used in the construction of the fitness function is assigned a value that contributes to the final score of the generated harmonization. These values indicate the importance of particular rules, due to their respective impact on the resulting fitness function evaluation. Although in musical practice, there is a certain level of subjectivity in evaluating the quality of the constructed harmonization, the essential evaluation based on a general position of the music community is usually unequivocal. The base values associated with particular rules included in the fitness function have been chosen so as to assure their compatibility with the evaluation used in musical practice. The fitness function can be divided into the following 3 main modules.

1. Strong constraints C_s (high penalty terms) - constraints stemming from the rules, that must absolutely be satisfied in the created harmonization for it to be considered correct.

2. Weak constraints C_w (lower penalty terms) - constraints derived from the rules that do not have to be satisfied in the created harmonization, but their non-fulfillment lowers the harmonization assessment.

3. Added value V_a (reward terms) - the rules that specify chord arrangements or connections between chords that improve the sound of the harmonization.

The fitness function f_t for a given individual X has the following form:

$$\begin{gathered} f_t(X) = V_a + C_w + (C \cdot t)C_s, \\ C_s = \sum_{i=1}^{m_s} \phi_i(X), \quad C_w = \sum_{j=1}^{m_w} \chi_j(X), \quad V_a = \sum_{k=1}^{m_a} \psi_k(X), \end{gathered} \tag{2}$$

where $\phi_i(X) < 0$ is the penalty for not fulfilling strong constraint $i, i = 1, \ldots, m_s$, $\chi_j(X) < 0$ is the penalty for not fulfilling weak constraint $j, j = 1, \ldots, m_w$, $\psi_k(X) > 0$ is the reward associated with the rule $k, k = 1, \ldots, m_a$, $t \leq n$ is the generation number, C is a constant parameter.

5. Fitness function – the rules

The fitness function must be appropriately matched to the type of harmonization being created. We apply the proposed algorithm for the two types of harmonization (tonal and modal). Accordingly, two sets of rules derived from music theory (respectively, tonal and modal) were created. Please consult the publicly-available source code [19] for detailed implementation of both fitness functions and their sub-functions.

5.1. Tonal harmonization

The set of considered rules for tonal harmonization includes those that are taught at music schools in the first years of harmonization classes. All of them come from a harmony textbook [20]. Similar rules can be found, for instance, in [21, 22].

As demonstrated in section 6 the selected set of rules allows for achieving effective harmonizations, highly graded by the human expert in terms of both formal and aesthetic aspects. In the created fitness function the values of $m_s = 9$, $m_w = 9$ and $m_a = 4$ are applied in Equation (2).

Strong constraints Strong constraints refer to harmonization acceptability. If any of these constraints is not fulfilled then harmonization cannot be considered correct. The following strong constraints are considered (selected examples of their violation are presented in Figure 2).

i) *Doubled prime in the first chord and the last chord* – The first and the last chord occurring in a harmonization are usually a tonic, so as to emphasize the key in which the harmonization is created.

ii) *Voices are not crossing* – The voices in the chords must not cross, that is, the highest note must be in the soprano, the lower one in the alto, yet the lower one in the tenor, and the lowest one in the bass.

iii) *Limited distances between voices* – The distance between the three highest voices should not exceed an octave interval. The distance could be up to two octaves between the two lowest voices.

iv) *No quint in the bass on strong downbeat* – Downbeats for the meter are given. Chords on the first given downbeat cannot have quint in the bass.

v) *Correct notes resolutions* – Some rules are specified for note resolution in chords: (a) a sixth must be resolved up by a second, (b) a seventh must be resolved down by a second, (c) a ninth must be resolved down by a second, (d) if the chord is dominant third must be resolved up by the minor second.

vi) *No parallel (or antiparallel) quints, octaves, or primes* – If there is a quint interval between two voices in a chord, there cannot be a quint interval between the same voices in the following chord. An analogous rule applies to octave and prime intervals.

vii) *Voices must move in different directions* – The voices, moving from one chord to the next, should move in different directions. The movement of all voices in one direction, up or down between consecutive chords is forbidden.

viii) *Penultimate chord the bass note* – The penultimate chord in harmonization is usually dominant or subdominant. It is important to emphasize the sound of the chord by doubling the prime (if possible) and not using a fifth in the bass.

ix) *No augmented interval moves* – There cannot be an augmented interval between two consecutive notes in one voice.

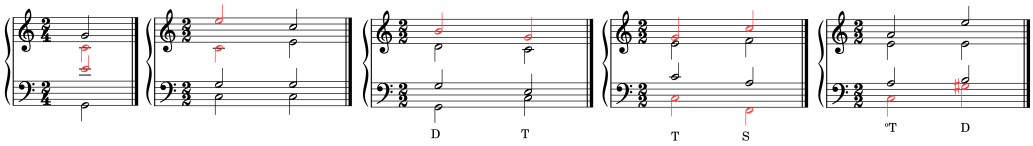


Figure 2: Examples of strong constraints violation. From left to right: ii) Crossed the alto and the tenor, iii) Distance between the soprano and the alto exceeds an octave, v) Incorrect resolution of thirds, vi) Antiparallel quints between the soprano and the bass, ix) Augmented jump in the bass.

Weak constraints Weak constraints do not have to be strictly satisfied, i.e. violating them does not make a harmonization unacceptable. However, violation of any such constraint lowers the harmonization evaluation. The following strong constraints are considered (selected examples of their violation are presented in Figure 3).

i) *Doubled quint in the bass* – When the quint is doubled in a chord, one of these quints should be in the bass.

ii) *No quint in the bass on on-beats* – On-beats are sorted by their importance. Quint in the bass on the on-beat is not preferable.

iii) *No tripled prime in tonic function* – It is permissible to triple prime in the last chord, although, it is not preferable.

iv) *No consecutive chords on quint* – Chords that have a fifth in the bass can occur, albeit, two such chords should not follow each other directly.

v) *The bass movement* – The lowest voice (the bass) is one of the most significant voices in a harmonization, hence its movement is preferred in chords connections.

vi) *Movement of at least two voices* – A movement of at least two voices between the two following chords is preferred so that the harmonization does not sound static.

vii) *No seventh interval* – There should not be a seventh interval between two consecutive notes in one voice or between three consecutive notes in total in one voice.

viii) *Melodic line smoothness* – The middle melodic lines, the alto and the tenor, should be conducted smoothly.

ix) *The bass movement restriction* – For the bass voice, the maximum interval it can take in two consecutive moves is a tenth.

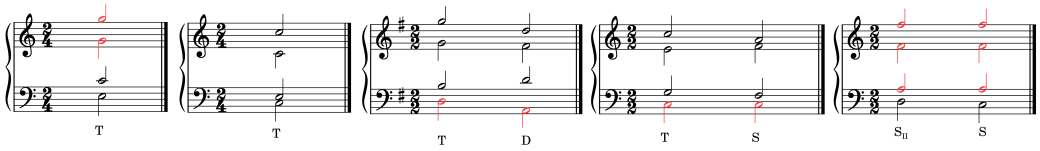


Figure 3: Examples of weak constraints violation. From left to right: i) Doubled quint but not in the bass, iii) A chord in tripled prime, iv) Two consecutive chords with quint in the bass, v) No bass movement, vi) Only one voice moved.

Added value Certain features of a harmonization improve its quality, and therefore, their occurrence positively contributes to the evaluation score.

i) *Parallel sixths* – If there is a sixth interval between two voices in a chord and a sixth interval between the same voices in the following chord.

ii) *Opposite movement of the soprano and the bass* – The soprano and the bass are the two most prominent voices in harmonization. For this reason, the opposite movement of these voices is preferred.

iii) *Opposite movement on a perfect interval* – Perfect intervals are octaves and quints. The opposite movement on such intervals is preferred.

iv) *Chord position* – Every chord can be in the closed or open position. The preferred position is open.

5.2. Modal harmonization

The problem of harmonizing modal music can be considered more complicated and advanced than harmonizing tonal music. Harmonization of modal music is characterized by a different frequency of chords in relation to the notes in the main melody. In tonal music samples, there is one chord per note in the melody. In modal music, one chord can correspond to more than one note (e.g. a sequence of notes). For this reason, condition (ii) defined in Section 4.3 must be adjusted. The note from the input voice to which the chord note (in the same voice) is matched is considered to be the first note from the sequence, which is defined by the chord label corresponding to this sequence. If any note from the sequence does not belong to the defined chord, the chord is drawn without considering the note in the input voice, but in the other voices each note must be unique (no doubling is allowed, cf. Section 4.1).

Analogously to the rules selected for tonal music (Section 5.1), the rules for modal music are also based on the modal harmony textbook [23]. Very similar or identical rules can also be found in other books on modal harmonization [11, 24]. It is worth mentioning that some rules used for modal harmonization are similar (or identical) to those used for tonal harmonization. These rules are not described again, but only their usage is indicated.

In the modal fitness function the values of $m_s = 6$, $m_w = 9$ and $m_a = 2$ are applied in Equation (2).

Strong constraints The following strong constraints are considered.

i) *No parallel (antiparallel) quints, octaves or primes* – The exception is the antiparallel intervals in the outer voices (melody and the bass).

ii) *Voices are not crossing*

iii) *No augmented interval moves*

iv) *No consecutive quart or quint interval in the bass* – If there is a quart (quint) interval between two consecutive notes, there cannot be another quart (quint) interval in the same direction between the following two notes.

v) *Correct doubled third* – If a chord is in the first inversion and has doubled third, the third can only be in the bass and the alto, or the bass and the tenor. If a chord is major, the third must not be doubled.

vi) *Correct the bass move in chord inversions* – If a chord is in the first inversion and the bass moves by quart interval, next the bass move must be in the opposite direction. If a chord is in the second inversion the bass must not move.

Weak constraints The following weak constraints are considered.

i) *Doubled prime in the bass in the first chord and the last chord*

ii) *The bass movement restriction*

iii) *Antiparallel quint and octaves in outer voices* – It is permissible for the melody and the bass to move in antiparallel quints or octaves, however, this is not preferable.

iv) *Doubled quint in chord* – When the quint is doubled in a chord, none of them should be in the bass.

v) *Melodic line smoothness*– The middle melodic lines, the alto and the tenor, should not include intervals bigger than quint between consecutive notes (in the same voice).

vi) *Limited distances between voices* – The distance between the three highest voices should not exceed an octave interval.

vii) *Chord inversion in the main cadence* – In the main cadence all chords should have prime in the bass.

viii) *Chord position before the main cadence* – Before the main cadence all chords should be in inversions or should not have prime in melody.

ix) *Movement on quint interval and octave interval* – If there is a quint (octave) interval between two voices and both voices move in the same direction on this interval, the higher voice should move by a maximum of a whole tone, lower one by at least third interval.

Examples of violation of selected strong and weak constraints are presented in Figure 4.

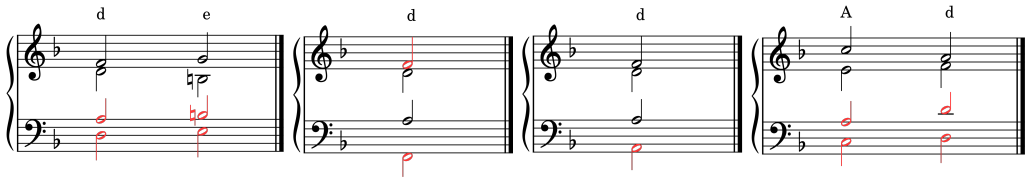


Figure 4: Examples of strong and weak constraints violation. From left to right: Strong constraints: i) Parallel quints between the bass and the tenor, v) Third incorrect doubled in a chord. Weak constraints: iv) Quint incorrect doubled in a chord, ix) Incorrect move on an octave

Added value The following added value rules are considered.

i) *Connections between chords* – If two consecutive chords have a common note, it is preferable for this note to remain in the same voice without movement. If two consecutive chords do not have a common note, it is preferable to move three voices in the direction opposite to the fourth.

ii) *Chords in the first inversion* – A chord without inversion is preferred before and after a chord in the first inversion.

6. Experimental setup and results – tonal harmonization

The examples used to tune and test the algorithm for tonal harmonization come from a harmony textbook [20]. Similar examples can be found in other harmony textbooks, as well. 18 examples (melodic lines with harmonic functions) were selected and divided into three groups:

Group 1: long examples (about 20 chords), using only basic functions, which define only three pitches (7 examples),

Group 2: short examples (about 10 chords), using basic and side functions and added pitches (4 examples),

Group 3: long examples (about 20 chords), using basic and side functions, and added pitches (7 examples).

Out of these 18 examples, 3 (one from each group) were used for parameters tuning and the remaining 15 were used in the final tests. All tests were run on a PC with IntelCore i7-9750H (2.6GHz) processor and 24GB RAM.

6.1. Parameterization

The basic values of the algorithm's parameters were selected in preliminary tests and tuned afterward. For each parameter, several values were tested (with the remaining parameters frozen at their basic values) on the 3 examples devoted to parameter tuning. Each test was run five times (with different random seeds) and returned values were averaged.

The following selections / ranges of parameters were tested (the finally selected values are bolded):

- s_p — (*population size*) — [10, 100, 500, **1000**, 1750, 2500, 3500, 5000];
- s_e — (*elite size*) — [0, **3**, 5, 10];
- p_c — (*crossover probability*) — [from 0 to 1 with step 0.1], **0.8**;
- p_m — (*mutation coefficient*) — [from 0 to l with step 1], **1** or **2**;
- p_m (additional fine-tuning) — [from 1 to 2 with step 0.1], **1.1**;
- p_s — (*selection pressure*) — [from 0.5 to 1 with step 0.1], **0.7**;
- n — (*number of generations*) — [1000, 3000, **5000**, 10000];
- t_s — (*tournament size*) — [2, **4**, 8, 10].

6.2. Number of required generations

The efficacy of the tonal version of the algorithm was checked on 15 samples that were harmonized. Table 1 shows the time (number of generations) required to find the first correct and the finally returned (best found)

solutions, respectively. In each run, the algorithm found the first correct harmonization (satisfying all strong constraints) within the first 86 generations (usually much faster).

The number of generations needed to find the correct solution varies between groups. Shorter problems, with fewer chords, were solved faster (cf. group 1 vs group 2). Likewise easier problems, using less advanced functions, turned out to be easier to solve (cf. group 1 vs group 3). Similar relationships can be observed among the finally returned solutions.

Table 1: The number of generations required to find a solution.

Group no.	Example no.	Generation number in which the result was found					
		first correct			best found (returned)		
		Mean	Min	Max	Mean	Min	Max
1	1	16.6	14	22	208.2	86	343
	2	16.8	13	22	268.2	129	744
	3	14.8	12	18	218.4	109	397
	4	16.6	14	19	390	90	803
	5	15.4	14	17	1414.5	145	3914
	6	11.2	7	14	909.6	105	2477
2	7	8.2	6	10	177.6	27	593
	8	3.2	1	5	24	13	36
	9	6.8	5	8	826.2	75	3200
3	10	33.6	19	86	1933.6	96	3576
	11	19.2	17	22	699.8	249	1224
	12	21	18	25	3098.6	2179	3838
	13	20.2	19	21	926.6	73	2507
	14	19.6	17	25	890.8	130	3334
	15	16.2	15	19	1411.6	198	2500

6.3. Running time

The running time of the algorithm is presented in Table 2, separately for each group. For testing purposes, group 4 was generated artificially by multiplying 10 times examples from groups 1 and 3 (making them 10 times longer). It can be observed from the table that the running time does not seem to depend on the complexity of the example (cf. Group 1 and Group 3), but only on its length. Furthermore, a rough comparison of the time relationship between groups 1, 2 and 4 suggests its quasi-linear dependence on the harmonization length.

Table 2: The average algorithm's running time in seconds (harmonization time) for each group.

Group 1			Group 2		
Mean	Min	Max	Mean	Min	Max
531.9	485.5	567.0	225.8	180.7	252.1
Group 3			Group 4		
Mean	Min	Max	Mean	Min	Max
536.2	508.7	582.4	5633.2	5126.5	6357.8

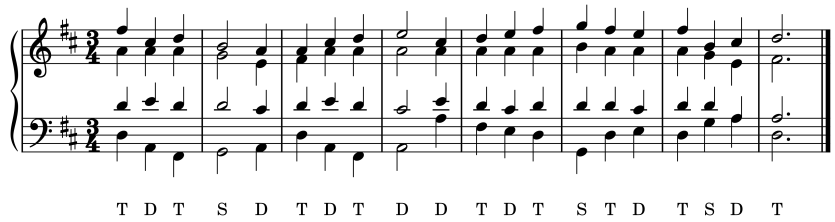
6.4. Human expert evaluation including aesthetic aspects

Every created harmonization has its score assigned as a result of the fitness function evaluation. However, this score only indicates how well a harmonization satisfies the *formal* fitness function requirements (the considered rules) and does not indicate directly how good is the harmonization in strictly musical terms (*how well does it sound*). For this reason, all 15 generated samples were additionally evaluated by a human expert - a harmony teacher. The teacher's method of evaluating the solutions was the same as when evaluating students' works. Harmonizations were evaluated on a 5-point scale, from 1 (the lowest score) to 5 (the highest score).

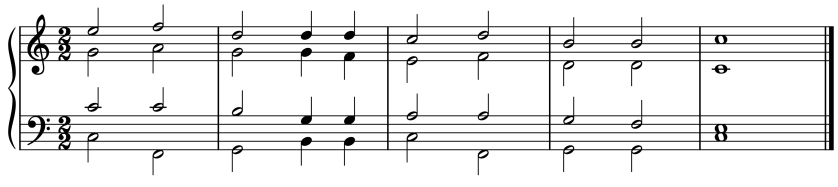
On the one hand, the expert evaluated the theoretical correctness of the constructed solutions, but on the other hand, based on many years of practical experience he/she was also able to evaluate aesthetic and creative elements. In other words, the expert's evaluation was comprehensive and concerned both major harmonization aspects: its construction and sound.

Out of all created harmonizations, eight were graded 5, six 4.5, and one 4. Two types of problems were identified by the expert in downgraded harmonizations. The first one was the lack of adherence to certain theoretical rules. The most common problem was reaching a fifth in the bass of a chord other than by a movement of a second interval. However, **none of the violated rules indicated by the experts was included in the fitness function**, which means that their fulfillment was not directly imposed, and the algorithm could only meet them by chance. In contrast, the rules used in the fitness function as strong constraints were strictly observed. This means that extending the fitness function with additional rules should potentially solve the problem highlighted by the expert.

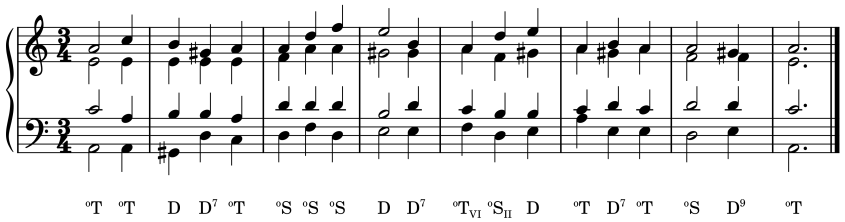
The remaining expert's remarks, formulated in a few cases, were related



(a) Long example, from the first group, graded 5.



(b) Short example, from the second group, graded 4.5.



(c) Long example, from the third group, graded 5.

Figure 5: Example harmonizations created by the algorithm.

to minor sound issues, mainly to chords combinations (most often $\mathbf{D}^7$ and $\mathbf{T}_{\mathrm{VI}}$). Requirements of this type are hard to be formally expressed in the fitness function which makes their enforcement in the resulting harmonizations difficult. However, also, in this case, an attempt could be made to formulate a rule and add it to the Added Value module, which could help eliminate this problem.

Figure 5 presents three examples of generated harmonizations rated 5, 4.5 and 5, respectively. The indicated place that could be harmonized differently in example 5b are chords fifth and sixth ($\mathbf{D}^7$, $\mathbf{T}_{\mathrm{VI}}$). The created solution is not incorrect, although a better sound would be achieved by placing the prime of $\mathbf{D}^7$ chord in the lowest voice.

To summarize, high grades given by the harmony teacher support the claim that the vast majority of created harmonizations are not only the-

oretically but also sonically correct. Seven harmonizations contain minor imperfections, some of which should be easily resolved by extending the fitness function. It is also worth mentioning that according to the expert's opinion, **the generated solutions do not expose any features of the automatic origin and fully correspond to the products of human harmonization.**

6.5. Modeling the solution

The fitness function consists of 22 smaller functions, each of which addresses and evaluates one particular aspect of harmonization. Each of these evaluations is multiplied by a respective weight (negative for a penalty and positive for a reward). Modifying these weights allows for modeling the solution by increasing/decreasing the relevance of a given aspect with respect to the others.

As an example, Figures 6a and 6b present two harmonizations of the same melodic line with different emphasis put on the reward for chords in the open position. In the first case, the base fitness function (the one used throughout the paper) was applied and in the second one, the respective coefficient was 3 times bigger, so as to reinforce the relevance of this feature. In both figures chords in the open position are marked in green. Indeed, the number of chords in the open position in Figure 6b is clearly greater than in Figure 6a.

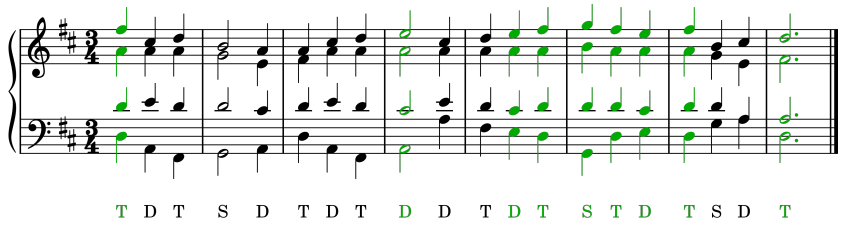
The above example confirms the possibility of modeling harmonization, so as to focus on specific aspects. However, it is important to note that too strong reinforcement of specific features may result in others not being met, despite the overall increase of the fitness value.

7. Experimental setup and results – modal harmonization

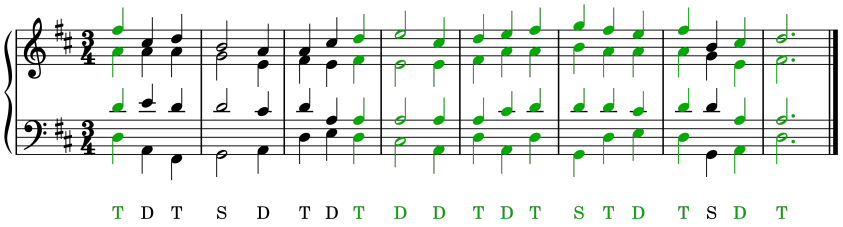
The harmonization of modal music is based on different rules than the harmonization of tonal music. For this reason, a readjustment of the parameters of the algorithm is required. All examples used for parameterization and testing were different from the examples used for tonal harmonization. All of them were taken from [25]. A total of 11 examples have been gathered, which were divided into 3 groups:

Group 1: short examples (about 10 chords) – 4 examples,

Group 2: medium examples (about 15 chords) – 3 examples,



(a) Harmonization created with a base fitness function.



(b) Harmonization created with an enhanced reward for the open position.

Figure 6: Modeling the solution. Chords in open positions are marked in green.

Group 3: long examples (about 20 chords) – 4 examples.

7.1. Parameterization

Parameterization of the algorithm was performed in exactly the same way as for tonal harmonization (Section 6.1). Three examples, one per group, were used for this paper. The remaining 8 was used exclusively in the test phase. The following ranges of parameters were tested (the finally selected values are bolded):

- s_p — (*population size*) — [10, 100, 500, **1000**, 1750, 2500, 3500, 5000];
- s_e — (*elite size*) — [0, **3**, 5, 10];
- p_c — (*crossover probability*) — [from 0 to 1 with step 0.1], **0.8**;
- p_m — (*mutation coefficient*) — [from 0 to l with step 1], **1** or **2**;
- p_m (fine-tuning) — [from 1 to 2 with step 0.1], **1**;
- p_s — (*selection pressure*) — [from 0.5 to 1 with step 0.1], **0.6**;
- n — (*number of generations*) — [1000, 3000, **5000**, 10000];
- t_s — (*tournament size*) — [2, **4**, 8, 10].

7.2. Number of required generations

The numbers of generations required to find the first correct and the finally returned (best found) solutions, respectively are presented in Table 3. In each run, the algorithm found a harmonization, which satisfies all strong constraints, within the first 20 generations. The number of generations needed to find the correct solution (and the best one) varies between groups. The longer the harmonization problem, the longer the time needed to find the correct or best solution.

Table 3: The number of generations required to find a solution.

Group no.	Example no.	Generation number in which the result was found					
		first correct			best found (returned)		
		Mean	Min	Max	Mean	Min	Max
1	1	5.2	3	9	341.2	27	1455
	2	6.6	5	8	44.8	28	67
	3	8.3	7	9	374.8	95	890
2	4	7	7	7	662.5	577	748
	5	14.4	10	19	163.6	92	340
3	6	11.4	9	13	1376.2	197	2930
	7	12.8	11	15	1781.4	233	4327
	8	14.2	11	18	1496.4	152	4448

7.3. Running time

The running times of the modal version of the algorithm are summarized in Table 4, separately for each group. Analogously to Section 6.3, group 4 was generated artificially by multiplying 10 times the examples from group 1 (examples created in this way were about 100 chords long). Since the groups differed only in length (the complexity of examples did not vary between groups) the running times suggest a quasi-linear dependence between the calculation time and the example length.

7.4. Human expert evaluation – technical and aesthetic aspects

As with the results for tonal harmonization (Section 6.4), an expert evaluation of generated modal harmonizations was performed. A harmony teacher evaluated all 8 test on a 5-point scale (the same as for tonal harmonization). Out of 8 harmonizations, one was graded 5, one – 4.5, three – 4, and three –

Table 4: The average algorithm's running time in seconds (harmonization time). Group 4 was generated artificially by multiplying 10 times examples from group 1.

Group 1			Group 2		
Mean	Min	Max	Mean	Min	Max
244.1	197.0	298.8	354.9	329.8	403.1
Group 3			Group 4		
Mean	Min	Max	Mean	Min	Max
476.9	422.2	544.6	2612.4	2073.2	3103.1

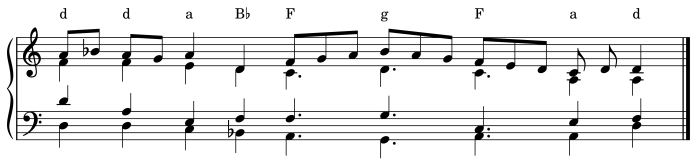
3.5, with an average grade of 4. Figure 7 presents 3 examples graded 5, 3.5 and 4.5, respectively.

The expert pointed out two general problems present in most of the resulting harmonizations. The first one concerned the leading of the tenor, the fluidity of which should have been better. The second one was related to the distance between the alto and the tenor, which usually should be smaller (preferably not exceeding a sixth). However, similarly to the algorithm version for tonal harmonization, **none of the issues pointed out by the expert, were explicitly addressed in the fitness function**. In both downgraded results presented in Figures 7b and 7c), respectively the identified problem can be observed. In the lowest rated example (Figure 7b) an additional concern is that the distance between the tenor and the bass is often too large (preferably should not exceed an octave).

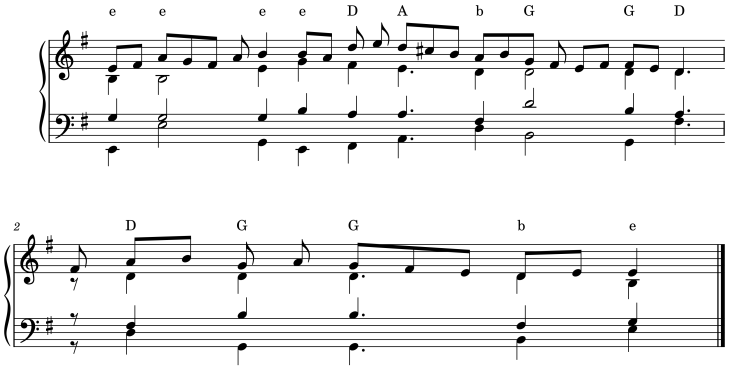
In summary, the assessments made by the expert support the statement that the harmonizations are for the most part theoretically correct but sometimes suffer from some lacks in the sonic domain. The most important complaint about sonic weaknesses was the lack of use of passing notes and chords with added notes. Well-sounding, man-made harmonizations contain such enhancements, making the sound of the whole harmonization better.

7.5. Enriching the harmonization through seventh chords

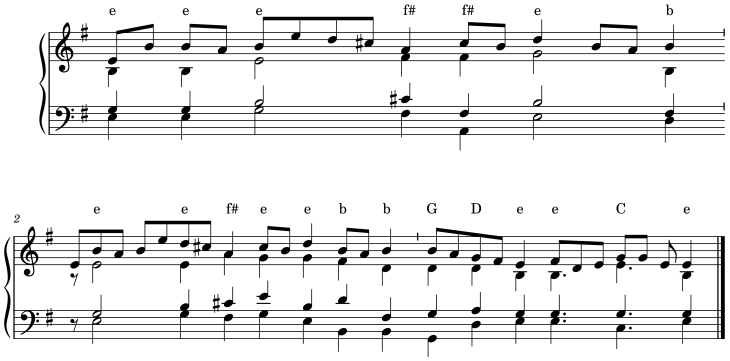
Following up on expert comments to enrich the sound of harmonization, an attempt was made to expand the used chords. One way to do it is to use more complex chords than the basic ones (consisting of more than 3 notes). As an extension of the baseline algorithm, an attempt was made to use chords with an added seventh. The addition of seventh chords to harmonization is based on probability. Changing a basic chord to a seventh chord (and vice



(a) Short example, from the first group, graded 5.



(b) Medium example, from the second group, graded 3.5.



(c) Long example, from the third group, graded 4.5.

Figure 7: Example modal harmonizations created by the algorithm.

versa) is done by means of the mutation operator. If a given chord is mutated, then with probability p_{sc} it can be converted to a seventh chord (if it was a basic chord) or to a basic chord (if it was a seventh chord). The probability $p_{sc} = 0.164$ was calculated based on the modal music harmonizations included in [25] and equaled the frequency of using seventh chords relative to all chords in harmonizations. Utilization of such chords requires an additional extension of the fitness function, since the use of seventh chords is

also bounded by certain rules. Accordingly, the fitness function was extended with the following rules.

Strong constraints The following strong constraints were added.

vii) *No seventh in the soprano or the bass* – Added seventh must always be in the alto or the tenor.

viii) *Seventh chord preparation* – The pitch of the added seventh note must be used in the previous chord and in the same voice.

ix) *No augmented quint in the seventh chord* – If the seventh chord is to be used, there must not be an augmented quint between any of its two pitches.

x) *Correct seventh resolution* – Seventh must move down a second or not move at all (only if the seventh chord is in the first inversion).

Weak constraints The following weak constraint was added.

x) *No second inversion* – If the seventh chord is used, it is preferable that it is without or in the first inversion (prime or third in the bass).

Added value The following added value rule was added.

iii) *Use of a seventh chord* – Since the use of a seventh chord enriches the harmonization, such chords are preferred.

The results generated by the algorithm with the extended fitness function are presented in Figure 8. Chords with added seventh are marked in green. Despite rules that severely restrict the use of chords with an added seventh, the algorithm has found positions where it is possible to use them in such a way that none of the rules is broken. It is also possible to encourage the algorithm to use more chords with an added seventh in the way described in Section 6.5. However, it is important to keep in mind that this can result in violating some weak constraints.

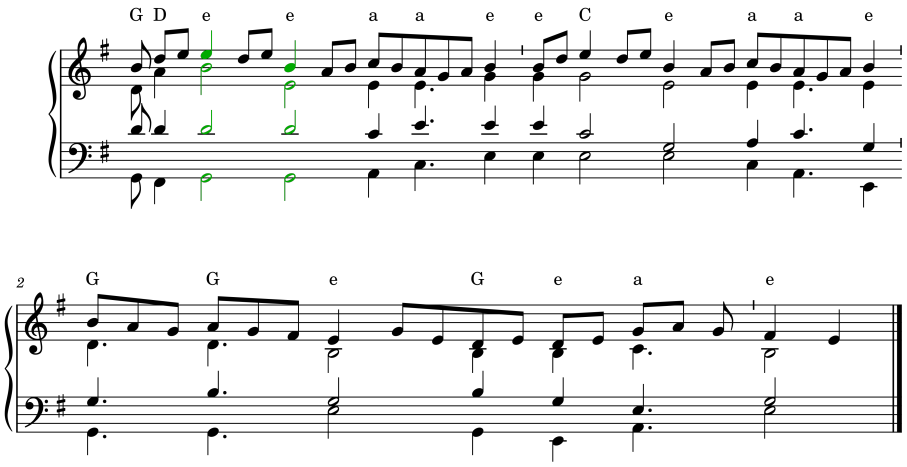


Figure 8: Enriching the harmonization. Chords with added seventh are marked in green.

8. Comparison of results

The results presented for both versions of the algorithm (modal and tonal) can be compared by considering three aspects: efficiency, running time and human expert evaluation.

In the case of efficiency, it can be seen that the algorithm very quickly finds the first correct solutions (regardless of the version). The best (returned) solutions were sometimes found much later (even after 4 000 generations), but both versions of the algorithm do not particularly differ regarding this aspect.

The running time is also similar for both versions of the algorithm. For the analogous length of examples, e.g. group 2 of the tonal version and group 1 of the modal version, the calculations took similar time. This suggests that the running time is not strongly dependent on the number of rules in the fitness function (22 for tonal and 16 for modal) nor on the used harmonic functions/chord labels (complexity of the problem). Keeping the rules relatively simple makes the running time mainly dependent on the length of the harmonized example.

The last aspect is the evaluation by the human expert. In this case, the algorithm performed quite well in both cases (a high rating for tonal harmonization and a lower, but still good rating for modal harmonization). It is worth noting that in no case did the algorithm break the defined rules,

and the observations made by the expert referred to the elements undefined (or differently defined) in the fitness function.

9. Conclusions and future work

The problem of melodic line harmonization considered in the paper is one of the stages of the music composition process and as such requires creativity. The outcome (a created harmonization) is generally hard to assess due to its subjective nature. With the above caution, this article points out that it is possible to achieve human-level performance in this task (melody harmonization) using evolutionary computation.

The proposed evolutionary algorithm creates harmonizations by means of carefully designed evolutionary operators and the fitness function that reflects music theory rules. The fitness function is composed of three general terms: (1) the rules that must be fulfilled if harmonization is to be considered correct, (2) additional rules that are expected to be fulfilled, otherwise the score of the harmonization is lowered, (3) the rules whose fulfillment further improves the resulting harmonization. Moreover, the modular design of the fitness function makes it easily extendable with other music rules and allows to emphasize various aspects in the resulting harmonization. Furthermore, the proposed algorithm does not require any training, which makes it independent from the styles/biases implicitly present in the training data.

The usefulness of the algorithm was demonstrated on two different types of harmonization (tonal and modal). In the case of tonal harmonization, the results generated by the algorithm are correct not only in terms of the music theory but also sonically. According to the expert's opinion, obtained solutions do not exhibit any features of artificial origin and fully correspond to the products of human harmonization. Additionally, the process of computationally generating harmonizations is relatively fast. In the case of modal harmonization the results generated by the algorithm are worse than for tonal harmonization, but still positively assessed. Moreover, an attempt has been made to address the problems identified by an expert related to the harmonization sound diversity.

Our future plans involve removing harmonic functions/chord labels added to the notes and performing harmonization of the melodic line itself, so as to enable the creation of choral adaptations in small ensembles or provide harmonization assistance for less advanced musicians.

References

[1] J. Mycka, A. Żychowski, J. Mańdziuk, Human-level melodic line harmonization, in: D. Groen, C. de Mulatier, M. Paszynski, V. V. Krzhizhanovskaya, J. J. Dongarra, P. M. A. Sloot (Eds.), Computational Science – ICCS 2022, Springer International Publishing, Cham, 2022, pp. 17–30.

[2] G. A. Wiggins, Searching for computational creativity, New Generation Computing 24 (3) (2006) 209–222.

[3] M. Kaliakatsos-Papakostas, A. Floros, M. N. Vrahatis, Artificial intelligence methods for music generation: a review and future perspectives, Nature-Inspired Computation and Swarm Intelligence (2020) 217–245.

[4] F. Carnovalini, A. Rodà, Computational creativity and music generation systems: An introduction to the state of the art, Frontiers in Artificial Intelligence 3 (2020) 14.

[5] O. Vechtomova, G. Sahu, D. Kumar, Lyricjam: A system for generating lyrics for live instrumental music, in: Proceedings of the 11th International Conference on Computational Creativity, 2021, pp. 122–130.

[6] J. Mańdziuk, M. Goss, A. Woźniczko, Chopin or not? a memetic approach to music composition, in: 2013 IEEE Congress on Evolutionary Computation, 2013, pp. 546–553.

[7] J. Mańdziuk, A. Woźniczko, M. Goss, A neuro-memetic system for music composing, in: L. Iliadis, I. Maglogiannis, H. Papadopoulos (Eds.), Artificial Intelligence Applications and Innovations, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 130–139.

[8] F. Pachet, P. Roy, Musical harmonization with constraints: A survey, Constraints 6 (1) (2001) 7–19.

[9] E. Agmon, The Languages of Western Tonality, Springer, 2013.

[10] R. L. Crocker, A History of Musical Style, Dover Publications, Inc., NY, 2018.

[11] H. Potiron, Treatise on the Accompaniment of Gregorian Chant, Society of St. John the Evangelist, Desclée, 1933.

[12] H. Hild, J. Feulner, W. Menzel, Harmonet: A neural net for harmonizing chorales in the style of J.S.Bach, NIPS'91: Proceedings of the 4th International Conference on Neural Information Processing Systems (1991) 267—274.

[13] A. Moray, C. K. I. Williams, Harmonising chorales by probabilistic inference., Advances in Neural Information Processing Systems 17 (2005) 25–32.

[14] L. Yi, J. Goldsmith, Automatic generation of four-part harmony., Proceedings of the Fifth UAI Bayesian Modeling Applications Workshop (2007).

[15] B. Evans, S. Fukayama, M. Goto, N. Munekata, T. Ono, Autochoruscreator : Four-part chorus generator with musical feature control, using search spaces constructed from rules of music theory, Proceedings ICMC (2014).

[16] A. Freitas, F. Guimaraes, Melody harmonization in evolutionary music using multiobjective genetic algorithms., Proceedings of the Sound and Music Computing Conference. (2011).

[17] R. D. Prisco, G. Zaccagnino, R. Zaccagnino, Evocomposer: an evolutionary algorithm for 4-voice music compositions., Evolutionary computation 28 (3) (2020) 489–530.

[18] O. Olseng, B. Gambäck, Co-evolving melodies and harmonization in evolutionary music composition., International Conference on Computational Intelligence in Music, Sound, Art and Design. (2018).

[19] J. Mycka, Melodic line harmonization - source code (2022).
URL `https://github.com/MelodicLineHarmonization/melodicLineHarmonization.git`

[20] K. Sikorski, Harmonia cz. 1, PWM, 2020.

[21] H. Benham, A Student's Guide to Harmony and Counterpoint, Rhinegold Publishing Limited, 2006.

[22] N. Rimsky-Korsakov, Practical Manual of Harmony, C. Fischer, 2005.

[23] W. Lewkowicz, Harmonia gregoriańska, Księgarnia św. Wojciecha, 1958.

[24] E. Lapierre, Gregorian Chant Accompaniment: A New And Simple Approach According To The Theory Of The Basic Modal Intervals, Literary Licensing, LLC, 2011.

[25] H. Potiron, Accompagnement du Kyriale Vatican par Le R.P. Dom Jean Hébert Desrocquettes: Société Saint Jean l'Évangéliste, Desclée et Cie, 1929.

Jan Mycka received his B.Sc. and M.Sc. degrees in Computer Science from the Faculty of Mathematics and Information Science of the Warsaw University of Technology (WUT), in 2020 and 2021. Currently Ph.D. student at WUT Doctoral School. His research interests include the application of artificial intelligence methods in music-related problems.

Adam Żychowski received his B.Sc. and M.Sc. degrees in Computer Science from the Faculty of Mathematics and Information Science, Warsaw University of Techonology, Warsaw, Poland in 2014 and 2015, respectively. He is currently pursuing PhD in Computer Science. His research interests include Game Theory and Computational Intelligence methods especially artificial neural networks and evolutionary algorithms.

Jacek Mańdziuk, Ph.D., D.Sc., IEEE Senior Member, received M.Sc. (Honors) and Ph.D. from the Warsaw University of Technology (WUT), Poland in 1989 and 1993, resp., and D.Sc. degree in Computer Science from the Polish Academy of Sciences in 2000. In 2011 he was awarded the title of Professor Titular. He is full professor at the Faculty of Mathematics and Information Science, WUT, Head of Division of Artificial Intelligence and Computational Methods, and Head of Doctoral Program in Computer Science at this faculty.

Prof. Mańdziuk was a recipient of the Fulbright Senior Research Award (UC Berkeley and ICSI Berkeley, USA) and the Robert Schuman Foundation Fellowship (CNRS, Besançon, France). He was a visiting professor at Nanyang Technological University (Singapore), University of New South Wales (Australia), Yonsei University (Korea) and University of Alberta (Canada).

His research interests include application of Computational Intelligence and Artificial Intelligence methods to games, dynamic and bilevel optimization problems, and human-machine cooperation in problem solving. He is also interested in the development of general-purpose human-like learning and problem-solving methods. He is the author of 3 books and 170+ research papers. For more information please visit http://www.mini.pw.edu.pl/∼mandziuk