Generating Piano Compositions Using GA

Projekat iz Računarske inteligencije
Matematički fakultet
Univerzitet u Beogradu

Ksenija Ivanović 135/2019
Dejana Kop 91/2019

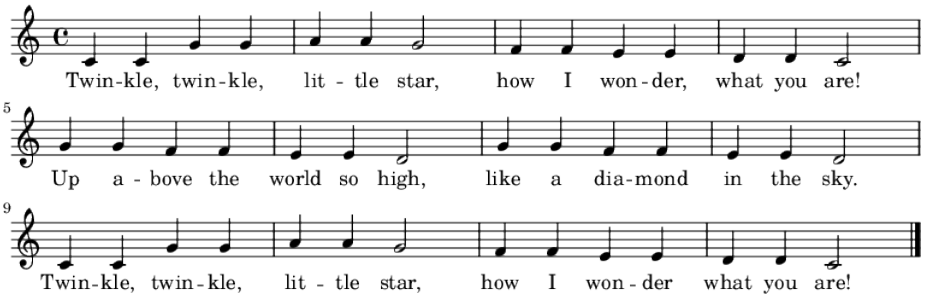
Maj 2024.

Sadržaj

1 Uvod

Cilj projekta je komponovanje kratkih klavirskih kompozicija primenom genetskog algoritma. Projekat se sastoji iz dve celine: generisanja ritma i dodavanja tonova na dobijeni ritam. Dodatna stavka je mogućnost čuvanja dobijenih melodija u tekstualnom formatu i njihovo kasnije učitavanje.

2 Potrebna muzička teorija i kodiranje problema



Dve osnovne osobine note su njeno trajanje i visina.

2.1 Ritam

U našem projektu smo se ograničile na četvoročetvrtinski takt. To znači da je osnovna merna jedinica vremena četvrtina note, a svaki takt može da se podeli na 4 jednaka dela. Tempo nam određuje koliko puta možemo da otkucamo četvrtinu note za 60s. To znači da je trajanje četvrtine note u sekundama

$$quarterNoteDuration = 60s/tempo,$$

odnosno trajanje bilo koje note:

$$noteDuration = 60s/tempo * 4 * noteLength.$$

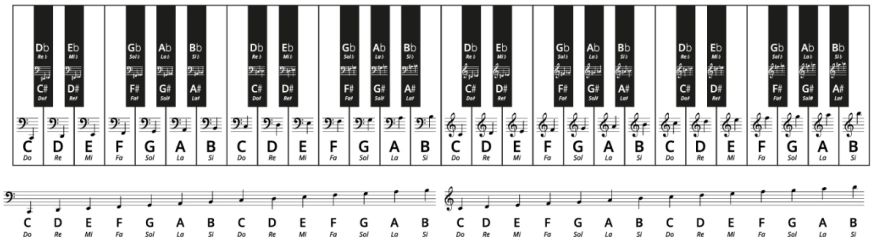
Ritam opisuje odnose u trajanjima uzastopnih nota. Takođe, uključuje i pauze. Za potrebe našeg projekta, izdvajamo dva moguća stanja:

1. 1 - početak nove note
2. 0 - i dalje se pušta prethodno započeta nota

Potrebno je definisati koja je vrednost note određena sa dve uzastopne jedinice. Iako je 1/4 osnova takta 4/4, odlučile smo se da ovo bude 1/8, kako bismo dobile interesantnije rezultate. To znači da nam je jedan takt određen nizom dužine 8.

2.2 Melodija

U današnjoj muzici se najčešće koristi sistem koji se sastoji od 12 tonova koji se mogu naći u više oktava.



Tih 12 tonova se mogu organizovati u lestvice - grupe tonova koje dobro zvuče zajedno. Lestvice su određene:

1. vrstom: u našem slučaju samo dur i mol.
2. početnim tonom - korenom

Za potrebe projekta smo se ograničile na korišćenje tonova iz 4. i 5. oktave. Jedan ton je kodiran u obliku: imeTona + brojOktave (npr. c4, cs4) Jedna nota je kodirana u obliku: [ton, vrednost] (npr. [c4, 3/8]).

3 Genetski algoritam

Odlučile smo se za korišćenje genetskog algoritma, koji je heuristička tehnika inspirisana procesom evolucije u prirodi.

3.1 Generisanje ritma

1. **Reprezentacija jedinki:** Svaka jedinka u populaciji genetskog algoritma predstavlja jedan mogući ritam. Ove jedinke predstavljene su nizom nula i jedinica, gde jedinica označava početak novog tona, a nula označava produžetak trajanja prethodnog tona za još 1/8.

2. **Generisanje početne populacije:** Na početku algoritma se generiše početna populacija jedinki.

   (a) **Ulazni parametri:**

      i. Veličina populacije

      ii. Dužina niza koji se generiše.

      iii. Maksimalni dozvoljeni broj uzastopnih nula u generisanom nizu.

   (b) **Inicijalizacija stringa:** Uvek počinjemo karakterom '1'. Zatim, u petlji se generišu preostali karakteri niza. Svaki karakter se nasumično bira iz skupa '01'. Ako je izabrani karakter '0', broj uzastopnih nula se povećava. Ukoliko broj uzastopnih nula premaši maksimalni dozvoljeni broj uzastopnih nula, karakter se postavlja na '1', čime se obezbeđuje da nema previše uzastopnih nula u nizu. Ako je izabrani karakter '1', broj uzastopnih nula se resetuje na nulu.

   (c) **Krajnji rezultat:** Niz generisanih jedinki.

3. **Ocena jedinki (fitness funkcija):** Korisniku se pušta generisani ritam i on mu dodeljuje ocenu od 1 do 5.

4. **Selekcija:** Odabir jedinki za reprodukciju se vrši na osnovu njihovih ocena. Jedinke koje bolje odgovaraju željenim kriterijumima imaju veću verovatnoću da budu odabrane za reprodukciju.

   (a) **Ulazni parametri:** Funkcija select_parents prima dva ulazna parametra:

      i. population: Lista svih jedinki u trenutnoj populaciji genetskog algoritma.

      ii. fitness_scores: Lista ocena (fitnes vrednosti) za svaku jedinku u populaciji, koja predstavlja koliko dobro svaka jedinka odgovara željenim kriterijumima.

   (b) **Izračunavanje verovatnoća selekcije:** Prvo se računa ukupna suma svih fitnes vrednosti u populaciji. Zatim se za svaku jedinku izračunava verovatnoća selekcije, koja je proporcionalna njenom fitnesu u odnosu na ukupni fitnes cele populacije. Ove verovatnoće se koriste za pravedno biranje roditelja, gde jedinke sa većim fitnesom imaju veću verovatnoću da budu izabrane.

(c) **Biranje roditelja:** Koristi se metoda ruletskog izbora (roulette wheel selection) kako bi se izabrali roditelji. To znači da se svaka jedinka u populaciji bira sa verovatnošću proporcionalnom njenom fitnesu. Funkcija random.choices se koristi za biranje k=2 roditelja iz populacije, pri čemu se koriste verovatnoće selekcije koje su izračunate prethodno.

(d) **Provera roditelja:** Nakon izbora roditelja, proverava se da li su izabrane dve različite jedinke kako bi se izbegla reprodukcija sa samim sobom. U slučaju da su izabrane dve iste jedinke, proces se ponavlja dok se ne izaberu dve različite jedinke.

(e) **Krajnji rezultat:** Na kraju funkcija vraća dve izabrane različite jedinke koje će biti korišćene kao roditelji za reprodukciju.

5. **Reprodukcija:** Odabrane jedinke se koriste za generisanje novih jedinki putem genetskog operatora kao što su ukrštanje (crossover) i mutacija. Ukrštanje kombinuje delove roditeljskih jedinki kako bi se generisala nova jedinka, dok mutacija menja neke delove jedinke na nasumičan način.

   (a) **Mutacija:**

      i. **Ulazni parametri:**

         A. individual: Jedinka koja se mutira, odnosno niz nula i jedinica koji predstavlja ritam.

         B. mutation_rate: Verovatnoća mutacije za svaki bit u jedinki. max_consecutive_zeros: Maksimalni dozvoljeni broj uzastopnih nula u mutiranom nizu.

      ii. **Inicijalizacija mutirane jedinke:** Počinje se sa praznim stringom mutated, koji će sadržati mutiranu jedinku.

      iii. **Mutacija svakog bita:** Za svaki bit u originalnoj jedinci, proverava se da li će biti mutiran. To se radi generisanjem slučajnog broja između 0 i 1, i upoređivanjem sa mutation_rate. Ako je generisani broj manji od mutation_rate, bit će mutiran.

      iv. **Ograničenje uzastopnih nula:** Ako je trenutni bit nula (bit == "0"), proverava se da li broj uzastopnih nula prelazi max_consecutive_zeros. Ako prelazi, trenutni bit se menja u jedinicu ("1") i brojač uzastopnih nula se resetuje na nulu. U suprotnom, brojač se povećava za 1. Ovo ograničenje garantuje da neće biti previše uzastopnih nula u mutiranoj jedinci.

      v. **Generisanje mutiranog bita:** Ako je bit mutiran, menja se u suprotni bit ("1"postaje "0"i obrnuto) i ažurira se brojač uzastopnih nula prema potrebi. U suprotnom, bit ostaje nepromenjen.

      vi. **Resetovanje početnog bita:** Kako bi se obezbedilo da svaki ritam počinje sa jedinicom, prvi bit u mutiranoj jedinci se postavlja na jedinicu.

      vii. **Krajnji rezultat:** Na kraju se vraća mutirana jedinka koja je rezultat ovog procesa.

6. **Evolucija:** Nakon što se generiše nova populacija, proces se ponavlja.

7. **Kriterijum zaustavljanja:** Korisnik u svakoj iteraciji donosi odluku da li želi da se generiše još jedna generacija.

8. **Rezultat algoritma:** Algoritam vraća niz jedinki koje su dobile ocenu 5. One će se koristiti u drugom delu algoritma.

3.2 Generisanje melodije

1. **Odabir lestvice:** Korisnik bira lestvicu koja će biti korišćenja u algoritmu.

2. **Reprezentacija jedinki:** Jedinka predstavlja niz oblika [ton, dužina] (npr. [c4, 1/8]).

3. **Generisanje početne populacije:** Na početku algoritma se generiše početna populacija jedinki.

   (a) **Ulazni parametri:**

      i. beats: Niz ritmova dobijenih u prvom delu.

   (b) **Inicijalizacija stringa:** Nasumično se bira jedan od ponuđenih ritmova, a zatim se svakom elementu tog niz pridružuje jedan nasumično izabran ton iz odabrane lestvice. Za svaki ton se slučajno bira oktava (4. ili 5.).

   (c) **Krajnji rezultat:** Na kraju se vraća generisani niz.

4. **Ocena jedinki (fitness funkcija):** Korisniku se pušta generisana melodija i on mu dodeljuje ocenu od 1 do 5.

5. **Selekcija:** Odabir jedinki za reprodukciju se vrši na osnovu njihovih ocena. Jedinke koje bolje odgovaraju željenim kriterijumima imaju veću verovatnoću da budu odabrane za reprodukciju. Na kraju funkcija vraća dve izabrane različite jedinke koje će biti korišćene kao roditelji za reprodukciju.

6. **Reprodukcija:** Ukrštanje kombinuje delove roditeljskih jedinki kako bi se generisala nova jedinka, dok mutacija menja neke delove jedinke na nasumičan način.

   (a) **Mutacija:**

      i. **Ulazni parametri:** Funkcija mutation prima dva ulazna parametra:

         A. individual: Jedinka koja se mutira.

         B. mutation_prob: Verovatnoća mutacije za svaku notu u jedinki.

      ii. **Mutacija svake note:** Za svaki bit u originalnoj jedinci, proverava se da li će biti mutiran. To se radi generisanjem slučajnog broja između 0 i 1, i upoređivanjem sa mutation_prob. Ako je generisani broj manji od mutation_prob, slučajno se bira novi ton iz lestvice i upisuje umesto trenutnog.

      iii. **Krajnji rezultat:** Na kraju se vraća mutirana jedinka koja je rezultat ovog procesa.

7. **Evolucija:** Nakon što se generiše nova populacija, proces se ponavlja.
8. **Kriterijum zaustavljanja:** Korisnik u svakoj iteraciji donosi odluku da li želi da se generiše još jedna generacija.
9. **Rezultat algoritma:** Algoritam vraća niz od 8 najbolje ocenjenih jedinki. One se spajaju u konačnu melodiju.

4 Snimanje i reprodukcija dobijenih rezultata

Rezultati se zapisuju u .txt fajlove. Za reprodukciju je korišćena biblioteka pygame i threading.

5 Alternativni pristupi i poređenje sa izabranim

1. **Jedan algoritam bira i dužinu trajanja i visinu note:** Prva ideja koju smo probale. Problem je manja kontrola nad dobijenim rezultatom. Često se dešavalo da je npr. ritam odličan, ali loši intervali (ili obrnuto), ali nije postojao način da se promeni samo jedan od ta dva.

2. **Niz tonova za svaki bit, uzastopni isti određuju trajanje:** npr. [c4, c4, c4, d4] bi značilo da imamo 3/8 c4, 1/8 d4. Problem sa ovim pristupom je kako kodirati situaciju poput 1/4 c4, 1/4 c4.

3. **Ritam je samo niz nula i jedinica bez dodatnih ograničenja:** Ovde je dolazilo do dva problema. Prvi je da niz počne nulom, odnosno pauzom. Drugi je da se pojavi predugačak niz nula. Zato su uvedena ograničenja da niz mora da počne jedinicom i da postoji maksimalan dozvoljen broj uzastopnih nula.

4. **Čuvaju se samo najbolji u trenutnoj generaciji:** Ovo je osnovni oblik genetskog algoritma. Prilikom kreiranja nove generacije čuva se određeni broj najboljih iz prethodne generacije, a ostali se zaboravljaju i budu zamenjeni novim jedinkama. Međutim, u našem problemu je velika verovatnoća da nove generacije daju lošije rezultate, tako da smo odlučile da čuvamo sve generacije (ovo možemo da posmatramo kao da "jedinke ne umiru").

5. **Algoritam vraća samo jednu najbolju jedinku:** Pokušaj da jedna jedinka bude cela kompozicija dovodi do loših rezultata. Umesto toga, odlučile smo se za pristup u kome čuvamo nekoliko najboljih rezultata koje kasnije kombinujemo u veću celinu.

6 Prednosti i mane izabranog pristupa

1. **Blisko ljudskom načinu razmišljanja:** Pokušaji slučajnog odabira na osnovu zadatih pravila koji se zatim iterativno popravljaju i kombinuju dobro opisuju ljudski pristup komponovanju.

2. **Interakcija između čoveka i računara:** Korisnik sve vreme učestvuje u procesu i usmerava program u pravom smeru. Mana je što zahteva korisničko vreme.

3. **Naivan pristup građenju većih celina:** Verovatno najzahtevniji deo ovog problema je kako izgraditi veće muzičke celine. Neophodno je imati određenu dozu ponavljanja, ali istovremeno i dovoljno varijacije. U našem slučaju, to smo pokušale da postignemo tako što imamo na raspolaganju više mogućih ritmova i na kraju najbolje melodije slažemo u jednu veću. Naravno, ovo je jako jednostavan pristup i ne garantuje dobre rezultate.

7 Literatura

1. Materijali sa predavanja
2. Tajčević M. (1962). Osnovna teorija muzike. Prosveta Beograd.
3. https://www.avid.com/resource-center/music-theory