Algoritmična kompozicija glasbe (končno poročilo)

Ivan Antešić
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani
Ljubljana, Slovenija
email: ivan.v.antesic@gmail.com

***Povzetek*—V okviru seminarske naloge sem razvil program, ki z algoritmično kompozicijo ustvari preprosto homofonično skladbo. V tem poročilu je predstavljen pristop z matematičnim modelom markovih verig, ki sem ga uporabil pri kreiranju mojega programa. Prav tako je na kratko predstavljeno kako je s teorijo glasbe možno določiti splošna pravila za delovanje algoritma. Na koncu končno rešitev in rezultate primerjam z drugimi algoritmi in z že obstoječo programsko opremo.**

I. Uvod

Algoritmična kompozicija je način generiranja glasbe z različnimi algoritmi [1], [2]. Generiranje glasbe s pomočjo pravil in naključja se je uporabljalo skozi celotno zgodovino glasbe. V 18. stoletju so bile na primer popularne glasbene igre kock ali kart, ki so lahko služile kot vir navdiha ali zabave [3]. Slavni skladatelji kot na primer Mozart, so pisali samostojne delčke skladbe, ki so jih nato lahko ljudje na podlagi naključja združevali skupaj v nove celote.

S razvojem tehnologije in algoritmov je danes možno ustvariti aplikacije[4], [5], ki so sposobne generirati kompleksne glasbene kompozicije. Kljub temu se še vedno večinoma uporabljajo za začetno kompozicijo, ki jo nadgradi človeški skladatelj. Teorija glasbe in kompozicije je obsežno področje, ki od skladatelja poleg poznavanja pravil glasbe zahteva še kreativnost in dolgoletne izkušnje, da lahko napiše kakovostno skladbo.

Ravno zato pa sem se odločil razviti svojo rešitev algoritmične kompozicije. Ker nimam glasbene izobrazbe in časa, da bi se izučil komponiranja, sem napisal program, ki namesto mene napiše preprosto skladbo.

Podobno kot glasbene igre preteklosti, rešitev deluje na principih naključnosti in rekombinacje. S preučevanjem glasbene teorije[6] sem določil osnovna pravila algoritma in z matematičnim modelom markovih verig na podlagi vhodne skladbe generiral novo različico. Cilj projekta je bil generirati različno glasbo, ki bi jo lahko uporabil za spremljavo preprostih računalniških iger. Pri tem sem se omejil na homofonične skladbe (ena melodija, podprta z harmonsko spremljavo) v pogostem štiričetrtinskem taktu.

V nadaljevanju poročila je opisan pregled že obstoječih algoritmov za kompozicijo. Nato je predstavljena moja implementacija rešitve z pripadajočimi metodami in glasbeno teorijo, ki se skriva v ozadju. Na koncu so vrednoteni rezultati rešitve in podani predlogi, kako bi lahko izboljšali algoritem.

II. Pregled področja

Področje algortimične kompozicije je dobro raziskano. Obstaja več pristop k razvoju algoritma, ki komponira glasbo - od strojnega učenja, genetskih algoritmov in različne kombinacije večih algoritmov. Spodaj so opisani najbolj razširjeni pristopi.

A. Fraktali in simulacije

Predvsem je razširjena uporaba fraktalov, kjer lahko z rekurzivnim ponavljanjem preprostih pravil generiramo zapletene vzorce. Prav tako je možno uporabiti simulacije npr. mulekularno dinamiko, kjer spreminjamo paremetre zvokov, ki se izvedejo v istem času [7]. Prednost takšnih algoritmov je preprosta implementacija za katero ni potrebno poznavanje glasbene teorije.

Čeprav s takšnim pristopom lahko ustvarimo harmonično glasbo, je ta precej različna od glasbenih del, ki jih ustvarjajo ljudje, saj ne upošteva točno določenih pravil komponiranje glasbe. Končna melodija takšnih algoritmov zveni tuje in nenavadno. Generiranju melodije z fraktali in simulacijami sem se zaradi tega izognil, služili pa so mi kot navdih pri eni od metod algoritma, kjer s ponavljajočim vzorcem delimo dobe takta skladbe.

B. Gramatike

Skozi stoletja so skladatelji razvili različna pravila in opazili različne vzorce, ki se pojavljajo pri komponiranju glasbe - nekatere zaporedne note se ujemajo bolje, kot druge, določeni akordi v slušatelju vzbudijo določene občutke. V želji, da bi generirali človeku domačo glasbo, nekateri pristopi uporabljajo algoritme, ki upoštevajo pravila in vzorce glasbene teorije. To je možno narediti z uporabo gramatike in L-sistemov, kjer definiramo slovnico ter nato na podlagi pravil gradimo našo skladbo. Slabost takšnih pristopov je deterministične rezultat (za nek vhod, vrne vedno isti izhod) in bolj zapletene implementacija.

Pristop z L-sistemi je uporabljen kot eden od algoritmov v delu Šest tehnik za algoritmično kompozicijo glasbe [8]. V članku so izpostavili, da generira dobre, kompleksne skladbe, vendar ima težjo implementacijo in zahteva pisanje velikega števila besed slovnice. Ker je moja ciljna melodija preprosta nisem uporabil pristopa z gramatiko.

C. Markove verige

Pri algoritmični kompoziciji se uporabljajo tudi različni matematični modeli, kjer z kontroliranjem naključnosti generiramo ali rekombiniramo dele skladb v zaključeno celoto. Najbolj razširjenj je model markovih verig, ki se uporablja v programskih opremah in knjižnjicah, kot je Jmusic[9], namenjenih računalniški kompoziciji glasbe[10].

Diskretni model opisuje verjetnost prehodov v naslednja stanja, glede na trenutno stanje v katerem se nahajamo. Njegova glavna lastnost je markov pogoj, ki pravi, da je naslednje stanje odvisno le od trenutnega stanja, kar lahko vidimo v spodnji enačbi, kjer X1,..., Xn predstavljajo naključne spremenljivke stanj, Pr pa pogojno verjetnost izbire naslednjega stanja.

$$Pr(Xn+1 = xn+1 | X1 = x1, ..., Xn = xn) \quad (1)$$
$$= Pr(Xn+1 = xn+1 | Xn = xn) \quad (2)$$

V različici s pomnenjem, ki se uporablja v Markovih verigah višjega reda, naslednje stanje ni odvisno samo od trenutnega ampak tudi od prejšnjih. Da to zadostuje Markovemu pogoju, zaporedja stanj združimo v novo stanje Yn.

$$Yn = (Xn-1, ..., X1) \quad (3)$$

$$Pr(Xn = xn | Xn-1 = xn-1..., X1 = x1) \quad (4)$$
$$= Pr(Xn = xn | Yn = yn) \quad (5)$$

Tako je lahko v markovi verigi npr. reda tri, verjetnost izbire naslednjega stanja odvisna od zaporedja prejšnih treh stanj, ki smo jih izbrali na poti do trenutnega stanja. To je še posebej priročno za algoritmično kompozicijo, kjer lahko z analaziranjem skladbe določimo vrednosti teh verjetnosti. Nato lahko algoritem vsakič naključno generira določene parametre glasbe (npr. višino not), še vedno pa se nagiba k temu, da sledi zaporedju tonov, ki so prisotni v vhodni skladbi, ki jo analiziramo. Zaradi teh lastnosti in nezapletene implementacije sem pristop z markovim verigam uporabil v moji rešitvi.

III. Implementacija

A. Uporabljene tehnologije

Rešitev sem implementiral v razvojnem okolju IntelliJ IDEA, v programskem jeziku Java. Za sintetiziranje zvoka sem uporabil Java Sound - nizkonivojski API, ki vsebuje Musical Instrument Digital Interface (MIDI) paket. Ta omogoča uporabo vmesnika za sintetiziranje in predvajanje zvoka z MIDI protokolom. MIDI je protokol, katerega sporočilo ne vsebuje dejanskega zvočnega zapisa ampak metapodatke, ki opisujejo kakšen zvok naj predvaja zvočna naprava, ki ji pošiljamo sporočilo. To pomeni, da je kvaliteta zvoka odvisna od strojne opreme, na kateri poganjamo rešitev, število inštrumentov pa omejeno, brez dodatnih paketov. Za potrebe moje implementacije to ni problematično.

B. Arhitektura

Na diagramu 1 je predstavljena arhitektura mojega programa. Sestoji iz treh razredov. MIDISynthesizer, z vmesnikom javax.sound.midi skrbi za predvajanje akordov in not. Note so objekti razreda Note, ki vsebujejo atribute potrebne za predvajanje zvoka z MIDI protokolom: ton, dolžino, intenziteto (kako močno je zaigrana nota) in barva zvoka. Glavni razred Composer z kontrolnimi spremenljivkami poganja glavne metode algoritma in generira kompozicijo.

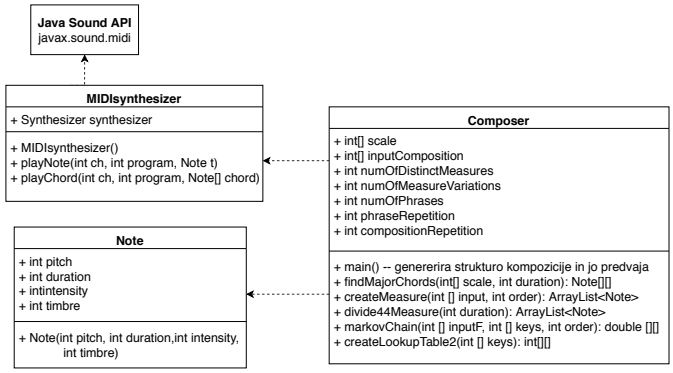

Figure 1. Diagram arhitekture programske rešitve.

C. Metode

1) Predvajanje zvoka: Ko želimo predvajati določeno noto pokličemo v metodo playNote, razreda MIDISynthesizer, ki na enega od 16 kanalov pošlje MIDI sporočilo noteOn, ki predvaja noto z določenim inštrumentom. Po premoru dolžine note, se na kanal pošlje sporočilo noteOff, ki prekine predvajanje. Predvajanje akordov deluje na podbone način le da naenkrat na enem kanalu prižgemo in ugasnemo tri note.

2) Markove matrike in generiranje tonov: V glasbeni teoriji je nota definirana, kot periodični zvok sestavljen iz barve, intezitete, dolžine in višine tona. Ton je odvisen od frekvence: na primer nota A4 je definirana z frekvenco 440Hz. Frekvenčna razdalja med toni imenovana interval, določa razmerja med notami, ki jih moramo upoštevati če želimo, da naša skladba zveni dobro. Pravimo, da so note med seboj konsonančne, ki zaporedoma zvenijo dobro, ali pa so disonančne, ko njihovo zaporedje zveni pokvarjeno. Namesto, da poskušamo v algoritem zapisati pravila o generiranju konsonančnih intervalov not lahko z markovimi verigami preprosto analiziramo že obstoječo skladbo in z računanjem verjetnosti zaporednih not avtomatsko sledimo tem pravilom. Poleg tega bomo tako generirali note ene lestvice (npr. D-dur lestvica not:D, E, F#, G, A, B, C#), kjer je večina not že konsonenčnih.

Algoritmu podamo vhodno skladbo v obliki zaporedja tonov, predstavljenimi s pripadajočimi številkami v MIDI protokolu. Kot primer sem izbral skladbo Marry had a little lamb, katere notni zapis je viden na sliki 2. Odvisno od izbranega reda verige (1 ali 2) bo analiziral zaporedje in izračunal z kakšno verjetnostjo določena nota sledi prejšni

noti (ali paru not). Te verjetnosti so predstavljene v markovi matriki, ki se generira ko pokličemo metodo markovChain. Spodaj je viden primer tabele (I) - če smo v stanju 62,60 je edino možno naslednje stanje 64 z verjetnostjo 1.

	60	62	64	67
64,62	0.6	0.4	0.0	0.0
62,60	0.0	1.0	0.0	0.0
60,62	0.0	0.0	1.0	0.0
62,64	0.0	0.25	0.5	0.25
64,64	0.0	0.4	0.6	0.0
62,62	0.0	0.33	0.66	0.0
67,67	0.0	0.0	1.0	0.0
67,64	0.0	0.0	0.0	1.0
64,67	0.0	1.0	0.0	0.0

Table I
PRIMER MARKOVE MATRIKE 2. REDA ZA VHODNO ZAPOREDJE TONOV SKLADBE MARRY HAD A LITTLE LAMB.

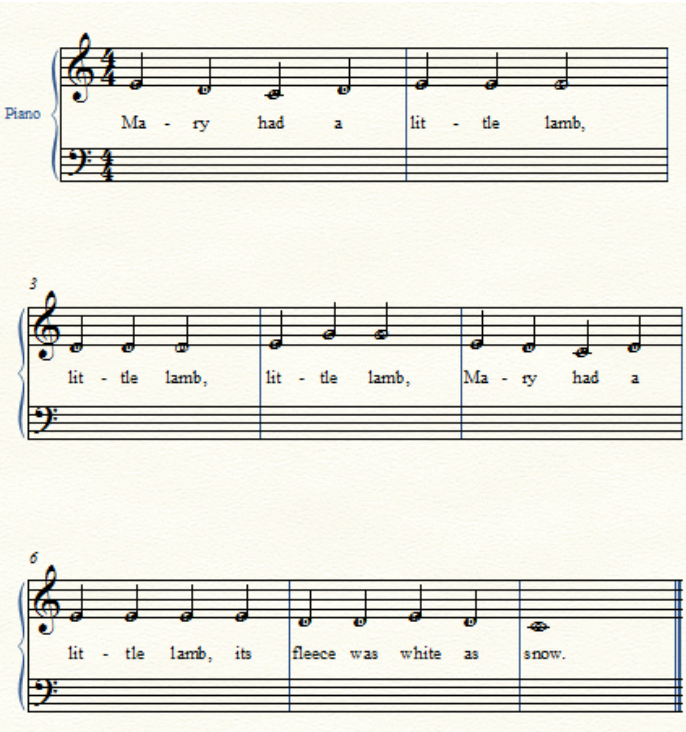

Figure 2. Notni zapis preproste skladbe Marry had a little lamb.

Višji kot je red verige, večje zaporedje not bomo analizirali, večja bo verjetnost, da nota sledi točno določeni noti in bolj bo generirana skladba podobna vhodni. Odločil sem se, da je za začetno analizo primerna veriga reda 2, saj imam za vhodni primer preproste, krajše skladbe, ki vsebujejo veliko ponavljanja.

3) Deljene dob takta: Poleg generiranja višin tonov, je potrebno določiti tudi čas trajanja note - njeno dolžino. V notnih zapisih pogosto zasledimo 4/4 takt, ki nam pove, da mora del skladbe med dvemi navpičnimi črtami vsebovati štiri enočetrtinske dobe. Vsaka doba je sestavljena iz not določenih dolžin (1, 1/2, 1/4, 1/8, ...), ki se lahko nato deli še naprej ali pa združi z novimi dobami. Veljati mora pogoj, da skupaj trajajo 4/4.

Z metodo divide44Measure v algoritmu določim dolžino not enega takta. Z vnaprej določeno verjetnostjo rekurzivno delimo takt na dve polovici dokler ne pridemo do dolžine 1/8 ali pa prej odnehamo. Verjetnosti sem določil da bo doba 1/4 najpogostejša medtem ko bosta 1/2 in 1/8 manj pogosti. Podobno velja za večino obstoječih skladb.

4) Generiranje strukture skladbe: Melodija je posledica kombinacij dolžin in višin zaporednih tonov. Če želimo, da je melodija tudi smiselna mora vsebovati neko strukturo. V glasbi poznamo več nivojev struktur: kratka zaporedja tonov enega takta so organizirana v motive, ki nato sestavljajo fraze. Ponavljajoče fraze nato sestavljajo stavke, itd. V glasbeni teoriji so strukture zaporedij točno določene, vsaka se uporablja za različne namene v različnih glasbenih obdobjih. Poznamo na primer binarno strukturo (zaporedja ABAB ali AA'BB'AA'BB', kjer ' označuje majhno variacijo enote), terniarno (ABA), ter druge.

Algoritem sem napisal tako, da celotno skladbo gradi iz ponavljajočih enot. Osnovno enota so motivi, ki jih generira metoda createMeasure. Ta z klicem metode divide44Measure, pridobi število not v taktu in njihove dolžine ter generira pripadajoče tone z markovo matriko drugega reda, ki jo pridobi od metode markovChain za dani vhod. Tako generira določeno število motivov in (če želimo) še njihove variacije - isti postopek požene z vhodom, ki je kar trenutni motiv in markovimi matrikami reda 1 (nižji red ker je vhodno zaporedje not kratko).

V metodi main nato motive in njihove variacije v binarni strukturi dodajamo zaporedja, ki sestavljajo fraze. V frazi lahko določimo tudi kolikokrat se bo ponovil določen motiv.

Algoritem nato iz dveh fraz A in B, sestavi končno melodijo kompozicije v terniarni obliki ABA. Končna dolžina in oblika melodije je odvisna od kontrolnih spremenljivk, ki določajo ponovitve, število motivov in njihovih variacij.

5) Harmonija: Če melodija predstavlja vodoravno dimenzijo skladbe, harmonija predstavlja vertikalo. Z dodajanjem različnih akordov v poslušalcu vzbudi različna čustva in občutke. Akord je največkrat zaporedje treh not, predstavljen z rimsko število od I do VII. Prva nota je imenovana koren in po njej je poimenovan akord - če izberemo prvo noto lestvice je to potem akord I. Druga nota akorda je od prve oddaljena za 2 tona lestvice od prve note. Tretja nota akorda je od prve oddaljena za 5 tonov lestvice. Vsak akord ima določen občutek in sledi drugim akordom z točno določenimi pravili.

Generirano melodijo mojega algoritma harmoniziram z preprostim dodajanjem akordov. Z metodo findMajorChords v podani lestvici poiščemo I, IV in V akord, saj so te v glasbi najbolj pogosti in vsebuje vse note lestvice. Med predvajanjem melodije v main metodi, glede na začetno noto vsakega takta dodamo primeren akord. Akord je primeren če je ena od njegovih treh not enaka izbrani noti, ki jo želimo harmonizirati. Pri tem upoštevamo pravilo, da lahko akordu I ali IV sledi katerikoli drug akord, akordu V pa mora slediti I, saj napetost

narašča z višjimi akordi in v poslušalcu vzbudi potrebo po zaključku.

IV. Rezultati

A. Vrednotenje rešitve

Skladba, ki jo generira algoritem je preprosta, vendar zveni pravilno, brez disonančnih not ali napačnih akordov, čeprav vsebuje nekaj ponavljajočih odsekov. Harmonizacija je preprosta z le enim akordom na takt, vendar si ti ustrezno sledijo, višajo ali razrešujejo napetost ter tako poglobijo občutek skladbe. Lahko je dolga le nekaj not, če izberemo malo število motivov brez variacij in ponavljanj, ali pa nekaj minut z izbiro večih motivov in variacij. Veljavne vhodne skladbe so enozvočne melodije, primerne so še posebej odseki namenjeni solo inštrumentu.

Pri daljših skladbah se opazijo pomanjklivosti algoritma. V primerjavi z drugimi, večjimi algoritmičnimi kompozicijami, kot je Abundant Music[4] in cgMusic[5], so skladbe preprostejše in ponavljajoče. Še posebej se to opazi pri harmoniji akordov, ki je precej statična in vsebuje le tri ponavljajoče akorde. Poleg tega moja rešitev generira le homofonije medtem, ko je Abundant Music zmožen generiranja kompleksnih polifoničnih melodij z frazami, ki imajo definiran začetek, jedro in zaključek. Moje fraze so nasprotno sestavljene iz naključnih motivov, ki ne zagotavljajo podobno zgradbo. Poleg tega ima moj algoritem le dva 2 nivoje strukture: motive in fraze, medtem ko imajo druge rešitve še stavke, refrene in druge strukture. Vsebujejo tudi večje število parametrov s katerimi je možno kontrolirati kompozicijo, na primer tempo, intenziteto, glasnost melodije in druge efekte.

Nepresenetljivo je, da je moja rešitev manj fleksibilna in obsežna kot ostale večje produkcije, saj sem jo razvijal sam le nekaj tednov z omejenim znanjem glasbene teorije. Rešitev ustreza prvotno zastavljenemu cilju, generiranja preproste smiselne melodije. V tem vidiku je boljša od preprostih rekombinacijski algoritmov, kot so glasbene igre[3], ki za delovanje zahtevajo primerno napisane odseke glasbe. Podobnejša je klasičnim, bolj smiselnim skladbam kot tiste, ki so generirane s pomočjo fraktolov ali simulacij[7], saj upošteva ustaljena glasbena pravila.

B. Vrednotenje algoritmov

Končni rezultat rešitve je odvisen od uporabljenih algoritmov metod. Celotni algoritem je hiter in generira skladbo dolgo okoli 200 not v približno petih sekundah. Daljših skladb ni smiselno generirati, saj se preveč ponavljajo.

Metoda markovih verig, ki generira tone not, deluje pričakovano dobro. Ohranja konsonačno zaporedje not originalne skladbe, hkrati pa generira nove kombinacije. Slaba stran metode je, da težko vplivamo na generirane struktire in težko dodamo željeno zgradbo, ki ni prisotna v vhodni skladbi. V tem vidiku so bolj deterministični algoritmi kot so L-sistemi[8] primernejši.

Metoda deljenja dob takta deluje popolnoma naključno, brez pomnenja zgodovine in zato ne prispeva k večji strukturi skladbe. Čeprav je ritem eden najpomembnejših vidikov kompozicije, ga je težko algoritmično definirati, saj poleg dolžine takta ne vsebuje drugih natančnih pravil. Zato tudi ostale rešitve nimajo dobro določenega ritma - pri nekaterih je preprost [8], pri drugih je konstanten[7], nekatere pa ga poskusijo zakriti z drugimi vidiki skladbe[4].

Generiranje končne strukture kompozicije iz fraz in dodajanje akordov predstavlja največjo oviro razgibanosti skladb. Ker obe metodi delujeta deterministično (struktura skladbe bo vedno ABA, akordi bodo vedno harmonizirali le prvo noto) so si generirane skladbe podobne med seboj.

C. Eksperiment

Rešitev sem preizkusil z bolj kompleksno vhodno skladbo Concerning Hobbits, ki vsebuje več not, višje oktave in drugo lestvico kot Marry had a little lamb (slika 3). Generirana skladba je razmeroma bolj kompleksna in vsebuje več smiselnih odsekov not, ki si sledijo v konjunktivnem zaporedju (postopoma naraščajo ali padajo). Z drugim inštrumentom in verjetnostjo deljenja dob se skladba razlikuje od prejšnih, vendar ne bistveno, predvsem zaradi harmonizacije, ki ostaja ista kot prej. Poleg tega algoritem za vse note višjih oktav določi I akord, ker generira akorde le ene oktave in posledično ne najde dugega primernega akorda, kot I, ki lahko sledi kadarkoli.

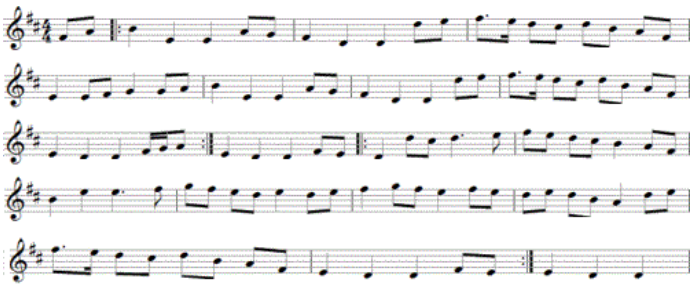

Figure 3. Notni zapis skladbe Concerning Hobbits (Howard Shore).

V. Zaključek

Predstavljena rešitev z algoritmično kompozicijo generira preprosto homofonično skladbo odvisno od vhodne, analizirane skladbe in kontrolnih spremenljivk. Algoritem upošteva glavna pravila glasbene teorije in kompzicije, da skupaj z matematičnim modelom markovih verig, doseže zadostno smiselnost izhodne skladbe. Rešitev bi bila primerna za generiranje originalnih kratkih skladb za preproste igre ali druge projekte, katerih razvijalci niso dovolj glasbeno vešči, da bi glasbeno spremljavo napisali sami.

Rešitev bi bilo smiselno razširiti z grafičnim vmesnikom in dodelati metode za generiranje strukture melodije ter akordov, tako da bi bile zmožne večih variacij. Povečali bi lahko na primer število možnih akordov ter naslednike izbrali na podlagi večih sosednjih not. Markove verige bi lahko uporabili še pri drugih vidikih skladbe, na primer analiziranju dolžin not in akordov. Rešitev bi lahko razširili tudi na polifonične skladbe.

LITERATURA

[1] Wikipedia, "Algorithmic composition," Dosegljivo: https://en.wikipedia.org/wiki/Algorithmic_composition, [Dostopano: 4. 11. 2018].

[2] M. Edwards, "Algorithmic composition: Computational thinking in music," *Commun. ACM*, vol. 54, no. 7, pp. 58–67, Jul. 2011. [Online]. Available: http://doi.acm.org/10.1145/1965724.1965742

[3] S. A. HEDGES, "Dice music in the eighteenth century," *Music and Letters*, vol. 59, no. 2, pp. 180–187, 1978. [Online]. Available: http://dx.doi.org/10.1093/ml/59.2.180

[4] P. Nyblom, "Abundant music," Dosegljivo: https://pernyblom.github.io/abundant-music/index.html, [Dostopano: 4. 11. 2018].

[5] M. Biedrzycki, "cgmusic - can computers create music," Dosegljivo: http://maciej.codeminion.com/2008/05/cgmusic-computers-create-music/, [Dostopano: 4. 11. 2018].

[6] S. Jarrett and H. Day, *Music Composition For Dummies*, ser. For Dummies. Wiley, 2008. [Online]. Available: https://books.google.si/books?id=z2pvq-XXJpwC

[7] D. Lockerby, "The molecular music box: Simple rules can create rich patterns," Dosegljivo: http://www.synthtopia.com/content/2014/12/22/the-molecular-music-box-how-simple-rules-can-lead-to-rich-patterns/, [Dostopano: 4. 12. 2018].

[8] P. Langston, "Six techniques for algorithmic music composition," in *Proceedings of the International Computer Music Conference*, vol. 60. Citeseer, 1989.

[9] A. B. Andrew Sorensen, "Jmusic," Dosegljivo: http://explodingart.com/jmusic/, [Dostopano: 14. 1. 2019].

[10] I. Xenakis, *Formalized music : thought and mathematics in composition / Iannis Xenakis*, rev. ed. ed. Pendragon Press Stuyvesant, NY, 1992.