UNIVERSITATEA BABEȘ-BOLYAI
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

Metode evolutive în compoziție muzicală

Rezumat teză de doctorat

Student doctorand: Sulyok Csaba
Coordonator științific: Prof. Dr. Czibula Gabriela

2019

UNIVERSITATEA BABEȘ-BOLYAI
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

Metode evolutive în compoziție muzicală

Rezumat teză de doctorat

Student doctorand: Sulyok Csaba
Coordonator științific: Prof. Dr. Czibula Gabriela

2019

Cuvinte cheie: compoziție algoritmică de muzică; programare genetică; programare genetică liniară; cunoștințe specifice de domeniu; evaluare automată de fitness

Cuprins

Cuprinsul tezei

Listă de publicații

Toate clasamentele sunt listate conform clasificării din 2014 a revistelor[1] și a conferințelor[2] din domeniul Informatică.

1. **Sulyok, C.**, McPherson, A., and Harte, C. (2015). *Corpus-taught Evolutionary Music Composition*. In Proceedings of the 13th European Conference on Artificial Life (ECAL), pages 587–594, York, UK. MIT Press. (**ISI Proceedings**)

   Categoria B, 4 puncte.

2. **Sulyok, C.** and Harte, C. (2017). *On Virtual Machine Architectures for Evolutionary Music Composition*. In Proceedings of the 14th European Conference on Artificial Life (ECAL), pages 577–584 Lyon, FR. MIT Press. (**ISI Proceedings**)

   Categoria B, 4 puncte

3. **Sulyok, C.** (2018). *Genetic Operators for Evolutionary Music Composition*. In Proceedings of the 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), pages 253–259, Timişoara, RO. (**ISI Proceedings**)

   Categoria C, 2 puncte

4. **Sulyok, C.**, McPherson, A., and Harte, C. (2019b). *Evolving the Process of a Virtual Composer*. Natural Computing, volume 18, issue 1, pages 47–60. Springer Netherlands. (**indexed WoS**)

   Categoria B, IF=0.778, 4 puncte.

5. **Sulyok, C.**, Harte, C., and Bodó, Z. (2019a). *On the Impact of Domain-specific Knowledge in Evolutionary Music Composition*. In Proceedings of the 2019 Genetic and Evolutionary Computation Conference (GECCO), pages 188–197, Prague, CZ. (**ISI Proceedings**)

   Categoria A, 8 puncte

[1] http://informatica-universitaria.ro/getpfile/16/CSafisat2.pdf;http://hfpop.ro/standarde/doctorat/2014-jurnale.pdf

[2] http://informatica-universitaria.ro/getpfile/16/CORE2013_Exported.xlsx;\http://hfpop.ro/standarde/doctorat/2014-conferinte.xlsx

6. **Sulyok, C.** (2019). *Towards Automated Quality Assessment Methods in Algorithmic Music Composition*. In Proceedings of the 21st International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC) *(accepted for publication)*, Timişoara, RO. (**ISI Proceedings**)

   Categoria C, 2 puncte

Punctaj: **24 de puncte**.

Capitolul 1

Introducere

Procesul de creare și evaluare a muzicii este subiectiv și greu de definit în mod fundamental. Capabilitatea compozitorului de a evalua calitatea unei noi idei muzicale va fi o decizie subiectivă bazată pe cunoștințe și evocarea pieselor anterioare (vezi Figura 1.1). Această recursiune a eului creativ sugerează utilitatea tehnicilor de calcul evolutiv în compunerea muzicală algoritmică.

Când proiectăm un sistem evolutiv, trebuie mai întâi să răspundem la întrebarea: „Ce ar trebui să evolueze acest sistem?" Majoritatea sistemelor de compunere prezente în literatură încearcă să evolueze direct piesele muzicale; acest proiect propune să evolueze în schimb *procesul* de compunere. Proiectăm pașii compunerii în baza unui proces care rulează pe o mașină virtuală (VM) Turing-completă. Această mașină conține un spațiu de instrucțiuni executate într-o anumită ordine în funcție de starea inițială a memoriei, și o modalitate de scriere a notelor pe un „scor muzical" ori de câte ori se găsește o instrucțiune de ieșire. Sistemul rezultat diferă de abordările anterioare din literatură prin încorporarea elementelor de programare genetică liniară (Brameier and Banzhaf, 2007).

Dezvoltarea abilităților devine apoi o problemă de programare genetică (Koza, 1992). Genotipul este șirul programatic prezentat mașinii virtuale, și fenotipul este ieșirea transformată în muzică. Într-un astfel de sistem, executarea pe mașina virtuală poate conține reguli latente de structurare care nu sunt vizibile în fenotipul muzical final.

Având în vedere percepția personală a muzicii sau a oricărei forme de artă în general, o definiție obiectivă a unui rezultat „bun" este dificilă de obținut (Waschka II, 2007). Procesul nostru de evaluare măsoară calitatea producției ca o similaritate distributivă cu piese existente într-un corpus de muzică reală. Deoarece în literatură rareori se folosesc evaluatori automați, descriem două metode noi de extracție și de evaluare a caracteristicilor, propunând și o metodă de comparare a acestora cu alte metrici automate. Ca un adaos suplimentar pe partea de evaluare automată, piesele din corpus nu sunt folosite pentru a forma sau influența populația inițială; în schimb ele ghidează numai testele de fitness, permițând un spațiu de căutare mai larg.

Lucrarea actuală se axează, de asemenea, pe *operatorii genetici* (Cavicchio Jr.,

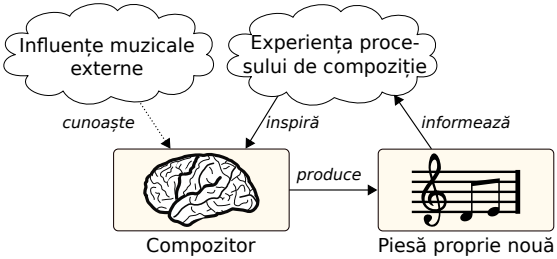

Figura 1.1. Procesul creativ al unui compozitor câștigă informații atât din experiența muzicii în general (influența externă a muzicii altora) cât și din experiența dobândită prin exersarea procesului de compoziție în sine.

1972)—setul de proceduri care creează generații ulterioare pornind de la cele existente. Prezentăm un studiu comparativ privind adaptarea dinamică a acestor operatori. Deși optimizarea online a hiperparametrilor a fost cercetată extensiv pentru abordări evolutive și algoritmi genetici, aceasta a fost rareori încercată împreună cu programarea genetică liniară.

Sistemul descris este potrivit pentru crearea oricărui produs similar unui set de intrări, atâta timp cât alegerea caracteristicilor extrase nu este influențată de natura modelului său. Cu toate acestea, se pune întrebarea: „Este util ca sistemul să fie informat într-o oarecare măsură despre natura propriei producții?" Propunem o comparație între setări care diferă în conștientizarea obiectivului final de a crea muzică. Este prezentat un limbaj specific domeniului (domain-specific language, DSL) (Fowler, 2010) cu instrucțiuni legate de muzică, ceea ce reduce gama largă de operații posibile la cele care se regăsesc deseori în procesul de compunere. De asemenea, propunem o reprezentare diatonică a înălțimilor notelor, care permite clarificarea pentru sistem a noțiunilor muzicale ca și gamă sau ton (Selfridge-Field, 2004).

Această teză propune următoarele contribuții originale în domeniu:

- O prezentare generală a abordării de programare genetică liniară pentru compunere algoritmică de muzică, propusă pentru prima dată în Sulyok et al. (2015), precum și explorarea spațiului parametrilor unui astfel de sistem (Secțiunea 3.2 și Sulyok et al. (2019b));
- O comparație a diferitelor arhitecturi și a seturilor de instrucțiuni legate de mașina virtuală (vezi Secțiunile 2.1, 3.3 și Sulyok and Harte (2017));
- O metodă obiectivă și complet automatizată de evaluare calitativă a datelor generate artificial, care se bazează pe similaritatea statistică raportată la un corpus de

date reale, alături de corpusuri multiple propuse, două metode de extracție de caracteristici (vezi Secțiunea 2.3 și Sulyok et al. (2019a)) și o propunere pentru un studiu comparativ cu alte abordări din literatura de specialitate (vezi Secțiunea 3.6 și Sulyok (2019));

- O comparație a operatorilor genetici și capacitățile lor adaptive in contextul de mai sus (vezi Secțiunile 1.1.3, 3.4 și Sulyok (2018));
- Explorarea impactului cunoștințelor specifice domeniului asupra sistemului, sub forma unui DSL cu inspirație muzicală și a unei reprezentări diatonice a pitch-ului (vezi Secțiunile 2.1, 3.5 și Sulyok et al. (2019a)).

Teza este structurată după cum urmează.

Capitolul 1 prezintă bazele cercetării noastre, inclusiv o prezentare teoretică a programării genetice, a subcategoriilor sale și a practicilor asociate. Secțiunea 1.2 oferă o imagine de ansamblu a literaturii de specialitate din domeniul metodelor evolutive, de evaluare a muzicii și a fitness-ului. Conceptele non-evolutive utilizate în cercetarea actuală, cum ar fi DSL-uri și n-grame, sunt de asemenea detaliate aici.

Capitolul 2 oferă o imagine metodologică de ansamblu asupra elementelor abordării propuse: detaliem funcționarea mașinii virtuale împreună cu seturile sale de instrucțiuni (Secțiunea 2.1), reprezentările noastre pentru muzică (Secțiunea 2.2) și metodele proprii de evaluare a similarității muzicale (Secțiunea 2.3).

Capitolul 3 prezintă mai multe seturi de experimente și rezultate pentru conceptele prezentate: Secțiunea 3.1 prezintă o dovadă a conceptului definit până acum; descrie experimente inițiale cu parametri parțial empirici pentru a obține un punct de pornire. Apoi, Secțiunea 3.2 se axează pe explorarea spațiului hiperparametrilor pentru a compara setări legate de dimensiunea populației, numărul de voci din corpus, precum și mecanismul de supraviețuire/reproducere abordat. O comparație aprofundată a diferitelor arhitecturi și dimensiuni de memorie ale mașinilor virtuale generale este prezentată în Secțiunea 3.3. O explorare a setărilor operatorilor genetici în manieră adaptivă este prezentată în Secțiunea 3.4. Secțiunea 3.5 se axează pe efectele încorporării cunoștințelor muzicale în VM și prezintă o comparație cu reprezentările uzuale. În cele din urmă, Secțiunea 3.6 propune un studiu (în derulare) de comparare a măsurilor de fitness cu alte abordări automate de evaluare din literatură.

Anexa A detaliază materialul auxiliar atașat la raportul curent, inclusiv locația depozitului software (repository) open-source, structura acestuia și modul de reproducere a experimentelor prezentate. Anexa B prezintă un număr de piese muzicale selectate aleatoriu, generate pe parcursul celor mai recente iterații experimentale.

Capitolul 2

Abordarea propusă

Sistemul nostru urmează o structură convențională de programare genetică (vezi Figura 2.1). O populație cu dimensiuni fixe este menținută pe parcursul unei iterații, reprezentată de genotipuri, fenotipuri și scoruri de teste de fitness ale indivizilor.

Genotipul este un vector de octeți care reprezintă în totalitate starea unei mașini virtuale, incluzând toate segmentele de memorie, regiștri și proprietăți—numim o astfel de structură de date un *șir genetic*. Orice valoare constituie o intrare validă pentru VM, prin urmare crearea unei generații zero este echivalentă cu generarea unor vectori aleatorii cu dimensiunile date. Șirurile genetice asigură starea inițială a VM, după care se execută programele, apoi octeții produși sunt colectați și transformați în fenotipuri: modele muzicale. Structura acestui model este complet independentă de structura și mecanismul VM. Această abordare în două etape a construirii de fenotip deviază de la sistemele muzicale evolutive anterioare, în sensul în care șirul genetic nu este folosit direct pentru a construi fenotipul, ci este interpretat de mașina virtuală.

2.1 Mașina virtuală

Așa cum am menționat anterior, interpretăm șirurile genetice folosind un VM—rezultatul este utilizat pentru a construi modelul muzical. Valorile de octeți întâlnite în memorie sunt mapate conform instrucțiunilor dintr-un set predefinit. Poziția inițială a indicelui de instrucțiune face parte din șirul genetic, la fel ca valoarea oricărui alt registru sau segment de memorie. Interpretorul citește octeți unul câte unul din RAM și execută instrucțiunea mapată la valoarea întâlnită. Un set de instrucțiuni poate conține mai multe tipuri de comenzi care manipulează datele în VM, cum ar fi transferul de date, aritmetice, condiționale etc. Pentru a produce date pentru construirea de fenotipuri, definim un tip special de instrucțiune de ieșire care furnizează ca ieșire unul sau mai mulți octeți din memorie sau registre.

Interpretarea unui șir genetic continuă până la îndeplinirea uneia dintre cele două condiții de oprire: fie se produce un număr preconizat de octeți, fie se atinge un număr maxim de cicluri de instrucțiune. Acesta din urmă este prezent ca un mecanism de

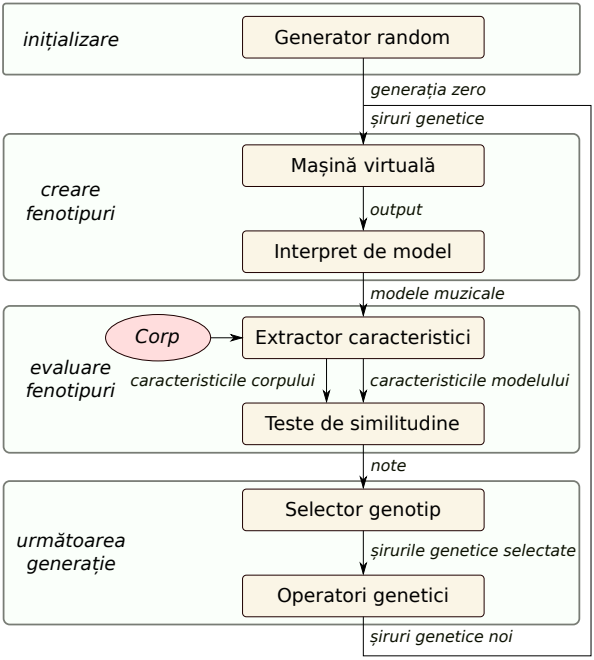

Figura 2.1. Fluxul de lucru al algoritmului propus: O populație de șiruri genetice este interpretată de mașina virtuală, și octeții rezultați sunt analizați pentru a construi modele muzicale. Caracteristicile relevante sunt extrase și comparate cu cele ale corpusului, oferind o măsură de fitness. Pe baza acestor scoruri, șirurile genetice sunt supuse operatorilor genetici pentru a produce generația următoare.

siguranță pentru a ieși din program atunci când este întâlnită o buclă infinită lipsită de instrucțiuni de ieșire.

Experimentele inițiale (Sulyok et al., 2015) propun o arhitectură de mașină virtuală bazată pe microprocesorul Intel 8080, cu adăugarea de instrucțiuni de ieșire. Acesta servește ca punct de plecare pentru explorare și comparație în etapele ulterioare ale cercetării. Aparatul include un spațiu de memorie RAM, care stochează atât instrucțiuni, cât și date, permițând rescrierea de sine în timpul executării. Fiind o mașină de 8 biți, 256 de instrucțiuni diferite sunt posibile. Indicatorul de instrucțiuni și un indicator de date auxiliare pot accesa orice octet în memorie; mutarea celei dintâi reprezintă un salt. Pe lângă memoria RAM, sistemul conține o stivă circulară de 256 de biți, accesată de un indicator de stivă pe 8 biți. Alți regiștri disponibili includ 8 regiștri de uz general, un acumulator și un carry flag. Includ comenzi tipice de transfer de date, aritmetice, logice,

ramificare, condiționale și control, alături de comenzile de ieșire dedicate.

Experimentele prezentate în Sulyok et al. (2019a) compară arhitecturi de mașini virtuale care se diferențiază prin cunoștințele încorporate referitoare la natura produselor lor. Acesta introduce un limbaj specific domeniului, proiectat în jurul cunoașterii producției sale, de aceea notele din memoria compozitorului virtual sunt distribuite sub formă de regiștri de 8 biți. Instrucțiunile de manipulare a datelor pentru regiștri de notă reflectă un impact specific domeniului: durata oricărei note poate fi dublată, înjumătățită sau punctată, iar înălțimea poate fi incrementată/decrementată pe scara diatonică.

2.2 Reprezentarea muzicii

Fenotipul nostru este un model care reprezintă o compoziție muzicală; interpretul de model ia șirul de octeți rezultați din VM pentru a produce un astfel de model. Pentru experimentele noastre, definim două reprezentări muzicale: una generală și permisivă, bazată pe MIDI (ne referim la acesta drept *modelul complex*), iar cealaltă este o variantă simplificată ghidată de statisticile unui corpus (*modelul redus*). Prima este reprezentată ca un set de voci, fiecare constând dintr-un șir de note. Fiecare notă are următoarele proprietăți:

1. *Interval inter-debut* - Perioada de timp dintre debutul notei anterioare și nota curentă dintr-o anumită piesă. Pentru prima notă dintr-o piesă, este intervalul de timp dintre începutul piesei și debutul notei.
2. *Durată* - Perioada de timp dintre debutul și decalarea notei.
3. *Înălțime* - O valoare numerică pe 7 biți (între 0 și 127) care reprezintă tonul definit în protocolul MIDI. Valoarea 69 este asociată cu frecvența de 440Hz (A concert), cu o creștere sau scădere unitară reprezentând o creștere sau o scădere a frecvenței respective cu un semiton.

Modelul redus este o versiune simplificată a reprezentării complexe, ghidată de statistica corpusului cântecelor populare folosit în experimentele noastre. Următoarele observații se referă la corpusul aflat la îndemână:

- Piesele sunt exclusiv monofonice, eliminând necesitatea mai multor voci.
- Piesele conțin melodii vocale fără retușuri sau note care se suprapun, prin urmare, timpul de debut și decalarea notelor poate fi complet încorporată în durată, adică fiecare notă începe la sfârșitul precedentei.
- 99,5% din valorile duratelor notelor se încadrează într-una din cele 8 valori comune, de aceea 3 biți sunt suficienți pentru reprezentarea lor.

Valori ale duratei	0	1	2	3	4	5	6	7
Valori Δ_t mapate din MIDI	1	2	3	4	6	8	12	16

Figura 2.2. Valori discrete ale duratelor notelor unui model redus, împreună cu valorile MIDI corespunzătoare, presupunând 4 unități pe sfert de notă și un tempo de 120BPM.

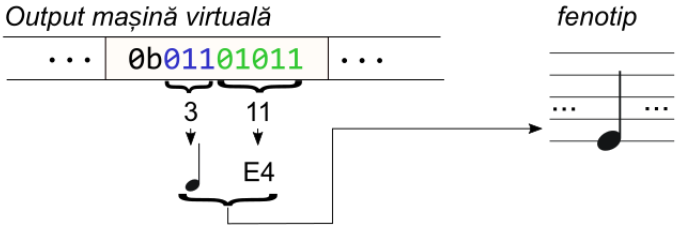

Figura 2.3. Construirea de modele reduse din octeții produși ai mașinii virtuale. Fiecare octet devine o notă cu cei mai semnificativi 3 biți fiind analizați ca durată, în timp ce ceilalți 5 devin înălțimea.

- În mod similar, 99,7% din înălțimea notelor pot fi mapate la 32 de valori cromatice contiguue, reprezentabile pe 5 biți.

Prin urmare, modelul redus devine o piesă muzicală cu o singură voce, care conține o serie de note, fiecare reprezentată complet de durată și înălțime.

Cele 8 valori de durată comune sunt mapate în unități temporale MIDI așa cum se vede în Figura 2.2. Valoarea înălțimii este reprezentată de un număr întreg în intervalul 0-31, cu o mapare flexibilă către înălțimile MIDI. Având în vedere numărul relativ mic de valori diferite pe care le permitem pentru durată și înălțime, putem reprezenta o notă în mod unic cu un singur octet. Pentru a folosi maximul posibil din informațiile produse de VM, interpretul de model creează 1 notă pe octet produs, folosind cei mai semnificativi 3 biți ca durată, iar restul de 5 ca înălțime (vezi Figura 2.3).

Rolul oricărei înălțimi de note este determinat de intervalul său raportat la tonica tastei armonice. Pentru a încorpora impactul contextual al tonului, definim diferite reprezentări în fenotipul redus, cu posibilitatea de conversie între fiecare. Acestea sunt:

- *cromatic standard* – pe scară echivalentă ca MIDI, dar redusă cu 53 pentru a se încadra în intervalul menționat mai sus; valorile extreme sunt deplasate cu octave pentru a se încadra în interval și a păstra consonanța, iar gama este ignorată.
- *cromatic derulat* – gama medie a membrilor corpusului este de 64 (E4). Versiunea

deplasată a acestui ton central (11) este desemnată drept tonică, și toate notele sunt transpuse în jurul acestei valori. Transpunerea nu afectează consonanța sau sentimentul general al piesei muzicale (Plomp and Levelt, 1965), iar această gamă permite valorilor de tonalitate să ia întotdeauna aceeași funcție, de ex. valoarea 18 reprezintă întotdeauna o cvintă perfectă.

- *diatonic* – Înălțimile sunt din nou centrate în jurul valorii de 11, dar sunt comprimate diatonic, astfel încât o creștere reprezintă un pas pe scara diatonică de 7 note. Această reprezentare interzice utilizarea accidentalelor, adică note cromatice „în afara locului" și, prin urmare, poate ajuta sistemul să faciliteze evoluția pieselor corecte din punct de vedere armonic.

2.3 Evaluarea calității muzicale prin similitudine

Fitness-ul oricărui fenotip interpretat este determinat de o serie de teste de similitudine. Toate testele urmăresc să evalueze cât de similar este un model din punct de vedere statistic celor din corpus. În acest scop, anumite caracteristici sunt extrase din corpus; aceleași caracteristici ale modelelor primite sunt comparate pentru a oferi valorile de fitness. Carasteristicile corpusului sunt grupate folosind algoritmul *k-means++* (Arthur and Vassilvitskii, 2007) pentru a controla extinderea spațiului de căutare.

Experimentele noastre folosesc două corpusuri diferite. Iterațiile de început folosesc *Inventions and Sinfonias* de la Bach[1], cuprinzând 30 de exerciții de claviatură. Acest catalog de piese muzicale a fost ales pentru lungime, tempo-ul constant, omogenitatea stilistică, precum și utilizarea lor anterioară în alte lucrări din literatură. Folosim în experimentele noastre atât o versiune cu o singură voce, cât și una cu două voci. Ambele conțin aceleași piese, fiecare cuprinzând același set de note, dar versiunile cu două voci reprezintă rezultatul unei separări bazate pe înălțime (împărțirea notelor executate cu mâna stângă și cea dreaptă).

Experimentele inițiale prezentate în Sulyok et al. (2019b) evidențiază că exercițiile Bach pot fi o alegere prea complexă pentru corpus. Piesele inerente implementează multe schimbări de gamă și șabloane ritmice complexe, ceea ce face posibil ca regulile generale să fie greu de dedus pentru un sistem pornind de la structuri complet aleatorii. Acest lucru sugerează că utilizarea unui corpus mai simplu ar putea să ofere beneficii. Prin urmare, experimentele privind impactul cunoștințelor specifice domeniului (Sulyok et al., 2019a) implementează colecția de cântece populare maghiare „Cimbalom"[2] ca și corpus. Fișierele sunt monofonice, cuprinzând doar aranjamente vocale fără instrumente însoțitoare.

[1] Works BWV 772-801 descărcate de pe `http://www.midiworld.com/bach.htm`
[2] Colecție descărcată de pe `http://www.cimbalom.nl/nepdalok.html`

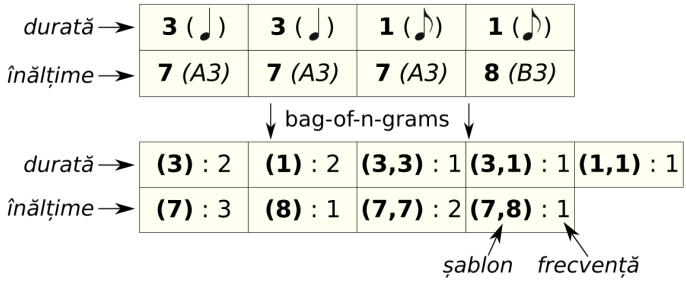

Figura 2.4. Demonstrarea construcției de n-grame (uni- și bigrame) ale unui fenotip, pentru comparație cu membrii corpusului.

Analizând entropia ambelor corpusuri în termeni de durată, înălțime și combinația ambelor relevă faptul că valorile urmează distribuții normale, a cărei medie este 45%. Entropia determină conținutul informațional al datelor: notele aleatorii ar produce valori ridicate, în timp ce repetiția constantă a aceleiași note are ca rezultat o entropie minimă; niciuna dintre aceste extreme nu sună plăcut într-o piesă de muzică. Astfel, ambele metode de extracție prezentate mai jos sunt asociate cu testarea entropiei.

Definim un descriptor ca fiind un rezultat al unei serii de transformări aplicate unui model. Testele de similitudine bazate pe descriptori vizează caracterul identic al descriptorilor intrării și cei ai corpusului. Patru transformări (histograme, diferențiala histogramelor, transformata Fourier și diferențiala transformării Fourier) sunt aplicate la fiecare dintre cele trei proprietăți ale unui model complex, rezultând în total douăsprezece teste de corelație. Calculăm corelația dintre doi descriptori linie cu linie folosind *coeficientul de corelație Pearson*.

Pentru a doua metrică de evaluare a similitudinii analizate, împrumutăm măsura de asemănare a cosinusului din domeniul regăsirii informațiilor (information retrieval): două melodii vor avea o similaritate maximă dacă unghiul închis de vectorii lor bag-of-n-grams este zero, adică aparițiile n-gramelor urmează aceeași distribuție. Funcția de fitness încorporează cele două proprietăți de bază ale modelelor reduse: durata și înălțimea. Cu toate acestea, valoarea de fitness bazată pe n-grame este derivată din șase componente. Acestea sunt n-grame de: (a) durate, (b) înălțimi, (c) perechi absolute de înălțime-durată, (d) diferențe între durate consecutive, (e) diferențe între înălțimi consecutive și (f) perechi de diferență de durată–diferență de înălțime. Figura 2.4 arată construcția caracteristicilor uni- și bigram pentru (a) și (b) din cadrul unei piese de exemplu.

Capitolul 3

Experimente și rezultate

Capitolul actual prezintă seturile de experimente efectuate pe parcursul cercetării, împreună cu rezultatele acestora, propunând, de asemenea, o comparație automată a măsurilor de fitness.

3.1 Evaluare de bază

Această secțiune prezintă testele inițiale care utilizează instrumentul de compoziție muzicală evolutivă. Configurațiile și rezultatele prezentate aici sunt transmise așa cum sunt prezentate în Sulyok et al. (2015). Executăm un număr de 40 de experimente, de fiecare dată permițând algoritmul să ajungă la 20.000 de generații; parametrii folosiți aici au fost derivați empiric din testele anterioare. Folosim o populație cu o mărime de 256 cu o rată de supraviețuire de 3% și un număr de clustere $k = 5$.

Figura 3.1 arată media și maxima notelor pe generație, calculând media pe cele 40 de rulări. Putem observa o creștere constantă a scorului maxim, dar stagnarea valorilor medii. Acest lucru poate fi explicat prin fragilitatea șirurilor genetice la operatorii genetici; chiar și o mică schimbare în genotip poate produce un model muzical complet diferit. Figura 3.2 arată distribuția de note a indivizilor cu cele mai mari scoruri în fiecare rundă. Demonstrează că algoritmul dă rezultate consistente pe diferite iterații. Deși proprietățile muzicale de repetiție și variație apar în piesele generate, alte proprietăți precum *armonie*, *melodie* și *entropie* lipsesc oarecum. Acest lucru sugerează necesitatea unor teste de fitness suplimentare inspirate de teoria muzicii.

3.2 Compararea parametrilor

Experimentele ulterioare explorează spațiul parametrilor posibili ai sistemului pentru a măsura impactul acestora asupra progresului procesului evolutiv. Experimentele transmise aici sunt în conformitate cu Sulyok et al. (2019b). Pentru fiecare set de parametri, executăm 20 de teste separate, permițând din nou sistemului să completeze 20.000 de generații. Parametrii menținuți constant pentru toate rulările includ o probabilitate

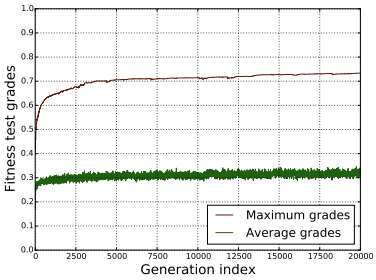

Figura 3.1. Progresul indicelui de fitness în cele 20.000 de generații: valorile medii și maxime pe generație.

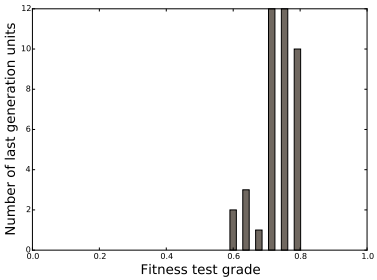

Figura 3.2. Distribuția notelor maxime în fiecare rulare. Valorile converg până la cel puțin 60%.

de supraviețuire de 15%, vârsta de supraviețuire maximă de 3, procentajul maxim de puncte de tăiere de 0,1%, și raportul de mutație maximă de 2%. Folosim 5 clustere de corp. Condițiile de oprire a mașinii virtuale sunt date de efectuarea a maxim 60.000 de instrucțiuni sau producerea a 2.600 octeți de ieșire.

Parametrii variați între încercări includ dimensiunea populației ($N \in \{2^x : 4 \leq x \leq 10\}$), numărul de voci din corpus (fie 1 sau 2), așa cum este discutat în Secțiunea 2.3, și mecanismul de supraviețuire: probabilistic sau determinist. Pot fi trase următoarele concluzii:

- Populații mai mari furnizează atât în medii mai bune, cât și în maxime mai ridicate. Estimarea unei curbe pentru notele disponibile (vezi Figura 3.3) sugerează că am obține numai câștiguri marginale prin creșterea continuă a dimensiunii populației.
- Folosind corpusul cu două voci se obțin valori medii ușor mai mici și maxime mai mari, probabil datorită creșterii dimensiunii inerente.
- O strategie probabilistică de supraviețuire previne elitismul și asigură o mai bună diversitate a populației: deși sunt produse note mai mici, se obțin medii semnificativ mai bune.
- Observând schimbările în ritmul apariției diferitelor tipuri de instrucțiuni (vezi Figura 3.4) observăm că algoritmul favorizează mai multe instrucțiuni de ieșire decât apar în datele aleatorii. Putem observa, de asemenea, o scădere a numărului de instrucțiuni de ramificare, care se poate datora efectului nociv al buclelor infinite.
- Evaluarea subiectivă a muzicii arată că piesele cu două voci sunt mai interesante, probabil din cauza natuii lor poliritmice inerente. Cu toate acestea încă lipsesc armonia și muzicalitatea generală.

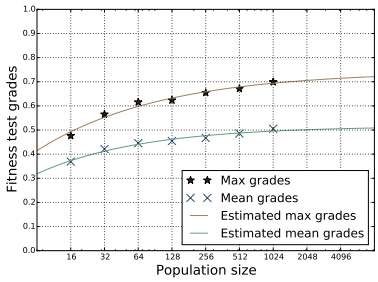

Figura 3.3. Curba de fitness estimată pe dimensiuni diferite de populație. Figura sugerează că o creștere suplimentară nu ar oferi rezultate semnificativ mai bune.

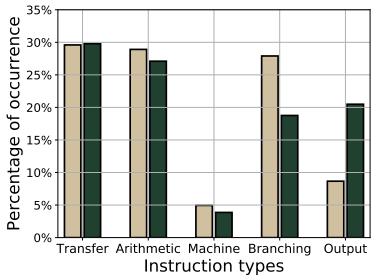

Figura 3.4. Rata de apariție a tipurilor de instrucțiuni întâlnite: barele din stânga arată generația zero (conținut VM aleatoriu), cele din dreapta sunt cele mai bune unități.

3.3 Arhitecturi pentru mașina virtuală

Capitolul actual compară diferite setări ale mașinii virtuale, a căror singură cerință este aceea de a putea emite octeți pe baza unui șir genetic: un vector de octeți care reprezintă în totalitate starea sa. Experimentele și rezultatele prezentate aici sunt transmise din Sulyok and Harte (2017). Parametrii testați includ:

- *arhitectura mașinii virtuale* - von Neumann sau Harvard;
- *diferite seturi de instrucțiuni* - sunt testate trei seturi diferite: cel complex utilizat în experimentele anterioare, un set OISC cu o singură instrucțiune `SBNZ`, și un set de instrucțiuni bazat pe stive;
- *dimensiunea memoriei* - 256, 4096 sau 65536 (adresabile folosind 8, 12 și, respectiv, 16 biți).

Aceste configurații rezultă în total în 18 scenarii diferite. 20 de experimente sunt executate pentru fiecare, solicitând piese de 30 de secunde și permițând algoritmului să ajungă la 10.000 de generații. Ceilalți parametri sunt aleși ca fiind optimi pe baza experimentelor anterioare. Pot fi trase următoarele concluzii:

- Rezultatele arhitecturii VM (vezi Figura 3.5) sugerează setul complex ca fiind cel mai bun, dar și cel mai vulnerabil la schimbare.
- Mașina cu o singură instrucțiune, deși a ajuns la cele mai mici fitness-uri, a funcționat mai bine decât se preconiza având în vedere complexitatea sa.

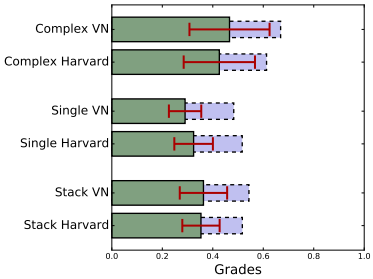

Figura 3.5. Generația finală grupată după setul de instrucțiuni și arhitectura VM. Barele late prezintă media și abaterea standard a întregii populații, în timp ce barele înguste prezintă valorile maxime.

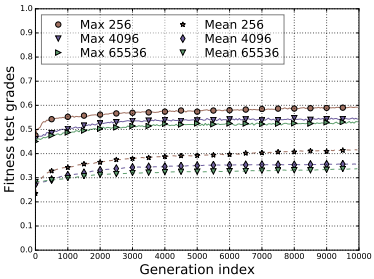

Figura 3.6. Progresul mediei și maximei grupat în funcție de dimensiunea memoriei. VM-urile care folosesc dimensiuni mai mici au obținut note mai mari, media crescând și în generațiile ulterioare.

- Arhitecturile von Neumann au un scor ușor mai mare decât cele Harvard, dovedind utilitatea exploratorie a capacității de autorescriere. Doar setul OISC a obținut rezultate mai bune folosind arhitectura Harvard; acest lucru poate fi cauzat de instrucțiunile sale îndelungate, ele fiind mai fragile în fața rescrierii în cazul von Neumann.

- Folosirea unei dimensiuni mai mici de memorie produce constant note mai mari (vezi Figura 3.6), cu toate acestea fișierele muzicale rezultate sunt subiectiv mai puțin atrăgătoare și excesiv de repetitive. Putem concluziona o deficiență în proiectarea testului de fitness: ele nu recompensează în mod corespunzător complexitatea inerentă a folosirii mai multor memorii.

3.4 Comparare a operatorilor genetici

În această secțiune cercetăm diferite abordări ale configurației hiperparametrilor operatorilor genetici în cadrul abordării noastre. Analizăm avantajele setării adaptive a distribuțiilor și ratelor operatorilor folosind metoda ascensiunii (hill climbing) (Russell and Norvig, 2016). Experimentele din această secțiune sunt prezentate din Sulyok (2018) - folosesc testele de corelație bazate pe descriptori, corpusul exercițiilor de pian Bach, și reprezentarea complexă a modelului.

Distribuțiile operatorilor preiau valori standard (8% reproducere, 90% încrucișare și 2% mutație, așa cum este propus de Koza (1992)), precum și valori adaptive folosind-o pe cea standard ca punct de plecare. Ratele testate includ numărul de puncte de tăietură

n_c, precum și numărul de octeți supuși mutației n_m, ambele reprezentate proporțional cu dimensiunea șirului genetic. Valorile care se modifică includ:

1. valorile constante utilizate în Sulyok et al. (2019b): $n_c = 0.1\%$, $n_m = 2\%$;
2. valori globale adaptive începând cu aceleași valori, adaptate o dată la nivel global la fiecare generație;
3. valori adaptive individualizate începând cu aceleași valori, moștenite și adaptate pe fiecare generație.

Pentru fiecare din cele 6 configurații rezultate, rulăm 20 de iterații, permițând algoritmului să ajungă la 10.000 de generații. Rezultatele sugerează că setările adaptive ale operatorilor oferă numai beneficii marginale și tind întotdeauna să crească numărul de entități supuse mutației cu costul unui număr mai scăzut de unități crescute prin încrucișare. Un rezultat obișnuit arată note maxime mari la costul mediilor mai mici–un rezultat surprinzător, deoarece numai mediile informează procesul de ascensiune. Adaptările individuale ale ratei nu oferă nicio îmbunătățire semnificativă față de nivelul de referință; acest lucru s-ar putea datora numărului copleșitor de dimensiuni de explorat.

3.5 Impactul cunoștințelor specifice domeniului

Secțiunea actuală investighează efectul integrării diferitelor niveluri de cunoștințe muzicale în arhitecturile mașinii virtuale (VM) și reprezentările de fenotip ale sistemului. Experimentele sunt prezentate în Sulyok et al. (2019a)– și folosesc testele de fitness bazate pe n-grame, corpusul cântecelor populare maghiare și reprezentarea redusă a modelului.

Examinăm două seturi de instrucțiuni care diferă în cunoștințele lor de structură muzicală. Primul este cel complex Turing-complet, utilizat în cercetările anterioare, care nu cunoaște natura producției sale; celălalt este un limbaj specific domeniului adaptat operațiunilor utilizate în mod obișnuit în procesul de compunere de muzică. Fenotipul este creat ca un model muzical redus cuprinzând o secvență de note reprezentate numai de durată și înălțime. Comparăm trei scheme de înălțime diferențiate prin cunoștințe de concepte tonale încorporate, cum ar fi tonica și gama.

Cu două arhitecturi VM diferite și trei scheme de înălțime, prezentăm și comparăm rezultate dintr-un total de șase configurații. Pot fi trase următoarele concluzii (vezi și Figura 3.7):

- Indiferent de reprezentarea înălțimii, DSL-ul obține rezultate mai bune decât mașina cu funcționalitate generală.
- Reprezentarea diatonică a înălțimilor duce în general la un rezultat mai slab decât celelalte două, posibil datorită spectrului său mai larg.

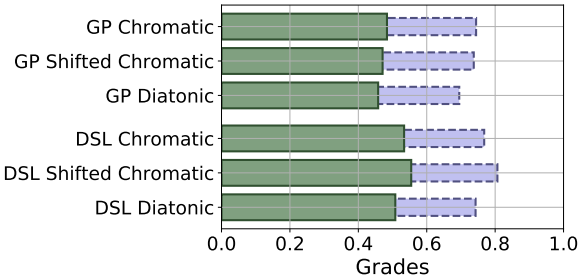

Figura 3.7. Valorile fitness medii și maxime de ultimă generație, grupate pe configurație

- Reprezentarea cromatică deplasată o întrece pe cea cromatică doar atunci când se utilizează DSL-ul. Acest lucru este de înțeles, deoarece DSL-ul conține instrucțiuni de manipulare a înălțimilor care păstrează corectitudinea armonică. Cu alte cuvinte, ajută la evitarea notelor cromatice disarmonice chiar dacă reprezentarea le-ar permite. Când se utilizează reprezentarea cromatică, membrii corpusului sunt răspândiți pe diferite tonici, de aceea notele consonante dintr-un model pot fi disonante în altul, iar noțiunile de gamă și armonie ar trebui să apară singure, fără să fie inerente în spațiul de căutare.

- Mașina DSL depășește deja aparatul GP în primele generații, dovedind eficacitatea DSL-ului chiar și pentru șiruri genetice aleatorii.

- O evaluare subiectivă arată multe rezultate plăcute armonic (vezi Figura 3.8); cu

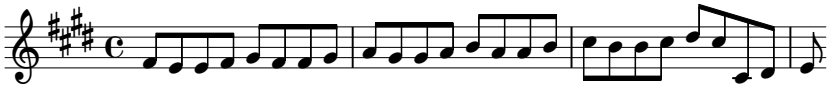

Figura 3.8. Fragment dintr-un rezultat care demonstrează variația pe un model mic. Acest model a obținut un indice de fitness de 73% .

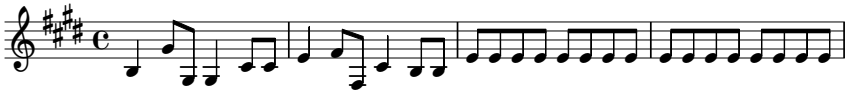

Figura 3.9. Fragment dintr-un rezultat care arată o primă parte supravariată, schimbându-se într-o buclă scurtă. Acest model a obținut un fitness de 79%.

toate acestea, testul de entropie este păcălit în multe cazuri, combinând segmente cu entropii extreme în aceeași piesă muzicală (vezi exemplu în Figura 3.9).

3.6 Propunere de comparare a metricilor de fitness

Secțiunea actuală prezintă propunerea unui set de experimente care încearcă să demonstreze viabilitatea celor două metode de evaluare automată a calității. În acest scop comparăm mecanismele între ele, precum și cu alte metrici propuse în literatura de specialitate. Experimentele în curs sunt prezentate aici cum sunt prezentate în Sulyok (2019).

Ca o primă etapă, intenționăm să reproducem câteva dintre cele mai acceptate sisteme de evaluare automată prezente în literatură și să le includem în sistemul LGP actual. Proiectarea modulară (așa cum se vede în Figura 2.1) permite o schimbare în stil black box al modulului de testare a calității pentru oricare alt bloc compatibil.

În timp ce orice fișier MIDI poate fi evaluat prin oricare dintre metricile propuse, o posibilă dovadă a calității ar fi corelarea cu păreri reale din partea unor ascultători umani. Pentru a colecta astfel de date numerice, propunem să folosim piața de crowdsourcing Amazon *Mechanical Turk* (MTurk) [1]. Alegând un set de fișiere MIDI, utilizatorii MTurk ar putea evalua numeric fiecare piesă, devenind astfel pseudo-funcția de referință a evaluării calității.

Propunem să selectăm modelele muzicale prezentate utilizatorilor pe baza următoarelor criterii: ar trebui să fie inclusă doar muzică generată, evoluată folosind toate funcțiile de fitness diferite, cu o variație mare a notelor. Modelele colectate ar fi prezentate utilizatorilor MTurk într-un mod în care fiecare utilizator ar asculta toate piesele și i-ar atribui o valoare numerică. Media valorilor de feedback ar constitui nota de referință pentru fiecare model din selecție. În cele din urmă, corelația poate fi măsurată între valorile atribuite de om și cele automate, oferind o dovadă numerică a eficienței acestora.

Propunem experimentul de mai sus cu următoarele valori: 5 funcții de fitness (dintre care 2 sunt cele prezentate anterior) folosite pentru a rula o simulare cu 500 de indivizi, ajungând la 1.000 de generații. Dintre ultimele 5 generații diferite, am alege 20 de modele așa cum au fost prezentate mai sus, rezultând 100 de piese scurte care pot fi prezentate utilizatorilor MTurk.

[1] Accesibil la https://www.mturk.com/

Capitolul 4

Concluzii

Rezultatele din literatura de specialitate demonstrează un potențial pentru compoziția algoritmică a muzicii, multe piese muzicale au fost compuse deja cu *asistența* algoritmilor evolutivi. Într-adevăr, Waschka II (2007) a văzut întotdeauna acești algoritmi ca instrumente auxiliare pentru inspirația compozitorilor în loc de compozitori virtuali independenți.

În cadrul cercetării propuse, am modelat cu succes procesul de gândire al unui compozitor virtual, separat de rezultatul lucrării sale prin intermediul mașinilor virtuale care produc octeți de produs pentru a fi interpretați ca și modele muzicale. Aceste piese au fost comparate cu membrii unui corp de muzică reală printr-o serie de teste de similaritate care implică transformări statistice, n-grame și entropie Shannon. Am evitat să stabilim condiții inițiale favorabile sistemului nostru, cum ar fi utilizarea corpusului ca populație inițială.

Setul de experimente a produs rezultate promițătoare, care demonstrează că metodologia reușește să creeze piese de muzică care converg către proprietățile corpusului ales. Piesele rezultate prezintă anumite calități muzicale (repetiție și variație) care nu sunt vizate în mod special de testele de fitness, apărând numai pe baza asemănărilor statistice. Deși proprietățile muzicale mai complexe cum ar fi armonia, lipsesc atunci când reprezentăm sistemul în mod general, aceste proprietăți evidențiate prin adăugarea de cunoștințe specifice domeniului.

În cele din urmă, întrucât sistemul a fost testat în mare măsură în mod izolat, fără o comparație aprofundată cu alte metode din literatură, am propus un set de experimente comparative în jurul unuia dintre cele mai importante aspecte ale cercetării: metoda de evaluare automată a calității. Prin urmare, am stabilit ca obiectiv primordial executarea și analiza rezultatelor acestor experimente.

Reprezentarea timpului poate fi de asemenea îmbunătățită, deoarece, în prezent datele furnizate testelor sunt o funcție a indicelui de notă. Aceasta permite buclele să apară pe piese cu un număr identic de note, dar cu o durată diferită, rezultând într-un poliritm.

Evaluarea subiectivă a rezultatelor arată numeroase șabloane interesante care nu sunt

prezente în corpus, dar care au proprietăți statistice similare. Cu toate acestea, multe rezultate combină segmente extrem de aleatorii cu porțiuni monotone, ceea ce sugerează că măsurătorile entropiei globale nu sunt întotdeauna adecvate pentru piese mai lungi. Prin urmare, propunem teste suplimentare care implică entropie instantanee măsurată în timp, așa cum este explorată de Manzara et al. (1992).

Deși combinația de n-grame și entropie produce rezultate interesante, experimente ulterioare ar putea fi efectuate pe metrica de evaluare a fitnessului. De exemplu, diferite scheme de ponderare pentru sub-teste sau mecanisme de normalizare pentru n-grame ar putea îmbunătăți calitatea rezultatelor. De asemenea, propunem experimente cu alți algoritmi de clustering, cum ar fi k-means kernelizat, clustering ierarhic sau spectral (Duda et al., 2000; Dhillon et al., 2004), studiind influența lor asupra pieselor muzicale generate.

Bibliografia tezei

Adler, D. (1993). Genetic algorithms and simulated annealing: a marriage proposal. In *IEEE International Conference on Neural Networks*, pages 1104–1109. IEEE.

Alfonseca, M., Cebrian, M., and Ortega, A. (2007). A simple genetic algorithm for music generation by means of algorithmic information theory. In *IEEE Congress on Evolutionary Computation*, pages 3035–3042. IEEE.

Angeline, P. J. (1996). Two self-adaptive crossover operators for genetic programming. In *Advances in genetic programming*, pages 89–109. MIT Press.

Arthur, D. and Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the 18th Annual Symposium on Discrete Algorithms (ACM-SIAM)*, pages 1027–1035, New Orleans, Louisiana. Society for Industrial and Applied Mathematics.

Bäck, T. (1992). Self-adaptation in genetic algorithms. In *Proceedings of the First European Conference on Artificial Life*, pages 263–271. MIT Press.

Bentley, J. (1986). Programming pearls: little languages. *Communications of the ACM*, 29(8):711–721.

Biles, J. A. (1994). GenJam: A Genetic Algorithm for Generating Jazz Solos. In *Proceedings of the 1994 International Computer Music Conference (ICMC)*, pages 131–137, Aarhus, Denmark. Michigan Publishing.

Bonissone, P. P., Subbu, R., Eklund, N., and Kiehl, T. R. (2006). Evolutionary algorithms + domain knowledge = real-world evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 10(3):256–280.

Brameier, M. F. and Banzhaf, W. (2007). *Linear genetic programming*. Springer Science & Business Media, 1st edition.

Cavicchio Jr., D. J. (1972). Reproductive adaptive plans. In *Proceedings of the ACM annual conference*, volume 1, pages 60–70, New York, New York, USA. ACM Press.

Chen, C.-C. J. and Miikkulainen, R. (2001). Creating Melodies with Evolving Recurrent Neural Networks. In *Proceedings of the INNS-IEEE International Joint Conference on Neural Networks*, pages 2241–2246, Piscataway, NJ. IEEE.

Chew, E. (2002). The Spiral Array: An Algorithm for Determining Key Boundaries. In *Proceedings of the Second International Conference on Music and Artificial Intelligence (ICMAI)*, pages 18–31, Edinburgh, Scotland. Springer Berlin Heidelberg.

Chuan, C.-H. and Chew, E. (2005). Polyphonic Audio Key Finding Using the Spiral Array CEG Algorithm. In *Proceedings of the 2005 IEEE International Conference on Multimedia and Expo (ICME)*, pages 21–24, Amsterdam, The Netherlands. IEEE.

Conklin, D. (2003). Music generation from statistical models. In *Proceedings of the 2003 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, pages 30—-35, Aberystwyth, Wales. AISB.

Costelloe, D. and Ryan, C. (2007). Towards models of user preferences in interactive musical evolution. In *Proceedings of the 9th Genetic and Evolutionary Computation Conference (GECCO)*, pages 2254–2254, London, UK. ACM Press.

Cover, T. M. and Thomas, J. A. (2012). *Elements of information theory*. John Wiley & Sons.

Cramer, N. L. (1985). A Representation for the Adaptive Generation of Simple Sequential Programs. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 183–187, Hillsdale, NJ, USA. L. Erlbaum Associates Inc.

Dahlstedt, P. (2007). Autonomous Evolution of Complete Piano Pieces and Performances. In *9th European Conference on Artificial Life*.

Davis, L. (1989). Adapting operator probabilities in genetic algorithms. In *proc. 3rd International conference on genetic algorithms*, pages 61–69.

De Jong, K. (1988). Learning with genetic algorithms: An overview. *Machine learning*, 3(2-3):121–138.

De Prisco, R., Zaccagnino, G., and Zaccagnino, R. (2011). A multi-objective differential evolution algorithm for 4-voice compositions. In *2011 IEEE Symposium on Differential Evolution (SDE)*, pages 1–8. IEEE.

Dhillon, I. S., Guan, Y., and Kulis, B. (2004). Kernel k-means, spectral clustering and normalized cuts. In *Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 551–556, Seattle, Washington, USA. ACM Press.

Dolin, B., Arenas, M. G., and Merelo, J. J. (2002). Opposites Attract: Complementary Phenotype Selection for Crossover in Genetic Programming. In *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN)*, PPSN VII, pages 142–152, London, UK. Springer-Verlag.

Donnelly, P. and Sheppard, J. (2011). Evolving Four-Part Harmony Using Genetic Algorithms. In *Applications of Evolutionary Computation*, volume 6625, pages 273–282. Springer, Berlin, Heidelberg.

Doraisamy, S. and Rüger, S. M. (2004). A Polyphonic Music Retrieval System Using N-Grams. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, pages 204–209, Barcelona, Spain. Universitat Pompeu Fabra.

Dostál, M. (2012). Musically meaningful fitness and mutation for autonomous evolution of rhythm accompaniment. *Soft Computing*, 16(12):2009–2026.

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern classification*. John Wiley & Sons.

Eigenfeldt, A. (2009). The Evolution of Evolutionary Software: Intelligent Rhythm Generation in Kinetic Engine. In *EvoWorkshops*, volume 5484, pages 498–507. Springer.

Esquivel, S. C., Leiva, A., and Gallard, R. H. (1997). Multiple crossover per couple in genetic algorithms. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 103–106. IEEE.

Fernández, J. D. and Vico, F. (2013). AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 48:513–582.

Fonseca, C. M. and Fleming, P. J. (1993). Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA)*, pages 416–423, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Fowler, M. (2010). *Domain Specific Languages*. Addison-Wesley Professional, 1st edition.

Gibson, P. M. and Byrne, J. A. (1991). NEUROGEN, musical composition using genetic algorithms and cooperating neural networks. In *Proceedings of the Second International Conference on Artificial Neural Networks (ICANN)*, pages 309–313, Bournemouth, UK. IET.

Gilreath, W. and Laplante, P. (2004). A one instruction set architecture for genetic algorithms. In *Biocomputing*, pages 91–113. Nova Science Publishers, Inc.

Gilreath, W. F. and Laplante, P. A. (2003). *Computer architecture: A minimalist perspective*, volume 730. Springer Science & Business Media.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

Goldberg, D. E. and Lingle, R. (1985). Alleles, loci, and the traveling salesman problem. In *Proceedings of an international conference on genetic algorithms and their applications*, volume 154, pages 154–159. Lawrence Erlbaum, Hillsdale, NJ.

Gomez, J. (2004). Self adaptation of operator rates in evolutionary algorithms. In *Genetic and Evolutionary Computation Conference*, pages 1162–1173. Springer.

Grefenstette, J. (1986). Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):122–128.

Harries, K. and Smith, P. (1997). Exploring alternative operators and search strategies in genetic programming. *Genetic Programming*, 97:147–155.

Hartmann, P. (1990). Natural Selection of Musical Identities. In *Proceedings of the 1990 International Computer Music Conference (ICMC)*, pages 234–236, Glasgow, Scotland. Michigan Publishing.

Hofmann, D. M. (2015). A Genetic Programming Approach to Generating Musical Compositions. In *Proceedings of the 4th International Conference on Evolutionary and Biologically Inspired Music, Sound, Art and Design (EvoMUSART)*, volume 9027, pages 89–100, Copenhagen. Springer, Cham.

Horner, A. and Goldberg, D. E. (1991). Genetic Algorithms and Computer-Assisted Music Composition. In *Proceedings of the 4th International Conference on Genetic Algorithms (ICGA)*, pages 437–441, San Diego, CA, USA. Morgan Kaufmann.

Horowitz, D. (1994). Generating Rhythms with Genetic Algorithms. In *AAAI*, volume 94, page 1459.

Hu, T., Banzhaf, W., and Moore, J. H. (2013). Robustness and evolvability of recombination in linear genetic programming. In *European Conference on Genetic Programming*, pages 97–108. Springer.

Jacob, B. (1995). Composing with Genetic Algorithms. In *International Computer Music Association*, pages 452–455.

Johanson, B. and Poli, R. (1998). GP-Music: An Interactive Genetic Programming System for Music Generation with Automated Fitness Raters. In *Proceedings of the Third Annual Conference on Genetic Programming*, pages 181–186, University of Wisconsin, Madison, Wisconsin, USA. Morgan Kaufmann.

Jones, D. W. (1988). A Minimal {CISC}. *SIGARCH Comput. Archit. News*, 16(3):56–63.

Julstrom, B. A. (1997). Adaptive operator probabilities in a genetic algorithm that applies three operators. In *Proceedings of the 1997 ACM symposium on Applied computing*, pages 233–238. ACM Press.

Kazarlis, S., Papadakis, S., Theocharis, J., and Petridis, V. (2001). Microgenetic algorithms as generalized hill-climbing operators for GA optimization. *IEEE Transactions on Evolutionary Computation*, 5(3):204–217.

Klinger, R. and Rudolph, G. (2006). Evolutionary composition of music with learned melody evaluation. In *Proceedings of the Int. Conference on Computational Intelligence, Man-machine Systems and Cybernetics (CIMMACS'06)*.

Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection*. MIT Press.

Krumhansl, C. L. (2001). A key-finding algorithm based on tonal hierarchies. In *Cognitive Foundations of Musical Pitch*, pages 77–110. Oxford University Press.

Li, X., Zhou, C., Xiao, W., and Nelson, P. C. (2005). Direct evolution of hierarchical solutions with self-emergent substructures. In *Machine Learning and Applications, 2005. Proceedings. Fourth International Conference on*, pages 6—-pp. IEEE.

Lidov, D. and Gabura, J. (1973). A Melody Writing Algorithm Using a Formal Language Model. *Computers in the Humanities*, 3-4:138–148.

Lipowski, A. and Lipowska, D. (2012). Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, 391(6):2193–2196.

Lo, M. and Lucas, S. M. (2007). N-gram fitness function with a constraint in a musical evolutionary system. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 4246–4251, Singapore. IEEE.

López, C. L. (2008). Heuristic Hill-Climbing as a Markov Process. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, pages 274–284. Springer Berlin Heidelberg.

Loughran, R., McDermott, J., and O'Neill, M. (2016). Grammatical Music Composition with Dissimilarity Driven Hill Climbing. In *Proceedings of the 5th International Conference on Evolutionary and Biologically Inspired Music, Sound, Art and Design (EvoMUSART)*, pages 110–125, Porto, Portugal. Springer.

MacCallum, R. M., Mauch, M., Burt, A., Leroi, A., and M (2012). Evolution of music by public choice. *Proceedings of the National Academy of Sciences*, 109(30):12081–12086.

Madsen, S. T. and Widmer, G. (2007). Key-finding with Interval Profiles. In *Proceedings of the 2007 International Computer Music Conference (ICMC)*, pages 212–215, Copenhagen, Denmark. Michigan Publishing.

Mahfoud, S. W. (1995). *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign.

Manaris, B., Vaughan, D., Wagner, C., Romero, J., and Davis, R. B. (2003). Evolutionary Music and the Zipf-Mandelbrot Law: Developing Fitness Functions for Pleasant Music. In *Workshops on Applications of Evolutionary Computation*, pages 522–534. Springer, Berlin, Heidelberg.

Manaris, B. Z., Roos, P., Machado, P., Krehbiel, D., Pellicoro, L., and Romero, J. (2007). A Corpus-Based Hybrid Approach to Music Analysis and Composition. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 839–845. AAAI Press.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

Manning, C. D. and Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

Manzara, L. C., Witten, I. H., and James, M. (1992). On the Entropy of Music: An Experiment with Bach Chorale Melodies. *Leonardo Music Journal*, 2(1):81.

Martins, J. M. and Miranda, E. R. (2007). Emergent Rhythmic Phrases in an A-Life Environment. In *Proceedings of ECAL 2007 Workshop on Music and Artificial Life (MusicAL 2007)*, pages 11–14.

Masataka, N. (2007). Music, evolution and language. *Developmental Science*, 10(1):35–39.

Mavaddat, F. and Parhami, B. (1988). URISC: the ultimate reduced instruction set computer. *International Journal of Electrical Engineering Education*, 25(4):327–334.

McCormack, J. (1996). Grammar based music composition. *Complex systems*, 96:321–336.

McIntyre, R. A. (1994). Bach in a box: the evolution of four part Baroque harmony using the genetic algorithm. In *Proceedings of the IEEE First World Congress on Computational Intelligence*, pages 852–857, Orlando, Florida, USA. IEEE.

McKay, R. I., Hoai, N. X., Whigham, P. A., Shan, Y., and O'Neill, M. (2010). Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines*, 11(3-4):365–396.

Mernik, M., Heering, J., and Sloane, A. M. (2005). When and how to develop domain-specific languages. *ACM computing surveys (CSUR)*, 37(4):316–344.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81.

Miranda, E. R. and Biles, J. A. (2007). *Evolutionary Computer Music*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Moroni, A., Manzolli, J., Von Zuben, F., and Gudwin, R. (2000). Vox populi: An interactive evolutionary system for algorithmic music composition. *Leonardo Music Journal*, 10:49–54.

Munroe, D. R. (2004). Genetic Programming: The ratio of Crossover to Mutation as a function of time. *Research Letters in the Information and Mathematical Sciences*, 6:83–96.

Murata, T. and Ishibuchi, H. (1995). MOGA: multi-objective genetic algorithms. In *Evolutionary Computation, 1995., IEEE International Conference on*, volume 1, page 289.

Nordin, P. and Banzhaf, W. (1995). Evolving Turing-Complete Programs for a Register Machine with Self-modifying Code.

Nordin, P., Banzhaf, W., and Francone, F. D. (1999). Efficient evolution of machine code for CISC architectures using instruction blocks and homologous crossover. *Advances in genetic programming*, 3:275–299.

Nuanáin, C. b., Herrera, P., and Jordá, S. (2015). Target-Based Rhythmic Pattern Generation and Variation with Genetic Algorithms. In *Proceedings of The 12th Sound and Music Computing Conference*, Maynooth, Ireland.

Oltean, M., Gro\csan, C., Dio\csan, L., and Mih\uail\ua, C. (2009). Genetic programming with linear representation: a survey. *International Journal on Artificial Intelligence Tools*, 18(02):197–238.

O'Reilly, U.-M. and Oppacher, F. (1995). Hybridized crossover-based search techniques for program discovery. In *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, volume 2, pages 573–578. IEEE.

Ortega, A., Alfonso, R. S., and Alfonseca, M. (2002). Automatic Composition of Music by Means of Grammatical Evolution. In *Proceedings of the 2002 International Conference on APL: Array Processing Languages: Lore, Problems, and Applications*, pages 148–155, Madrid, Spain. ACM Press.

Patterson, D. A. and Hennessy, J. L. (2013). *Computer organization and design: the hardware/software interface*. Newnes.

Pearce, M. T. (2005). *The construction and evaluation of statistical models of melodic structure in music perception and composition*. PhD thesis, City University London.

Perkis, T. (1994). Stack-based genetic programming. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pages 148–153 vol.1.

Phon-Amnuaisuk, S., Law, E. H. H., and Kuan, H. C. (2007). Evolving Music Generation with SOM-Fitness Genetic Programming. In *Applications of Evolutinary Computing*, pages 557–566. Springer Berlin Heidelberg, Berlin, Heidelberg.

Phon-Amnuaisuk, S., Tuson, A., and Wiggins, G. (1999). Evolving Musical Harmonisation. In *Proceedings of the 1999 International Conference on Artificial Neural Nets and Genetic Algorithms*, pages 229–234, Portorož, Slovenia. Springer.

Piszcz, A. and Soule, T. (2006). Genetic Programming: Analysis of Optimal Mutation Rates in a Problem with Varying Difficulty. In *FLAIRS Conference*, pages 451–456.

Plomp, R. and Levelt, J. M. (1965). Tonal Consonance and Critical Bandwidth. *The Journal of the Acoustical Society of America*, 38(4):548–560.

Poli, R. and Langdon, W. B. (1998). On the search properties of different crossover operators in genetic programming. *Genetic Programming*, pages 293–301.

Poli, R., Langdon, W. B., McPhee, N. F., and Koza, J. R. (2008). *A field guide to genetic programming*. Lulu. com.

Reddin, J., McDermott, J., and O'Neill, M. (2009). Elevated Pitch: Automated Grammatical Evolution of Short Compositions. In *Applications of Evolutionary Computing*, volume 5484 of *Lecture Notes in Computer Science*, pages 579–584. Springer Berlin Heidelberg.

Renders, J.-M. and Bersini, H. (1994). Hybridizing genetic algorithms with hill-climbing methods for global optimization: two possible ways. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence.,*, pages 312–317. IEEE.

Roy, D. B., Das, P., and Mukhopadhyay, D. (2015). ECC on Your fingertips: a single instruction approach for lightweight ECC design in GF (p). In *International Conference on Selected Areas in Cryptography*, pages 161–177. Springer.

Russell, S. J. and Norvig, P. (2016). *Artificial Intelligence: A Modern Approach*. Malaysia; Pearson Education Limited.

Ryan, C., Collins, J. J., and O'Neill, M. (1998). Grammatical evolution: Evolving programs for an arbitrary language. In *Proceedings of the First European Conference on Genetic Programming (EuroGP)*, pages 83–96, Paris, France. Springer.

Selfridge-Field, E. (2004). Music Theory for Computer Applications.

Spector, L. and Robinson, A. (2002). Genetic Programming and Autoconstructive Evolution with the Push Programming Language. *Genetic Programming and Evolvable Machines*, 3(1):7–40.

Srinivas, M. and Patnaik, L. M. (1994a). Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4):656–667.

Srinivas, M. and Patnaik, L. M. (1994b). Genetic Algorithms: A Survey. *Computer*, 27(6):17–26.

Sulyok, C. (2018). Genetic Operators for Evolutionary Music Composition. In Abraham, E., Negru, V., Petcu, D., Zaharie, D., Ida, T., Jebelean, T., and Watt, S., editors, *Proceedings of the 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 253–259, Timisoara, RO.

Sulyok, C. (2019). Towards Automated Quality Assessment Methods in Algorithmic Music Composition. In *Proceedings of the 21st International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC) (accepted for publication)*, Timisoara, RO.

Sulyok, C. and Harte, C. (2017). On Virtual Machine Architectures for Evolutionary Music Composition. In *Proceedings of the 14th European Conference on Artificial Life (ECAL)*, pages 577—-584, Lyon. MIT Press.

Sulyok, C., Harte, C., and Bodó, Z. (2019a). On the Impact of Domain-specific Knowledge in Evolutionary Music Composition. In *Proceedings of the 2019 Genetic and Evolutionary Computation Conference (GECCO)*, pages 188–197, Prague, CZ. ACM Press.

Sulyok, C., McPherson, A., and Harte, C. (2015). Corpus-taught Evolutionary Music Composition. In *Proceedings of the 13th European Conference on Artificial Life (ECAL)*, pages 587–594, York, UK. The MIT Press.

Sulyok, C., McPherson, A., and Harte, C. (2019b). Evolving the process of a virtual composer. *Natural Computing*, 18(1):47—-60.

Swain, J. P. (1986). The need for limits in hierarchical theories of music. *Music Perception: An Interdisciplinary Journal*, 4(1):121–147.

Temperley, D. (2001). *The cognition of basic musical structures*. MIT Press.

Thywissen, K. (1999). GeNotator: An Environment for Exploring the Application of Evolutionary Techniques in Computer-assisted Composition. *Org. Sound*, 4(2):127–133.

Tokui, N. and Iba, H. (2000). Music composition with interactive evolutionary computation. In *Proceedings of the Third International Conference on Generative Art*, volume 17, pages 215–226, Milan, Italy.

Towsey, M., Brown, A., Wright, S., and Diederich, J. (2001). Towards Melodic Extension Using Genetic Algorithms. *Educational Technology and Society*, 4(2):54–65.

Turing, A. M. (1950). Computing Machinery and Intelligence. In *Mind*, volume 59, pages 433–460.

Tuson, A. and Ross, P. (1998). Adapting operator settings in genetic algorithms. *Evolutionary computation*, 6(2):161–184.

Ukkonen, E., Lemström, K., and Mäkinen, V. (2003). Geometric Algorithms for Transposition Invariant Content-Based Music Retrieval. In *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR)*, pages 193–199, Maryland, USA. Johns Hopkins University Press.

Unemi, T. (2002). SBEAT3: A Tool for Multi-part Music Composition by Simulated Breeding. In *Proceedings of the 8th international conference on artificial life*, pages 410–413. MIT Press.

Vafaee, F. and Nelson, P. C. (2009). Self-adaptation of genetic operator probabilities using differential evolution. In *Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems. SASO'09.*, pages 274–275. IEEE.

Vafaee, F., Nelson, P. C., Zhou, C., and Xiao, W. (2008). Dynamic adaptation of genetic operators' probabilities. *Studies in Computational Intelligence*.

Waschka II, R. (2007). Composing with Genetic Algorithms: GenDash. In *Evolutionary Computer Music*, pages 117–136. Springer London.

Whorley, R. and Conklin, D. (2016). Music Generation from Statistical Models of Harmony. *Journal of New Music Research*, 45(2):160–183.

Whorley, R., Rhodes, C., Wiggins, G., and Pearce, M. (2013). Harmonising Melodies: Why do we add the bass line first? In *International Conference on Computational Creativity*, pages 79–86, Sydney.

Wiggins, J. (2007). Compositional process in music. In *International handbook of research in arts education*, pages 453–476. Springer Netherlands.

Wolkowicz, J., Heywood, M., and Keselj, V. (2009). Evolving indirectly represented melodies with corpus-based fitness evaluation. In *Workshops on Applications of Evolutionary Computation*, pages 603–608, Tübingen, Germany. Springer Berlin Heidelberg.

Wolkowicz, J., Kulka, Z., and Keselj, V. (2008). N-gram-based approach to composer recognition. *Archives of Acoustics*, 33(1):43–55.

Worth, P. and Stepney, S. (2005). Growing Music: Musical Interpretations of L-Systems. In *EvoWorkshops 2005: Applications of Evolutionary Computing*, pages 545–550. Springer, Berlin, Heidelberg.

Wu, C. L., Liu, C. H., and Ting, C. K. (2014). A novel genetic algorithm considering measures and phrases for generating melody. In *Proceedings of the 2014 Congress on Evolutionary Computation (CEC)*, pages 2101–2107, Beijing, China. IEEE.