Music composition using CSP and RL algorithms

Fidaa Naffaa
The Rachel and Selim Benin School of Computer Science and Engineering
Hebrew University
Jerusalem, Israel
fidaa.naffaa@mail.huji.ac.il

Chiho Song
The Rachel and Selim Benin School of Computer Science and Engineering
Hebrew University
Jerusalem, Israel
chiho.song@mail.huji.ac.il

Ariel Zaken
The Rachel and Selim Benin School of Computer Science and Engineering
Hebrew Univerisy
Jerusalem, Israel
ariel.zaken@mail.huji.ac.il

***Abstract*—Our AI has a job to improvise the music. We considered using the Constraints Satisfaction Problem algorithm and Reinforcement Learning algorithm. For Constraints Satisfaction Problem algorithm, we considered one note as one node. If CSP agent assigned the node of note, then adjacent nodes can't be filled with same note as we assigned before. For Reinforcement Learning algorithm, RL agent evaluates every 4 bars that AI have assigned. According to the evaluation function, RL agent updates the policy.**

Keywords—AI, Reinforcement Learning (RL), Agent, Policy, Constraints Satisfaction Problem (CSP)

I. Introduction

Music consists of "bar", "bit", "chords", and "notes". According to "Bar" and "Bit", the AI generates a state board. "Notes" and "Chords" are actions. AI improvise the music by choosing the notes(actions). This is the basic structure of our idea. This project will use 2 algorithms such as Constraints Satisfaction Problem algorithm and Reinforcement Learning algorithm.

II. Constrinats Satisfaction Problem

A. Background

Constraint satisfaction problem(CSP) algorithm consists of 3 sets. There are variables, domain of each variable, and constraints. Our CSP algorithm gets the input from user that chords progression (For example, G, C, D, Em). After then, AI makes the board of nodes that each note is a variable. According to the chord, every node has relevant domains. (For example if chord is G, then it has three notes and not playing note such as 'G', 'B', 'D', and 'empty'). And like coloring problem that we learnt in class. We used same constraints. Each note that is closed to each other can't be same note.

B. Implementation

- Node.py: Each node is variable in CSP. It is an object that contains the information of music order and chord.
- Board.py: Board is consisting of node objects. It represents the current state of notes that we choose. If it is not consistent with constraints, AI can't put the nodes on state board.
- CSP_Agent.py: it contains the algorithm of the CSP backtracking. This file manages the CSP.

C. Result

Output of the AI is following (Please listen to our result):

- https://youtu.be/rRY_6ZkWPgc

Output file has good sound because domain of variable only contains relevant notes of chord that is assigned. We conclude that even though, it sounds good and harmony, our AI is not really improvising the music. So, we thought that if we use reinforcement learning, we could get better music than this result.

III. Reinforcement Learning

A. Background

For reinforcement learning, we used a Q-learning agent that is "self-taught", as in, instead of learning from other music pieces, it learns by taking a simple four-chord progression and using a reward function to determine which notes to play on which beat, we chose to implement it by beginning the training run with a very high epsilon, and letting it decay over time until it reaches zero by the end, that is because at the start we want to explore as many states and options as possible, and towards the end we want to write a melody with the states that we established give the highest reward.

We chose Q-learning because we felt it best reflected the process of music composition, there is an element of exploration and randomness to it that can be refined in order to produce the best results, which goes hand in hand with the artistic elements that go into music writing, while at the same time providing the necessary computational power that can solve the search problem that is finding the most suitable thing to play in every moment.

B. Implementation

First off, the environment we used was an abstract music player that deals with music as a collection of symbolic characters and strings and numbers, it moves from state to state by changing the note being played on that state and incrementing the beat, if the beat changes from 4 to 1 then it changes chords as well.

Our learner works as follows:

Given a four-chord progression with a standard 4/4 time signature, we constructed the agent as follows:

- Actions: Every possible combination of the twelve chromatic scale notes A, A#, B, C, C#, D, D#, E, F, F#, G, and an additional "empty" note signified by $ where nothing is played, from a single quarter note up to a group of four sixteenth notes, amounting to around 31000 possible actions.
- States: A state consisted of a chord, beat and a note that is played on that beat, starting with a default "$" signifying that nothing is being played on that beat.
- Reward: The reward function gives a higher reward to chord notes and to runs starting with chord notes, especially on the first beat.

C. *Result*

Here are some of the results of our Q-Learning agent, they all use a chord progression of G,C,D,Em on 120 BPM with a time signature of 4/4:

- Maximum of one note per beat:

  https://youtu.be/BZhrmBrKOao

- Maximum of two notes per beat (**please listen!**):

  https://youtu.be/KSsTSPfDUYU

- Maximum of three notes per beat:

  https://www.youtube.com/watch?v=Nz-995zdxPI

- Maximum of four notes per beat:

  https://www.youtube.com/watch?v=8Ok9aG2hinU

IV. Result Analysis

We have scored 200 wav files for each of the CSP and RL algorithms, we based our evaluation function on the one used in another study[1] which also used Q-Learning to compose music, albeit their algorithm took an existing melody and added a harmony to it, but we found that the same principles they used to evaluate their work applied to ours as well, we scored by the following criteria:

Every piece started out with a score of 100.0, from there we penalized by these two metrics :

1) Harmony : For each note that added dissonance to any of the three chord notes, we subtracted one point divided by the number of notes in that particular beat.

2) Novelty : If the top two most used notes in the 4-bar sequence constituted more than 50% of all used notes, we deducted 20 points, if they constituted more than 40%, we deducted 15 points, if they constituted more than 30%, we deducted 10 points and if they constituted more than 25% then we deducted 5 points.

The third criteria used in the article we referenced was Coherency, we found that it was irrelevant to our program's music since all of our notes were from the same octave.

The figures below contain the score distribution for each algorithm

Comparison:

We can see that RL is much more consistent than CSP and it follows a normal distribution, as opposed to the wild discrepancies we can see with CSP where at certain points it drops to the low 60's in score, that is likely due to our decision to penalize repetition and overusing certain notes, while RL likely suffers from dissonance on occasions because of the tendency to explore.

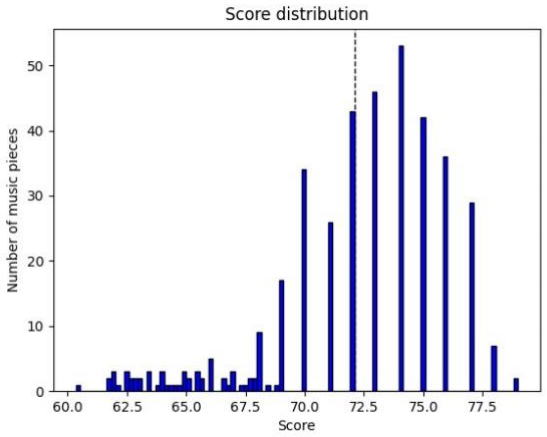

Figure 1- CSP score – mean score: 72.09

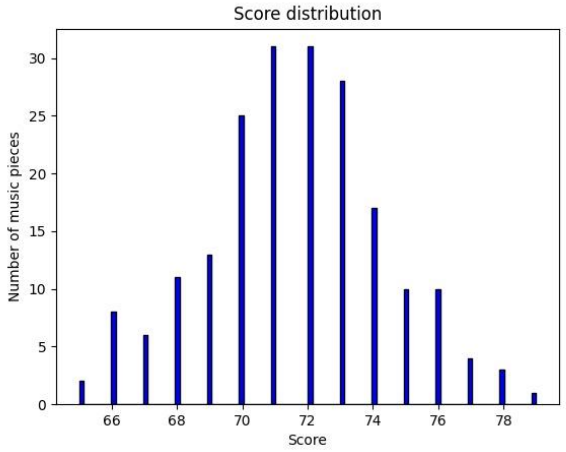

Figure 2 - RL score - mean score: 71.56

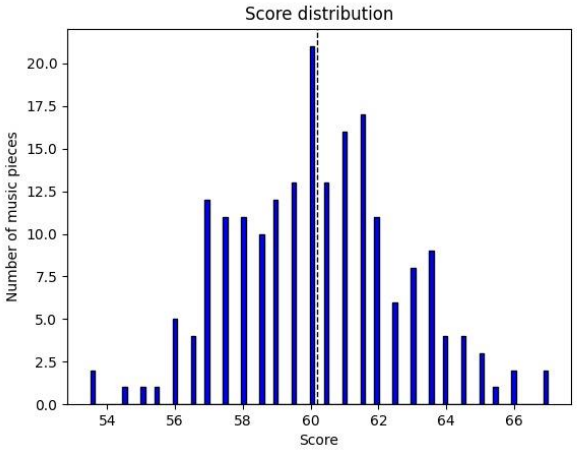

Figure 3 - 10 iteration training

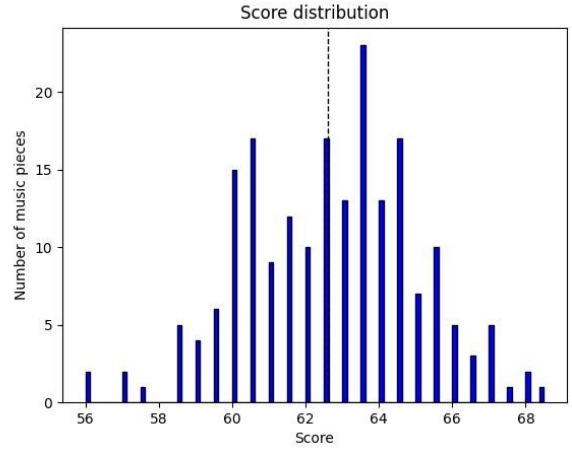

Figure 4 - 100 iteration training

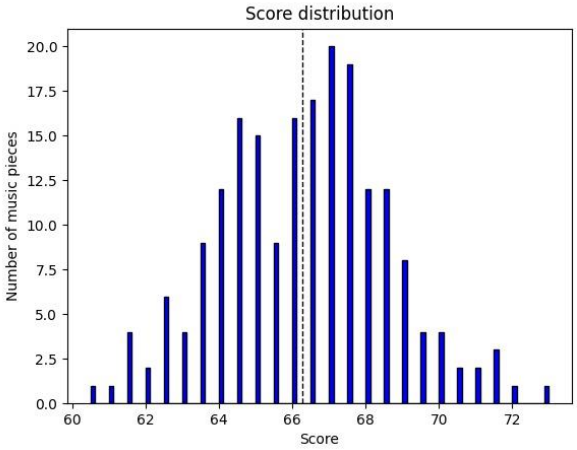

Figure 5 - 1000 iteration training

According to "Figure 3, 4, and 5", we can see that more training affects the score of the music.

V. Conclusions

Although according to our results, CSP slightly outperforms RL when using a maximum of one note per beat, however, we can still conclude that if we can find a reasonable reward function, RL works better to improvise music, that is due to a number of factors:

1- RL is a more natural fit for music composition, since music needs an element of randomness in order to break out of familiar patterns of notes, while with CSP the number of choices at each node is very limited and will lead to predictable music, even if it did indeed adhere to sound compositional practices.

2- With RL it is much easier to generate more dynamic music and to do so deliberately, using a reward function that encourages variety when choosing how many notes to play on the next beat, while with CSP, even if we were to find a domain where we can choose between multiple note subdivisions, the choice is likely to be random and not reflect a deliberate attempt at rewarding dynamism.

3- CSP only contains relevant notes of chords. So, CSP algorithm output sound was better than we expected. But RL contains others notes that can go along with the chord. In other words, RL has much more potential for exploration than CSP. Therefore, RL has a bigger chance of making more varied and interesting sounding melodies.

VI. Criticisms and Limitations

Firstly, our biggest limitation with Q-Learning was finding a natural reward function that takes into account long term compositional value, in our tweaking of the reward function we could not find a payoff that would prevent the algorithm from eventually settling on a looping melody, unless we prevented our epsilon from decaying to 0, which still brought dissonance as the choice of actions in that case was random.

Secondly, also when using Q-Learning, we were limited by the range of octaves we could use, we eventually settled on using only one, that is because the amount of actions when taking into account 16^{th} notes was in the region of 31000, and that is only taking into account one octave, if we were to use more we would have had a very computationally inefficient algorithm, therefore a smarter use of states and actions should have been made, I believe there are ways to integrate more octaves without sacrificing runtime.

Lastly, when we run the program with CSP algorithm, we only had constraints that if the action has same chord notes and it is not repeated with adjacent nodes. There can be more various constraints to set with musical knowledge but we couldn't find more.

Acknowledgment

Special thanks to Jeffrey Rosenschein, Noa Moriel, Reshef Mintz, Roy Friedman, Yoni Sher and the course staff.

References

[1] Mint Lin, Yuchen Sun, and Simon Zhu, "Music Generation Using Q-Learning" (original link)