Possible Voice Leadings Generation of First-Species Counterpoint with Depth-First Search and Backtracking

Rayhan Hanif Maulana Pradana - 13521112
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521112@std.stei.itb.ac.id

***Abstract*—Counterpoint is a form or a technique in musical composition regarding the interaction of polyphony, which means the simultaneous combination of two or more melodic lines. The species counterpoint is a step-by-step guide of writing contrapuntal music by giving rules to writing tonal voice leading, that is, the movement of the melodic lines, from the least complex, the First-Species Counterpoint to Fifth-Species Counterpoint. The counterpoint of a given melodic line can be generated by a computer, which will be demonstrated by using the Depth-First Search algorithm with Backtracking. This does not mean however, that budding composers should neglect the practice of writing counterpoint. The computer in this regard is used as a tool to help the practice and analysis of contrapuntal writing.**

Keywords—counterpoint; musical composition; polyphony; tonal voice leading; species counterpoint; first-species counterpoint; depth-first search algorithm; backtracking algorithm

I. Introduction

Species counterpoint is a step-by-step rule or guide of writing contrapuntal music, a form of music where two or more melodic lines move independently but are coherent with the rest of the lines and forms a harmonic interaction. Species Counterpoint works by giving certain rules on how to write a tonal voice leading, the movement of individual melodic lines adhering to classical tonality. Species counterpoint was first introduced by Johann Joseph Fux (1680-1741), an Austrian composer from the late baroque era in his book Gradus ad Parnassum or "Steps to Parnassus". It contains the First-Species Counterpoint to the Fifth-Species Counterpoint and its exercises for students of music composition. The First-Species Counterpoint, also known as the 1:1 Counterpoint, is the least complex of the five. It only has two rules and several suggestions or best practices of writing a voice leading. Moreso, the First-Species Counterpoint only has two melodic lines, one is a given, or known as the Cantus Firmus, and the other one is the melodic line in which the practicing student must write on his own. And lastly, the corresponding melodic lines is written in one-by-one, or note-against-note, hence the name 1:1 Counterpoint.

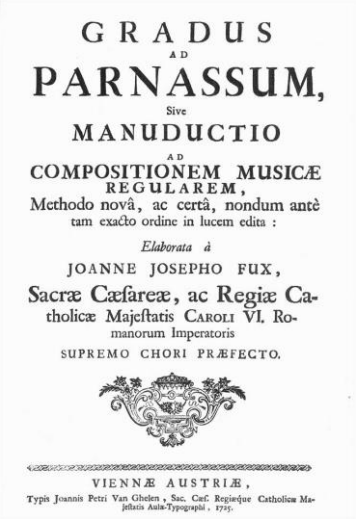
GRADUS
AD
PARNASSUM,
Sive
MANUDUCTIO
AD
COMPOSITIONEM MUSICÆ
REGULAREM,
Methodo novâ, ac certâ, nondum antè
tam exacto ordine in lucem edita :
Elaborata à
JOANNE JOSEPHO FUX,
Sacræ Cæsareæ, ac Regiæ Ca-
tholicæ Majestatis CAROLI VI. Ro-
manorum Imperatoris
SUPREMO CHORI PRÆFECTO.
VIENNÆ AUSTRIÆ,
Typis Joannis Petri Van Ghelen, Sac. Cæs. Regiæque Catholicæ Ma-
jestatis Aulæ Typographi. 1725.

Figure 1.1 The Gradus ad Parnassum book

(Source: https://www.google.com/url?sa=i&url=https%3A%2F%2Fcommons.wikimedia.org%2Fwiki%2FFile%3AFux-Gradus-ad-Parnassum.jpg&psig=AOvVaw2axFulBQ-dLjWU5DY8mqDD&ust=1684768102816000&source=images&cd=vfe&ved=0CBEQjRxqFwoTCKDHwbPYhv8CFQAAAAAdAAAAABAE)

Species Counterpoint were studied by great composers such as Haydn, Mozart, Schubert, Schoenberg, and Beethoven. The mastery of counterpoint in these composers is apparent. Counterpoint gives insight on how harmony really works, which is why it is still taught today to students.

II. Theoretical Basis

A. Musical Notes and Intervals

There are a total of twelve notes in diatonic scale. These notes are the following: C, C♯ or D♭, D, D♯ or D♭, E, F, F♯ or G♭, G, G♯ or A♭, A, A♯ or B♭, and B.

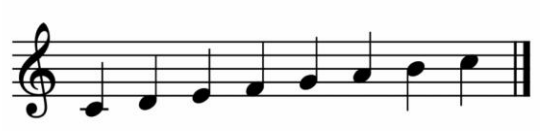

Figure 2.1 The C Major Diatonic Scale

(Source: https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.skoove.com%2Fblog%2Fthe-diatonic-scale%2F&psig=AOvVaw2lA2mM-fRDO89aCMSYrxKt&ust=1684769022710000&source=images&cd=vfe&ved=0CBEQjRxqFwoTCPD3nIbchv8CFQAAAAAdAAAAABAE)

The individual notes can be stacked together, forming a chord, or be put in a sequence of notes, forming a melody or a melodic line. The chord and the melody can be seen as a relationship between the notes.

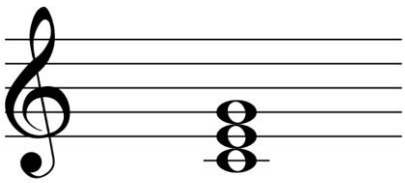

Figure 2.2 C, E, and G stacked together forming a C Major Triad chord

(Source: https://www.musictheory.net/lessons/40)

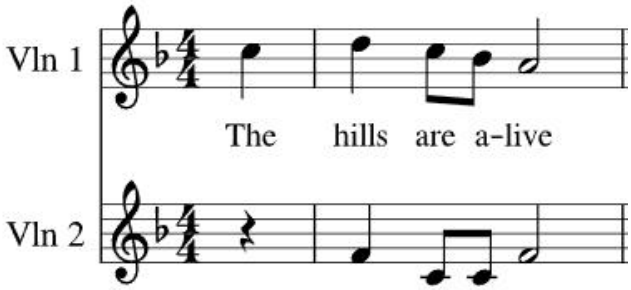

Figure 2.3 Excerpt from The Sound of Music. The notes in the upper bar (Vln 1) form a melody or a melodic line

(Source: https://musescore.com/user/1089156/scores/5383655)

In defining the relationship between the notes, it is useful to define the smallest relationship between them. The smallest relation possible is between two notes, called an interval. Intervals can be counted using half-steps. The half-step between the notes is the following: C-C♯, C♯-D, D-D♯, D♯-E, E-F, F-F♯, F♯-G, G-G♯, G♯-A, A-A♯, A♯-B, and B-C. Note that this is a two-way relationship. For example, the E-F means that E to F is a half-step, and F to E is a half-step as well. Finally, we can define the intervals as follows:

Interval	Half-Steps
Unison	0
Minor Second	1
Major Second	2
Minor Third	3
Major Third	4
Perfect Fourth	5
Augmented Fourth	6
Diminished Fifth	6
Perfect Fifth	7
Minor Sixth	8
Major Sixth	9
Minor Seventh	10
Major Seventh	11
Octave	12

Table 2.1 Intervals

From Octave, it is possible to have farther intervals, such as the Major Ninth. However, the concept is the same, adding half-step each interval. Notice that the intervals can be grouped into a Unison, Second (2^{nd}), Third (3^{rd}), Fourth (4^{th}), Fifth (5^{th}), Sixth(6^{th}), Seventh (7^{th}), an Octave (8ve), Ninth (9^{th}), Tenth (10^{th}), and so on. Intervals can also be grouped according to their quality: Consonant and Dissonant. Consonance happens when two or more notes complement each other such that it is "pleasing" to the ears, while dissonance happens when these notes "collide" with each other thus producing a "less pleasing" sound. Although dissonance is associated with "bad sounding" harmony, it is used quite a lot in many works of music. Consonant and Dissonant intervals can be further grouped into Perfect Consonances, Imperfect Consonances, and Dissonances.

Perfect Consonances	Imperfect Consonances	Dissonances
Unison, 5^{th}, 8ve	3^{rd}, 6^{th}, 10^{th}	2^{nd}, 4^{th}, 7^{th}

Table 2.2 Consonant and Dissonant intervals

Finally, a harmonic interval is an interval between two stacked together notes, while a melodic interval is an interval between a succession of two notes, hence it is called melodic.

B. Melodic Motions and Voice Leading

A melodic motion is a succession of two notes in a melodic line in reference to another melodic line. In this case, we are only interested in two melodic lines. There are three types of melodic motion: Direct Motion, Contrary Motion, and Oblique Motion. A direct motion happens when a succession of two notes in different melodic lines have the same direction, while a contrary motion happens when a succession of two notes in different melodic lines have different directions. An oblique

motion is when a succession of two notes in a melodic line does not change direction, while succession of two notes in the other melodic line changes direction.

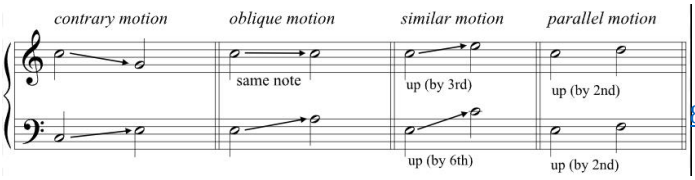

Figure 2.4 Melodic motions. Similar motion and parallel motion are direct motions.

(Source: https://musictheory.pugetsound.edu/mt21c/TypesOfMotion.html)

A voice leading is the linear progression of melodic lines and their interaction with each other. A voice leading contains many melodic motions. A melodic line can also be called a voice, hence the name voice leading. The upper bar melodic line (Vln 1) in Figure 2.3 actually interacts with the lower bar melodic line (Vln 2). The upper bar melodic line can be seen as the voice leading for the lower bar melodic line and vice versa.

C. Counterpoint

Counterpoint is a compositional technique of writing two or more independent melodic lines but and such that they complement each other, forming a coherent harmony or sound. Counterpoint is often encountered in music and is encountered quite a lot in western classical music, specifically the Fugue form.

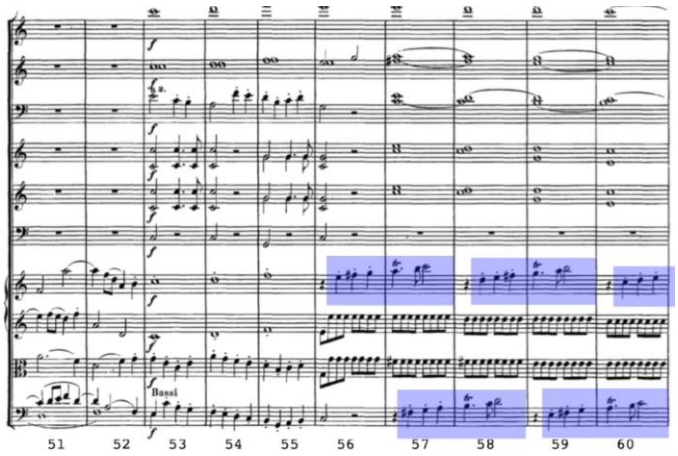

Figure 2.5 A contrapuntal passage (marked with light purple) from the Exposition section in the Finale of Mozart's Jupiter Symphony

(Source: https://www.youtube.com/watch?v=YTxYykhQZbI&t=257s)

D. Cantus Firmus

A cantus firmus is a given reference melodic line in which students will perform an exercise by writing the voice leading for it. The voice leading must adhere to the rules of the species-counterpoints.

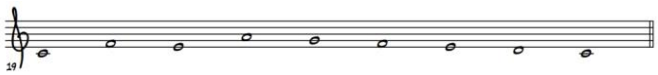

Figure 2.6 A Cantus Firmus given by Antonio Salieri to his student Franz Schubert

(Source: https://48a396c9-e039-4167-9469-820fc5572658.filesusr.com/ugd/cf79fa_272174fb4d9f4e24aa9ce71afc6dcb91.pdf)

E. First-Species Counterpoint

The First-Species Counterpoint is the least complex of the five species counterpoints. Species-counterpoint is a step-by-step guideline and exercise to writing a contrapuntal passage. Each species-counterpoint has rules that become increasingly complex from the first to the last species-counterpoint.

The First-Species Counterpoint is a note-by-note writing, which means for every note in the Cantus Firmus, there is exactly one note that corresponds to it in the other melodic line. The First-Species Counterpoint mentions best practices: stepwise motions are preferred in order to produce smooth voice leading, contrary motion is best, direct motion is less preferred. But the compulsory rules are:

1. No Dissonant Harmonic Intervals
2. No Direct Motion to a Perfect Consonance

For simplicity's sake, we will ignore the best practices and only pay attention to the rules.

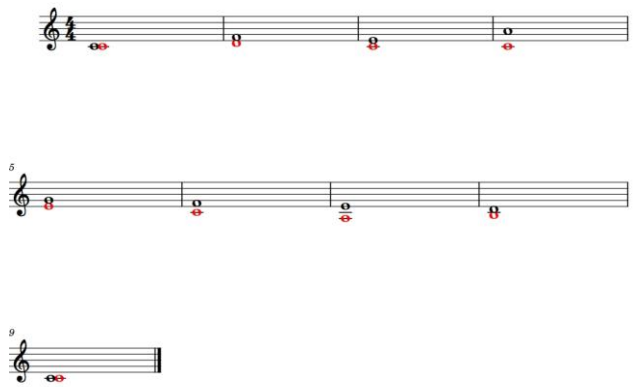

Figure 2.7 One of the voice leading solutions (in red) given by Schubert to the Cantus Firmus by Antonio Salieri

(Source: Author)

F. Graph, Circuit, and Tree

A graph is a mathematical structure used to model pairwise relationships between models. A graph has vertices or nodes, and edges, which can be written as:

$$G = (V, E)$$

G = Graph
V = Vertices/Nodes
E = Edges

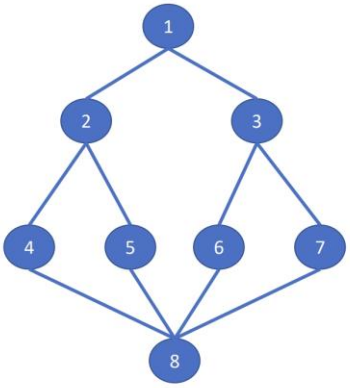

Figure 2.8 A graph with eight nodes and ten vertices

(Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf)

A path is the possible sequence of nodes from one node to another node through the adjacent edges, and a circuit is a path from a node and back to it. For example, 3 – 7 – 8 – 6 - 3 is a circuit. A tree is simply a graph without a circuit. Nodes in a tree may or may not have a parent or children, or both. If a node has no parent, it is called the root node. And if a node has a parent but no child, then it is a leaf node. A tree has levels, in which the root node is at level 0, and its children are at level 1, and so on. Nodes with the same parent are called siblings. A dynamic tree is a tree generated by traversal algorithms such as Depth-First Search algorithm.

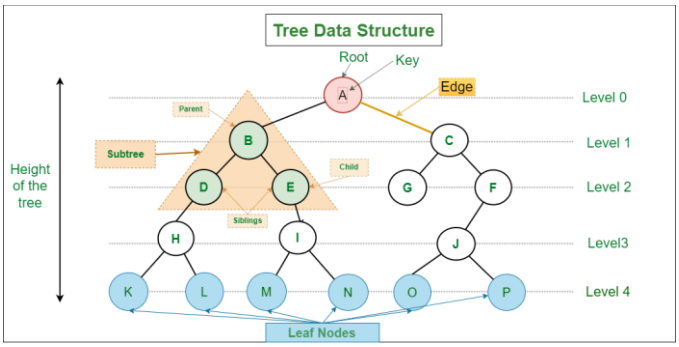

Figure 2.9 A tree and its properties

(Source: https://www.geeksforgeeks.org/introduction-to-tree-data-structure-and-algorithm-tutorials/)

G. Depth-First Search

Depth-First Search (DFS) is a graph traversal algorithm which starts at a designated node and explores as far as possible along each edge before backtracking. The algorithm is as follows:

1. Visit the starting node v.
2. Visit a neighboring node w (nodes that connected to v through an edge). The neighboring node w must be unvisited.
3. Repeat the second step until it reaches a node u where all of its neighboring nodes are already visited.
4. Backtrack to the last node with unvisited neighboring nodes.
5. DFS ends if there are no reachable nodes from the starting node v left unvisited.

One example of a DFS traversal of the graph shown in Figure 2.8 is: 1 – 2 – 4 – 8 – 7 – 3 – 6 – 3 (backtrack) – 7 (backtrack) – 8 (backtrack) – 5.

H. Backtracking

Backtracking is simply going back to the previous node in a graph traversal. It is used when there are no nodes left to visit in the case of DFS, or when a certain condition is met or not met. Backtracking when a certain condition is met or not met is called pruning or cut-off. In our case, the backtrack is a phase in the DFS algorithm.

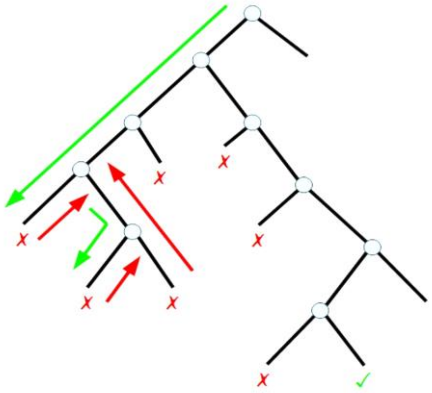

Figure 2.10 Backtracking example. The red cross indicates a cut-off or pruning

(Source: https://www.w3.org/2011/Talks/01-14-steven-phenotype/)

III. Implementaion of DFS and Backtracking to Generate Voice Leadings in First-Species Counterpoint

The implementation of DFS and Backtracking to generate all of the possible voice leadings in First-Species Counterpoint will be coded in Python. The program will receive a Cantus Firmus in the form of a list and using a set of notes in certain range, which will be explained. The possible voice leadings, in the form of a list for each voice leading, will then be outputted to a text file.

A. Data Structures

The data structure in this implementation concerns the musical notes and their relationship with each other, or the intervals. The notes, of course, are related to all the other notes. That is, if we model the graph of note relationships in an octave:

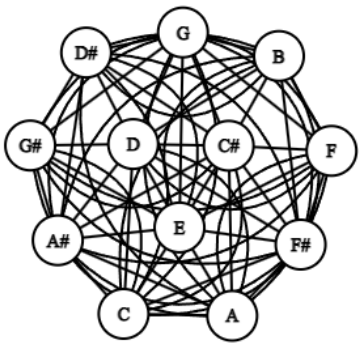

Figure 3.1 Graph of note relationships in an octave

(Source: Author)

Of course, a voice leading cannot be limited to one octave. And we can do more than one octave. But due to the computation cost, we will only use an octave with additional notes just below and just above our octave. We will use the International Organization for Standardization system in naming the note in each octave. The notes used in this implementation are: A3, A♯3, B3, C4, C♯4, D4, D♯4, E4, F4, F♯4, G4, G♯4, A4, A♯4, B4, C5, C♯5, D5, D♯5, E5. For each note, we will assign an integer as shown below:

noteDictionary.py

```
notes = {
    -2 : "A3",
    -1 : "A#3",
    0 : "B3",
    1 : "C4",
    2 : "C#4",
    3 : "D4",
    4 : "D#4",
    5 : "E4",
    6 : "F4",
    7 : "F#4",
    8 : "G4",
    9 : "G#4",
    10 : "A4",
    11 : "A#4",
    12 : "B4",
    13 : "C5",
    14 : "C#5",
    15 : "D5",
    16 : "D#5",
    17 : "E5"
}
```

Table 3.1 The Hash data structure to represent the notes

In the context of the DFS and Backtracking algorithm, the notes can be structured as a dynamic tree. Because we want to find all of the possible voice leadings, the traversal will not stop until all possible solutions path are found in the tree. The path in the tree represents the possible voice leading. The tree generated during the traversal may look like:

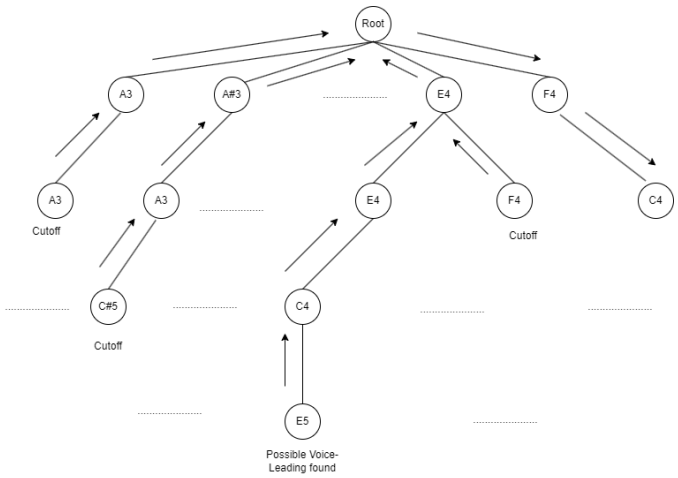

Figure 3.2 An illustration of the tree generated by the DFS and Backtracking

(Source: Author)

We will structure each node, or note in this case:

note.py

```
class Note:
    def __init__(self, noteType):
        self.noteType = noteType
        self.parent = None

    def setParent(self, parent):
        self.parent = parent
```

Table 3.2 The Note class

The Cantus Firmus will be modeled using a list:

algo.py

```
cantusFirmus = [1, 6, 5, 10]
```

Table 3.3 The Cantus Firmus using a list. The elements 1, 6, 5, and 10 represents C4, F4, E4, and A4

Each note in each level except for the level 0 (the root note, which can be any note), will correspond and be compared to the Cantus Firmus at index [level – 1]. Note that in this implementation, we will use the Cantus Firmus given by Antonio Salieri in Figure 2.6. We limit the Cantus Firmus to the first four as well, so it does not become too computationally expensive. Of course, we can do the full Cantus Firmus if we wish to.

B. Algorithm

The algorithm in the program uses standard DFS implementation with Backtracking and pruning using recursion.

algo.py

```
from note import Note
from noteDictionary import notes

upperNoteNumber = 17
lowerNoteNumber = -2

cantusFirmus = [1, 6, 5, 10]

result = [[]]
resultCounter = 0


def dfs(cantusFirmus, parentNote, childNote, level):
    # parentNote is an instance of Note, childNote is an integer

    currentNote = Note(childNote)
    currentNote.setParent(parentNote)
    # You, 18 hours ago • fix: Add pruning constraints
    interval = abs(currentNote.noteType - cantusFirmus[level]) % 12 # This calculates the half-steps of the current harmonic interval
    if(interval == 1 or interval == 2 or interval == 5 or interval == 6 or interval == 10 or interval == 11): # No Dissonant Harmonic Intervals
        return  # Cutoff

    steps = currentNote.noteType - parentNote.noteType  # This calculates the half-steps of the melodic interval between current and previous note
    if(steps == 0 and level > 0):
        if(interval == 0 or interval == 6 or interval == 7): # No Direct Motion to a Perfect Consonance
            return  # Cutoff

    if(level == len(cantusFirmus) - 1):
        constructCounterpoint(currentNote)
    else:
        level += 1

        for i in range(lowerNoteNumber, upperNoteNumber + 1):   # Iteration of neighbors
            dfs(cantusFirmus, currentNote, i, level)
```

```
def constructCounterpoint(note):
    global result
    global resultCounter

    while note.parent is not None:
        result[resultCounter].append(notes[note.noteType])
        note = note.parent

    result[resultCounter].reverse()
    result.append([])
    resultCounter += 1


def printResult():
    file_path = "output/result.txt"
    file = open(file_path, 'w')

    file.write("There are/is " + str(len(result)) + " voice leading(s) possible\n")
    file.write("Possible Voice Leadings:\n")
    for i in range(len(result)):
        file.write(str(result[i]) + "\n")

    file.close()


def main():
    rootNote = Note(1)
    for i in range(lowerNoteNumber, upperNoteNumber + 1):
        dfs(cantusFirmus, rootNote, i, 0)
    result.pop()
    printResult()


main()
```

Table 3.4 The implementation of DFS and Backtracking

IV. Result

The result are as follows:

No.	Voice Leadings
1.	['A3', 'A3', 'C4', 'A3']
2.	['A3', 'A3', 'C4', 'C4']
3.	['A3', 'A3', 'C4', 'C#4']
4.	['A3', 'A3', 'C4', 'D4']
…	…
1267	['C4', 'D4', 'C4', 'C4']
1268	['C4', 'D4', 'C4', 'C#4']
1269	['C4', 'D4', 'C4', 'D4']
1270	['C4', 'D4', 'C4', 'F4']
…	…
9591	['E5', 'D5', 'C5', 'C5']
9592	['E5', 'D5', 'C5', 'C#5']
9593	['E5', 'D5', 'C5', 'E5']
…	…
9609	['E5', 'D5', 'E5', 'F#4']
9610	['E5', 'D5', 'E5', 'A4']
9611	['E5', 'D5', 'E5', 'C5']
9612	['E5', 'D5', 'E5', 'C#5']

Table 4.1 All possible first four voice leadings in the given Cantus Firmus by Antonio Salieri

We can see from Table 4.1, that one of the possible voice leadings, the 1267th voice leading, is the same as the first four voice leading notes given by Schubert in Figure 2.7. This confirms that Schubert correctly wrote it although probably not the most optimal way. Let's see the 9591st voice leading generated:

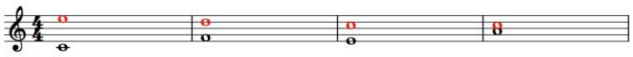

Figure 4.1 The 9591st voice leading generated (the red notes)

(Source: Author)

By analyzing the melodic motions, there are no problems whatsoever. The motions are as follows: Contrary motion from the 1st to the 2nd bar, Direct motion from the 2nd to the 3rd bar with imperfect consonance (Minor Sixth), and finally an oblique motion from the 3rd bar to the 4th bar. The harmonic intervals are as follows: Major Eleventh in the 1st bar, Major Sixth in the 2nd bar, Minor Sixth in the 3rd bar, and finally Minor Third in the 4thbar, all imperfect consonance.

For further experimenting, the code can be accessed via the author GitHub repository:

https://github.com/rayhanp1402/MakalahStima_13521112

V. Conclusion

Sometimes algorithms can be used unexpectedly in some problems, such as DFS and Backtracking in the generation of voice leading in contrapuntal music. While this implementation limits some aspect of counterpoint writing, it can be expanded easily to include all aspects of counterpoint writing, which can be extremely powerful. Although it is much more computationally efficient than Brute Force algorithm by the usage of pruning and Backtracking, it is still computationally expensive to use for large passages of counterpoint. It is definitely not the best algorithm for voice leading generation in contrapuntal writing. The more efficient algorithm would be the Genetic Algorithm, a method that can be used for solving constrained optimization problems based on biological natural selection.

VI. Acknowledgement

I would like to begin by acknowledging Allah SWT., who has provided me with guidance, the will to learn, knowledge, and determination to go through the discrete math class in this third semester and complete this paper. I express my deep appreciation to Dr. Nur Ulfa Maulidevi. who provided her students with the necessary knowledge in algorithm, pique her students' interest in algorithm, and who has trained her students enough to increase our problem-solving skills, particularly in algorithmic related problems which is essential in the completion of this paper. And last but not least, I would like to express my gratitude to the mathematicians and computer scientists before us who have advanced the field of algorithm and the analysis of algorithm to where it is now. Computational algorithms have provided us with tools that makes numerous problems possible to solve, especially in the computational fields.

References

[1] Schoenberg, Arnold, Theory of Harmony, 100TH Anniversary ed, University of California Press, 2010

[2] Fux, Johann, The Study of Counterpoint: From Johann Joseph Fux's Gradus Ad Parnassum, Revised ed, W. W. Norton & Company, 1965

[3] Rosen, Kenneth H., Discrete Mathematics and Application to Computer Science, 8th Edition, Mc Graw-Hill, Inc, 2018

[4] Munir, Rinaldi. 2021. "Breadth / Depth First Search (BFS/DFS) (Bagian1)", https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf. Accessed 21 May 2023, 12.28 GMT+7.

[5] Munir, Rinaldi. 2021. "Algoritma Runut-balik (Backtracking) (Bagian1)", https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf. Accessed 21 May 2023, 12.51 GMT+7.

[6] musictheory.net, https://www.musictheory.net/lessons/40. Accessed 21 May 2023, 21.23 GMT+7.

[7] Atkinson, Richard. 2017. "Magnificent Counterpoint in the Finale of Mozart's Jupiter Symphony. https://www.youtube.com/watch?v=YTxYykhQZbI&t=257s. Accessed 22 May 2023, 10.32 GMT+7

[8] Gran, Jacob. 2020. "How to Compose 1:1 Counterpoint || Tonal Voice Leading 1". https://www.youtube.com/watch?v=b5PoTBOj7Xc&t=750s. Accessed 21 May 2023, 09.02 GMT+7

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Mei 2023

Rayhan Hanif Maulana Pradana dan 13521112