A Hybrid Computational Intelligence Approach for Automatic Music Composition

Giovanni Acampora*, Jose Manuel Cadenas†, Roberto De Prisco*,
Vincenzo Loia*, Enrique Muñoz‡ and Rocco Zaccagnino*
*Dept. of Computer Science, University of Salerno, Fisciano, Italy 84084
Email: {gacampora, robdep, loia, zaccagnino}@unisa.it
†Dept. of Information and Communications Engineering, University of Murcia, Espinardo, Spain 30003
Email: jcadenas@um.es
‡European Centre for Soft Computing, Mieres, Spain 33600
Email: enriquemuba@um.es

***Abstract*—The use of computers in the production of artifacts has drawn the attention of both artists and computer scientists. Among the different art disciplines, music is one of the arts that most benefited from the use of computers. There are many works which demonstrate the great synergy between these two fields. In this paper we will focus on a specific music composition problem: the figured bass problem, in which we have to automatically generate a 4 voice piece of music, starting from an input the bass line. To solve this problem we use a hybrid strategy, in which different metaheuristics cooperate to find high quality solutions. The cooperation is controlled by means of the combination of fuzzy control and knowledge obtained through Data Mining. As will be shown in the experimental results section, this hybrid strategy is capable of finding musical solutions with an acceptable quality and never discordant which, according to experts, are sound and adhere to scholastic rule.**

***Index Terms*—Computer Music, Evolutionary Computation, Adaptive Memetic Algorithms.**

I. Introduction

The use of computers as means to produce artifacts has been subject of study since the beginning of the computer era. Both artists and computer scientists have been interested in this particular use of the computer. Among the art disciplines, music is one of those that have most benefited from computers. Automatic composition by means of a computer program is almost as old as the computer itself: the ILLIAC suite [1], [2] is a musical piece composed by a computer program and dates back to 1957 just a few years after the introduction of the first computer.

The use of computers in the field of music encompasses many aspects, such as notation programs, sound synthesis, real-time performances, digital instruments, and much more. In this paper we are interested in the aspect of *algorithmic (or automatic) music composition*, that is in the development of computer programs able to automatically generate music, without human intervention.

Algorithmic music composition has fascinated both computer scientists and musicians. We can find works dating back to well before the computer era, which can be considered algorithms for automatic composition (e.g. Mozart's dice game). More recent works include those developed by E.R. Miranda [3] and D. Cope [4], [5], [6].

The tools used to obtain algorithmic composers include many techniques, such as random numbers, formal grammars, cellular automata, fractals, evolutionary algorithms [7][8]. An interesting review of the field can be found in the book by Miranda [9].

Music has a long and interesting history. Music practices have varied during the course of human history developing in various directions also depending on the local culture. This resulted in several formal frameworks for the development of music. The most used one is the so called tempered music system. This system is based on several formal rules that are well suited to be handled by computer programs.

In this paper we are concerned with a very specific tempered music composition problem, known as the *figured bass problem*. A figured bass is a written melody for the bass voice, with indications (numbers or symbols) about the chords to be used. In order to play a figured bass the performer becomes a composer because it is necessary to improvise the other voices in order to obtain a full 4-voice piece of music with the constraints given by the chords indications. The figured bass appeared in the XVI century and was very popular in baroque music. Nowadays it is used (figured or also unfigured) as an exercise for aspiring composers.

So the goal of our automatic composer is that of producing 4-voice piece of music starting from a figured bass line. Formidable examples of 4-voice compositions are Bach's chorales.

Algorithms are frequently used to find optimal or good solutions, where optimal and good are defined according to a precisely defined measure. However, for musical compositions this precise measure does not exist. This is a serious problem, because there is no easy way to decide if a solution is "better" than other. The quality of a musical composition (as the quality of any artistic artifact) is subjective. For the specific problem that we consider in this paper, well known harmony rules can help, although they do not constitute a precise measure. The existence of such a metric would reduce the problem of automatic music composition to an optimization problem with a very large search space. Indeed the figured bass problem, and any other music problem, can be seen as a combinatorial problem where one has to place a finite number of notes on a finite grid (the stave).

In order to "compare" solutions we can exploit harmony rules.

As previously said, there is a broad variety of tools and techniques that can be used to design an algorithmic composer. In this paper we propose the use of a hybrid intelligent tool based on a multi-agent framework and metaheuristics. In this framework different agents analyze a figured bass problem instance and solve it in a parallel way, cooperating between them. This cooperation is performed by jointly exploiting data mining, together with a decision making framework exploiting fuzzy methodologies.

In order to solve the problem each agent implements a metaheuristic. Metaheuristics [10] are intelligent strategies used to design and improve very general heuristic procedures with a high performance. Within metaheuristics, different works [11] have underlined that they can benefit from its combination with fuzzy technologies and data mining improving their results. In this paper we present an adaptive cooperative metaheuristic able to solve figured bass problem. This strategy is able to adapt its behavior depending of the instance being solved. To do that it exploits a set of knowledge models able to code the knowledge obtained from previous executions of the individual metaheuristics. Metaheuristics have already been used for several other problem; to the best of our knowledge, this is the first attempt to use them for a musical problem.

The remainder of the paper is organized as follows. In Section II we introduce some musical concepts necessary to understand the figured bass problem. After that, in Section III the proposed composer is presented. Next, in Section IV we show some results obtained using the composer, and finally in Section V we present some conclusions.

II. Music Background

In this paper we consider the tempered music system, used in western music. This system start from a reference sound (a given frequency, normally 440Hz) and defines *ocatves* as the ranges of frequencies between those sounds obtained by doubling/halving the reference sound. Each octave is divided in 12 equally spaced notes, denoted by the letters A, A♯ or B♭, C, C# or D♭, D, D# or E♭, E, F, F# or G♭, G and G# or A♭. More precisely, if a note has frequency f the "next" note has frequency $f \cdot 2^{\frac{1}{12}}$.

Tempered music is based on the notion of tonality. Roughly speaking, a tonality is a group of notes which form a scale. Using each one of the 12 notes of a scale as starting point we can obtain different tonalities (there are different kinds of tonality for each note, like major, minor, ...). For example, C major scale is C, D, E, F, G, A, B, while D major scale is D, E, F#, G, A, B, C#.

The notes of a scale are often denoted as I, II, III, IV, V, VI, VII, specially when the emphasis is given to the degree of the scale and not to the particular note, which depends on the tonality. Such a notation is possible because all notes are equally spaced.

The interested reader should refer to [12] to get a deeper insight of tempered music.

Usually a tonality is considered the main tonality for a piece, and thus, the notes belonging to the corresponding scale are considered more important than those not belonging to it. Western music, starting from the common practice period, is based on well established harmonic and melodic rules for the tempered music system. We refer the interested reader to any good harmony book as [13] for a discussion about such rules.

In our problem a musical composition is made up of 4 independent voices, called *bass*, *tenor*, *alto*, *soprano*. The bass line is given as input together with numbers, one for each note, that determine the chords to be used.

A musical piece is composed of a sequence of *measures*, each one divided into a given number of *beats*. In each beat the 4 voices play a note (this is a simplification of what really happens, since a composition may also have passing notes, which are not part of the harmony). The vertical set of notes in a beat is a *chord*. The notion of chord is fundamental for this work. There are many types of chords.

In this paper we consider 3- and 4-note chords. Chords are built upon each degree of the scale, that is, I, II, III, IV, V, VI, VII. The note upon which the chord is built, called *root note*, is often given to the bass; however the bass can play any other note which is called chord inversion. Chord inversions usually denoted by a superscript which indicates the inversion (e.g., III^{6}, V^{46}, I^{357}).

The chords considered on this work are described in Table I (for each one of them we give an example in the tonality of C).

TABLE I
Chords considered - with examples in the tonality of C

Chord	Root note	Set of notes
Major triad	C	C, E, G
Minor triad	E	E, G, B
Major seventh	C	C, E, G, B
Minor seventh	D	D, F, A, C
Dominant seventh	G	G, B, D, F
Half-diminished seventh	B	B, D, F, A

One of the main rules of tempered music system is that the vertical set of notes has to belong to any of the allowed chords. Other rules may affect sequences of chords. Some sequences are "better" than others, where better is difficult to define, as it is a subjective evaluation. In any case it is largely accepted that particular sequences of chords work better than others. Some chords are "more important" than others because they suggest, prepare, device or enforce tonal centers. The art of tonal music consists precisely in arranging chords in such a way that their interplay is pleasant and significant.

In practice, this is translated into simple rules as the following [13]:

I is followed by IV o V, sometimes VI, less often II o III.
...
...
II is followed by V, sometimes VI, less often I,III o IV.

Besides the rules concerning sequences of chords, we can also find rules about melodic lines. A melodic line is the sequence of notes played by each voice. Rules about melodic lines can refer to the movement of a single voice (for example, a jump bigger than an octave is normally not allowed) or also to the movements of two voices (for example, two voices that proceed by parallel fifth are not allowed). Again, we refer the interested reader to a standard textbook, like [13] for a better and more detailed explanation.

An example of a figured bass problem is provided in Figure 1. Part a of the figure provides the input: only a bass line with indications about the chord; part b provides a solution.

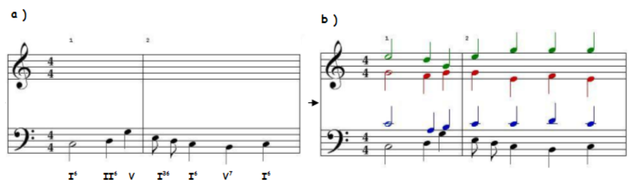


Fig. 1. Chromosome representation

III. An Automatic Music Composer

We propose the use of an adaptive cooperative metaheuristic to solve the figured bass problem. In order to describe the metaheuristic it is necessary to introduce three main aspects: the architecture of the strategy, the cooperation scheme which allows to create a synergy among the different metaheuristics, and the intelligent knowledge extraction process that obtains the data mining models which give adaptability to the strategy.

The development of cooperative metaheuristics lies in the combination of existing metaheuristics to obtain hybrids, in which the different metaheuristics cooperate in a parallel way. Due to their capacity of cooperating in a parallel way, cooperative metaheuristics are closely related to parallel metaheuristics. Many studies have demonstrated [11], [14], [15] that parallel metaheuristics can obtain better results than their sequential components, even when they are allowed to use less time. These studies have shown that the combination of different threads, each one using a different metaheuristic, increments the robustness of the global search independently of the instance being solved.

The proposed strategy combines different metaheuristics which are executed in a parallel way while they exchange information. Information exchange is controlled using previous knowledge about the performance of the individual metaheuristics.

A. Architecture

The strategy is modeled using a multiagent system where we can distinguish two types of agents: *optimization agents* and a *coordinator agent*. Each optimization agent is in charge of executing a metaheuristic, and the coordinator agent is responsible of three tasks: to configure the parameters of the optimization agents depending on the instance being solved, to initiate their parallel execution, and to control their cooperation by deciding how and when they have to exchange information.

Figure 2 shows the architecture of the proposed strategy.

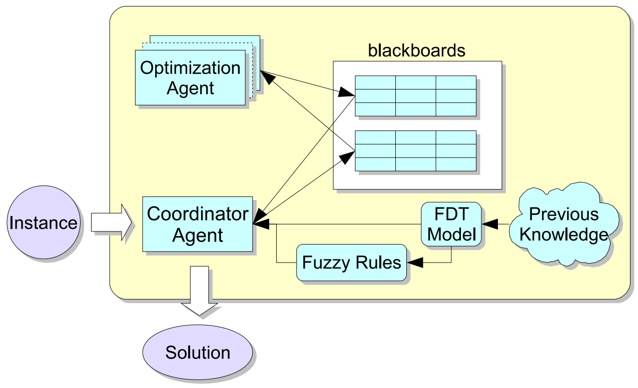


Fig. 2. Architecture of the cooperative metaheuristic strategy

The adaptability of the coordinator is achieved using the knowledge obtained by means of computational learning and coded using a set of trees. In particular, the trees help the coordinator to:

1) choose the best parameters values for each metaheuristic, and
2) sort the suitability of each metaheuristic to solve the instance under consideration, using a set of weights in the range $[0, 1]$.

By analyzing the weights, the coordinator can find out if a metaheuristic is obtaining a poor behavior (according to some measure of performance, e.g. the value of the objective function). As a consequence it can decide to send some solutions to the “bad” metaheuristic from a metaheuristic with a better behavior.

Optimization agents execute in an asynchronous way their associated metaheuristics while send and receive information. The exchange of information is carried out using a blackboard. In detail, two blackboards are used, the first is used by optimization agents to publish the solutions they have found, the second is used by the coordinator in order to indicate the instructions for each agent. Specifically, after a specified period of time, each optimization agent stops and publishes its current solution/population (depending on its type, trajectory or population based). Then the coordinator agent analyzes the behavior of each metaheuristic, using its previous knowledge, and decides if any strategy needs to improve its performance. In this case, the coordinator writes in the second blackboard the new solution/population that the metaheuristic has to use.

B. Cooperation Scheme

The cooperation scheme used by the strategy is mainly defined by the control rules used by the coordinator and the trees that guide its adaptivity mechanism. These components have to help the coordinator agent to:

- choose the best parameters values for each metaheuristic.
- find out when a metaheuristic is obtaining poor behavior and has to receive new solutions from other metaheuristics with a better behavior.

The first task is carried out using a set of trees obtained through a supervised learning process. Each one of the trees is related to a parameter of a metaheuristic.

The coordinator is able to perform the other task using the information contained in the rest of trees. These trees analyze the instance at hand and sort the different cooperating metaheuristics according to their suitability to solve it. In particular the trees obtain a set of weights $\Omega = \{\omega_i, i = 1, \ldots, n\}$ where n is the number of metaheuristics, $\omega_i \in [0, 1]$ and $\sum_{i=1}^{n} w_i = 1$. Each weight ω_i is associated to a metaheuristic m_i and represents its suitability to solve the instance at hand.

In order to control the exchange of solutions, the coordinator uses the following set of fuzzy rules (this set of rules is replicated once for each metaheuristic):

if $(\omega_1 \cdot \delta_1)$ is *enough* **then**
send solutions from m_1 to m_h

$\ldots \ \ldots$

if $(\omega_n \cdot \delta_n)$ is *enough* **then**
send solutions from m_n to m_h

where:

- n is the number of metaheuristics.
- m_h is the metaheuristic being evaluated.
- $\delta_i = (\xi_{m_i} - \xi_{m_h})/maximum(\xi_{m_i}, \xi_{m_h})$, where ξ is a measure of performance.
- $\omega_{m_i} \in [0, 1]$ where $\sum_{i=1}^{n} \omega_{m_i} = 1$ and represents the weight of metaheuristic m_i.
- *enough* is a fuzzy set with trapezodial membership function (with support $\in [0, 1]$) defined by:

$$\mu(x, a, b, c, d) = \begin{cases} 0 & x \leq a \ or \ x \geq d \\ \frac{x-a}{b-a} & x \in (a, b] \\ \frac{d-x}{d-c} & x \in [c, d) \\ 1 & x \in [b, c] \end{cases}$$

with $(a, b, c, d) = (0, 0.1, 1, 1)$.

Using this set of rules the coordinator is able to know when it has to send new solutions to a given metaheuristic. In detail, it changes the position in the search space of a metaheuristic which is showing a bad performance for a position near the position of another metaheuristic with a better behavior. The firing of the rules is controlled using an *α-cut* defined by the user. In the event that more than a rule is fired, the coordinator applies all of them.

To decide which solutions are sent to a metaheuristic, we consider the following situations:

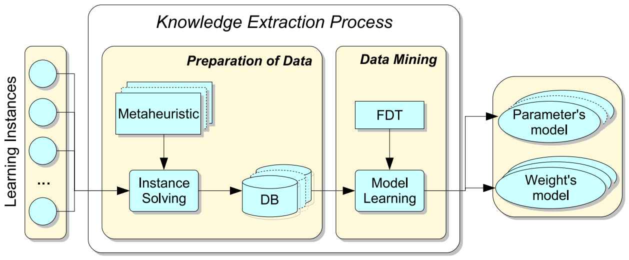


Fig. 3. Obtaining models: Knowledge extraction process

- $m_{receiver}$ is based on trajectories. A solution "near" the best solution obtained by the metaheuristics that have fired the rule is sent. Where "near" means that we apply $\frac{1}{2\omega_{sender}}$ times a mutation operator.
- $m_{receiver}$ is population based:
  - m_{sender} is trajectory based. A proportion of the worst individuals of $m_{receiver}$'s population is substituted by a set of solutions near m_{sender}'s best solution.
  - m_{sender} is population based. A proportion of the worst individuals of $m_{receiver}$'s population is substituted by a set of the best members of m_{sender}.
  - Different metaheuristics have to send solutions. A proportion of the worst individuals of $m_{receiver}$'s population equals to sum of $m'_{sender}s\ \omega_i$ is substituted by a set of solutions where each m_{sender} chooses its solutions as described before.

C. Tree Construction

In order to obtain the models related to weights and parameters we use an intelligent knowledge extraction process, as the one shown in Figure 3.

This process is divided in two different phases, data preparation and data mining. In addition, the process is supervised and previous to strategy's operation.

1) Data Preparation Phase: In this phase the different metaheuristics that compose the strategy are executed several times to solve diverse training instances.

In particular, each metaheuristic is executed using different values for each one of its parameters. Using the information generated in these executions we obtain a dataset which describes the behavior of each metaheuristic.

In detail, we obtain two types of datasets. The first type is related to metaheuristics' parameters and we need one dataset for each parameter of each metaheuristic. The second type is related to metaheuristics' weights and we need one dataset for each metaheuristic.

Focusing on parameters' datasets, each one of them stores for each training instance: a description of the instance and the parameter value that obtained the best performance after solving it. On the other hand, each weights' dataset stores for each training instance: a description of the instance and a weight obtained as

$$\omega_{m_i} = \frac{fit_{best}^{q,i}}{\sum_{l=1}^{n} fit_{best}^{q,l}}$$

where $fit^{q,i}_{best}$ is the best value of the objective function obtained by metaheuristic i for instance q.

2) Data Mining Phase: Once we have obtained the datasets, the data mining phase extracts a collection of fuzzy decision trees which model the information contained in them. Once again we can distinguish two types of models, *parameter models* and *weight models*. The first type is related to the choice of parameters values carried out by the coordinator, and there is one model for each parameter of each metaheuristic. On the other hand, the second type of models is related to the weights Ω, and there is one model for each metaheuristic.

After completing this phase we obtain a set of trees that guide the adaptation of the coordinator. That way it is available to control optimization agents cooperation.

IV. Experimental Analysis

In this section we perform tests in order to asses the validity of the proposed strategy. Each strategy takes as input a figured bass line and produces a complete 4-voice harmonization of the bass line. Some examples of MIDI files generated during the experimental sessions can be listened at http://music.dia.unisa.it/Musimatica/Download.html.

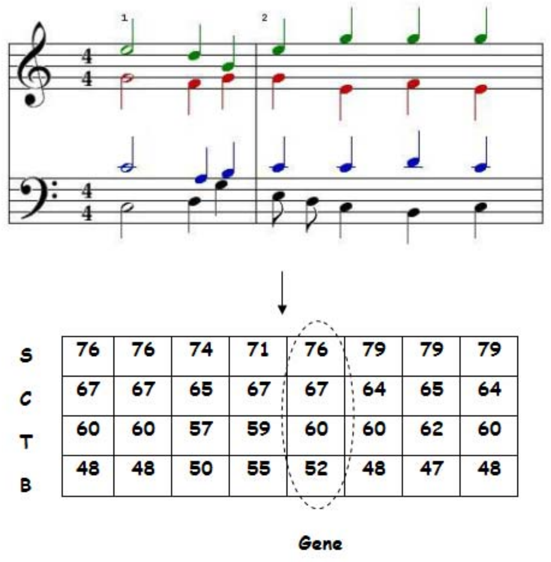
A. Chromosome and gene representation

Figure 4, shows the MIDI-code representation of a chromosome. The time signature $\binom{n_b}{b_\ell}$ is part of the input; we consider as basic time unit the length of the beats. We allow notes of the input to have a length greater than (actually, a multiple of) the beat length. However, if an input note is shorter than the beat length then we consider it only if the note is placed on the beat (otherwise it is only a passage note).

Let n_m denote the number of measures in the input bass line. We can represent a chromosome as an array C of dimension $4 \times n$, where $n \leq n_b \times n_m$. Element $C(i,j)$ contains the j^{th} note for voice i. We place the bass voice in the lower row and the soprano voice in the upper row. Notes are represented using MIDI codes; this allows us to easily play the chromosomes. In the previous example (Figure 4) we have $n_b = 4$, $b_\ell = 1/4$, $n_m = 2$ thus $n = 2 \times 4 = 8$. Notice that this representation doesn't contain information about the notes' length; indeed we do not use such information for the metheuristic. Obviously, we keep the information about the length of the notes in a separate data structure.

For each bass note b_i our goal is to select 3 notes to be played along with the bass note b_i. We will denote such notes with t_i, c_i, s_i. The set of notes $\{b_i, t_i, c_i, s_i\}$ is the chord at position i. We can also think of a (candidate) solution to our problem as an array of $4 \times n$ cells, where each column i contains the four notes b_i, t_i, c_i, s_i. The set of these 4 notes must be one of the allowed chords (see Table I) and must match the indication given in the input.

Since the bass is figured choosing the chord is the easy step, since the input provides already an indication. After choosing the chord we have a huge number of possibilities to place the notes of the chord. Only the bass note is already fixed; notice that the bass not does not necessarily correspond to the root note of the chord.



S	76	76	74	71	76	79	79	79
C	67	67	65	67	67	64	65	64
T	60	60	57	59	60	60	62	60
B	48	48	50	55	52	48	47	48

Fig. 4. Chromosome and MIDI representation

B. Fitness function

Our evaluation function takes into account musical composition rules from the common practice period. We consider a simple set of rules, which are detailed in Table II.

To assign a value to a chromosome we start from a reference value which depends on the sequence of chords and thus, depends only on the input, i.e., it is fixed and is based upon harmonic considerations (appropriate sequences of chords);

Given a chromosome x, we assign a reference value by giving a weight for each pair of genes. For each two consecutive bass notes we look at the sequence of the two chords C_1 and C_2 given in x to those two notes and we give the following weights:

- C_1 and C_2 are chords in the same major tonality. The weights are given in Table III.
- C_1 and C_2 are chords in the same minor tonality. The weights are given in Table IV.
- C_1 and C_2 modulate from a major to a major tonality or from a minor to a minor tonality. The weights are given in Table V.
- C_1 and C_2 modulate from a major to a minor tonality or from a minor to a major tonality. The weights are given in Table VI.

The reference value is given by the sum of the weights for pair of genes of x. The weights have been chosen so that good harmonic pattern have heavier weights.

Then to obtain the fitness value of a particular solution we subtract a given cost for each rule violation present in the solution from the reference value. The relations between two consecutive genes (chords) allows us to decide whether the chromosome is good or not. For example, if a voice makes an augmented fourth jump we subtract 40 from the initial value; if two voices make a hidden fifth we subtract 60. If two notes within a chord are at distance greater than an

TABLE II
RULES TABLE

Single voice error	Cost	Type		Two voices error	Cost	Type
sixth	30	normal		hidden unison	40	normal
aug. fourth	40	normal		hidden fifth	60	normal
aug. fifth	40	normal		hidden octave	50	normal
seventh	80	critical		unison	100	critical
$>$ octave	100	critical		parallel fifth	100	critical
				parallel octave	100	critical
Single voice cost	**Cost**	**Type**		**Error within chord**	**Cost**	**Type**
jump	interval	no error		octave leap	30	normal

TABLE V
WEIGHTS FOR MAJOR-TO-MAJOR OR MINOR-TO-MINOR MODULATION.

	C	G	D	A	E	B	G♭	D♭	A♭	E♭	B♭	F
C	2000	0	-500	-1000	-1500	-2000	-2500	-2000	-1500	-1000	-500	0
G	0	2000	0	-500	-1000	-1500	-2000	-2500	-2000	-1500	-1000	-500
D	-500	0	2000	0	-500	-1000	-1500	-2000	-2500	-2000	-1500	-1000
A	-1000	-5000	0	2000	0	-500	-1000	-1500	-2000	-2500	-2000	-1500
E	-1500	-1000	-1000	0	2000	0	-1000	-1000	-1500	-2000	-2500	-2000
B	-2000	-1500	-1500	-5000	0	2000	- 0	-500	-1000	-1500	-2000	-2500
G♭	-2500	-2000	-2000	-1000	-500	0	2000	0	-500	-1000	-1500	-2000
D♭	-2000	-2500	-2500	-1500	-1000	-500	0	2000	0	-500	-1000	-1500
A♭	-1500	-2000	-2000	-2000	-1500	-1000	-500	0	2000	0	-500	-1000
E♭	-1000	-1500	-1500	-2500	-2000	-1500	-1000	-500	0	2000	0	-500
B♭	-500	-1000	-1000	-2000	-2500	-2000	-1500	-1000	-500	0	2000	0
F	0	-500	-1000	-1500	-2000	-2500	-2000	-1500	-1000	-500	0	2000

TABLE VI
WEIGHTS FOR MAJOR-TO-MINOR OR MINOR-TO-MAJOR MODULATION.

	C	G	D	A	E	B	G♭	D♭	A♭	E♭	B♭	F
C	-500	-200	0	500	0	-200	-500	-700	-1000	-1200	-1000	-700
G	-700	-500	-200	0	500	0	-200	-500	-700	-1000	-1200	-1000
D	-1000	-700	-500	-200	0	500	0	-200	-500	-700	-1000	-1200
A	-1200	-1000	-700	-500	-200	0	500	0	-200	-500	-700	-1000
E	-1000	-1200	-1000	-700	-500	-200	0	-500	0	-200	-500	-700
B	-700	-1000	-1200	-1000	-700	-500	-200	0	-500	0	-200	-500
G♭	-500	-700	-1000	-1200	-1000	-700	-500	-200	0	-500	0	-200
D♭	-200	-500	-700	-1000	-1200	-1000	-700	-500	-200	0	-500	0
A♭	0	-200	-500	-700	-1000	-1200	-1000	-700	-500	-200	0	-500
E♭	-500	0	-200	-500	-700	-1000	-1200	-1000	-700	-500	-200	0
B♭	0	-500	0	-200	-500	-700	-1000	-1200	-1000	-700	-500	-200
F	-200	0	-500	0	-200	-500	-700	-1000	-1200	-1000	-700	-500

TABLE III
WEIGHTS FOR STEPPING BETWEEN CHORDS IN THE SAME MAJOR TONALITY.

	I	ii	iii	IV	V	vi	vii^o
I	250	200	50	200	250	50	10
ii	100	100	100	150	2000	150	10
iii	100	100	100	200	100	250	10
IV	250	150	100	200	1500	100	10
V	2000	100	100	100	250	150	10
vi	100	200	150	150	200	200	10
vii	1000	50	150	50	50	100	200

TABLE IV
WEIGHTS FOR STEPPING BETWEEN CHORDS IN THE SAME MINOR TONALITY. THE CHORDS ARE DENOTED $2^*, 3^*$ ETC. BECAUSE IN A MINOR TONALITY ON THESE DEGREES WE CAN HAVE SEVERAL TYPE OF CHORDS (E.G. ON THE SECOND DEGREE WE CAN HAVE ii OR ii^o).

	i	2*	3*	4*	5*	6*	7*
i	250	200	50	200	250	50	10
2*	100	100	100	150	2000	150	10
3*	100	100	100	200	100	250	10
4*	250	150	100	200	1500	100	10
5*	2000	100	100	100	250	150	10
6*	100	200	150	150	200	200	10
7*	1000	50	150	50	50	100	200

octave we subtract 30 (except for the couple bass-tenor). For some violations (which we call critical errors) we have bigger penalties. For example for a voice jump bigger than an octave or for two voices that create exposed parallel fifths we have a penalty of 100. Table II provides also the cost that we used for each rule. We construct the possible chords always putting voices in the order bass, tenor, alto and soprano; hence no voice crossing is allowed.

Let's consider an example and again take the one presented in Figure 1. The reference value is the sum of the weights of the chords sequence I - ii - V - I - I - V - I, that is $200 + 2000 + 2000 + 250 + 250 + 2000$, the total is 6700.

To obtain the fitness value of a particular solution, we start from the base value and subtract all the costs listed in Table II.

In this particular case the overall single voices cost is 48, the cost for errors within a chord is 90, the cost of single voices errors 0 and also the cost of two voices errors is 0. Total cost is 138. The fitness value is 6562.

C. Configuration of the Strategy

We will build a strategy using four different individual metaheuristics:

- Genetic Algorithm,
- Tabu Search,
- Simulated Annealing, and
- Particle Swarm Optimization.

After deciding the composition of the metaheuristic, we have to apply the intelligent knowledge extraction process. We will give a brief description of the decisions taken to apply it to a real problem.

- As a first step we have decided to use 36 training instances, in particular this instances have been chosen among J.S. Bach's chorales.
- Each one of them has been solved using each metaheuristic configured with different combination of parameters values. In particular, for each parameter of each metaheuristic we have chosen 4 different values on average. To choose these values we have used previous knowledge about their behavior, however, if there is no previous knowledge we recommend to use, at least, 10 different values.
- After choosing parameters values, we solve each instance 10 times with each possible combination of parameters values, to obtain more precise values.
- Next we obtain parameters and weights datasets.
- Finally we use a fuzzy decision tree construction technique in order to obtain parameters and weights models.

D. Configuration of the Experiments

In order to test the validity of the approach we compare the proposed strategy with the individual metaheuristics that comprise it. To do so, we perform tests with 4 different instances chosen among J.S. Bach's chorales, in particular chorales 26, 112, 266, 606.

Each strategy was executed during 100000 evaluations of the objective function and each instance was solved 10 times. The results show the averages. For the cooperative strategies one can resort to parallel schemes if time is important, or one can simulate the parallelism in a one-processor computer. This is the strategy taken here and the procedure is extremely simple.

We construct an array of solvers and we run them using a round-robin schema. This implementation uses an asynchronous communication mode that is simulated in this way: solvers are executed during a certain number of evaluations of the objective function. This amount of evaluations is a random number in the interval $[5000, 5500]$ and after this period of time, information exchanges are performed. These steps are repeated until the stop condition, given in terms of objective function evaluations, is fulfilled.

The use of the number of objective function evaluations to control the execution of the strategies is broadly extended as a way to perform fairer comparisons with existing techniques. Because in that way we can compare the results independently of the programming language used, the architecture of the computer utilized or the ability of the programmer to optimize his code.

However, we consider that the time can be considered as an important measure of performance, and in that sense we have to point out that the overhead derived from the execution of the coordinator is negligible and that the execution time of the cooperative strategies is always smaller than that of the slower individual metaheuristic for the same number of evaluations of the objective function.

All tests were executed on an Intel Pentium 4 3.20Ghz with 1GB of Memory.

E. Results

The results obtained are shown in Table VII. For each strategy and each instance we present the average percentage of error obtained with respect to the best solution obtained.

From these results we can observe that the proposed strategy (CS) is the one which obtains the lowest error. In order to demonstrate this conclusion we have carried out a statistical analysis [16] of the results. First, we have applied Friedman test to ascertain if there are any differences among the results obtained by all strategies, concluding with a confidence level of 99% that differences do exist. After that we have applied a post-hoc test (Benjamini-Hochberg test together with Wilcoxon test) to study if the error obtained by the proposed strategy is less than that obtained by the rest.

TABLE VII
RESULTS OBTAINED WITH 4 DIFFERENT INSTANCES CHOSEN AMONG J.S. BACH'S CHORALES.

Instance	GA	PSO	SA	TS	CS
Bach's chorale 70	32,03%	26,71%	0,20%	1,41%	0,04%
Bach's chorale 215	39,70%	35,47%	0,88%	5,15%	0,04%
Bach's chorale 266	43,79%	39,38%	12,82%	5,86%	0,02%
Bach's chorale 3604	21,95%	25,66%	4,26%	11,21%	0,02%

The results of these statistical tests indicated with a confidence level of 99% that the results obtained by the cooperative strategy are better.

From these results we can observe the utility of the cooperation between metaheuristics and the adaptive selection of parameters by means of data mining models.

F. Musical analysis

After performing a performance study with respect to the value of the objective function and checking the superiority of the proposed strategy we want to analyze its results from a musical point of view.

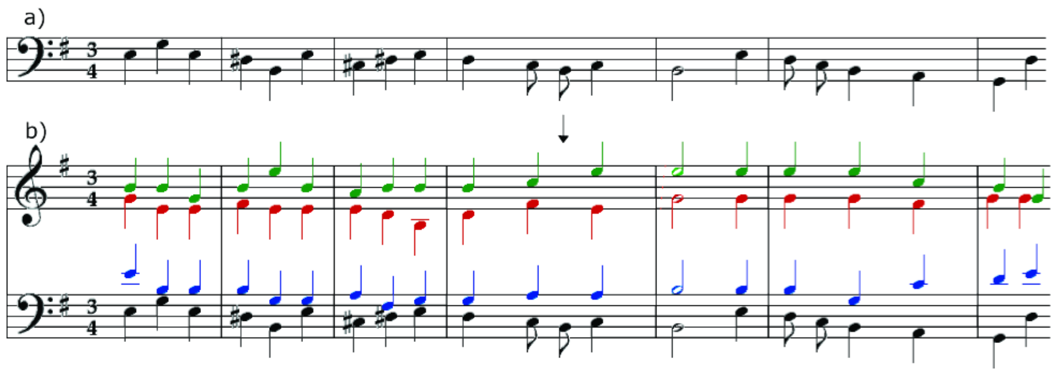


Fig. 5. An example of input - Bach's chorale 266 - (part a) and final solution (part b)

As we have pointed out in the introduction, the quality of a musical composition is a subjective judgment. However the solutions provided do not have any of the errors that guided the search performed by the cooperative metaheuristic.

We have also performed listening tests where we had people, with a music background, listen to the pieces composed by our automatic composer. The avarage response that we had is that the compositions are sound and adhere to scholastic rule. A human composer would probably do better in terms of originality, but most people were surprised that a computer could actually produce such outputs.

V. Conclusions

Figure 5 shows an example of the results obtained by the strategy with Bach's chorale 266. Part a of the figure shows bass line (input) and part b the final solution obtained.

The results obtained fulfill our expectations, returning solutions with an acceptable quality and never discordant. Such results are possible because the cooperative metaheuristic is able to eliminate all critical errors and many of the errors allowed by the used harmonic rules.

In this paper we have presented an automatic composer based on an adaptive cooperative metaheuristic strategy. This composer is able to generate a musical composition without human intervention.

The specific problem under consideration is figured bass problem, in which the objective is the generation of a 4 voice piece starting from bass line. We have tackled the problem using an adaptive cooperative metaheuristic strategy. In this strategy a set of different metaheuristics cooperate under the supervision of a coordinator. This coordinator is modeled using the knowledge extracted by means of a supervised learning process, which is applied to historical data of the execution of individual metaheuristics.

To test the validity of the strategy we have performed different tests obtaining interesting results.

As future work we want to study the similar (but harder) problem called unfigured bass where no indications about the chords is provided in the input. Another interesting direction for future work is the use of an extended set of harmony rules.

Acknowledgements

The authors thank the "Ministerio de Educación y Ciencia" of Spain and the European Fund of Regional Development for the support given to this work under the project TIN2008-06872-C04-03. And the "Fundación Séneca, Agencia de Ciencia y Tecnología de la Región de Murcia (Plan Regional de Ciencia y Tecnología 2007/2010)" which give support under the "Programa de Ayudas a Grupos de Excelencia de la Región de Murcia" of Spain.

References

[1] L. Hiller, Computer music, *Scientific American*, 201(6):109–120, Scientific American Inc., 1959.
[2] L. Hiller, L.M. Isaacson, *Experimental music: Composition with an Electronic Computer*, McGraw-Hill, New York, 1959.
[3] http://neuromusic.soc.plymouth.ac.uk
[4] D. Cope, *Experiments in Musical Intelligence*, Computer Music and Digital Audio Series 12, A-R editions, Madison, WI, 1996.
[5] D. Cope, *The Algorithmic Composer*, Computer Music and Digital Audio Series 16, A-R Editions, Madison, WI, 2000.
[6] D. Cope, *Virtual Music: Computer Synthesis of Musical Style*, MIT Press, Cambridge, 2004.
[7] Y.-p. Chen, "Interactive Music Composition with Evolutionary Computation," *ACM SIGEVOlution*, vol. 2, no. 1, pp. 9-16, 2007.
[8] M. Marques, V. Oliveira, S. Vieira, and A. C. Rosa, "Music composition using genetic evolutionary algorithms," in *Proceedings of the 2000 Congress on Evolutionary Computation (CEC-2000)*, pp. 714-719, 2000.
[9] E.R. Miranda, *Composing Music with Computers* , Music Technology series, Focal Press, 2001.
[10] B. Melián, J.A. Moreno, J.M. Moreno, *Metaheurísticas: una visión global*, Inteligencia Artificial 19:7–28, 2003.
[11] J.M. Cadenas, M.C. Garrido, E. Muñoz, Using machine learning in a cooperative hybrid parallel strategy of metaheuristics, *Information Science*, 179(19):3255–3267, Elsevier, 2009.
[12] G. Loy, *Musimathics vol. 1: The Mathematical Foundations of Music*, The MIT Press, Cambridge, 2006.
[13] W. Piston, M. DeVoto, *Harmony*, Norton, New York, 1987.
[14] T.G. Crainic, M. Toulouse, Parallel Strategies for Metaheuristics, In F. Glover and G.A. Kochenberger, editors, *Handbook of Metaheuristics*, pages 475–513, Kluwer Academic Publisher, 2003.
[15] D. Pelta, C. Cruz, A. Sancho-Royo, J.L. Verdegay, Using memory and fuzzy rules in a cooperative multi-thread strategy for optimization, *Information Sciences*, 176(13): 1849–1868, Elsevier, 2006.
[16] S. García, A. Fernández, J. Luengo, F. Herrera, A study statistical of techniques and performance measures for genetics-based machine learning: accuracy and interpretability, *Soft Computing*, 13(10):959–977, Springer, 2009.