Available online at www.sciencedirect.com

ScienceDirect

Mathematics and Computers in Simulation 80 (2010) 1200–1211

MATHEMATICS AND COMPUTERS IN SIMULATION

www.elsevier.com/locate/matcom

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

Received 3 December 2007; received in revised form 27 March 2009; accepted 13 April 2009
Available online 23 April 2009

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.

Keywords: 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 14th century of a clockwork bell which marked time.

* Corresponding author. Tel.: +34 95 2 13 27 63; fax: +34 95 2 13 27 66.
E-mail address: jl_galan@uma.es (J. Luis Galán).

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 [11].

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 (16th 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 DERIVEusing 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 mathe-