Generating Mean Jazz Music Chord with Greedy Algorithm

Rayza Mahendra G.H / 13517073
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517073@std.stei.itb.ac.id
rayzamgh@gmail.com

Abstract*— Music is one of the oldest forms of art. The basic building blocks of music is the creation and composition of melodies in certain succession to create a tone that sounds and feels good to the human ear. Though the compositional preferences themselves vary from person to person, a very particular genre of music called jazz stands out in this matter. The writer has selected jazz as his subject of experimentation due to its spontant nature, hence there are very few jazz music that follows exact chord patterns. With this information in mind this article will show the use of greedy algorithm to determine the average form of jazz music through selective chord pattern generator.

Keywords : jazz ; greedy ; chord ; pattern

I. Introduction

Music is defined as a selected repetition in the form of sounds that creates patterns, songs, and harmony to the human ear. These harmonic patterns are achieved through vibration generated from musical instrument or the vocal chord of a singer. Music as one of the oldest form of art, estimated to have originated from 50000 years ago, has come a very long way from strictly being played on acoustic rocks banging on to each other and flutes made from carved bones of large mammals into the modern form of electronically generated songs we are familiar with in this day of age. Though the pitch and general feeling of music from those years ago sounds different then what we're used to today, the notes played on those very rocks are fundamentally composed from the same vibration of air that modern electrically generated music uses.

Vibrations generates pitches and audios that are audible in the range of 20Hz - 20000Hz to the human ear. Most of the sound and vibration between this interval will sound gibberish and unsettling at times, but certain frequencies do create harmonies akin to that of a melody.

A melody (from Greek μελῳδία, melōidía, "singing, chanting"), also tune, voice, or line, is a linear succession of musical tones that the listener perceives as a single entity[1]. These melodies can then be further composed with slight variations in tempos, pitches, progression to create the many genres of musical compositions such as pop, metal, rock, ragtime, and including but not limited to jazz.

Modern day musics typically rocks and pop songs uses near identical composition of chord. Though the melodies may differ significantly, the sound and general feel of most of these music can be classified to the "popular progression" which is a chord progression that dominates the music industry today. this repetitive progression of chord does not hold true to jazz music.

Jazz believed to be related to "jasm", a slang term dating back to 1860 meaning "pep, energy"[6] was a very popular music during the late 19th to 20th centuries in the African-American communities in New Orleans, United States. At the time it was seen as the american take on the classical music genre brought forth by their ancestor from the european continent in the late 17th century. Jazz quickly rose into popularity in the late 1900s as the genre was often played during happy hours in night clubs and bars around the United States.

Jazz often involves improvisation in their play as to not bore the audiences during the performance. These improvisations follows a sets of rules which generally translates into musical chords we are familiar with.

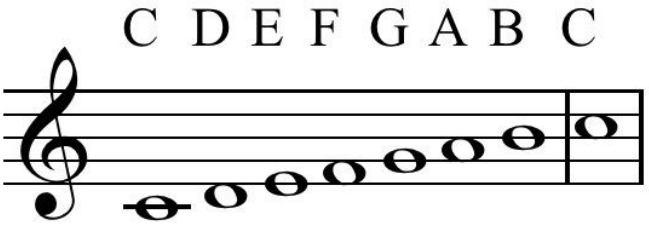

Image 1 "Standard Octave scale on a G Clave"

source : https://en.wikipedia.org/wiki/Jazz

Because of this sets of rules the writer was then able to extract chord data from various jazz musics and generate a form of "The Average" of jazz music with the use of Greedy Algorithm.

II. Greedy Algorithm

A. General Concept.

Greedy algorithm is an algorithmic method that has the ultimate goal of optimising an outcome through also optimising a locally made decision from a decision tree [3]. though its use may be very popular, it often does not lead to the optimal result due to there being certain variables that ultimately affects the outcome in unforseen way from the perspective of the algorithm.

Nevertheless its use can be seen with problems not requiring much thought and problems which are very specific regarding its desired return value.

The focus of greedy algorithm is to find the best yield result, regardless of minimization or maximization.

Greedy algorithm has five typical components:

1. A candidate set, from which a solution is created
2. A selection function, which chooses the best candidate to be added to the solution
3. A feasibility function, that is used to determine if a candidate can be used to contribute to a solution
4. An objective function, which assigns a value to a solution, or a partial solution, and
5. A solution function, which will indicate when we have discovered a complete solution [5]

One of its most popular use is with integer knapsack problem example.

B. Example Problem.

Suppose there are six items that are available in a certain house. Each of those items has weight and value in money assigned to them. Your job is to find the maximum value that you can carry in one trip given that you can only carry so much weight.

maximumWeight = 10

id	Weight	Value
1	1Kg	4
2	3Kg	9
3	2Kg	8
4	5Kg	14
5	2.5Kg	2
6	1Kg	3

There are three ways to approach this problem with greedy algorithm:

1. Weight Minimization
2. Value Maximization
3. Density Maximization

These three approach also has different greedy components mainly in the selection function. Approach number one has the selection function of choosing the lightest available item in the house. This will generate this result:

chosen item	Weight	Value
1	1Kg	4
6	1Kg	3
3	2Kg	8
2	3Kg	9
5	2.5Kg	2
-	total = 9.5Kg	total = 26

While approach two and three chooses the largest value and density value by weight per item and generates these results:

chosen item	Weight	Value
4	5Kg	14
2	3Kg	9
3	2Kg	8
-	total = 10Kg	total = 31

chosen item	Value/Weight	Weight	Value
1	4Value/Kg	1Kg	4
3	4Value/Kg	2Kg	8
2	3Value/Kg	3Kg	9
6	3Value/Kg	1Kg	3
5	0.8Value/Kg	2.5Kg	2
-	-	Total =9.5Kg	total = 26

From these tables of solutions we can infer that the optimal sets of item to carry is given by table number two.

As the reader can see, different sets of value for components of a greedy algorithm, though implemented to the same problem may yield different result or identical result. As such a solution implementing the concept of greedy algorithm is heavily influenced by the components on which it was constructed from.

III. Generating Jazz Music with Greedy

A. *Extracting Chord Data*

In order to generate the music, a large amount of data is required. This data is in the form of hundreds of jazz music chord extracted from websites, [4] which provides free musical data in the form of midi files and sheet musics.

Data in the form sheet musics are then extracted manually while data from midi files are extracted with the help of python using pymidi.

These data is extracted in the form of .txt files which contains chord progression data of a song per line of text:

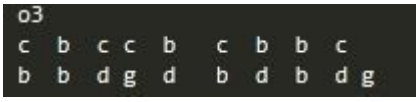

Image 2 "Chord data from Debussy_quartet"

source : MyPc

B. *The Algorithm*

To generate the music, the program implements an algorithm similar to that of the integer knapsack problem. the program defines the maximum number of a value of a chord that a music can have. These values of notes are determined from the extracted data with this rule:

for each instance of chord (ex : C F# E), initiate a value of chords that counts the number of chord of the same type that follows the origin chord, this sentence is best explained by an example such as Image 2:

ignoring o3 as definer for octave number 3

for every unique character in Image 2, assign a list for every chord that follows it, if the list already exists then, increment the value of it.

for example take the valueofChord[‘c’]

valueofallChord is a dictionary that contains :

{valueofChord[‘c’], valueofChord[‘b’], valueofChord[‘g’], etc..}

valueofChord[‘c’] itself is a list of list that contains the number of instance of chords that follows it, it looks like this:

valueofChord[‘c’] = [[‘b’ , 4] , [‘c’ , 1]]

this means that there are 4 instance of chord C followed by chord B in the data, and there are 1 instance of chord C followed by chord C in the data as well.

the full data extracted from **image 2** would contain the following:

valueofChord[‘c’] = [[‘b’ , 4] , [‘c’ , 1]]

valueofChord[‘d’] = [[‘g’ , 2] , [‘b’ , 1]]

valueofChord[‘b’] = [[‘c’ , 3] , [‘b’ , 2] , [‘d’ , 3]]

valueofChord[‘g’] = [[‘d’ , 1]]

this essentially give value for every single combination of 2 chords that exists in image2 that will the be relevant to the next step.

C. *Assigning Each Component to The Greedy Algorithm*

As mentioned in the passage before, a greedy algorithm requires five total input as its parameter components. In order to extract "Mean" data of jazz chords each component has been assigned the following value:

1. Candidate set, the set of chords that represents a fully generated jazz music.
2. Selection function, chose the key with the maximum value in value of chord, with regards to the key previously chosen by the algorithm as the dictionary key to the list to search in.
3. Feasibility function, checks if the number of chord in the candidate set falls below the maximum amount of allowed chord.
4. Objective function, the sum of all the chord in a candidate set.
5. Solution function, if the sum of all the chord returns the maximum number of chords allowed.

With this components in place the only parameters needed to generate a music would be the first chord of the music and the allowed number of chord per music.

D. Problems With Applicating the Program

As the reader might have surmised there are quite a few number of problems that may occur from this algorithm. This portion of the paper will now explain the already found problems during the experimentation phase and the workaround that the writer has implemented as a complement to the pure greedy algorithm.

1. Looping chord:

   if the example in point III.B were run as is, it would return a chord as follows = [C, B, C B, C, B….]

   even though this will not result in an error in the program. This will however create a tedious and boring music, therefore to avoid this problem the writer has implemented a function to decrease the value in valueofChord[x] by 1 if the key was taken as an element in the candidate set. This however raises the next error.

2. Empty chord:

   if not enough chord data was given to the program then the algorithm would stop as is. Regarding this error the writer does not implement a solution, taking into mind that the program was fed over hundreds of music samples.

E. Example

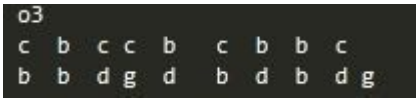

Image 2

given the data taken from image2 and the valueofallChord as follows:

valueofChord[‘c’] = [[‘b’ , 4] , [‘c’ , 1]]

valueofChord[‘d’] = [[‘g’ , 2] , [‘b’ , 1]]

valueofChord[‘b’] = [[‘c’ , 3] , [‘b’ , 2] , [‘d’ , 3]]

valueofChord[‘g’] = [[‘d’ , 1]]

with paramater maximumchord = 6, and initialchord = ‘c’ we can generate the music following this step by step example:

CS = CandidateSet = [c]

currentChord = ‘c’

maxkey

maxchord = 6

step 1

iterate valueofChord[‘c’], found maxkey = b

CS = [c, b]

currentChord = ‘b’

valueofChord[‘c’] = [[‘b’ , (4-1)] , [‘c’ , 1]]

//SolutionFunction

CS.len == maxchord returns false

step 2

iterate valueofChord[‘b’], found maxkey = c

CS = [c, b, c]

valueofChord[‘b’] = [[‘c’ , (3-1)] , [‘b’ , 2] , [‘d’ , 3]]

currentChord = ‘c’

//SolutionFunction

CS.len == maxchord returns false

step 3

iterate valueofChord[‘c’], found maxkey = b

CS = [c, b, c , b]

currentChord = ‘b’

valueofChord[‘c’] = [[‘b’ , (3-1)] , [‘c’ , 1]]

//SolutionFunction

CS.len == maxchord returns false

step 4

iterate valueofChord[‘b’], found maxkey = d

CS = [c, b, c , b, d]

currentChord = ‘d’

valueofChord[‘b’] = [[‘c’ , 2] , [‘b’ , 2] , [‘d’ , (3-1)]]

//SolutionFunction

CS.len == maxchord returns false

step 5

iterate valueofChord[‘d’], found maxkey = g

CS = [c, b, c , b, d, g]

currentChord = ‘g’

//SolutionFunction

CS.len == maxchord returns true

solution = CS = [c, b, c , b, d, g]

a new set of chords has been generated from the algorithm as CS = [c, b, c, b, d, g]

these list of new chords can then be parsed into an alda file and run as it was a normal midi file.

V. Analysis

Below are snippets of txt files generated by the above program:

Image 4 "Generated Sample 1"

Image 4 was an output example of running the program with parameters, initialchord = 'g', and maximumChord = 300

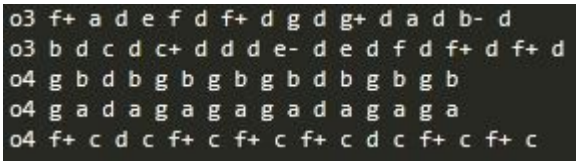

Image 5 "Generated Sample 2"

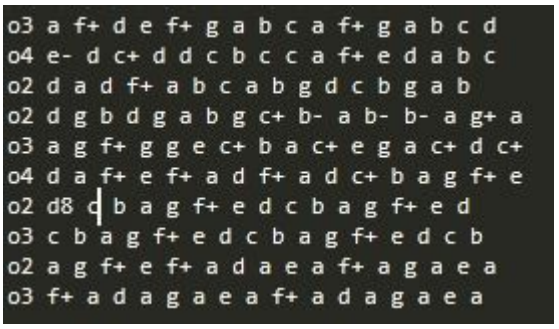

Image 6 "Generated Sample 2"

Image 5 was an output example of running the program with parameters, initialchord = 'f#', and maximumChord = 80

Image 6 was an output example of running the program with parameters, initialchord = 'a', and maximumChord = 150

the generated text has been tested using audible means and the result was to be expected. While testing was possible, the writer was unable to get objective opinions regarding the produced result.

VI. Conclusion

While it has been shown that generating new sets of chords is possible using the greedy algorithm, the true nature of jazz music itself was as an improv play. A procedurally generated copy can only mimic creativity to an extent. As such even though it is possible to generate the music, the result left much more to be desired.

References

[1] Berg, P (1996). "Abstracting the future: The Search for Musical Constructs". *Computer Music Journal*. MIT Press. **20** (3): 24–27 [11]. doi:10.2307/3680818. JSTOR 3680818.

[2] ^ Baofu, Peter (3 January 2013). *The Future of Post-Human Performing Arts: A Preface to a New Theory of the Body and its Presence*. Cambridge Scholars Publishing. ISBN 9781443844857.

[3] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/Algoritma-Greedy-(2019).pdf

[4] https://freemidi.org/genre-jazz

[5] Hazewinkel, Michiel, ed. (2001) [1994], "Greedy algorithm", Encyclopedia of Mathematics, Springer Science+Business Media B.V. / Kluwer Academic Publishers, ISBN 978-1-55608-010-4

[6] Wilton, Dave (6 April 2015). "The Baseball Origin of 'Jazz'". OxfordDictionaries.com. Oxford University Press. Retrieved 20 June2016.

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, 29 April 2012

Rayza Mahendra, 13517073