Human-level Melodic Line Harmonization

Jan Mycka[1[0000−0003−2250−021X]], Adam Żychowski[1[0000−0003−0026−5183]], and Jacek Mańdziuk[1,2[0000−0003−0947−028X]]

[1] Warsaw University of Technology, Warsaw, Poland
[2] AGH University of Science and Technology, Krakow, Poland

Abstract. This paper examines potential applicability and efficacy of Artificial Intelligence (AI) methods in automatic music generation. Specifically, we propose an Evolutionary Algorithm (EA) capable of constructing melodic line harmonization with given harmonic functions, based on the rules of music composing which are applied in the fitness function. It is expected that harmonizations constructed in accordance to these rules would be formally correct in terms of music theory and, additionally, would follow less-formalised aesthetic requirements and expectations. The fitness function is composed of several modules, with each module consisting of smaller parts. This design allows for its flexible modification and extension. 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 modelled by means of adjusting the relevance of particular fitness function components. The proposed method generates solutions which are technically correct (i.e. in line with music harmonization theory) and also "nice to listen to" (i.e. aesthetically plausible) as assessed by an expert - a harmony teacher.

Keywords: evolutionary algorithm · harmonization · music generation

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* [19] to mimic creative behaviors of AI agents. One of the fields in which AI creativity has been intensively developed is music [9]. AI methods are applied to create new compositions [4] or complement / expand existing pieces [18]. In both tasks they managed to demonstrate human-level performance. Another line of research is imitation of a style of a particular composer, e.g. F. Chopin [10,11]. 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 the notes, both vertically (simultaneous sound) and horizontally (time sequence).

Harmonization is a creative process in which the main role is played by intuition, talent and experience of a musician [14]. The process is constrained by

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_2

various rules resulting from music theory and centuries of musical practice. 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. fulfil music harmonization rules; (2) to this end we design a specific fitness function that corresponds to theoretical rules of harmonization and can be easily extended or tuned to reflect the desired aspects of the resulting harmonizations; (3) the proposed fitness function covers not only harmonic but also melodic aspects (voices leading) of created harmonizations; (4) through modification of the weights of individual components of the fitness function, solutions can be tuned to meet the desired requirements; (5) the resulting harmonizations are evaluated by an (human) expert and assessed as both technically correct and **possessing human-like characteristics**.

2 Problem definition

Harmonization is a process of creating an accompaniment for a given melody line. Created accompaniment consists of three new melodic lines (voices). Usually, the highest voice (soprano) is the base for a harmonization and the other three voices (alto, tenor and bass) are to be created. Four notes, one from each voice, form a chord. Creating harmonization is mostly based on musician's experience and intuition. However, throughout the years many theoretical rules that harmonization has to fulfill were developed [2, 5]. 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. The harmonization is generated not for the raw melody, but for the melody extended with harmonic functions assigned to each note. The harmonic functions determine which notes (pitches) should constitute a chord. 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 [8] or hidden Markov models [12]. 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.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_2

The algorithm described in this paper relies on a different method, the evolutionary algorithm, 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 [20] and evaluates connections between two consecutive chords. The evaluation rules are based on music theory. Yet another work [6] hybridizes heuristic rules with dynamic programming method.

The use of EAs in the melody harmonization problem has been considered in a few recent works [7,13,15], however each of them addresses a slightly different problem formulation rendering a direct comparison impossible. In [7] 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 [13], only chords are created, not entire melodic lines. In [15], 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 what renders a direct comparison impossible. Hence, the assessment of the resulting melody lines proposed in the paper is two-fold: (1) by means of a numerical fitness value assigned by the algorithm, and (2) by a human expert - a harmony teacher.

4 Proposed method

To address the harmonization problem we propose the EA that maintains a population of individuals (candidate solutions). In each generation, the current candidate solutions 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.

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

4.1 Problem search space - *admissible* chords

The input data is a soprano melodic line (the highest voice) with a particular harmonic function assigned to each note. This function indicates at least three and at most five pitches, which have to be used in a chord. If the function

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_2

indicates only three pitches, 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 (soprano). In addition, an ambitus (the lowest and highest possible pitch of a voice) is defined for each voice, derived from the theory. Thus, for each harmonic function it is possible to define a set of all admissible chords corresponding to this function. The evolutionary operators (mutation, crossover), as well as the construction of the initial population are based on these pre-defined sets of admissible chords associated with particular harmonic functions.

4.2 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 function assigned to the completed note,
(ii) the note given in the input voice is located in the chord in the same voice (soprano).

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

The first population is generated randomly. Every individual consists of randomly chosen 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 individuals found in the entire run of the algorithm will not be lost. The rest of the population is generated by means of selection procedure and genetic operators (mutation and crossover), according to Algorithm 1.

```
1  GenerateNewPopulation (P)
2      CalculateFitnessValues(P)  // calculates fitness of each individual
3      P_new ← GetElite(P, s_e)    // population of new individuals
4      while |P_new| < |P| do
5          c_1 ← Selection(P)
6          if rand([0,1]) < p_c then // crossover
7              c_2 ← Selection(P)
8              c_new = Crossover(c_1, c_2)
9          else
10             c_new ← c_1
11         end
12         c_new ← Mutation(c_new)
13         P_new = P_new ∪ {c_new}
14     end
15     return P_new
```

Algorithm 1: Next generation population procedure.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_2

4.3 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 is calculated. Let's define by x_i the i-th individual of the tournament according to the fitness value 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*.

4.4 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 and p_m is the mutation coefficient (algorithm's parameter). Mutating a chord consists in its replacement by another chord uniformly sampled among those that meet requirements (i)-(ii). An example of mutation is shown in Fig. 1.

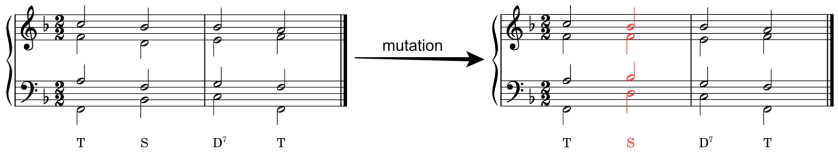


Fig. 1: Example of mutation. Second chord has changed.

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. The crossover method and its example application are presented in Fig. 2.

4.5 Fitness function

The fitness function is used 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). The set of considered harmonization rules includes those that are taught at music schools in the first years of

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_2

Crossover$_2$ (c_1, c_2)
  $k \leftarrow \text{rand}(1, \ldots, l)$
  for $i \in [1, \ldots, l]$ **do**
    if $i < k$ **then**
      $c[i] = c_1[i]$
    else
      $c[i] = c_2[i]$
    end
  end
  return c

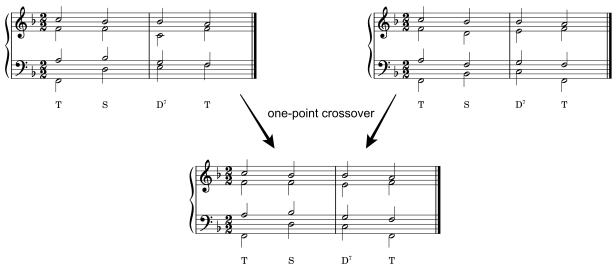


Fig. 2: One-point crossover: left - pseudocode, right - example with $k = 2$.

harmonization classes. All of them come from a harmony textbook [17]. Similar rules can be found, for instance, in [3, 16].

As demonstrated in section 5 the selected set of rules allows achieving effective harmonizations, highly graded by the human expert in terms of both formal and aesthetic aspects. Additionally, a modular design of the fitness function allows its easy extension by means of adding new rules, if required.

Each rule used in the construction of the fitness function is assigned a value that affects the final score of the generated harmonization. These values indicate the importance of particular rules. 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 3 main modules, each referring to specific rules.

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 as correct.
2. Weak constraints C_w (lower penalty terms) - constraints derived from 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:

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

where $\phi_i(X) \leq 0$ is the penalty for not fulfilling strong constraint $i, i = 1, \ldots, m_s$, $\chi_j(X) \leq 0$ is the penalty for not fulfilling weak constraint $j, j = 1, \ldots, m_w$,

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_2

$\psi_k(X) \geq 0$ 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, $m_s = 9$, $m_w = 9$, $m_a = 4$. Please consult the publicly-available source code [1] for a detailed implementation of the above 22 fitness sub-functions.

Strong constraints Strong constraints define harmonization in terms of acceptability. If any of these constraints is not fulfilled then harmonization cannot be considered correct. The following strong constraints are considered (selected examples are presented in Fig. 3).

i) Doubled prime in the first and last chord – The first and last chord occurring in a harmonization usually is 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 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 septim must be resolved down by a second, (c) a ninth must be resolved down by a second, (d) if chord is dominant third must be resolved up by minor second.
vi) No parallel (or antiparallel) quints, octaves or primes – If there is a fifth interval between two voices in a chord, there cannot be a fifth 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 bass note – The penultimate chord in harmonization is usually a 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.

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 are presented in Fig. 4).

i) Doubled quint in bass – When the quint is doubled in a chord, one of these quints should be in bass.
ii) No quint in bass on on-beats – On-beats are sorted by their importance. Quint in bass on on-beat is not preferable.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_2

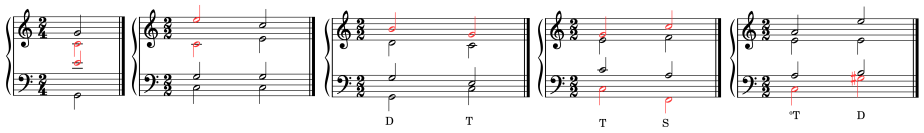


Fig. 3: Examples of strong constraints violation. From left to right: ii) Crossed alto and tenor, iii) Distance between soprano and alto exceeds an octave, v) Incorrect resolution of thirds, vi) Antiparallel quints between soprano and bass, ix) Augmented jump in bass.

iii) No tripled prime in tonic function – It is permissible to triple prime in the last chord. However, this is not preferable.
iv) No consecutive chords on quint – In harmonization chords that have a fifth in the bass can occur. However, two such chords should not follow each other directly.
v) Bass movement – The lowest voice (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 two following chords is preferred, so that the harmonization does not sound static.
vii) No septim interval – There should not be a septim 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, alto and tenor, should be conducted smoothly.
ix) Restricting bass movement – For the bass voice a maximum interval it can take in two consecutive moves is tenth.

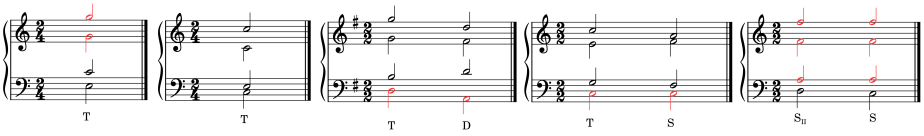


Fig. 4: Examples of weak constraints violation. From left to right: i) Doubled quint but not in bass, iii) A chord in tripled prime, iv) Two consecutive chords with quint in bass, v) No bass movement, vi) Only one voice moved.

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.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_2

ii) Opposite movement of soprano and bass – Soprano and 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 move on such intervals is preferred.
iv) Chord position – Every chord can be in the closed or open position. The preferred position is open.

5 Experimental setup and results

The examples used to tune and test the algorithm come from a harmony textbook [17]. 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:
1. long examples (about 20 chords), using only basic functions (7 examples),
2. short examples (about 10 chords), using basic and side functions and added pitches (4 examples),
3. long examples (about 20 chords), using basic and side functions and added pitches (7 examples).

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

5.1 Parameterization

The algorithm parameters were chosen experimentally based on preliminary tests. For each parameter several values were tested (with the remaining parameters frozen at their basic values) on the 3 examples devoted for 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 (fine tuned) — [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].

5.2 Efficacy of the algorithm - formal aspects

The efficacy of the algorithm was checked on 15 samples which were harmonized by the algorithm. Table 1 shows the time required to find the first correct and

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_2

the finally returned (best found) solutions, resp. In each run the algorithm found the 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 fewer 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.

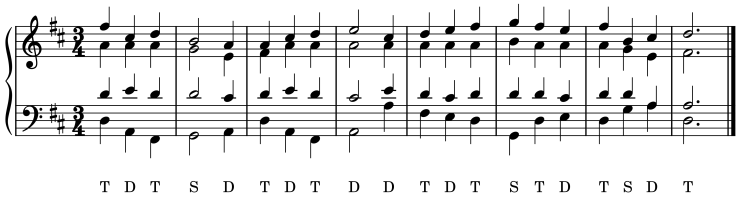
Group no.	Example no.	Generation number in which the result was found					
		correct			returned		
		Mean	Min	Max	Mean	Min	Max
1	1	16.6	14	22	208.2	86	343
	2	16.8	13	22	268.2	129	744
	3	14.8	12	18	218.4	109	397
	4	16.6	14	19	390	90	803
	5	15.4	14	17	1414.5	145	3914
	6	11.2	7	14	909.6	105	2477
2	7	8.2	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

5.3 Human expert evaluation including aesthetic aspects

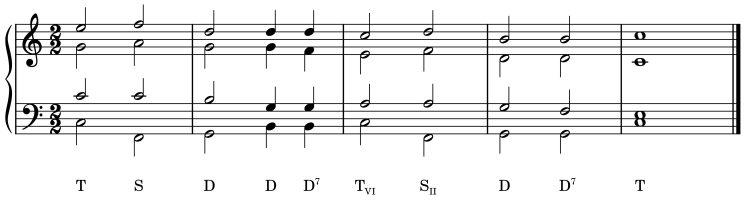
Every created harmonization has its score assigned as a result of the fitness function evaluation. However, this score only indicates how well a harmonization satisfies the *formal* fitness function requirements and does not indicate directly how good is the harmonization in strictly musical terms (*how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

On the one hand, the expert evaluated theoretical correctness of the constructed solutions, but on the other hand, based on many years of practical experience he/she also evaluated aesthetic and creative elements. In other words, the expert's evaluation was comprehensive and concerned both major harmonization aspects: its construction and sound. Figure 5 presents three examples

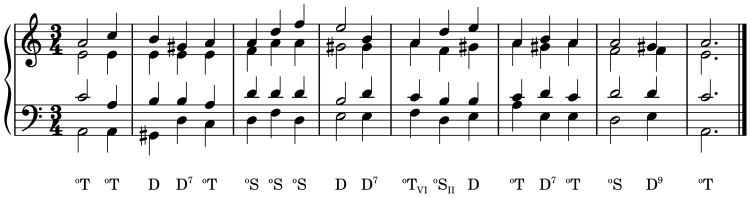
ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_2



(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.

Fig. 5: Example harmonizations created by the algorithm.

of generated harmonizations rated 5, 4.5 and 5, resp. Out of all created samples, eight were graded 5, six 4.5, and one 4. Two types of problems were distinguished in downgraded harmonizations. 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 a movement by a second. However, **none of the violated rules identified by the experts were 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 indicated in the fitness function as strong constraints were strictly observed. This means that extending the fitness function with additional rules should potentially solve this problem.

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

From the presented examples two were rated 5 (examples 5a, 5c), and one 4.5 (example 5b). 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

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_2

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 theoretically but also sonically correct. The 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 automatic origin and fully correspond to the products of human harmonization.**

5.4 Running time

The running times of the algorithm are summarized in Table 2, separately for each group. It can be observed from the table that the running time does not depend on the complexity of the example, but only on its length. Furthermore, a rough comparison of 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). Group 4 was generated artificially by multiplying 10 times examples from groups 1 and 3.

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

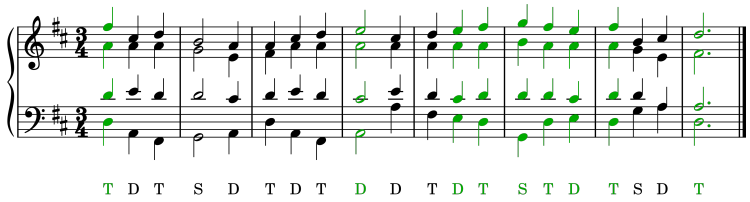
5.5 Modeling the solution

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.

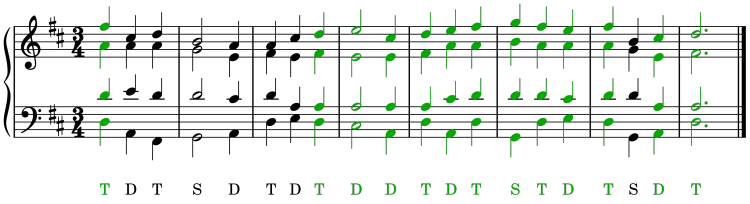
As an example, Figs. 6a and 6b present two harmonizations of the same melodic line with different emphasis put on the reward for chords in the open position. In the first case the base fitness function (the one used throughout the paper) was applied and in the second one the respective coefficient was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

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_2



(a) Harmonization created with a base fitness function.



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

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

6 Conclusions and future work

The problem of melodic line harmonization considered in this paper is part of the music composition process and as such requires creativity. The outcome (a melody) 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) the rules that should be fulfilled, otherwise the score of harmonization is lowered, (3) rules whose fulfillment improves harmonization. The 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.

Harmonizations generated by the algorithm are correct not only in terms of 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, as presented in Table 2, the process of computationally generating harmonizations is relatively fast.

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

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_2

References

1. https://github.com/MelodicLineHarmonization/melodicLineHarmonization.git
2. Agmon, E.: The Languages of Western Tonality. Springer (2013)
3. Benham, H.: A Student's Guide to Harmony and Counterpoint. Rhinegold Publishing Limited (2006)
4. Carnovalini, F., Rodà, A.: Computational creativity and music generation systems: An introduction to the state of the art. Frontiers in Artificial Intelligence **3**, 14 (2020)
5. Crocker, R.L.: A History of Musical Style. Dover Publications, Inc., NY (2018)
6. Evans, B., Fukayama, S., Goto, M., Munekata, N., Ono, T.: Autochoruscreator : Four-part chorus generator with musical feature control, using search spaces constructed from rules of music theory. Proceedings ICMC (2014)
7. Freitas, A., Guimaraes, F.: Melody harmonization in evolutionary music using multiobjective genetic algorithms. Proceedings of the Sound and Music Computing Conference. (2011)
8. Hild, H., Feulner, J., Menzel, W.: 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 pp. 267—274 (1991)
9. Kaliakatsos-Papakostas, M., Floros, A., Vrahatis, M.N.: Artificial intelligence methods for music generation: a review and future perspectives. Nature-Inspired Computation and Swarm Intelligence pp. 217–245 (2020)
10. Mańdziuk, J., Goss, M., Woźniczko, A.: Chopin or not? a memetic approach to music composition. In: 2013 IEEE Congress on Evolutionary Computation. pp. 546–553 (2013)
11. Mańdziuk, J., Woźniczko, A., Goss, M.: A neuro-memetic system for music composing. In: Iliadis, L., Maglogiannis, I., Papadopoulos, H. (eds.) Artificial Intelligence Applications and Innovations. pp. 130–139. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
12. Moray, A., Williams, C.K.I.: Harmonising chorales by probabilistic inference. Advances in Neural Information Processing Systems **17**, 25–32 (2005)
13. Olseng, O., Gambäck, B.: Co-evolving melodies and harmonization in evolutionary music composition. International Conference on Computational Intelligence in Music, Sound, Art and Design. (2018)
14. Pachet, F., Roy, P.: Musical harmonization with constraints: A survey. Constraints **6**(1), 7–19 (2001)
15. Prisco, R.D., Zaccagnino, G., Zaccagnino, R.: Evocomposer: an evolutionary algorithm for 4-voice music compositions. Evolutionary computation **28**(3), 489–530 (2020)
16. Rimsky-Korsakov, N.: Practical Manual of Harmony. C. Fischer (2005)
17. Sikorski, K.: Harmonia cz. 1. PWM (2020)
18. Vechtomova, O., Sahu, G., Kumar, D.: Lyricjam: A system for generating lyrics for live instrumental music. In: Proceedings of the 11th International Conference on Computational Creativity (2021)
19. Wiggins, G.A.: Searching for computational creativity. New Generation Computing **24**(3), 209–222 (2006)
20. Yi, L., Goldsmith, J.: Automatic generation of four-part harmony. Proceedings of the Fifth UAI Bayesian Modeling Applications Workshop (2007)

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_2