Automated generation of contrapuntal musical compositions using probabilistic logic in Derive

Gabriel Aguilera José Luis Galán Rafael Madrid
Antonio Manuel Martínez Yolanda Padilla Pedro Rodríguez

Department of Applied Mathematics
University of Málaga (Spain)
jl_galan@uma.es

Abstract

In this work, we present a new application developed in Derive 6 to compose counterpoint for a given melody ("cantus firmus"). The result is non-deterministic, so different counterpoints can be generated for a fixed melody, all of them obeying classical rules of counterpoint. In the case where the counterpoint cannot be generated in a first step, backtracking techniques have been implemented in order to improve the likelihood of obtaining a result. The contrapuntal rules are specified in Derive using probabilistic rules of a probabilistic logic, and the result can be generated for both voices (above and below) of first species counterpoint.

The main goal of this work is not to obtain a "professional" counterpoint generator but to show an application of a probabilistic logic using a CAS tool. Thus, the algorithm developed does not take into account stylistic melodic characteristics of species counterpoint, but rather focuses on the harmonic aspect.

The work developed can be summarized in the following steps:

(1) Development of a probabilistic algorithm in order to obtain a non-deterministic counterpoint for a given melody.

(2) Implementation of the algorithm in Derive 6 using probabilistic Logic.

(3) Implementation in Java of a program to deal with the input ("cantus firmus") and with the output (counterpoint) through inter-communication with the module developed in Derive. This program also allows users to listen to the result obtained.

Key words: CAS, contrapuntal musical compositions, automated compositions, probabilistic logic

1 Introduction

1.1 Historical background

Humans have been attempting to automate the creation and execution of music for a very long time. One of the first successful attempts to do so resulted in the development in the 14$^{\text{th}}$ century of a clockwork bell which marked time. In 1650, the *musarithmica mirifica*, a machine for composing music, was described by Athanasius Kircher (inspired by Ramon Lull); it can be seen in the Pepys Museum in Cambridge [9].

Great composers such as Haydn, Mozart or Philipp Emanuel Bach used techniques such as rolling dice in the study of random music compositions [19]. Some methods describing this technique can be found in [13].

Many scholars, such as Babbage or Hofstadter [15], have shown the possibility of developing machines able to compose music automatically. One example of these machines was ILLIAC, a computer that in 1955 composed music in the style of Palestrina (16$^{\text{th}}$ century) [14].

Nowadays, probability has a very important role in automatic music composition. Some works in this area can be seen in [7], [16], [18].

1.2 "Cantus Firmus" and Counterpoint

A polyphonic composition consists of two or more independent melodic voices. A "cantus firmus" is a fixed melody to which one or more voices is added in order to obtain a polyphonic composition. So a "cantus firmus" is the basis for a polyphonic composition. A counterpoint is the relationship between the "cantus firmus" and the other voices with a harmony dependence [12].

Strict two-voice counterpoint is a pedagogical method to compose a polyphonic composition with two voices (one is the given "cantus firmus" and the other is obtained attending to different rules). This type of counterpoint can be classified in five groups of increasing complexity called species (first, second, third, fourth and the most complex species called florid counterpoint).

The counterpoint takes into account the current note in the "cantus firmus" (or even the previous ones) and based on this, the note (or notes) in the generated voices are selected.

In this work, we will focus only on the first species counterpoint (note against

note). This kind of counterpoint establishes a way of combining the current note in the "cantus firmus" with a note for the generated voice. We have considered two different types of generated voice: the above and the below voices against the "cantus firmus".

The main goal of this work is to show an application of a probabilistic logic using a CAS tool. Specifically, the developed application is an automated non-deterministic counterpoint generator for a given "cantus firmus". Thus, since the main goal is not to obtain a "professional" counterpoint generator, the algorithm developed does not take into account stylistic melodic characteristics of species counterpoint, but rather focuses on the harmonic aspect.

1.3 Work developed

Two different applications have been developed. The first is a new probabilistic algorithm to obtain a non-deterministic counterpoint (first species) for a given melody. This algorithm has been implemented in DERIVE using probabilistic logic. The application can generate both voices, the above and below ones, against the "cantus firmus". The second application is a JAVA program, as the interface with the user, which deals with the input ("cantus firmus") and with the output (counterpoint). This program also allows one to listen to the result obtained by the first application (by means of midi sound).

Since the algorithm for generating the counterpoint is a non-deterministic one, a standard backtracking technique has been implemented in order to backtrack from a point from which it is not possible to continue. This fact improves the likelihood of obtaining a result. On the other hand, the non-deterministic character of the algorithm, led us to obtain different possible results for a given "cantus firmus" for different executions of the algorithm. Optionally, the program can obtain all the different solutions for a given input. An alternative method to the one we have developed would be to generate all the different solutions and select the one which best fits to the given "cantus firmus". However, this method would take a considerable amount of time since it has an exponential complexity.

1.4 DERIVE *6 and* JAVA

From among the huge amount of software now available on the market, we have chosen the program DERIVE for several reasons:

(1) The wide variety of mathematical resources DERIVE offers together with

the possibility of using a variable with different types, make this software one of the most powerful tools to use in algorithms dealing with mathematical concepts such as the probabilistic logic which is used as the core in our algorithm for generating the counterpoint.

(2) In order to use the automated counterpoint composer program with our Computer Science students, we had to choose software they were used to dealing with. In the mathematical engineering subjects, we use DERIVE to develop different programs to solve specific problems. For example, we have developed different utility files using DERIVE to assist in the teaching-learning process, files such as Multiple Integration [2], Line Integration [6], Complex Analysis [11], Random Variables [4], Automated Theorem Provers [3] and Graph Theory [5]. Our students therefore, are used to dealing with this software.

On the other hand, the main reasons we took into account when we chose DERIVE to work with our students were:

- First of all, this software is, in our view, easier to use than other "more complex" mathematical programs as it operates with a very simple syntax.
- Because of this simple syntax, the student is able to start solving problems using the program within a short period of time, since basic functions and operations are available in several menus.
- It has few requirements, with regard either to memory or the physical space necessary for installing it.
- The DERIVE license is much cheaper than others.

(3) Since DERIVE performs only simple file management (one can save and load only the DERIVE files), we wanted to develop an application able to communicate as much as possible with DERIVE so that this application can serve as a starting point in the future for other works which intercommunicate with DERIVE. That is the main reason why we developed the two different applications mentioned in section 1.3 and fully explained in sections 2 and 3: the first, developed using DERIVE, consists mainly of developing the algorithm for generating the counterpoint, while the second one, developed in JAVA, is focussed on managing the environment and the inter-communication between the first application and the user.

(4) DERIVE is one of the most popular CAS (Computer Algebra System) used nowadays in the teaching of Mathematics for Engineering and in High School Education [10].

On the other hand, one of the main reasons for choosing JAVA to implement the interface with the user is because of the ease with which it deals with midi sounds in order to play the "cantus firmus" and/or the generated counter-

point. Other important reasons for this choice were: JAVA is a multi-platform programming language, it is object oriented and is widely used.

2 Description of the algorithm

2.1 The process

The input of the generator is the sequence of notes called "cantus firmus", and the program generates another sequence corresponding to the above or below voice (or both). In this work, both voices are calculated for the first species counterpoint. So, in this type of counterpoint only one note is generated for each note in the "cantus firmus". The input sequence has to be in a given tonality and mode (C major in our case), and the process is similar to how human composers work, where they have to follow some contrapuntal rules. These rules are not deterministic so several different choices can be selected among in each case. Not all the possibilities are equally suitable, and sometimes the preferences of different rules are contradictory. In some cases, it is not possible to obey the contrapuntal rules and so no result can be obtained. The choice of one note changes the different possibilities for the next one. So the trace of the selections can be shown by means of a graph, specifically by a tree.

The notation used for the natural scale is "c d e f g a b", while the above and below scales are "c' d' e' f' g' a' b' " and "C D E F G A B" respectively. For implementation reasons, we assign a number to each note in order to be able to calculate distances between them by means of subtractions. This assignment has not been made using pitch classes and interval classes [20], mainly because we do not consider either sharps or flats since, in this work, we only consider the key of C major. Future work, as stated in section 5, will be the extension to other keys.

2.2 Rules

In order to generate a first species counterpoint for a given "cantus firmus", we have to take into account different rules (some of them are common for all species and other are specific for each species) [17]. Rules can be different in number and meaning for different authors.

It is important to remark that generally rules are non-deterministic because

they provide some hints on the behavior of the melody to be obtained, but they do not fix one result at each step of the process. In fact, some possibilities are good for some rules and not so good for others.

In order to clarify the non-deterministic and vague character of these rules, let us show some examples of rules given by J. J. Fux [21], [8]:

- Avoid moving in parallel thirds or sixths for very long.
- Attempt to keep any two adjacent parts within a tenth of each other, unless an exceptionally pleasing line can be written by moving outside of that range.
- Avoid having any two parts move in the same direction by skips.
- Attempt to have as much contrary motions as possible.

For these reasons, the rules we have considered take into account some of these counterpoint rules when assigning probabilities to notes. So, if a high probability is assigned to a note, it has more probability of being chosen than another note with smaller probability. This means that not always the "best" option is chosen (in this case, the algorithm would be deterministic and we always would obtain the same result for a given "cantus firmus"). The algorithm developed allows us, optionally, to obtain all the possible outputs for a given "cantus firmus", but it is not the main goal of this work, since we would like to obtain a non-deterministic output.

The assignment of probabilities depends on the previously chosen notes and also on the distance to the corresponding note in the input. Therefore, probability is assigned to each note in accordance with some rules. Finally a probabilistic logic, similar to the one described in [1], chooses the final probability for each note. The classical rules of the first species counterpoint [21], [8], [17] have been condensed in our work into the following four rules: Initial Point, Final Point, Precedence (1, 2 and 3) and Distance.

Let us now describe these rules for the above voice. The rules for the below voice are defined symmetrically.

- The Initial Point rule assigns probabilities 0.25, 0.30 and 0.45 to C, G and c respectively for the first note that is chosen.
- The Final Point rule assigns 0.75, 0.20 and 0.05 to B, A and F respectively for the penultimate note. The final note must be c.
- The Precedence rules assign probabilities depending on the previously generated notes. That is, precedence 1, 2 or 3 take into account only one, two or three previous notes according to the probability functions described below.

◦ Precedence 1.

Precedence 1 is based on the fact that the nearer a note is to the previous one (but different), the greater its probability is. To compute these probabilities, we have used the scheme shown in figure 1 in which we consider, as an example, c to be the current note. In this figure we can see that we give a probability x_1 for the two nearest notes (distance = 1) to the actual one; probabilities $\frac{x_1}{2}$, $\frac{x_1}{4}$, $\frac{x_1}{8}$, $\frac{x_1}{12}$, 0 for distance 2, 3, 4, 5 or greater respectively. On the other hand, distance 0 is not allowed and thus its probability is 0.

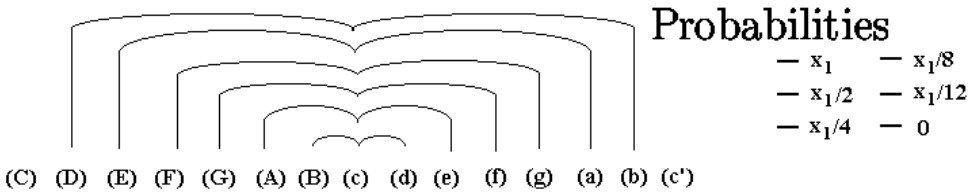

Fig. 1. Probability generation for Precedence 1.

To compute the value x_1 we establish that the sum of the probabilities must be equal to one.

$$\frac{x_1}{12} + \frac{x_1}{8} + \frac{x_1}{4} + \frac{x_1}{2} + x_1 + x_1 + \frac{x_1}{2} + \frac{x_1}{4} + \frac{x_1}{8} + \frac{x_1}{12} = 1$$

So, we obtain that $x_1 = \frac{12}{47}$.

Formally, let X be the last note chosen. The probability y of the note Y being the next selection is given according to the following probability function:

$$f_1(y) = \begin{cases} 0 & \text{if } |Y - X| = 0 \text{ or } |Y - X| \geq 6 \\ \frac{12}{47} & \text{if } |Y - X| = 1 \\ \frac{6}{47} & \text{if } |Y - X| = 2 \\ \frac{3}{47} & \text{if } |Y - X| = 3 \\ \frac{3}{94} & \text{if } |Y - X| = 4 \\ \frac{1}{47} & \text{if } |Y - X| = 5 \end{cases}$$

Note that the sum of these values is equal to 0.5 since each leaf of the density function has two different possibilities because of the absolute value function.

In order to avoid dissonances, augmented fourths are not allowed. Thus, the leaps involving “F” and “B” (or “f” and “b”) have been forbidden since these are the only possibility to get an augmented fourth in C major.

Therefore:

$$f_1(y) = 0 \qquad \text{if} \begin{cases} X = 4 \text{ and } Y = 7 \\ \text{or} \\ X = 7 \text{ and } Y = 4 \\ \text{or} \\ X = 11 \text{ and } Y = 14 \\ \text{or} \\ X = 14 \text{ and } Y = 11 \end{cases}$$

- Precedence 2.

  This rule can only be applied when at least two notes have been previously generated.

  Let X and Y be the last two generated notes. The probability z of the note Z being the next selection is given according to the following probability function:

$$f_2(z) = \begin{cases} 0.7 & \text{if } \operatorname{sgn}(Z - Y) = \operatorname{sgn}(Y - X) \\ 0.3 & \text{if } \operatorname{sgn}(Z - Y) \neq \operatorname{sgn}(Y - X) \end{cases}$$

  This rule can be read as continued ascending or descending is preferred to changing the line determined by the two previous notes. This behavior is shown in figure 2.

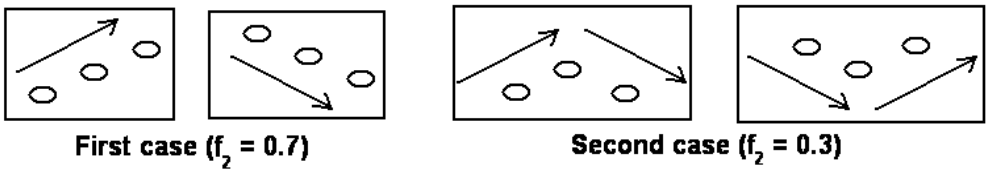

Fig. 2. Probability generation for Precedence 2.

- Precedence 3.

  This rule can only be applied when at least three notes have been previously generated.

  Now we have three possibilities: the four notes (including the current note) are in an ascending or descending scale (the preferred option), there is one change (the second preferred option) and there are two changes (the worst option). These three cases with their different situations can be seen in figure 3.

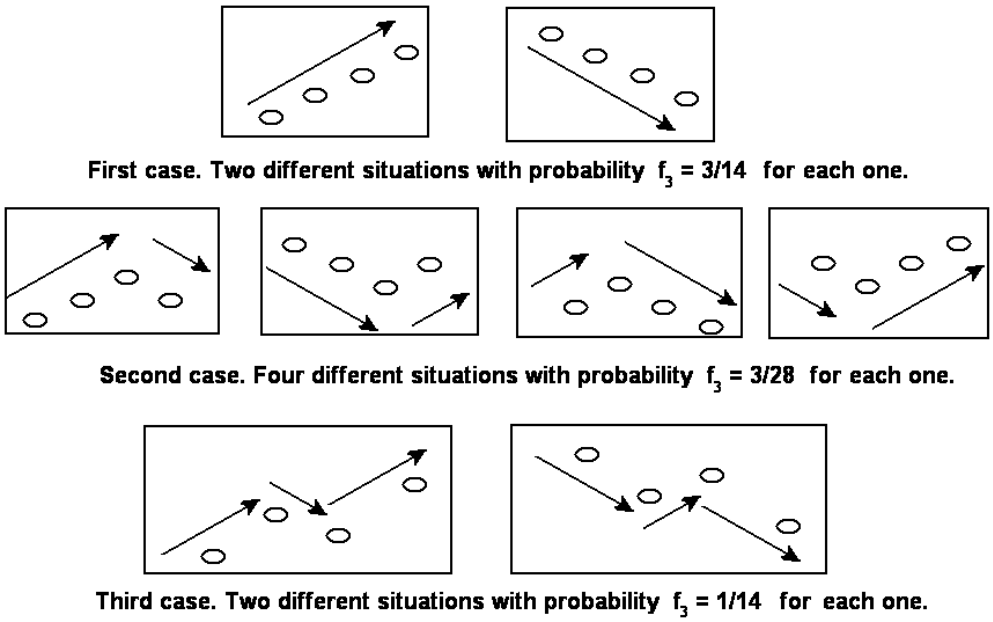

Fig. 3. Possibilities for the triple function.

Formally, let W, X and Y be the last three generated notes. The probability z of the note Z being the next selection is given according to the following probability function:

$$f_3(z) = \begin{cases} \frac{3}{14} & \text{if } \operatorname{sgn}(X-W) = \operatorname{sgn}(Y-X) = \operatorname{sgn}(Z-Y) \\ \frac{3}{28} & \text{if } ((\operatorname{sgn}(X-W) = \operatorname{sgn}(Y-X)) \\ & \qquad \text{and } (\operatorname{sgn}(Y-X) \neq \operatorname{sgn}(Z-Y))) \\ & \quad \text{or}((\operatorname{sgn}(X-W) \neq \operatorname{sgn}(Y-X)) \\ & \qquad \text{and } (\operatorname{sgn}(Y-X) = \operatorname{sgn}(Z-Y))) \\ \frac{1}{14} & \text{if } \operatorname{sgn}(X-W) = \operatorname{sgn}(Z-Y)) \\ & \qquad \text{and } (\operatorname{sgn}(Y-X) \neq \operatorname{sgn}(Z-Y)) \end{cases}$$

Note that there are two different situations for case 1, four for case 2 and two for case 3. So, the sum of the corresponding probabilities is:

$$2 \cdot \frac{3}{14} + 4 \cdot \frac{3}{28} + 2 \cdot \frac{1}{14} = 1$$

- The Distance rule assigns probability y to a note Y in the output sequence depending on the corresponding note X in the input sequence.

Let d be the distance defined by $d(X,Y) = ((Y-X) \bmod 7)+1$. This function provides us with the distance between notes with octave equiva-

lence since we do not use either sharps or flats because we only consider the key of C major.

The probability y of the note Y being the next selection is given according to the following probability function:

$$f_4(y) = \begin{cases} 0.35 \text{ if } d = 3 \\ 0.35 \text{ if } d = 6 \\ 0.25 \text{ if } d = 5 \text{ and } d' \neq 5 \\ 0.05 \text{ if } d = 1 \text{ and } X \neq Y \end{cases}$$

where d' is the distance between the previous note in the "cantus firmus" X' and the last generated note Y'.

2.3 Example

In order to clarify how the algorithm works using the previous rules, we show the following example:

Let us consider the "cantus firmus" "CEDGc" (in C major), shown in figure 4.

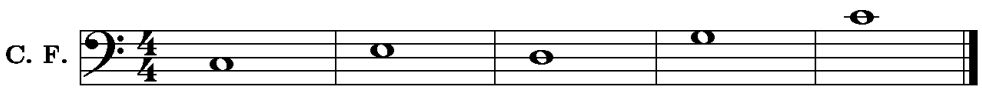

Fig. 4. "Cantus firmus".

(1) The Initial Point rule is the only one which can be applied to obtain the first note. Note also that this rule will not be used again in order to obtain other notes in the output sequence. This rule offers three possible notes: c, G and C with probabilities 0.45, 0.3 and 0.25 respectively. As we can see, although these three notes are valid, not of all them are equally preferred. For example, let us suppose that C is the selected note. This selection will influence the choice of the second note for the output sequence.

(2) In order to obtain the second note, two different rules can now be applied to find out which note can be selected: Precedence 1 and Distance. Precedence 2 and Precedence 3 can not be applied since only one note has been generated. Since this second note is neither the last nor the penultimate note, Final Point can not be applied.

In general, if a rule assigns probability 0 to a note, this note can not be chosen in the current step.

In this case, Precedence 1 assigns probability 0 to any note whose

horizontal distance with C (the previously selected note) is equal to 0 or greater than or equal to 6. Thus, the possible notes for this rule are: D (horizontal distance 1); E (horizontal distance 2); F (horizontal distance 3); G (horizontal distance 4) and A (horizontal distance 5).

On the other hand, the Distance rule assigns probability 0 when the vertical distance is 2, 4 or 7. The unison sound is not allowed and vertical distance 5 is also not allowed if the previous vertical distance was 5. Thus, the only possible note for this rule among the five previous notes is G. Then, G is the selected note for this second step.

(3) In order to obtain the third note, three different rules can now be applied to find out which note can be selected: Precedence 1, Precedence 2 and Distance. Precedence 3 can not be applied since only two notes have been generated. Since this third note is neither the last nor the penultimate note, Final Point can not be applied.

In this case, the possible notes for Precedence 1 are: F, A (horizontal distance 1); E, B (horizontal distance 2); D, c (horizontal distance 3); C, d (horizontal distance 4) and e (horizontal distance 5).

Precedence 2 only assigns probability 0 when the note is the same as the previous one. This case is always considered also in Precedence 1. Thus, Precedence 2 only affects the final probability but it does not provide any additional restriction for the possible notes.

The possible notes for the Distance rule among the nine previous notes are: d, F, A, B. Let us suppose that d is the selected note for this third step.

(4) Since the fourth note is the penultimate, A, B and F are the only notes allowed using the Final Point rule. Nevertheless, F is not possible because Precedence 1 and Distance rules assign probability 0 to it. A is also not allowed since the vertical distance with the corresponding note in the "cantus firmus" is equal to 2, thus, the Distance rule assigns probability 0 to A. Therefore, the only possible note in this step is B.

(5) Since the fifth note is the last one, the Final Point rule imposes that c be the chosen note.

Thus, the output sequence would be "CGdBc" which is shown in figure 5.

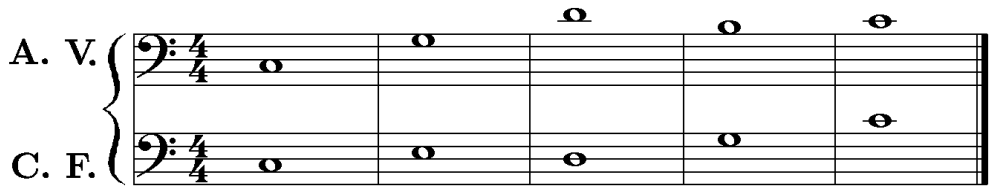

Fig. 5. Output of the example.

The process can be described by means of the graph (tree) shown in figure 6.

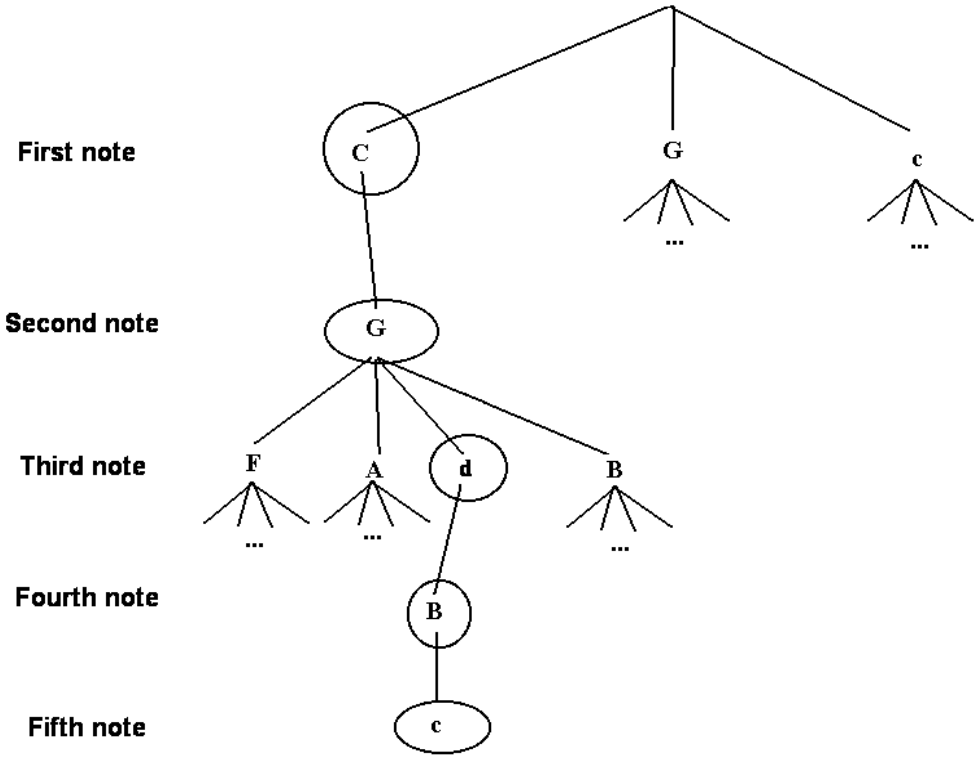

Fig. 6. Trace tree of the example.

This is the result obtained for a single execution of the algorithm. Since the algorithm is non-deterministic, other executions of it can provide different above voices. For example, in this case, after executing the program 100 times for this "cantus firmus" we obtained the following results: "C G d B c" (23 times), "G c A B c" (17 times), "G c d B c" (13 times), "G e A B c" (3 times) and "c B d B c" (44 times). The first one is the output obtained in the example described below. Note that this output is not the one most frequently generated by the algorithm.

2.4 Backtracking

It is possible that when applying the rules described in section 2.2, we obtain a note for which there is no way of finding the corresponding note for the output. In this case the program tries other possibilities with a standard backtracking technique (going back to the previous node in the trace tree when no solution has been found for the current node).

Finally, if all possibilities have been considered and no solution has been found, the result of the algorithm would be *no counterpoint is possible for this "cantus firmus"*.

3 Description of the environment

In this section, we present how to use the environment developed in JAVA.

3.1 Menu bar

The menu bar consists of 5 different menus: `File, Edit, Counterpoint, See` and `Help`. Let us now see the different options in each menu.

`File` **Menu**

- Open File: allows us to load a musical composition.
- Save: allows us to save the current musical composition in `TXT` format.
- Derive path: allows us to set the path where the program DERIVE is located.
- Exit: allows us to exit the application.

`Edit` **Menu**

- Input: allows us (when on) to edit the "cantus firmus" directly in the main window.
- Output: allows us (when on) to edit the generated Counterpoint directly in the main window.
- Trace: allows us (when on) to run the DERIVE application with the trace option in order to follow the execution of the algorithm.

`Counterpoint` **Menu**

- Above voice: selects the option of finding an above voice when running the DERIVE algorithm.
- Below voice: selects the option of finding a below voice when running the DERIVE algorithm.
- Both voices: selects the option of finding both the above and below voices when running the DERIVE algorithm. When choosing this option, the DERIVE algorithm obtains an above voice and a below voice for the given "cantus firmus" but it does not take into account any harmonic or melodic compatibility between the two voices. Furthermore, the algorithm automatically transposes the "cantus firmus" up or down an octave where needed in order to obtain the below and the above voices.

`See` **Menu**

- Input editing: displays an option box in the main window to set the editing of the "cantus firmus" to on or off.
- Output editing: displays an option box in the main window to set the editing of the counterpoint to on or off.
- Trace edition: displays an option box in the main window to set the trace option of the generator algorithm to on or off.

`Help` **Menu**

- Help: shows a small help.
- Notes equivalence: shows the notation used for the notes.
- About Musical Counterpoint: shows the credits of the software developed.

3.2 Real time modifications

It is also possible, from the main window, to make some real time modifications:

(1) Before, after or while a composition is playing, different instruments can be chosen for each voice ("cantus firmus", above and/or below voices).
(2) We can also change the volume in real time.
(3) Finally, the *tempo* (Adagio, Andante, Allegro, Presto, Vivace and different combinations of them) can also be changed in real time.

3.3 Inter-communication with Derive

Once the "cantus firmus" is entered and the different options for the counterpoint are selected, we can generate the below, the above or both voices with or without the trace option by executing the application developed in Derive. This can be done by pressing the *Counterpoint* button in the main window. Once this is done, Derive is opened using our algorithm with the given "cantus firmus" and the selected options. Then we have to execute the highlighted line in Derive's work window and finally exit Derive answering *Yes* when we are asked to save the changes.

In order to get the generated counterpoint in the main window of the Java environment, we have to press the *Import* button and then, the corresponding counterpoint will appear either above or below the "cantus firmus".

3.4 Playing the composition

Once the counterpoint has been imported, we can play the composition by pressing different buttons. The different options are: playing the "cantus firmus" alone, each voice (the below or above one) individually, either voice (below or above) with the "cantus firmus", both voices (below and above) together, but without the "cantus firmus" and, finally, both voices together

with the “cantus firmus” [1] .

4 Results

We show in this section some examples of the results obtained after using the algorithm for different “cantus firmus”.

4.1 Example 1: Above voice against “Cantus firmus”

After executing the algorithm for the “cantus firmus” “C G E D E F E D C” we obtained the above voice “c B c B G A c B c”

The composition is shown in figure 7.

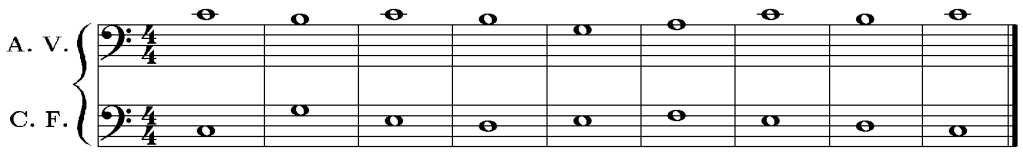

Fig. 7. Example 1: Above voice against “Cantus firmus”.

4.2 Example 2: Below voice against “Cantus firmus”

After executing the algorithm for the “cantus firmus” “c d e g f e d e c” we obtained the below voice “C F G c B G F A c”.

The composition is shown in figure 8.

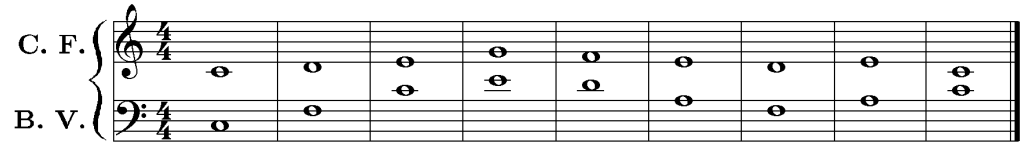

Fig. 8. Example 2: Below voice against “Cantus firmus”.

[1] See the restrictions explained in option “both voices” of the `Counterpoint Menu` in section 3.1 when playing the below and above voices together.

4.3 Example 3: Above and below voices against "Cantus firmus"

After executing the algorithm for the "cantus firmus" "C E G A G F E D C" we obtained the above voice "C G B c B d c A c".

Considering the same "cantus firmus" (one octave above), "c e g a g f e d c", we obtained the below voice "C G B c e d A B c".

The compositions are shown in figure 9.

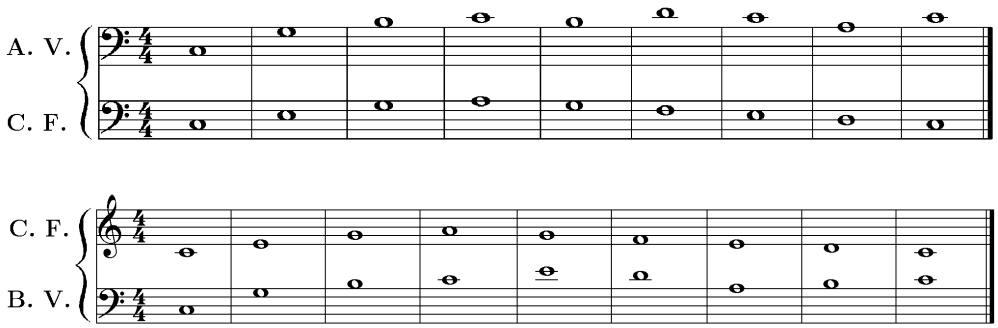

Fig. 9. Example 3: Above and below voices against "Cantus firmus".

5 Conclusions and future work

The results obtained using our new algorithm have been checked by teachers from the Conservatory of Music of Málaga. They agreed that the automated counterpoints generated were good considering the restrictions imposed.

Some future work directly related with this application will be:

- Translation of the user interface of the application to different languages. Nowadays a Spanish version of the whole application is available together with a user's manual and an installation manual at
  `http://www.satd.uma.es/matap/jlgalan/contrapunto/`
  When the multilingual version is ready, it will be available at
  `http://www.satd.uma.es/matap/jlgalan/conterpoint/`
- Development of algorithms for generating second, third, fourth and florid species counterpoints.
- "Cantus firmus" in different keys and modes. In this case, we will use pitch classes and interval classes.
- Development of friendly interfaces which inter-communicate with our applications developed in DERIVE.

Finally, we thank the anonymous reviewers for their valuable comments and suggestions which have led us not only to improve the article but even to find some errors in the program code.

References

[1] G. Aguilera. Rewriting in a probabilistic logic with imprecise probabilities. *Proceedings of Applications of Logic and Mathematics to Programming*. Alicante. 2003.

[2] G. Aguilera, C. Cielos, J.L. Galán, M.A. Galán, A. Gálvez, A.J. Jiménez, Y. Padilla and P. Rodríguez. A basic course of multiple integrals with a computer algebra system. *Proceedings of the 3rd International Conference on the Teaching of Mathematics an the Undergraduate Level*. Istanbul. 2006.

[3] G. Aguilera, J.L. Galán, A. Gálvez and P. Rodríguez. ATPCL.mth: Automated Theorem Provers for Propositional Classical Logic with Derive. *Proceedings of TIME 2004*. Montreal. 2004. I.S.B.N.: 3-901769-59-5.

[4] G. Aguilera, J.L. Galán, Y. Padilla and P. Rodríguez. Random_distributions.mth: Random samples from distributions with Derive. *Proceedings of TIME 2004*. Montreal. 2004. I.S.B.N.: 3-901769-59-5.

[5] G. Aguilera, J.L. Galán and P. Rodríguez. GRAPH_ALGORITHMS.MTH: Graph algorithms using display step in Derive 6. *Proceedings of DES-TIME-2006*. Dresden. 2006. I.S.B.N.: 3-901769-74-9.

[6] C. Cielos, J.L. Galán, M.A. Galán, A. Gálvez, A.J. Jiménez, Y. Padilla and P. Rodríguez. Line integrals with computer algebra systems. *Proceedings of the 5th International Conference APLIMAT*. Bratislava. 2006. I.S.B.N. 80-967305-6-8.

[7] M. Farbood and B. Schoner. Analysis and Synthesis of Palestrina-Style Counterpoint Using Markov Chains. *Proceedings of International Computer Music Conference*. Havana. 2001.

[8] J. J. Fux. Study of Counterpoint. *Alfred Mann (Editor) . W. W. Norton & Company*. 1990

[9] M. Gardner. The Colossal Book of Mathematics. *W. W. Norton & Company*. 2001.

[10] J.L. Galán. The role of computer algebra systems in mathematics teaching for Engineering degrees. *Plenary lecture of the 5th International Conference APLIMAT*. Bratislava. 2006. I.S.B.N. 80-967305-6-8.

[11] J.L. Galán, M.A. Galán, Y. Padilla and P. Rodríguez. RESIDUES.MTH: Solving Problems of integration using the residue theorem. *Proceedings of TIME 2004*. Montreal. 2004. I.S.B.N.: 3-901769-59-5.

[12] I. Girton. An Introduction to Species Counterpoint. *http://www.listeningarts.com/music/general_theory/species/menu.htm*. 2003.

[13] S. A. Hedges. Dice Music in the Eighteenth Century. *Music and Letters. Volume 59, Number 2*. 1978.

[14] L. Hiller and L. Isaacson. Musical Composition with a High-Speed Digital Computer. *Journal of the Audio Engineering Society*. 1958.

[15] D.R. Hofstadter. Godel, Escher, Bach: An Eternal Golden Braid. *Penguin Books*. 2000.

[16] R. Klingerf and G. Rudolph. Automatic Composition of Music with Methods of Computational Intelligence. *WSEAS Transactions on Information Science and Applications*. 2007.

[17] S. G. Laitz The Complete Musician An Integrated Approach to Tonal Theory, Analysis, and Listening. *Oxford University Press*. 2007.

[18] E. Miranda. Composing Music with Computers. *Focal Press*. 2001.

[19] Z. Ruttkay. Composing Mozart Variations with Dice. *Teaching Statistics. Volume 19, Number 1*. 1997.

[20] J. N. Straus. Introduction to post-tonal theory. *Prentice Hall*. 2004

[21] Wikipedia. Counterpoint. First Species *http://en.wikipedia.org/wiki/Counterpoint#First_species*.