Improvised Traditional Greek Songs Composed by Genetic Algorithms

DIONYSIOS POLITIS, PANAGIOTIS LINARDIS*, MICHAEL DIMOPOULOS
Department of Informatics
Aristotle University of Thessaloniki
University Campus, GR-540 06 Thessaloniki
GREECE
*linardis@csd.auth.gr http://www.csd.auth.gr

Abstract: - A Genetic Algorithm is used in this paper to compose music exhibiting the characteristics of traditional Greek music, and especially Greek carols. The GA has the ability to explore a vast region of the composition space through its mutation parameters. At the same time the algorithm is carefully directed, through its fitness function, to observe the composition rules of the prototyped melody. These features make the algorithm a good tool for simulating the procedure of improvisation. Since the fitness function of the GA is derived from the modal characteristics evolving from the knowledge base of the context of the prototyped melodic lines, the resulting music is bounded by the main characteristics of the prototype in terms of scales and rhythm. The proposed algorithm is transformative and generative as well.

Key-Words: - Improvised Computer Generated Music Composition, Context of Traditional Greek Songs, Genetic Algorithms.

1 Introduction

Computer generated musical compositions have been widely applied in recent years aiming in simulating the synthesis procedure at a melodic level. The main axes of synthesis are *transformation* and *generation* [1][2] and the whole procedure can be adequately defined by coordinates of this vector space. Since the world of music is not uniform nor unified, it consists of various segmented systems diversified on matters of scales, rhythms and transitional phenomena. Apart from the instrumental differences there exist *prosodic* depictions of the accompanying lyrics on the harmonic ingredients of melody that entrench the context of each musical tradition [3]. Consequently it is rather impossible to plan a universal composer since the knowledge base involved would be immense.

As a result, the generation algorithm copes with specific music entities. The specific one encountered is comprised by traditional Greek songs and especially carols. From this sample space the adequate fitness criteria are extrapolated for the generation of the improvised outcome.

2 Problem Formulation

The sample space used consists of 10 Greek traditional songs in mode D. The most characteristic of them is a traditional carol, which uses 7-tone diatonic scales and can be perceived as a Dorian [4] scale variation of C major with some affinity with D minor. A synoptic sample line with 3 measures can be seen at fig. 1.

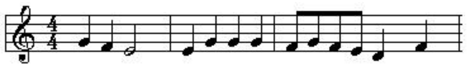

Fig. 1. Exemplar melody from the Christmas carols.

The next step is to transform the melody to a proper set of numerics and infer an adequate computability scheme. This is not obvious, since Greek melodies are not composed in the usual notation and the transformation from their original system to the 12 semitone chromatic scale inserts a degree of uncertainty and fuzziness [5]. The transformation procedure is described in fig.2.

Each note i from the stave system is conceived as a two dimensional vector (x_i,y_i), with the first coordinate denoting pitch and the second duration. The musical surface for x is limited to 2.5 octaves and for y to [𝅝, 𝅗𝅥, …,𝅘𝅥𝅯]. This range is considered capacious for the specific musical patterns. In order to cope with discontinuities and segmentation of the

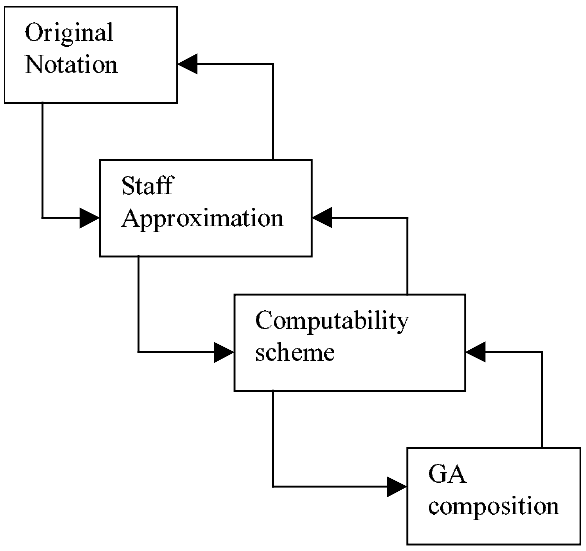

Fig. 2. The transformation procedure.

musical structure at the generation stage, the temporal organization uses 32 quanta of [♪] for each rhythmic measure imposing constraints on the fluctuation Δx of successive vectors within $\pm\epsilon$. This yields n-tuples of generated vectors $(v_1, v_2, \ldots v_n)$, with π_r the projection to axis r, that have

$$\pi_x v_i = \pi_x v_j, \forall (i,j) \in [1, n] \qquad (1)$$

In such a case, the duration of x = n · duration(♪).

In most cases (1) is very stringent, and usually it is used as,

$$\pi_x v \cong \pi_x v, \forall (i,j) \in [1, n] \qquad (2)$$

at reverse staff approximation with an extend of ambiguity $\pm\epsilon$. This ambiguity is normalized according to the fuzziness used at the transformation of original notation to the staff approximation [6] (fig. 2). If the original notation is considered (fig. 3), then *max* (Δx) = 2 commas for the segment of fig. 1.

If (2) holds, then the n-tuple corresponds to

$(v\ ,v\ ,.\ \ldots v\ ) \to V_n^1$

$$\pi_y V_n^1 = \sum \pi_x v = n \cdot \text{duration}(♪) \qquad (3)$$

where V_n^1 is a vector that that replaces the n-tuple and has the same duration.

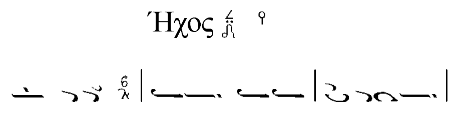

Fig. 3. Original carol notation.

Example: if 4 successive ♪ are generated in the same pitch level, we quote a ♩.

2.1 The GA-Music Algorithm

A description of the Genetic Algorithm used in this paper is shown in Fig. 4.

```
/* Initialization */
Create_random_population.
Evaluate the fitness value of every individual.
Sort population with respect to fitness value in
descending order.

/* Create generations */
ngen=0;

do {
    /* crossover */
     for (j=0,i=0; i < ncross; j +=2, i++) {
        cross_over(Individual[j],
                   Individual[j+1],
                   child1, child2);
        Evaluate(child1);
        Evaluate(child2);
     }
    /* mutation */
     for (i=0; i < nmut; i +=2) {
        mutation(Individual[0], child);
        mutation(Individual[1], child1);
        Evaluate(child);
        Evaluate(child1);
     }
     Sort_Population();
     ngen++;
} while (ngen < MAX_GENERATIONS);
```

Fig. 4. Genetic algorithm for composition

2.2 Individuals

The individual is a bit string (chromosome). Chromosome values are binary coded. Each chromosome represents 12 measures arranged in-groups of three, where one measure is composed of 32 pitch levels. We choose to encode each pitch with five bits so as to have 32 discrete levels. Chromosome length is 12 ·32 · 5 = 1920 bits.

2.3 Fitness Function

The formal characteristics of each individual are compared to those of the prototyped melody [7]. The fitness function makes the comparison and ranks each individual according to its "proximity" to the prototype. However, finding a good evaluation function is a complex task. In this work we propose a fitness function F of the form:

$$F=\sum_{i=1}^{n} c_i R_i \tag{4}$$

where:

R_i is the ranking value showing how close is the individual to fulfilling rule i, and c is a coefficient indicating the importance (weight) of rule i in the composition. This flexible form of F allows the easy experimentation with many rules and weights. In the present implementation we used only three rules (n=3) and found that the combination of weights:

$$c = c = 5, c_3 = 1 \tag{5}$$

gives acceptable results.

2.3.1 Rules

The rules that every individual should obey are:

R_1: The first $4/32$ of a measure are the same.

R_2: The next $4/32$ are same or form groups of two. For each group holds: $|Aver| \leq 3 \cdot |Aver_1|$ where $Aver$ is the absolute mean value of the group and $Aver_1$ the mean absolute value of all the previous pitches. We discriminate between groups based on how "close" their mean value is to the mean value of all the previous ones.

R_3: The rest $24/32$ of the measure are in groups of two or in fours and for each group: $|Aver| \leq 5 \cdot |Aver_1|$.

We applied the above rules for the first three measures of each individual.

2.4 Genetic Operators

The creation of an offspring is accomplished with the help of a cross-over operator that merges the genes of two individuals. The cross-over operator used here is a one-cut cross-over. To ensure diversity, mutation is applied with a probability p_m. Mutation is applied here to the best 2 individuals in the population. Three different mutation operations are used each one with different probability.

a) Single-bit mutation: it randomly selects a bit and complements it.
b) 1-point-mutation: it randomly selects a pitch and replaces it with a random selected one.
c) 2-point-mutation: it randomly selects a pitch and replaces both it and its neighbor with a randomly generated one.

3 Experimental Results

For the experiments we used the following parameter values (fig. 5):

```
POPULATION = 800
MAX_GENERATIONS = 200
PCROSSOVER = 0.4
PMUTATION = 0.1
PMUTATION1= 0.02
PMUTATION2 = 0.002
```

Fig. 5. Parameter values at the generation stage.

For the transformation stage we have used the encoding scheme of table 1.

Table 1.

C	C$^{\#}$	D	D$^{\#}$	E	F
1,2,3	4,5	6,7,8	9,10	11,12,	14,15,16
F$^{\#}$	G	G$^{\#}$	A	A$^{\#}$	B
17,18,19	20,21,22	23,24	25,26,27	28, 29	30,31,32

The samples notes have pitches ranging from C_4 up to B_4. The transformation values can be seen in table 1. Using rules R_1 R_2 R_3, the genetic algorithm produces a highly fit solution. The reverse transformation is based on table 1 led to the following melodic line.

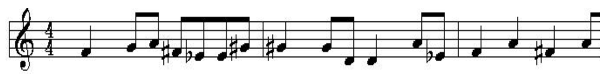

Fig. 5. GA composed melody.

4 Conclusions

One of the challenges faced by many expert system designers is that they are often met with opposition and skepticism from existing human experts. The designers find it important to emphasize that they are not trying to replace human beings with machines, but rather create tools and methodologies that are meant to either make the job of the experts less cumbersome or to help educate would-be experts [7]. This improvising GA rather than seeking to replace composers, relies on them for their knowledge.

Although the outcome of the algorithm relies on its proximity to the prototype, its musical structure has to comply with more subtle factors of composition and human perception. Operating in this direction, the authors of this paper are increasing the number of rules and exploring different sets of weights. They are also formulating rules that comply with the acceptable standards of musical categorization.

References:

[1] D. Goldberg, *Genetic algorithms in Search, Optimization, and Machine Learning,* Addison-Wesley, 1989.

[2]Michalewicz, Zbigniew, G*enetic Algorithms + data structures = evolution programs*, Springer-Verlag, New York, 1996.

[3] Politis, D., Tsoukalas, A., Linardis, P., "VIDI-A Voice Instrument Digital Interface for Byzantine Music", *International Computer Music Conference ICMC97*, Thessaloniki, Proceedings, pp. 403-407, September 25-30, 1997.

[4] Fels, S., Nishimoto, K., Mase, K., "Musikalscope: A Graphical Musical Instument", *IEEE Multimedia,* Vol.5, No.3, 26-35, July-September 1998.

[5]Rao, V., Rao, H., *GC++ Neural Networks and Fuzzy Logic,*MIS:Press, 1995.

[6] Mastorakis, N.E., Gioldasis, K.D., Koutsouvelis, D. and Theodorou, N.J., " Study and Design of an Electronic Musical Instrument which accurately produces the spaces of the Byzantine music", *IEEE Transactions on Consumer Electronics,* Vol.41, No.1, pp.118-124, February 1995.

[7] Biles, J., "GenJam: A Genetic Algorithm for Generating Jazz Solos", *International Computer Music Conference ICMA*, San Fransisco, 1994.