SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Skladanje glazbe korištenjem genetskog algoritma

Marko Đurasević

Voditelj: Doc. dr. sc. Domagoj Jakobović

Zagreb, travanj, 2011.

Sadržaj

1. Uvod

Tijekom povijesti, ljudi su stalno pokušavali riješiti probleme koji su ih okruživali. Sa samim razvojem civilizacije, problemi koje su ljudi pokušavali riješiti postajali su sve složeniji i napredniji. Rješavanje takvih problema uskoro je postalo prezahtjevno, štoviše, čak i nemoguće za ljude. Međutim, u današnje vrijeme s iznimno brzim razvojem računala, moguće je riješiti vrlo složene probleme, koje je bilo nemoguće riješiti prije samo pedesetak godina. No bez obzira na stalna poboljšanja računala i danas postoji velik broj problema koji su presloženi da bi ih se točno riješilo u razumnom vremenskom periodu i pomoću računala. Takvi problemi se rješavaju uz pomoć takozvanih heuristika (grč. Heuriskein – umijeće pronalaska novih strategija (pravila) za rješavanje problema), algoritama koji ne garantiraju optimalno rješenje nekog problema, ali pronalaze zadovoljavajuća rješenja u razumnom vremenskom razdoblju. Genetski algoritam, uz mnoge druge heurističke metode (npr. genetsko programiranje, evolucijsko programiranje, evolucijsku strategiju) spada u podgranu heurističkih metoda koja se naziva evolucijski algoritam.

Iako taj algoritam poglavito služi za rješavanje problema optimizacije i pretraživanja, genetski algoritam se može upotrijebiti i u umjetničke svrhe. Tako primjerice, možemo koristiti genetski algoritam u svrhu komponiranja glazbenih djela. Iako korištenje genetskog algoritma u tu specifičnu svrhu nema neku praktičnu primjenu, ljudska znatiželja, koja je kroz povijest motivirala ljude da isprobaju različite metode i načine komponiranja glazbe, je također inspirirala ljude da primjene genetske algoritme i u svrhu skladanja glazbe. Iako ovaj način komponiranja glazbe nije još jako poznat niti popularan, posljednjih godina se sve više proučavaju metode pomoću kojih se mogu komponirati što kvalitetnija i uhu ugodnija glazbena djela[1][2][3]. Za pretpostaviti je da će se daljnjim razvojem i optimiranjem genetskih algoritama, također razvijati i njihova primjena u različitim granama umjetnosti poput ove.

Cilj ovog rada je dati kratki uvod u osnove genetskog algoritma, kao i objasniti osnovne pojmove vezane uz genetski algoritam (mutacija, križanje, populacija, itd.), opisati problem komponiranja glazbe korištenjem genetskog algoritma te prikazati jednostavno programsko rješenje koje korištenjem genetskog algoritma komponira donekle jednostavnu monofonu melodiju.

2. Genetski algoritam

Ocem genetskih algoritama smatra se John H. Holland, koji ih je predložio u ranim sedamdesetima. Spomenuti genetski algoritam je heuristička metoda koja imitira proces prirodne evolucije. Nad populacijom potencijalnih rješenja provode se različite operacije genetskog algoritma (mutacija, križanje i selekcija), pomoću kojih se genetski materijal pojedinih jedinki populacije mijenja što rezultira evolucijom same populacije. Nakon što se ispuni uvjet prekida algoritma, na temelju trenutne populacije, traži se jedinka koja ima najveći faktor dobrote te upravo ona predstavlja najbolje rješenje nekog problema u zadanoj populaciji (ne nužno i općenito najbolje rješenje). Genetski algoritam se koristi za pronalazak točnih ili približno točnih rješenja, prvenstveno za probleme optimizacije i pretraživanja. U posljednja tri desetljeća, genetski algoritmi su se pokazali kao veoma moćan alat za rješavanje mnogih inženjerskih problema.

2.1 Prikaz rješenja

U genetskom algoritmu, rješenje problema, je prikazano pomoću jednog kromosoma. Kromosom, u ovom slučaju, može biti bilo kakva struktura podataka koja opisuje svojstva jedne jedinke. Važno je pritom da kromosom predstavlja moguće rješenje nekog problema i da primjenom genetskih operatora ne dolazi do stvaranja novih kromosoma; koji predstavljaju nemoguća rješenja. Također je vrlo bitan način koji će se koristiti za prikaz rješenja u genetskom algoritmu, budući da može uvelike utjecati na složenost i učinkovitost genetskog algoritma.

Danas se za prikaz rješenja najčešće koristi binarni prikaz, u kojem je svaki kromosom predstavljen nizom od n bitova. Duljina binarnog niza definira broj rješenja koja se mogu prikazati. U ovom načinu, kromosom je prikazan kao vektor binarnih brojeva. Ovaj način prikaza rješenja omogućuje vrlo jednostavnu implementaciju genetskih operatora križanja i mutacije, zbog toga što su svi kromosomi u ovom prikazu nužno jednake duljine.

Slika 2.1. Primjer zapisa kromosoma binarnim kodom

Osim binarnog prikaza rješenja, postoji još mnogo drugih načina pomoću kojih se kromosomi mogu kodirati. Tako je za prikaz kromosoma moguće koristiti i nizove cijelih brojeva, nizove znakove ili slično.

2.2 Inicijalizacija početne populacije

Populaciju potencijalnih rješenja predstavlja određeni broj kromosoma koji se generira na temelju zadane veličine populacije. Veličina populacije je najčešće konstantna tijekom čitavog izvođenja programa, no može se zadati da se veličina populacije mijenja kroz generacije. Veličina populacije će ovisiti o prirodi konkretnog problema, no najčešće se sastoji od nekoliko stotinjaka ili tisuća kromosoma. Ona se najčešće generira sasvim slučajno, tako da jedinke poprimaju vrijednosti iz skupa svih mogućih rješenja. Međutim, moguće je i modificirati inicijalizaciju tako da kromosomi ne poprimaju vrijednosti iz cijelog skupa mogućih rješenja, već samo iz određenih podskupova u kojima je pojava optimalnog rješenja najvjerojatnija.

2.3 Funkcija dobrote

Zadaća funkcije dobrote je ocjenjivanje dobrote pojedinog rješenja (kromosoma). Dobrota (eng. *fitness*) pojedinog rješenja je vrijednost koja određuje koliko je neko rješenje kvalitetno. Na temelju dobrote, jedinke se mogu međusobno uspoređivati, te se tako mogu odrediti koje su jedinke bolje od drugih, što nam je od iznimne važnosti u selekciji samih jedinki. Pri završetku algoritma, traži se jedinka koja ima najveću dobrotu, koja onda predstavlja optimalno rješenje nekog problema iz zadane populacije. Funkcija dobrote je vrlo važan dio genetskog algoritma i zbog toga je bitno što bolje definirati funkciju dobrote za pojedine probleme, jer će o funkciji dobrote uvelike ovisiti i sama kvaliteta rješenja. Nažalost, za mnoge probleme, definiranje funkcije dobrote nije baš sasvim trivijalno.

2.4 Selekcija

Selekcija je proces kojim se osigurava da se dobra svojstva jedinki očuvaju i prenesu na novu generaciju jedinki, dok loša svojstva lagano odumiru i nestaju. Selekcijom se odabiru dobre jedinke koje će se međusobno križati i na taj način svoj genetski materijal prenijeti na iduću generaciju. S obzirom na vrstu selekcije, genetske algoritme dijelimo na generacijske i eliminacijske.

Generacijske selekcije odabiru određen broj najboljih jedinki iz trenutne populacije i od njih stvore novu populaciju koja ima jednak broj jedinki kao i populacija od koje je nastala. Nedostatak jedinki nadoknađuje se tako da se neke jedinke koje su bile odabrane da tvore novu populaciju pojavljuju više puta. Odabir jedinki od kojih se stvara nova populacije temelji se na vrijednostima dobrote jedinki. Veliki nedostatak ove selekcije je upravo spomenuto višestruko pojavljivanje jednakih jedinki u novoj populaciji, što umanjuje genetsku raznolikost populacije i može dovesti do zagušenja populacije, što posljedično usporava algoritam.

Eliminacijske selekcije odabiru određen broj loših kromosoma koji se eliminiraju i zamijene novim kromosomima koji su nastali primjenom genetskih operatora nad dobrim kromosomima. U ovoj selekciji, vjerojatnost pojave jednakih kromosoma u novoj populaciji je puno manja, zbog čega ova selekcija, osim što joj je

vrijeme izvođenja dosta brzo,obično daje puno bolje razultate nego što ih daje generacijska selekcija.

2.5 Genetski operatori

2.5.1 Križanje

Križanje je genetski operator u kojem sudjeluju dvije jedinke, koje se nazivaju roditelji, i u kojem nastaje jedna ili dvije nove jedinke koje pak nazivamo djeca. Bitno je da djeca nastaju kombiniranjem svojstava obaju roditelja. Ako su roditelji bili dobri, vrlo je vjerojatno da će i dijete biti dobro, ako ne i bolje od roditelja, no moguće je i suprotno. Vrsta križanja koja se primjenjuje ovisi o načinu prikaza rješenja. Neka od najčešćih križanja su: križanje s jednom točkom prekida, križanje s dvije ili više točaka prekida i uniformno križanje. Križanje s jednom ili dvije točke prekida se koristi kod velikih populacija i kromosoma većih duljina dok se uniformno križanje koristi prvenstveno kod manjih populacija.

2.5.2 Mutacija

Mutacija je genetski operator koji djeluje nad jednom jedinkom i to tako da promijeni jedan ili više bitova te jedinke. Najjednostavniji oblik mutacije je promjena jednog bita unutar kromosoma. Ostali oblici mutacije su: miješajuća mutacija, invertirajuća mutacija i potpuna miješajuća mutacija. Učestalost mutacije će ovisiti o samoj vjerojatnosti mutacije (koja je najčešće jedan od ulaznih parametara genetskog algoritma), te obično poprima vrijednost između 0.001 i 0.01. Ukoliko vjerojatnost mutacije teži jedinici, genetski algoritam se pretvara u algoritam slučajne pretrage prostora rješenja, dok ukoliko vjerojatnost mutacije teži prema nuli, postupak optimiranja će najvjerojatnije zastati u nekom od lokalnih optimuma. Važna uloga mutacija je u obnavljanju izgubljenog genetskog materijala i izbjegavanju lokalnih optimuma.

2.6 Uvjet zaustavljanja

Uvjet zaustavljanja određuje kada će se rad genetskog algoritma zaustaviti. Najčešće se kao uvjet zaustavljanja algoritma zadaje određen broj iteracija kroz koje algoritam mora proći. Drugi mogući uvjeti zaustavljanja mogu biti zadani vrijednošću funkcije dobrote koju jedna od jedinki mora dostići, brojem iteracija u kojima populacija nije napredovala, vremensko ograničenje i slično.

2.7 Parametri genetskog algoritma

Svaki genetski algoritam ima sljedeće parametre: veličina populacije, broj iteracija (generacija) i vjerojatnost mutacije. Ovisno o vrsti genetskog algoritma i prirodi problema koji se rješava, genetski algoritam može imati još neke dodatne parametre. Konačni rezultati izvođenja će uvelike ovisiti o samim parametrima genetskog algoritma. Međutim, veliki je problem kako za pojedini problem odrediti

idealne parametre za izvođenje algoritma. Optimiranje parametara može se postići kroz izvođenje velikog broja eksperimenata i uspoređivanjem rezultata koji su dobiveni za pojedine parametre. Ponekad se i sami genetski algoritmi primjenjuju kao metoda traženja idealnih parametara.

2.8 Prednosti i mane genetskih algoritama

Velika prednost genetskog algoritma je ta što je primjenjiv na širok skup problema, a ne samo na određene probleme kao neki drugi algoritmi. Genetski algoritmi nude veliki stupanj slobode i nadogradnje koji mogu povećati efikasnost samog algoritma. Rezultat genetskog algoritma ne mora biti nužno samo jedno rješenje, već skup rješenja. Sam algoritam možda neće biti u stanju naći optimalno rješenje nekog problema, ali može dati neko dovoljno dobro rješenje problema. Nadalje, lako je primjenjiv na višedimenzijske probleme.

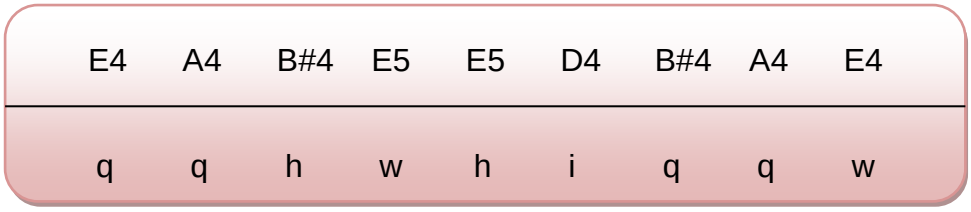
Iako je genetski algoritam primjenjiv na širok skup problema, često se ti problemi trebaju modificirati i prilagoditi kako bi se genetski algoritmi mogli primijeniti na njih. Parametri također imaju jako veliki utjecaj na izvođenje genetskih algoritama pa je zbog toga potrebno odrediti idealne parametre za pojedini problem, što vrlo često nije trivijalno i potrebno je mnogo testiranja i uspoređivanja rezultata da se oni odrede. Uz to, genetski algoritmi nemaju 100% učinkovitost te je njihovo izvođenje veoma sporo.

3. Primjena genetskog algoritma na skladanje glazbe

Prvi objavljeni članak koji se bavi tematikom komponiranja glazbe uz pomoć genetskih algoritama objavljen je 1991. godine[3]. Iako su prošla već dva desetljeća od tada, broj članaka na ovu tematiku, kao i broj programskih rješenja je još uvijek vrlo skroman. Rezultat genetskih algoritama su, najčešće, kratke monofone (jednoglasne) melodije. U nastavku je opisan jedan način implementacije genetskog algoritma koji komponira jednu takvu jednostavnu monofonu melodiju na temelju proizvoljnih ulaznih parametara.

3.1 Prikaz rješenja

Rješenje ovog problema, podrazumijeva se, bit će niz tonova određenih trajanja koji će predstavljati melodiju. Jedinka koja predstavlja rješenje problema, u ovoj implementaciji biti će prikazana pomoću podatkovne strukture, koja sadrži oznaku dobrote, duljine melodije (broj tonova) i dvije povezane liste cijelih brojeva. Oznaka dobrote se računa pomoću funkcije dobrote te nam služi za međusobno uspoređivanje jedinki. Oznaka duljine predstavlja broj tonova u jedinki i jedan je od ulaznih parametara algoritma. Prva lista sadrži cijele brojeve koji predstavljaju adresu elemenata u polju tonova. Polje tonova je niz znakova koji predstavljaju tonove od tona A4 do tona G#5. Druga lista, također sadrži cijele brojeve koji predstavljaju adresu elementa u polju trajanja. Podržana trajanja su: cijela nota (w), polovinka (h), četvrtinka (q) i osminka (i).



Slika 3.1. Primjer jedinke (gornji dio pretstavlja listu tonova, a donji listu trajanja)

3.2 Inicijalizacija populacije

Populacija se generira nasumično tako da se liste jedinke ispune slučajno generiranim cijelim brojevima iz definiranih intervala. Pri generiranju slučajnog niza tonova, cijeli tonovi imaju tri puta veću vjerojatnost da se pojave nego povišeni, odnosno sniženi tonovi, dok se kod generiranja slučajnog niza trajanja u većoj mjeri preferiraju četvrtinke i polovinke, a u manjoj mjeri osminke i cijeli ton. Veličina populacije zadaje se prilikom pokretanja algoritma i mora se sastojati od minimalno tri jedinke.

3.3 Funkcija dobrote

Najveći utjecaj na kvalitetu dobivene melodije ima upravo funkcija dobrote. Funkcija dobrote na temelju intervala susjednih tonova određuje njihovu međusobnu

dobrotu, a ukupna dobrota jedinke se određuje kao zbroj dobrota svih intervala u jedinki. Iz tog se može zaključiti da će dobrota čitave jedinke ovisiti isključivo o intervalima susjednih tonova. Stoga je dovoljno odrediti dobrotu pojedinih intervala i na temelju njih vrlo se jednostavno određuje ukupna dobrota jedinke. Nažalost, određivanje dobrote intervala potpuno je subjektivno te je stoga nemoguće univerzalno odrediti vrijednosti tih vrijednosti. Tablica 3.1 prikazuje samo jednu mogućnost odabira dobrote intervala. Disonantni intervali imaju manju dobrotu, dok intervali veći od oktave imaju dobrotu jednaku nuli, iz razloga da se smanji broj intervala koji su veći od jedne oktave.

Tablica 3.1. Dobrote intervala

Interval	Dobrota
Prima	30
Mala sekunda	20
Velika sekunda	80
Mala terca	40
Velika terca	90
Kvarta	100
Povećana kvarta (tritonus)	10
Kvinta	130
Mala seksta	70
Velika seksta	120
Mala septima	60
Velika septima	40
Oktava	110
Iznad oktave	0

3.4 Selekcija

Selekcija koja se koristi u ovom algoritmu je troturnirska eliminacijska selekcija (engl. *steady state tournament*). Iz populacije se nasumično odaberu tri jedinke. Od

tih triju jedinki odredi se jedinka koja ima najmanju dobrotu te se ona potom izbriše. Izbrisana jedinka nadomještava se novom jedinkom koja nastaje primjenom genetskog operatora križanja nad preostale dvije jedinke. Nakon toga se nad novom jedinkom, koja sadrži kombinaciju genetskog materijala obaju roditelja, primjenjuje genetski operator mutacije koji može dodatno promijeniti genetski izgled jedinke. Nakon toga se naposljetku, uz pomoć funkcije dobrote odredi dobrota novonastale jedinke. Proces se ponavlja dok se ne zadovolji uvjet zaustavljanja koji je u ovom algoritmu zadan samim brojem iteracija algoritma.

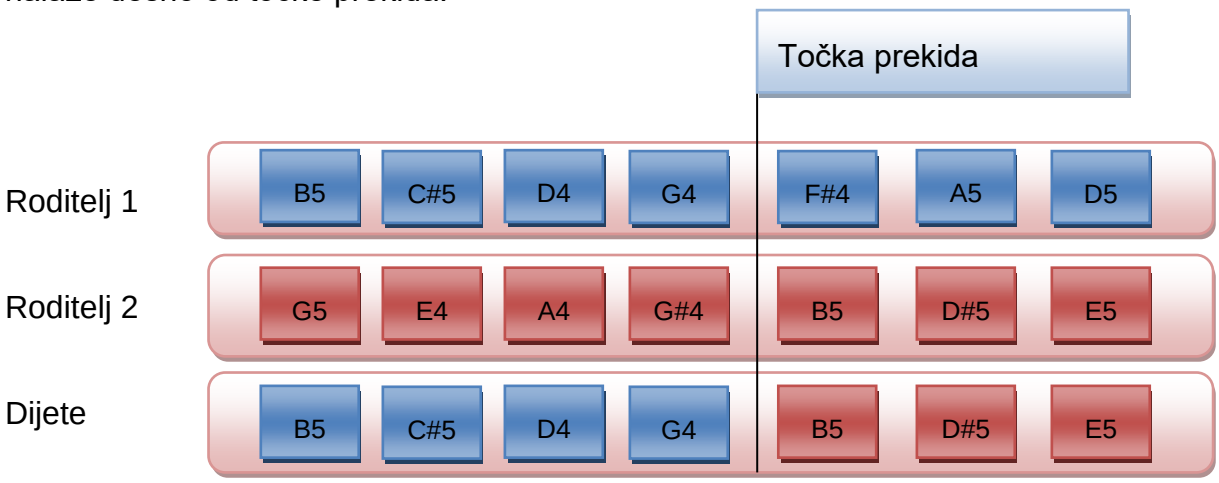
3.5 Križanje

Križanje se obavlja nad dvjema jedinkama koje su odabrane u selekciji i imaju najveću dobrotu. Pošto su sve jedinke jednake duljine, križanje je vrlo lako za implementirati. Bitno je napomenuti da se križanje obavlja samo nad listom tonova, no ne i nad listom trajanja. Nakon obavljanja operacije križanja računa se dobrota djeteta.

3.5.1 Križanje s jednom točkom prekida

Za veće populacije i veće jedinke implementirano je križanje s jednom točkom prekida. Točka prekida odabire se slučajno iz intervala [0, duljina-1], gdje je duljina broj tonova u jedinki. Nadalje, slučajnim odabirom određuje se od kojeg će se roditelja uzeti dio jedinke koji se nalazi lijevo od točke prekida, a od kojeg roditelja dio jedinke desno od točke prekida. Konačno dobiva se nova jedinka koja ima potencijal da bude bolja od oba roditelja.

Slika 3.2 prikazuje primjer ovakvog križanja. Jedinka dijete je prva četiri tona, odnosno one tonove koji se nalaze lijevo od točke prekida, naslijedila od prvog roditelja, dok je od drugog roditelja naslijedila ostale tonove, odnosno one koji se nalaze desno od točke prekida.

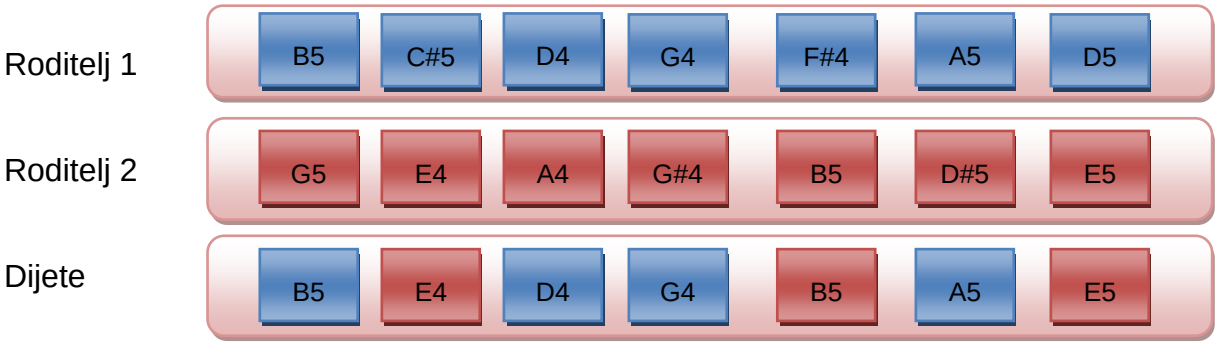


Slika 3.2. Križanje s jednom točkom prekida

3.5.2 Uniformno križanje

Ovo križanje koristi se pretežito kod manjih populacija. Križanje funkcionira tako da se za svaki gen (u ovom slučaju ton), slučajno odabere hoće li se on naslijediti od jednog ili drugog roditelja. Vjerojatnost nasljeđivanja od jednog, odnosno drugog roditelja je potpuno jednaka.

Slika 3.3. prikazuje primjer uniformnog križanja. U ovom slučaju, dijete je prvi, treći, četvrti i šesti ton naslijedilo od prvog roditelja, dok je od drugog roditelja naslijedilo drugi, peti i sedmi ton. Kao što se vidi, nema nikakvog pravila koje određuje će se ton naslijediti od kojeg roditelja.



Slika 3.3. Unirormno križanje

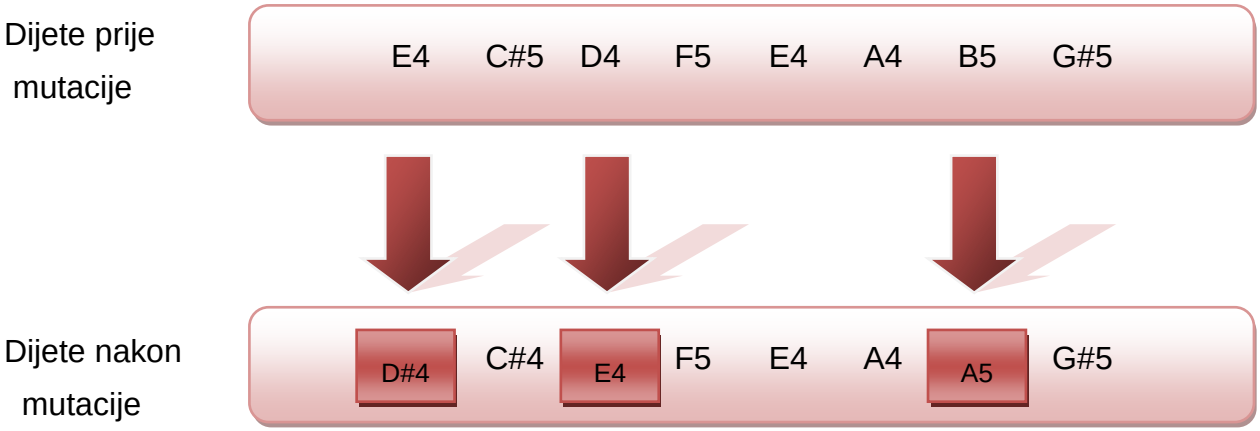
3.6 Mutacija

Nakon što se križanjem dviju jedinki dobije nova jedinka, nad njom se primjenjuje genetski operator mutacije. Na temelju zadane vrijednosti vjerojatnosti mutacije, za svaki gen jedinke ispituje se da li će se taj odabrani gen mutirati ili ne. Za ovaj konkretan problem, moguće je implementirati veliki broj različitih mutacija. U ovoj implementaciji korištenoj za gornji primjer su odabrane konkretno dvije mutacije: promjena tona za stepen/polustepen te zamjena mjesta dvaju susjednih tonova.

3.6.1 Promjena tona za stepen/polustepen

Ova vrsta mutacije prolazi kroz svaki ton jedinke i na osnovu zadane vjerojatnosti mutacije odlučuje hoće li pojedini ton promijeniti za polustepen ili stepen gore, odnosno dolje. Odluka o tome da li će se ton povećati ili sniziti, donosi se slučajno i obje mutacije mogu se dogoditi s jednakom vjerojatnošću. Međutim, kod odluke hoće li se ton promijeniti za polustepen ili stepen, vjerojatnosti nisu jednake. Veću vjerojatnost ima mutacija koja će ton promijeniti za stepen, i to tri puta veću vjerojatnost, nego što je vjerojatnost mutacije koja će ton promijeniti za polustepen. Uz pomoć ove mutacije se održava svojevrsna genetička raznolikost jedinki jer se nakon svakog križanja mogu pojaviti neki novi tonovi u jedinki, koji se možda nisu mogli naslijediti od roditelja te se tako sprječava da algoritam prebrzo završi u nekom globalnom optimumu. Također se može dogoditi da mutiranjem tona dođe i do poboljšanja intervala između tog tona i njemu susjednih tonova te se na taj način može i dodatno poboljšati ukupna dobrota jedinke. Zbog toga je potrebno nakon mutacije, korištenjem funkcije dobrote, ponovo odrediti ukupnu dobrotu jedinke.

Slika 3.4 prikazuje primjer takve mutacije. Na primjeru se vidi da su mutirali prvi, treći i sedmi ton. Prvi ton je mutirao za jedan polustepen prema dolje, treći ton je mutirao za jedan stepen gore, dok je sedmi ton mutirao za jedan stepen prema dolje. Također, vidljivo je, da je mutacijom došlo do pojave čak dva nova tona u jedinki (D#4 i A5). Uz to su se promijenili i pojedini intervali između tonova, u ovom slučaju intervali između prvog i drugog tona, drugog i trećeg tona, trećeg i četvrtog tona, šestog i sedmog tona, kao i sedmog i osmog tona. Kao što se vidi, odabir mutacija i tonova je sasvim proizvoljan. Ukoliko se mutira ton čijom bi se mutacijom prešlo preko raspona dozvoljenih tonova, tada se ton dobiven mutacijom određuje tako da se ciklički, na temelju liste tonova, odredi novi ton u koji se prelazi mutacijom.

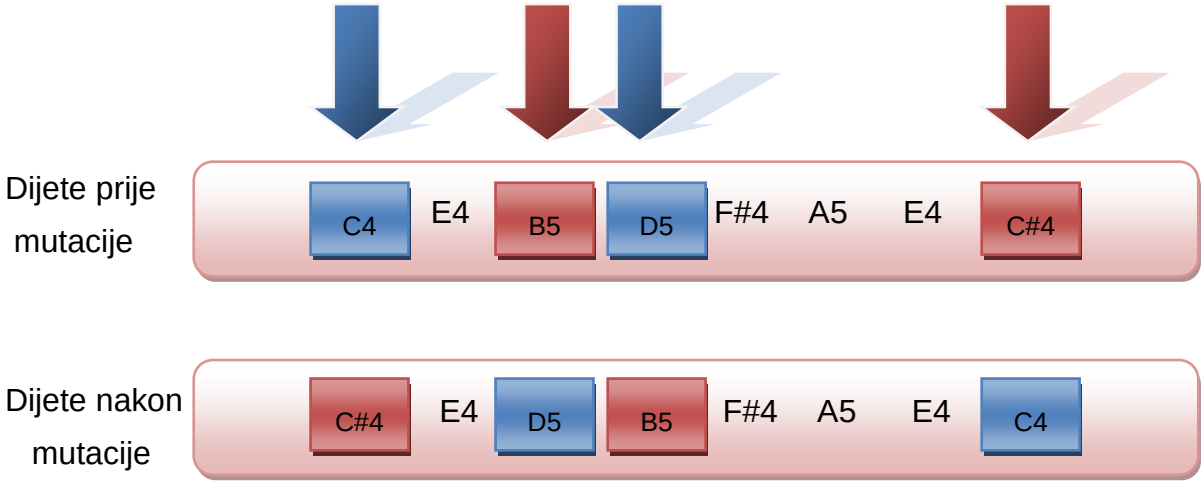


Slika 3.4. Mutacija tonova za stepen/polustepen

3.6.2 Zamjena mjesta dvaju susjednih tonova

U ovoj mutaciji se također prolazi kroz svaki ton jedinke te se za zadanu vjerojatnost odlučuje hoće li se odabrani ton i njemu sljedeći ton zamijeniti za mjesta. U ovoj mutaciji ne dolazi do pojave novih tonova u djetetu nakon što je ono nastalo križanjem. Pomoću ove mutacije može se jedino pokušati poboljšati intervale između tonova koji su susjedni onim tonovima koji su se međusobno zamijenili za mjesta, pošto se tada dobivaju novi intervali između zamijenjenih tonova i njima susjednim tonovima, koji mogu imati veću dobrotu nego što su ju imali prethodni intervali i na taj način povećati ukupnu dobrotu jedinke. Kao i u prethodnoj mutaciji, nakon njenog obavljanja mora se ponovo izračunati ukupna dobrota jedinke.

Slika 3.5. Prikazuje primjer upravo ovakve mutacije. Crvene strjelice označavaju tonove koji su odabrani za mutaciju, dok plave strjelice označavaju tonove koji će zamijeniti mjesta s tim tonovima. Kao što vidimo, za mutaciju su odabrani treći i osmi ton. Treći ton (B#5) se bez problema zamijeni se sljedećim tonom (D5), dok osmi ton (C#4) nema sljedećeg tona. U ovom slučaju se za sljedeći ton uzima prvi ton u listi (C4), te prvi i osmi ton zamijene mjesta. Nakon obavljanja mutacije, promijenili su se intervali između prvog i drugog tona, drugog i trećeg tona, četvrtog i petog tona, kao i između sedmog i osmog tona. Kao što vidimo u primjeru, nakon mutacije jedinka se sastoji od istih tonova, od kojih su neki zamijenili mjesta te nije došlo do pojave novih tonova.



Slika 3.5. Mutacija tonova zamjenom mjesta

3.7 Parametri algoritma

Za ovaj genetski algoritam određena su četiri parametra o kojima će izvođenje algoritma ovisiti. Parametri su sljedeći:

1) Broj iteracija algoritma određuje koliko će se puta provesti selekcija nad populacijom.
2) Faktor mutacije određuje vjerojatnost mutiranja pojedinog gena unutar jedinke. Za ovaj parametar unose se vrijednosti iz intervala od nula do jedan.
3) Veličina populacije određuje broj jedinki koje će se stvoriti na početku algoritma. Ovaj parametar mora imati minimalnu vrijednost tri, zbog toga što su potrebne minimalno tri jedinke za implementiranu selekciju.
4) Veličina jedinke određuje veličinu, odnosno broj tonova od kojih se sastoje jedinke.

Uz to je, pri pokretanju algoritma, još dodatno moguće odabrati da li će se provoditi križanje sa jednom točkom prekida ili uniformno križanje, kao i da li će se koristiti mutacija promjene tona za stepen/polustepen, mutacija zamjene mjesta dvaju susjednih tonova ili kombinacija obiju mutacija.

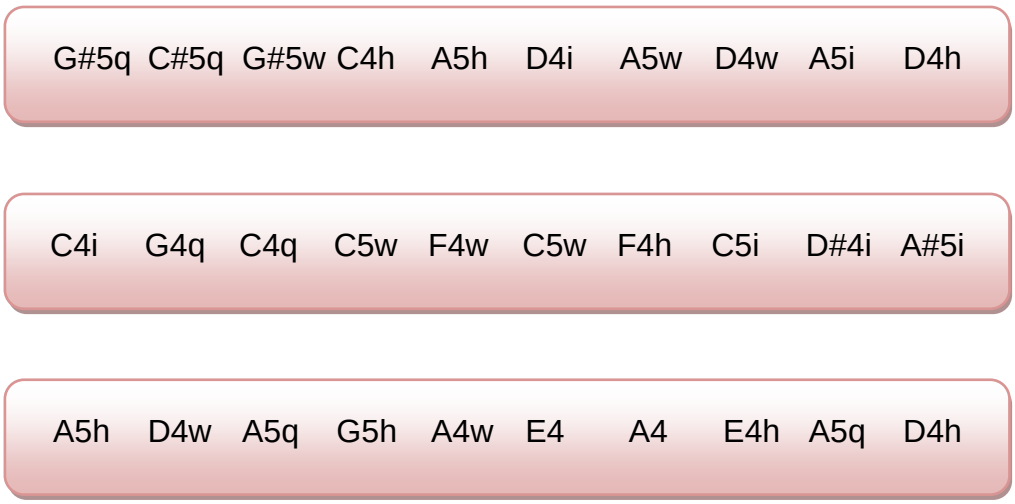
3.8 Rezultati

Algoritam je testiran na računalu sa četverojezgrenim procesorom Intel i7-920 @ 2.67 GHz sa 6 GB RAM memorije na frekvenciji 1.066 GHz. Korišteni operacijski sustav bio je 64-bitni Windows 7 Professional. Svi rezultati dobiveni su na temelju prosjeka od sto izvođenja algoritma.

Stupci tablice predstavljaju:

- B.I. – Broj iteracija algoritma
- F.M. – Faktor mutacije

- V.P. – Veličina populacije
- B.N. – Broj nota jedinki
- V.K. – Vrsta križanja, 0 za uniformno, 1 za križanje s fiksnom točkom
- V.M. – Vrsta mutacije, 0 za promjenu tona za stepen/polustepen, 1 za zamjenu mjesta dvaju tonova, 2 za kombinaciju obiju mutacija
- P.P. – Prosjek najboljih početnih jedinki
- MAX – Najbolja jedinka unutar svih izvođenja algoritma za zadane parametre
- P.MAX – Prosjek najboljih jedinki
- MIN – Najgora jedinka unutar svih izvođenja algoritma za zadane parametre
- P.MIN – Prosjek najgorih jedinki
- V.I. – Vrijeme izvođenja algoritma (u milisekundama)



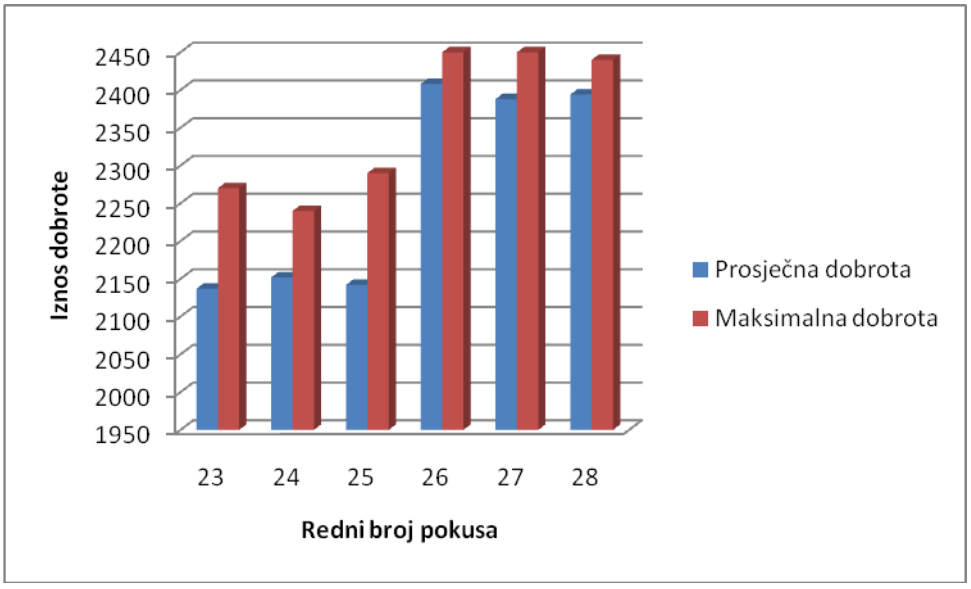
Slika 3.6. Primjer nekoliko najboljih jedinki dobivenih uzvođenjem algoritma

Tablica 3.2. Rezultati izvođenja algoritma

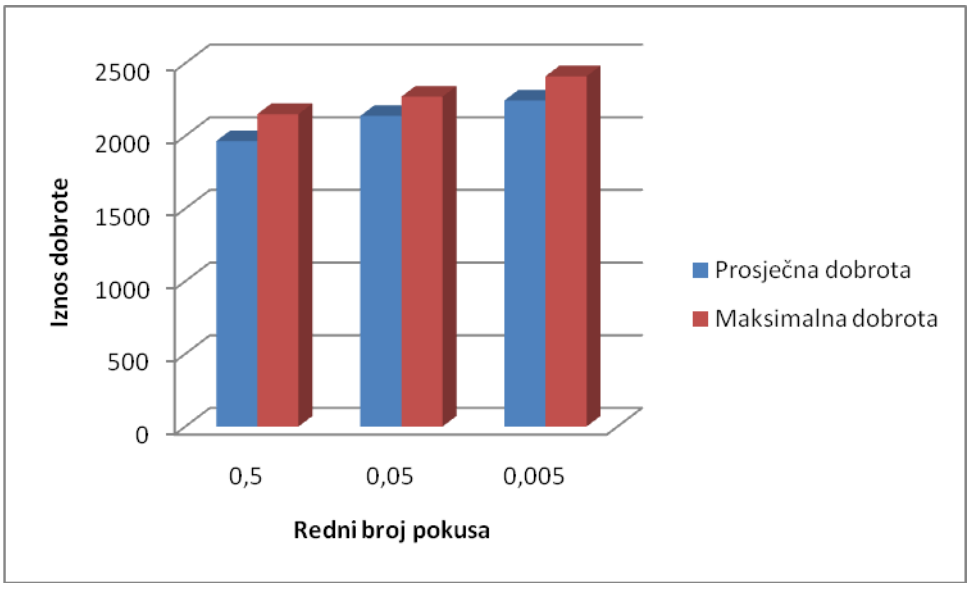
	B.I.	F.M.	V.P	B.N.	V.K.	V.M.	P.P	MAX	P.MAX	MIN	P.MIN	V.I.
1.	1000	0,1	50	10	0	0	824	1160	1091	500	811	740
2.	1000	0,1	50	10	0	1	815	1170	1092	450	669	747
3.	1000	0,1	50	10	0	2	806	1170	1093	430	706	747
4.	1000	0,1	50	10	1	0	804	1170	1088	630	834	740
5.	1000	0,1	50	10	1	1	821	1160	1090	550	813	740
6.	1000	0,1	50	10	1	2	810	1170	1099	600	795	737
7.	1000	0	50	10	0	-	808	1130	1030	710	998	744
8.	1000	0	50	10	1	-	815	1130	1004	860	860	734
9.	1000	1	50	10	0	0	812	1080	971	220	362	751
10.	1000	1	50	10	0	1	802	1170	1088	260	546	766
11.	1000	1	50	10	0	2	817	1040	959	190	361	778
12.	1000	1	50	10	1	0	818	1100	991	190	369	767
13.	1000	1	50	10	1	1	794	1170	1123	640	814	762
14.	1000	1	50	10	1	2	810	1090	982	130	382	769
15.	100	1	50	10	0	2	811	990	903	230	413	746

16.	100	1	50	10	1	2	811	1090	928	270	493	745
17.	10000	0,1	250	10	0	2	817	1170	1153	460	756	799
18.	10000	0,1	250	10	1	2	876	1170	1156	520	901	808
19.	10000	0,1	700	10	0	2	914	1160	1105	210	379	807
20.	10000	0,1	700	10	1	2	914	1170	1151	420	586	813
21.	10000	0,05	700	10	0	2	913	1170	1133	200	474	810
22.	10000	0,05	700	10	1	2	910	1170	1157	490	700	816
23.	50000	0,05	3000	20	0	0	1761	2270	2137	710	901	903
24.	50000	0,05	3000	20	0	1	1759	2240	2152	780	950	911
25	50000	0,05	3000	20	0	2	1765	2290	2142	690	923	919
26.	50000	0,05	3000	20	1	0	1759	2450	2408	1430	1633	922
27	50000	0,05	3000	20	1	1	1762	2450	2388	1300	1535	957
28.	50000	0,05	3000	20	1	2	1751	2440	2394	1330	1544	983
29.	50000	0,5	3000	20	0	0	1765	2150	1964	550	674	979
30.	50000	0,005	3000	20	0	0	1767	2410	2243	810	1061	939

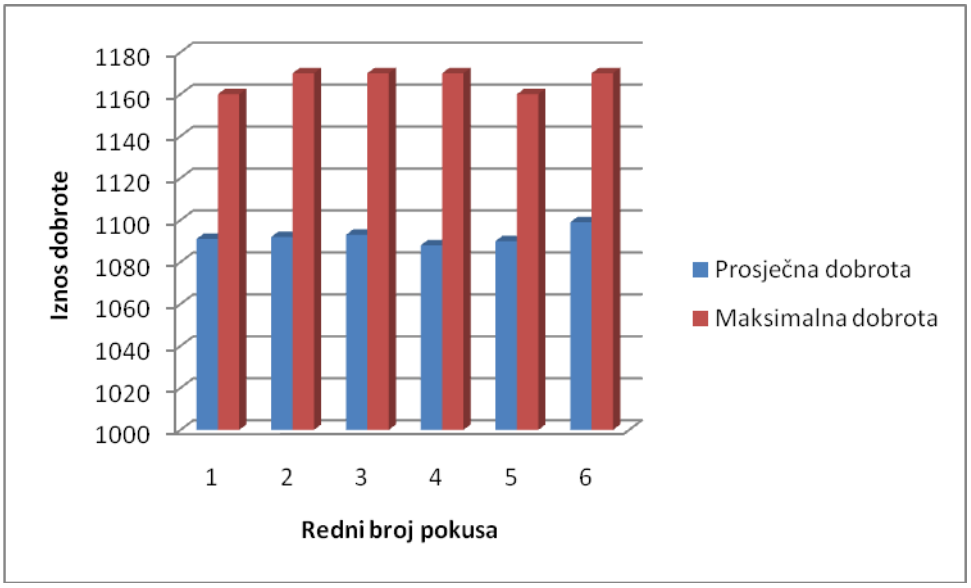
Rezultati nam pokazuju koliko je, u pojedinim slučajevima, velika ovisnost rezultata o ulaznim parametrima i korištenim genetskim operatorima. Iz rezultata je vidljivo da povećanjem broja iteracija i veličine populacije dobivamo sve bolja rješenja, što je bilo i za očekivati, dok smanjen broje iteracija se dobivaju lošiji rezultati. Nadalje, mutacija ima također vrlo velik utjecaj na kvalitetu dobivenih rezultata, zbog toga je potrebno izabrati vjerojatnost mutacije u ovisnosti o veličini populacije i broja tonova jedinke. Za iznos vjerojatnosti mutacije od jedan, rezultati dobiveni za vrstu mutacija 0 i 2 su dosta loši, dok su rezultati dobiveni za vrstu mutacije 1 iznenađujuće dobri. Rezultati koji su dobiveni za vjerojatnost mutacije od 0 su nešto bolji od rezultata kada je vjerojatnost mutacije iznosila 1, no opet su lošiji, nego kad je korištena neka umjerena vjerojatnost. Pri manjim populacijama, vrsta križanja nema neki veliki utjecaj na dobivene rezultate. Tek pri radu s većim populacijama, sa većim brojem tonova, može se primijetiti da križanje s jednom točkom prekida daje dosta bolje rezultate od uniformnog križanja. Također je pri većim populacijama bolje smanjiti iznos vjerojatnosti same mutacije.



Slika 3.7. Usporedba dobrota za različite kombinacije vrsta mutacija i križanja pri većim populacijama



Slika 3.8. Usporedba rezultata za različite vjerojatnosti mutacije



Slika 3.9. Usporedba dobrota za različite kombinacije vrsta mutacija i križanja pri manjim populacijama

3.9 Usporedba sa drugim postojećim implementacijama

U članku *„A Genetic Algorithm for Composing Music“* opisan je ponešto drugačiji oblik implementacije genetskog algoritma za skladanje glazbe. Ova implementacija je specifična po tome što uopće nema operator križanja, nego samo tri različita operatora mutacije. Mutacije koje se koriste su: zamjena mjesta dvaju uzastopnih tonova, koja funkcionira jednako kao istoimena mutacija opisana u ovom radu; promjena jednog tona koja za razliku od ranije opisane mutacije ne mijenja ton za

polustepen, odnosno stepen gore ili dolje, već ga zamijeni jednim proizvoljnim tonom; promjena tona za oktavu koja, mijenja odabrani ton za jednu oktavu. Još jedna vrlo bitna značajka ove implementacije je postojanje referentne jedinke. Referentna jedinka se koristi kako bi se generiranoj melodiji dao prepoznatljiv ritam i kako bi se spriječilo nasumično generiranje ritma. Zbog toga će tijekom inicijalizacije populacije nastati jedinke koje imaju ritam vrlo sličan onome kojeg ima referentna jedinka. U ovoj se implementaciji koristi eliminacijska selekcija, u kojoj se još dodatno prije brisanja jedinki s malom dobrotom brišu duplikati, odnosno jedinke s istom dobrotom. Nadalje, ukoliko najbolja jedinka nakon određenog broja iteracija nema zadovoljavajuću dobrotu, ta se jedinka briše, a jedinka koja je bila druga po dobroti postaje najbolja jedinka.

U članku *„Cohesive Music Generation with Genetic Algorithms“* je opisana implementacija koja tijekom skladanja melodije koristi čak dva genetska algoritma. Prvi genetski algoritam generira izvornu melodiju, s kojom se sve ostale melodije pokušaju harmonizirati. Nakon toga drugi genetski algoritam generira višeglasne varijacije ove melodije. Ovaj algoritam pristupa skladanju melodije na izrazito inovativan način, naime algoritam melodiju sklada po dijelovima. Tako se zasebno skladaju uvod, pripjev, kao i ostali dijelovi melodije. Zbog toga ova implementacija zahtijeva dodatne ulazne parametre, odnosno nekakav predložak po kojem će se skladati melodija. Tako je na primjer potrebno zadati tonalitet i trajanje za svaki pojedini dio melodije. Ova implementacija se također razlikuje i po zapisu tonova u podatkovnoj strukturi, naime dodana je još jedna lista koja sadrži oznake koje određuju kojoj oktavi pripada pojedini ton. Zbog toga su proširени i genetski operatori, od kojih neki djeluju i nad ovom drugom listom. Funkcija dobrote u ovoj implementaciji je također mnogo opsežnija, zbog toga što uzima puno više parametara u obzir pri ocjenjivanju jedinki.

Treći primjer implementacije genetskog algoritma za skladanje glazbe je nešto jednostavniji. Ovaj algoritam ima dva ista ograničenja kao što ima i algoritam opisan u ovom radu, a to su: algoritam generira samo jednoglasne melodije; melodija je samo niz tonova bez neke specifične mjere. Podatkovna struktura koja predstavlja jedinku također sadrži listu tonova i listu trajanja, a još dodatno sadrži i listu oznaka oktava kojim pojedini tonovi pripadaju. Za razliku od funkcije dobrote opisane u ovom radu, koja dobrotu ocjenjuje samo na temelju dobrota intervala susjednih tonova, funkcija dobrote u ovom primjeru ocjenjuje dobrotu na temelju još nekih dodatnih parametara (melodija mora biti u C-dur ili C-mol ljestvici, melodija se proteže kroz najviše dvije oktave i slično).

4. Zaključak

Na temelju dobivenih rezultata vidljivo je da algoritam, primjenom genetskih operatora, pokušava optimirati melodiju, prema zadanim vrijednostima dobrota pojedinih intervala. Zbog toga će krajnja melodija koju ćemo dobiti kao rezultat ovisiti ne samo o ulaznim parametrima genetskog algoritma, nego i o vrijednostima dobrota intervala koje su bile unaprijed zadane. Prema vrijednostima koje su bile zadane za ispitivanje ovog algoritma vidljivo je da će se preferirati intervali poput kvarte, kvinte i oktave, dok će se intervali poput tritonusa, prime, kao i intervali iznad oktave izbjegavati.

Postoji mnogo mogućnosti kako proširiti i poboljšati ovaj algoritam u budućnosti. Ovisno o vrsti glazbe koju se želi komponirati trebalo bi detaljnije razraditi dobrote pojedinih intervala. Također bi bilo dobro detaljnije se usredotočiti i na samo trajanje tonova i generiranje njihova trajanja. Primjerice, jedno veliko proširenje algoritma predstavljalo bi komponiranje polifonih melodija, a ne samo onih monofonih. S druge strane, ta promjena zahtijevala bi dodatno proširenje funkcije dobrote, kao i slaganje nove liste dobrota intervala za tonove koji se izvode u isto vrijeme. Kao još jedno moguće proširenje nameće se i komponiranje pratnje za glavnu melodiju. To proširenje bi također zahtijevalo određene promjene u funkciji dobrote i još nekim dijelovima algoritma.

Kao što je vidljivo, prostora za poboljšanja ima mnogo te bi upravo zbog te činjenice daljnji rad na ovom algoritmu bio višestruko isplativ.

5. Literatura

[1] Craane, J., Melody composition using genetic algorithms, 2009.

http://jcraane.blogspot.com/2009/06/melody-composition-using-genetic.html

[2] Pigg, P., Cohesive Music Generation with Genetic Algorithms, 2002.

http://web.mst.edu/~tauritzd/courses/ec/fs2002/project/Pigg.pdf

[3] Matić, D., A genetic algorithm for composing music, Yugoslav Journal of Operations Research, 2010.

http://www.doiserbia.nb.rs/img/doi/0354-0243/2010/0354-02431001157M.pdf

6. Sažetak

Genetski algoritam je heuristička metoda koja imitira proces prirodne evolucije. Primjenom genetskih operatora mutacije, selekcije i križanja nad populacijom potencijalnih rješenja nekog problema, pokušava se dobiti što bolje rješenje određenog problema. Genetski algoritam uglavnom se primjenjuje za rješavanje problema optimizacije i pretraživanja.

Predmet proučavanja ovog rada bili su sami genetski algoritmi, a posebice njihova primjena u umjetnosti, u ovom slučaju u komponiranju glazbe. Kroz ovaj rad predstavljena je jedna moguća implementacija genetskog algoritma koja kao rezultat daje jednostavnu monofonu melodiju te je upravo uspjeh pri rješavanju ovog problema pokazao da se ti predstavljeni genetski algoritmi zbog svojih specifičnih svojstava mogu efikasno iskoristiti i čak i u ovu, pomalo apstraktnu, namjenu.