# Preface

The purpose of this essay is the development of an A.I application called “Counterpoint” that will move the music industry a step further. This first experiment will hopefully help computer musicians, students and composers to develop compositions from given melody fast and efficiently. This attempt is based upon a maze-solving algorithm which finds the best contrapuntal solution to a 1st  species melodic line (cantus firmus). Because of the [n.] in music technology, there are also **[v.]** to computer musicians of no or little music knowledge who they like to compose music. They see the creative part of music and they don’t want to get involved in rules and restrictions. There is a great deal of computer musicians who ignore the semiology of music. For them it’s “I hit this button and this sound comes up

New drum machines, such as…provide many features that composers of the past would definitely kill to have. When I first saw a certain drum machine, I thought “This reminds me a maze. What if I find a second melody applying a maze-solving algorithm?”

# Introduction

User-driven system that examines the style of user and generates the best counterpoint solution. an algorithm much simpler and quicker, which could somehow be considered “smart” and music-equipment friendly.

Real-time solution under demands of limited memory.

The algorithm takes into consideration basic rules. It is important to note that the algorithm creates free counterpoint which is not based on strict harmonic rules. In other words, the algorithm does not follow chord progressions or any harmonic patterns. Free counterpoint became more and more popular after the last quarter of 19th century, where the new generation of composers, such as Mozart, Beethoven, Schumann etc. were looking for ways to expand (or even bypass) the strict style of counterpoint.

In addition, we consider that the key is always modal C Ionian (or C major in major/minor western system)

The source code of “Counterpoint” can be applied to any open-source score writer (such as musescore), or even to a DAW as a plug-in.

# Literature review

The literature was more than helpful for me, as I gained a thorough understanding of the topic I would like to write about. Every relevant source fit the missing points to my research and [**v**.] me to create an efficient Expert System and a more efficient algorithm.

Starting over, the very interesting report of (Schottstäd, 1984) has put Fux counterpoint rules under the viewpoint of a rule-based Expert System. All counterpoint rules have been represented in IF…THEN statements, in order for the computing system to find a valid countermelody in agreement to cantus firmus.

(Gwee, 2002) in her dissertation examines the process of composition from the point of computational theory and Machine Learning. She suggests a quite interesting rule-based Expert System with an algorithm that decides, among a set of rules, which one is necessary and which is not, in order for the best possible countermelody –among a set of valid countermelodies- to be chosen. Use of scoring. In case of failure, she also suggests a Recovery Rule that reconsiders the importance of those rules. Additionally, she applies some stylistic rules with fuzzy logic to results, in order for countermelody to sound more human-like […] optimization and heuristic algorithms for her application, such as best-first search with branch-and-bound pruning algorithm, genetic algorithms, exhaustive search

(Tanaka, Nishimoto, Ono, & Sagayama, 2010) proposed a stochastic approach to counterpoint using Markov chains to create a two-voice free counterpoint composition (not first-species) based on the style of imitation. Both cantus firmus and countermelody are produced randomly and adjust by following certain rules stored in Knowledge Base. The cantus firmus exposes the main theme (a series of notes of certain pitch and duration which contribute the rhythmic pattern) and then the countermelody imitates (repeats) this theme in later measures, while cantus firmus develops free counterpoint.

A quite interesting meta-heuristic approach aiming to automated scoring can be found in the working paper of (Herremans & Sörensen, 2011). In sort, an algorithm simultaneously generates a first-species cantus firmus of random fragments and a countermelody, which adheres to Fux’s rules in a strict manner. Just like (Gwee, 2002), for every fragment there is a score that has been applied according to the extent that the rules are followed. The restrictions refer to both horizontal and vertical aspect of melody just like (Schottstäd, 1984) did. The total sum of those scores form the objective function and therefore a possible melody. In order to reduce much *randomness* to the final output, a Local Search heuristic has been applied. This algorithm searches to the neighborhood of the generated melody and optimizes the solution by modifying some notes moving them to other directions inside the same neighborhood. Since heuristics do not always return the most optimized solution, they introduced *meta-heuristics* for high-quality melodies. More precisely, they developed Variable Neighborhood Search algorithm (V.N.S), which proposes movements to other neighborhoods that may reduce the score of the objective function even more.

In the inspiring work of (Illescas, Rizo, Inesta, & Ramirez, 2012) there is the powerful and handy tool of automatic melody analysis from the perspective of counterpoint. Once the melody is inserted, a well-trained model examines it in tonality, harmony and style and classifies each note to harmonic or non-harmonic (i.e., passing tone, appoggiatura, suspension, escape, anticipation or pedal).

(Madsen, 2013) created an Expert System that produces fair two-voice counterpoint melodies of all species in the style of da-Palestrina. His Knowledge Base contains rules based on the book of professor Knud Jeppsen and they are split into two big categories: rules that concern harmony and rules that concern the melodic contour. Moreover, some of these rules have been described as *strict*, while others as features a composition is nice to have (*softer* rules). Such features are **countermovement** of two voices and **cadences**. It is the first paper that refers to such musical elements. As for the search mechanism, he uses an evolutionary algorithm which evaluates a set of randomly generated countermelodies in order to find the best counterpoint. The evaluation process follows a penalty strategy every time a rule is violated. The aim is to be used in further music development beyond counterpoint.

(Samory, Mandanici, Canazza, & Peserico, 2014) also made a meticulous comprehensive survey upon the importance of counterpoint rules moving the rule system of (Gwee, 2002) a step further. From their perspective, the significance of all rules changes from time to time and from species to species. Actually, they differ among composers and experts as well. So, they categorized counterpoint rules to *absolute* (rules that are generally considered as necessary or fundamental), *majority* (most of the composers agree that are necessary), *undefined* (the interpretation of the same rule differs from author to author) and these of *implicit* (which are more subjective).

(Komosinski & Szachewicz, 2015) followed the same pattern with (Herremans & Sörensen, 2011) and (Madsen, 2013), suggesting dominance relation method for composing first species counterpoint. Their formula is based on penalty-or-reward strategy for every time a rule is violated (or satisfied accordingly). The algorithm generates a set of potential countermelodies and in the end decides which countermelody is “better” or “more interesting” than another.

All the above surveys, as well as the exhaustive comprehensive survey of (Fernández & Vico, 2013), pointed out that everybody keeps their Intelligent Systems in the context of fixing, improving or rejecting countermelodies that are arbitrarily generated by computer itself. In essence, this means that their Systems are trained to accept or produce melodies sticked to “essential” counterpoint rules, whose significance changes from time to time and from style to style (i.e., Fux, da Palestrina, Schönberg etc.). In this point I will highlight the words of (Fux, 1725) that are also referred to (Schottstäd, 1984); *rules should be seen as guidelines and not like absolutes*. (Komosinski & Szachewicz, 2015) also made an excellent point, mentioning that there is no music Literature that defines the significance of such rules in an absolute way.

So far, no research has been done about a System that learns from user’s style and tries its best to accompany his/her melody with a free first species counterpoint following a set of fundamental, general-accepted rules (i.e., no hidden/exposed/parallel fifths and eights, no intervals of 2nd, 7th and perfect fourth etc.). The fact that there is always some kind of judgement about the cantus firmus melody or the one is produced, also led me to infer that the aforementioned research is not encouraging creativity at any level and holds user away from self-expression and creative process. Above all, my vision is to develop an application which won’t judge the user, but will learn from him instead.

To this point I must underline the absence of time interval and limited memory as restrictions. The only work that refers to a quick solution to a *reasonable amount of time* is the work of (Herremans & Sörensen, 2011). Because of the fact that Machine Learning and evolutionary algorithms improve relatively slowly and attractthe full attention of CPU, they are inappropriate for real-time solutions. Similarly, technological equipment such as drum pads cannot store huge sets of potential melody paths and additional memory will increase their cost. (Barnouti, Al-Dabbagh, & Naser, 2016) noted that such limitations are the center of interest for video games with embedded A.I. More precisely, role-playing games, strategy games and maze-solving games usually contain an artificially intelligent hero or agent starting from a specific point aiming to reach another point (that is called destination) through the shortest or the most beneficial path. This type of A.I is applied with the help of special algorithms called “shortest path algorithms”

Let’s alter our viewpoint a little bit and place an imaginative character upon the very top left cell of the grid as shown in [figure]. Let us now suppose that the character is aiming at the most top right cell [lloks like a dungeon with obstacles and try to find the sortest path among the available cells]. If we replace those cells with music notes, our hero will try to find the most beneficial melodic path starting from measure 1 aiming to reach the last measure avoiding the obstacles (created by fundamental rules). Concerned that there is no relevant research so far that approaches automated counterpoint solution from the perspective of maze-solving algorithms, I decided to solve that specific problem with one of those techniques. In addition, (Barnouti, Al-Dabbagh, & Naser, 2016) also referred that the shortest path algorithms are very common not only in game programming, but also in almost every real-time solving problems including logistics, management systems, analytics, networking, and design. By this approach of counterpoint problem, I can show how shortest-path algorithms can be used in music as well.

To summarize, in this research I will develop an Expert System that produces a free counterpoint from a cantus firmus given by the user. All rules will be stored in a Knowledge Base and the algorithm I will implement for the is the A\* and fill this gap not only in literature, but also in music market. Create an optimized countermelody adapted to user’s input.

# Implementation

[**image**] input->[black box] -> output

[By and large], the algorithm for automated first species counterpoint consists of [**number**] modules:

1. *Generator:* Transforms the grid to semi-directed weighted graph.
2. *Reviewer*: Studies the cantus firmus inserted by the user and extracts a report for cantus firmus melody concerning motions and intervals between melody laps. Then it applies the additional cost to the nodes of the graph.
3. *Search* *Engine*: Finds the path with the minimum cost among the nodes, which is the optimized countermelody.
4. *Recovery Tool*: Generates an inversion of the given melody in case an optimized solution cannot be found.

In the following paragraphs I will [**v.**] more details about the assumptions have been made to this research, semiotics, knowledge representation and the algorithm behind Generator, Reviewer and Search Engine.

## Assumptions

According to (Fux, 1725), writing above a cantus firmus is slightly different from writing below of it to ways that are beyond the context of this research.

Drum pad is a block, thus grid-like. Array XxY where x is the notes and y the mesaure [**image**]. Let us assume that every melody will be written in simple key of no sharps or flats [**image]**. To this key correspond the modal scales[[1]](#footnote-1), such as C Ionian (identical to C major scale), D Dorian, E Frygian, F Lydian, G Myxolydian, A Aeolian (identical to A natural minor scale), and B Locrian. In order to evaluate the performance of the algorithm in demanding situations, we also assume that the drum pad is big enough to carry 15x8=120 pads.

There must be exactly two notes in every measure. One note for the cantus firmus and one for the countermelody. In other words, none of the aforementioned voices can have a rest and there must not be an empty stave.

To enhace creativity, I decided not to follow the standpoint of (Fux, 1725) concerning the harmonic aspect of first species counterpoint and let the user produce a free form of counterpoint.

Unlike to (Schottstäd, 1984), the cantus firmus and the first note of countermelody will be defined automatically. In other words, the algorithm will decide the path it should be followed and therefore which note to be chosen in the beginning (i.e. the same note to cantus firmus -unison- or an octave higher).

## semiotics

The pad will be represented as a grid of 15 rows and 8 columns. Counting from bottom to top, the last row refers to the note C…C (middle C)…C [**image**]. Since in first species counterpoint only whole notes are allowed, we assume that the mark ‘\*’ denotes a whole note. As for empty staves (lines and gaps), they will be represented with ‘.’ character. Concerning time signature (4/4, ¾, etc.), counterpoint was primarily developed during the middle ages, where modern metric system hasn’t been invented yet. Thus, there is nothing more than whole notes in a one-measure pentagram, so time signature is none of our concern.

## knowledge representation

This implementation was designed under the state that the user is free to press any button he desires and expects for the machine to produce a melody inside a very short time interval. Deciding which set of rules is redundant and which indispensible is quite hard to tell, though, as the set of valid countermelodies is exponential in Cantus Firmus length. Besides, the nature of Cantus Firmus in our case will not be as expected in books most of the times and it, therefore, exceeds school counterpoint. Because of user’s freedom to construct any melody he desires, there is big possibility that the melody may not start and end to the same note, which is a fundamental rule for key and scale definition.

In addition, deciding which rule to follow and which to reject is - from music perspective- highly dependable from the cantus firmus contour itself. Fux rules may prohibit intervals of 7 or three skipwise motions in a row, but this could be the characteristic that makes user’s melody unique. Furthermore, Gwee’s Recovery Rule[[2]](#footnote-2) will consequently start over the search for a valid countermelody and therefore, it requires more time. For all the above reasons, I suggest the alternative that the algorithm will produce an inversion of the given melody.

As for the set of rules I followed the paradigm of (Schottstäd, 1984), but considering the demands of my research I proposed an alternative. What I stored in Knowledge Base is mainly a description of possible motions a melody can do and the intervals are produced by the convolution of Cantus Firmus and countermelody. I did not take into consideration the quality of the intervals (i.e. major, minor, augmented, diminished), since the algorithm does not count the distance between notes in semitones. In short, these rules are the following:

Table 1: Single voice motion description

|  |  |
| --- | --- |
| **Motion** | **Description** |
| Lap | The distance between two successive notes in the same voice. |
| Stepwise motion | The voice moves in neighbor notes where the lap between them is 2. |
| Skipwise motion | The voice moves in laps more than 2. |
| Repetitive motion | The voice repeats the same note (the lap is 1) |
| Ascending motion | The voice is moving upwards. |
| Descending motion | The voice is moving downwards. |
| Symmetric motion | The voice makes a skipwise motion from one direction and then a stepwise motion to the opposite and vice versa. |

Concerning the motions when the two voices sound together:

Table 2: Two-voice motion description

|  |  |
| --- | --- |
| **Motion** | **Description** |
| Interval | The distance between two notes that sound together. |
| Similar (also direct) motion | The two voices move to the same direction (upwards or downwards) but by different intervals. |
| Parallel motion | The two voices move to the same direction (upwards or downwards) and by same intervals |
| Contrary motion | The two voices move to different directions (one upwards the other downwards and vice versa) |
| Oblique motion | The one voice moves upwards or downwards and the other stays in the same note. |
| Unison | The two voices lie in the same pitch note |

Because of the big diversity among the importance level of each counterpoint rule, I decided to go along with (Samory, Mandanici, Canazza, & Peserico, 2014). I implemented and categorized the following rules to *fundamental* (rules that the general concensus approves as fundamental for writing music correctly) and *redundant* or *stylistic* (Gwee, 2002).

Table 3: Funtamental rules

|  |  |
| --- | --- |
| **Rule** | **Description** |
| The first note of counterpoint should be either the same to cantus firmus (unison), or an octave higher. | [the algorithm automatically applies a ‘@’ to every node but the same and an ocate higher in the first and last collumn.] |
| The final note (finalis) should be approached by step, either with ascending or with descending motion (leading note approach). | [the algorithm automatically applies a ‘@’ to every node but the same and an ocate higher in the first and last collumn.] |
| The leading note must not be doubled but only sound together with the other leading note approach. |  |
| No intervals of 2, 4 and 7between two voices | [the algorithm automatically applies a ‘@’ to every 2,4 and 7 interval between notes.] |
| Overlap | A skipwise motion where the voice makes an intervalic lap greater than an octave[[3]](#footnote-3). |
| Parallel fifths and eights | The two voices make parrallel motion by an interval of 5 (or 8 accordingly) from measure to measure. |
| Concecutive or exposed fifths and eights | The two voices reach the measure before the last with contrary motion and then they move in parallel creating an interval of 5 (or 8 accordingly) to the last measure. In the same pattern, from an interval of 5 (or 8 accordingly) the two voices move in similar motion creating an interval different from 5 or 8. |
| Voice crossing | A lower voice sings to a pitch higher than the higher voice (creates registral confusion). |

Table 4: Redundant or stylistic rules

|  |  |
| --- | --- |
| **Rule** | **Description** |
| Leap of seven | A skipwise motion where the voice creates an intervalic lap of seven. |
| Dissonant leap | A voice sings a diminished or augmented interval. In C major scale these are B-F (5th diminished) and F-B (4th augmented). |
| Dissonant interval | The two voices create a diminished or augmented interval. In C major scale these are B-F (5th diminished) and F-B (4th augmented). |
| No more than 3 oblique motions in a row |  |
| Parallel thirds and sixths | The two voices make parrallel motion and they form an interval of 3 (or 6 accordingly) from measure to measure. |

There are some rules like voice crossing and no intervals of 2, 4 and 7between two voices, which I didn’t store in Knowledge Base. The reason I did this is because I wanted to save some time from long rule checks. Besides, their construction is not dependable from concurrent melody motions, like parallel/exposed 5ths and 8ths. It is more like a choice and in counterpoint such intervals are generally avoided.

Since I ensured that … some more ‘stylistic’ rules need to be applied. Stylistic rules make the difference between machine and human approach to counter melody. [**to what extend? What a machine does that human wouldn’t do?**]. Complying to strict rules may produce something predictable. We generally need a solution very close to human, so as an expert cannot tell the difference -at least quite easily. Thus, I don’t want just a solution (even if it is monotonous or with many oblique motions), but a solution with some stylistic approach as well. What it needs to be discussed further is: “what are those stylistic rules?”.

To begin with, the style is more like a manner, a habit, or a way someone chooses to adopt in his writing [**source? Who said that? Is it my oppinion?]**. It is true that there are many stylistic approaches out there, from medieval and gregorian to classical, jazz, modern and pop. Supposing a manufacturer company releases a drum pad with a counterpoint feature, it is more than clear that is impossible to cover all stylistic demands. So the company will inevitably apply a specific style, according to the expert musician who was hired to contribute the Expert System. That sounds like a great idea, especially if this expert is a well-known artist (such as Will.i.am, Tiesto, or Armin van Buuren etc.), or even a celebrity. Many followers and fans who somehow immitate or are inspired by their style will **[v.]** to own it. My impression is that such applications should not rely on specific style to find a solution for two main reasons. First, as (Deliège & Wiggins, 2006) also stated, this does not let much to imagination and therefore to creativity and development of personal style. Second, it may not be comfortable to a certain group of users (potential customers) that do not follow expert’s style. Given the above, I decided that the algorithm should rely on user’s style only and change the importance of rules depending on the case.

In order to achieve this goal, the algorithm should examine user’s input according to its melodic ‘contour’. By contour we mean the melodic movement and all intervallic leaps… additional cost according to cantus firmus motion…For instance, [**pic of a score with a melody**]. As (Gwee, 2002) mentioned, stylistic rules are of heuristic nature. Since I decided to implement A\* heuristic algorithm… the algorithm gathers some data… and using this formula… will cause an increment of additional node cost. In this way I ensure that the countermelody won’t follow the motion pattern of cantus firmus, but it will convolute in a more symmetric [**explain**] manner. [reduntant rules some fuzziness]

As mentioned before, inspired by the impressive idea of (Gwee, 2002) for a Recovery Rule in the event of failure, my algorithm suggests an inversion of cantus firmus instead of separating redundant from indispensible rules.

1. Without their stylistic demands [↑](#footnote-ref-1)
2. (Gwee, 2002) [↑](#footnote-ref-2)
3. This rule is indispensable when it comes to choral counterpoint, but it is generally allowed to other instruments. [↑](#footnote-ref-3)