SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Rješavanje problema rubikove kocke evolucijskim algoritmima

Vinko Kolobara
Voditelj: *prof.dr.sc. Domagoj Jakobović*

Zagreb, svibanj 2017.

SADRŽAJ

1. Uvod

Rubikova kocka je logička igra jednostavnih pravila, ali s velikim brojem mogućih stanja igre i složenim (za običnog čovjeka) algoritmima za rješavanje. Zbog brojnosti stanja naivne metode za rješavanje su nepodobne i neefikasne.

U ovom radu će se taj problem pokušati riješiti evolucijskim algoritmima, točnije genetskim algoritmom i genetskim programiranjem. Kako su se ti algoritmi pokazali efikasnim za rješavanje složenih problema (generiranje rasporeda, aproksimacije funkcija, učenje neuronskih mreža ...), tako bi se mogli pokazati efikasnima i za problem rubikove kocke.

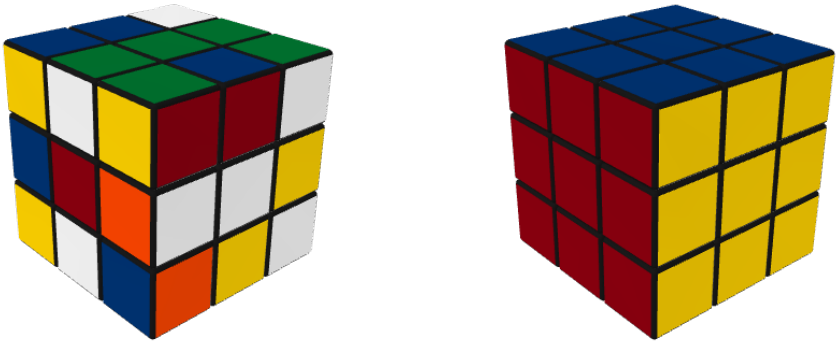
Za pristup genetskim algoritmom, ideja je učiti rješenje za samo jednu kocku koje će se prikazati kao niz rotacija (što kraći mogući) koje će dovesti do ciljnog stanja ili što bliže ciljnom stanju.

S druge strane, pokušaj genetskim programiranjem bit će usmjeren na pronalaženje genetskog programa koji će bilo koju rubikovu kocku naći rješenje (ili doći dovoljno blizu rješenju) u što manjem broju rotacija. Točnije, za određeno stanje će biti u stanju odrediti koja rotacija će ga dovesti u najbolje sljedeće stanje.

Ovaj problem otežava činjenica što je za navedeni problem jako teško naći funkciju koja će odrediti kvalitetu neke rotacije u nekom stanju. Teško je garantirati da se već ispravno pozicioniranih par strana kocke, neće pokvariti u procesu postavljanja ostalih strana.

2. Problem rubikove kocke

2.1. Definicija



Slika 2.1: Primjer nekog nasumičnog početnog stanja rubikove kocke (lijeva slika) i prikaz ciljnog stanja (desna slika)

Rubikova kocka je logička igra općenito dimenzija $N \times N \times N$ u kojoj je cilj pomoću osnovnih rotacija strana kocke postići da svaka strana bude iste boje. U ovom radu će se nadalje govoriti o kocki dimenzija $3 \times 3 \times 3$. Za svaku stranu moguće su dvije vrste rotacije, u smjeru kazaljke na satu i obrnuto. Sve dozvoljene rotacije označavat će se:

- F - prednja strana u smjeru kazaljke sata,
- F' - prednja strana obrnuto od kazaljke sata
- B - stražnja strana u smjeru kazaljke sata,
- B' - stražnja strana obrnuto od kazaljke sata
- U - gornja strana u smjeru kazaljke sata,
- U' - gornja strana obrnuto od kazaljke sata
- D - donja strana u smjeru kazaljke sata,

– D’ - donja strana obrnuto od kazaljke sata

– L - lijeva strana u smjeru kazaljke sata,

– L’ - lijeva strana obrnuto od kazaljke sata

– R - desna strana u smjeru kazaljke sata,

– R’ - desna strana obrnuto od kazaljke sata

Često se uvode i dodatne rotacije, koje označavaju uzastopno izvođenje jedne od jednostavnih rotacija:

– F2 - prednja strana dva puta u smjeru kazaljke sata,

– B2 - stražnja strana dva puta u smjeru kazaljke sata,

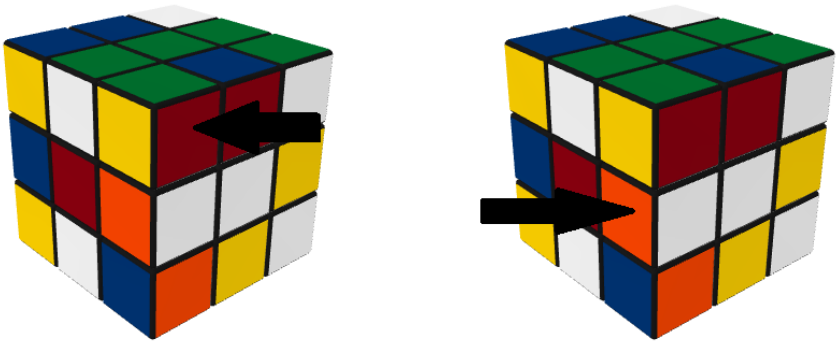
– U2 - gornja strana dva puta u smjeru kazaljke sata,

– D2 - donja strana dva puta u smjeru kazaljke sata,

– L2 - lijeva strana dva puta u smjeru kazaljke sata,

– R2 - desna strana dva puta u smjeru kazaljke sata

Svaka strana se može prikazati kao $N \times N$ matrica sa određenim bojama na poljima. Dodatno, može se uvesti i pojam kockice (engl. *cubie*), i to dvije vrste, kutna (engl. *Corner Cubie*) i rubna (engl. *Edge Cubie*). Svaka kutna kockica na sebi ima 3 boje, i može biti u 2 osnovna stanja: ispravno orijentirana (kada su sve boje na pravim mjestima) i pogrešno orijentirana. Svaka rubna kockica sastoji se od 2 boje, i može biti u ista 2 stanja kao i kutna. Na kocki postoji 8 kutnih i 12 rubnih kockica.



Slika 2.2: Primjer kutne (lijeva slika) i rubne kockice (desna slika)

2.2. Standardni algoritmi za rješavanje

Broj stanja u kojima se rubikova kocka može naći iznosi oko 43 kvintilijuna što je popriličan broj i obična metoda grubom silom bi bila neuspješna. Zbog toga postoje brojni uspješni algoritmi kojima se može brzo i efikasno doći do rješenja.

Najuspješniji algoritmi koriste teoriju grupa i činjenicu da kada dođu u određeni skup stanja (grupu), ako se uzme određeni podskup rotacija, nemoguće je "pogoršati" rješenje već samo prijeći u sljedeću grupu u kojoj ima manje stanja i koja je bliža ciljnom rješenju. Primjeri takvih algoritama su Thistlewaite, Kociemba, Korf... [2]

Kako je ovo kombinatorni problem i jako složen za uobičajeno rješavanje, pokušaj u ovom seminaru bit će usmjeren na rješavanje rubikove kocke upotrebom samo evolucijskih algoritama, bez korištenja dodatnih informacija iz teorije grupa i sličnog.

2.3. Programska reprezentacija

Rubikova kocka se u implementaciji sastoji od 6 strana, od kojih je svaka strana $N \times N$ matrica koja sadrži oznaku boje pojedinog elementa. Oznaka boje je cijeli broj iz intervala $[0, 5]$. Svaka kocka može vratiti i i-tu rubnu ili kutnu kockicu, gdje je i iz intervala $[0, 11]$ za rubne, a iz intervala $[0, 7]$ za kutne.

Kroz sučelje je omogućeno i rotiranje kocke na 2 načina: predavanjem indeksa stranice i oznake radi li se u smjeru kazaljke ili obrnuto od kazaljke na satu; ili predavanjem cijelog broja od 0-17 za kojeg metoda automatski računa koju stranu i u kojem smjeru rotira.

3. Genetski algoritam

3.1. Opis genetskog algoritma

Genetski algoritam pripada evolucijskim algoritmima i pokušava korištenjem genetskih operatora (križanje, mutacija, selekcija) doći do rješenja problema. Sastoji se od populacije od N jedinki, svaka jedinka sadrži dobrotu (engl. *fitness*) koja označava koliko je ta jedinka blizu ciljnom rješenju, a svaka jedinka se sastoji od m gena koji predstavljaju dio rješenja.

Selekcijom biramo roditelje za križanje, križamo ih, obavljamo mutaciju i to sve ponavljamo dok se ne ostvari uvjet zaustavljanja (obično broj iteracija ili pronalazak rješenja ili dostignuta određena dobrota).

Algoritam 1 Primjer eliminacijskog genetskog algoritma

```
1: P <- generiraj nasumično početnu populaciju
2: dok nije zadovoljen uvjet zaustavljanja radi
3:     R1, R2, W <- selekcija(P)      ▷ Izaberi 2 roditelja i jedinku za eliminaciju
4:     D <- križaj(R1, R2)
5:     D <- mutiraj(D)
6:     P(W) <- D                      ▷ Zamijeni najgoreg novonastalim djetetom
```

3.2. Primjena GA na problem rubikove kocke

3.2.1. Reprezentacija rješenja

Poznato je da se svaka rubikova kocka može riješiti u maksimalno 20 rotacija[6], zato će svaka jedinka biti predstavljena kao niz od 40 gena (cijelih vrijednosti) koji predstavljaju, redom, slijed rotacija koje je potrebno izvesti. Dozvoljeni brojevi za svaki gen su iz raspona $[0, 17]$. Vrijednosti iz raspona $[0, 12]$ predstavljaju obične operacije

2	0	...	7	15

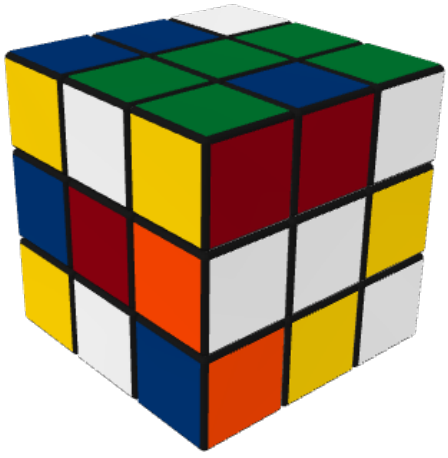
Slika 3.1: Primjer rješenja genetskog algoritma

(rotacija svake strane u bilo kojem smjeru), dok vrijednosti iz raspona $[13, 17]$ predstavljaju složene operacije (dvostruke rotacije pojedine strane).

3.2.2. Genetski operatori i dobrota

Korištena je troturnirska selekcija, za križanje je korišteno križanje s jednom i dvije točke prekida, a za mutaciju je implementiran poseban operator uz korištenje i nasumične mutacije gena. Koristi se operator mutacije koji će odrediti nakon koje je rotacije u kromosomu najbolje stanje kocke, i sve rotacije nakon tog stanja nasumično generira (pohlepna mutacija).

Posebnost ovog problema i ovakve implementacije operatora je to što se mutacija i križanje ne koriste u kontekstu u kojem se inače koriste u genetskom algoritmu. Naime, križanje uobičajeno ne mijenja previše rješenje i služi za lokalno pretraživanje prostora stanja, a mutacija je zadužena da rješenje odvede u sasvim drugom smjeru. U ovom slučaju, događa se upravo obrnuta stvar, križanje obavlja posao mutacije, a mutacija posao križanja.



Slika 3.2: Prva dobrota: brojanje koliko je oznaka na pravom mjestu (centralna kockica je nepomična i ona određuje boju cijele strane) = 7 (za prikazane 3 stranice, analogno se računa i za ostale)
Druga dobrota: brojanje ispravno pozicioniranih rubnih i kutnih kockica = 1 * 6 (kutna na gornjem desnom rubu, gore zelena, dolje bijela)

Implementirane su dvije različite dobrote. Prva jednostavno broji koliko je boja na pravim mjestima kocke (ne brojeći centralni element svake strane jer je on fiksan). Maksimalna vrijednost ove mjere je 48, ali se skalira na $[0, 1]$.

Druga mjera broji koliko je kutnih i rubnih kockica na pravim mjestima u pravoj orijentaciji i na to jos nadoda vrijednost prve mjere. Maksimalna vrijednost ove mjere je 148, također skalirano na $[0, 1]$ (svaka ispravna rubna kockica vrijedi 4, dok svaka ispravna kutna kockica vrijedi 6, napravljeno zato da u slučaju kombinacije sa prvom mjerom svaka podmjera sudjeluje u jednakom broju).

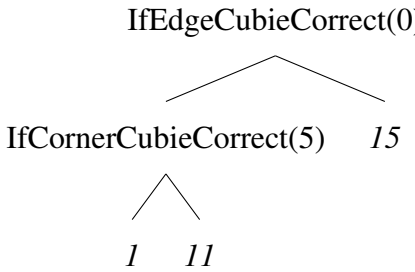
4. Genetsko programiranje

4.1. Opis genetskog programiranja

Genetsko programiranje također pripada evolucijskim algoritmima i koristi iste funkcije kao i genetski algoritam. Najveća razlika je prikaz rješenja. U genetskom algoritmu se rješenje uglavnom prikazuje kao niz gena, dok se u genetskom programiranju rješenje prikazuje kao stablo. Svako stablo se sastoji od funkcijskih (engl. *Function*) i završnih (engl. *Terminal*) čvorova. Završni čvorovi se nalaze u listovima stabla, dok funkcijski nešto rade sa podstablima kako bi odredili konačan rezultat.

Rad algoritma je jednak genetskom, uz uvođenje dodatnih složenosti kod stvaranja inicijalne populacije te složenijih operatora križanja i mutacije.

4.2. Primjena GP na problem rubikove kocke



Slika 4.1: Primjer jednog dobivenog rješenja genetskim programiranjem

Pokušaj genetskim programiranjem, za razliku od prethodnog pokušaja, za cilj ima pronaći najbolji program koji će u što manje koraka riješiti bilo koju konfiguraciju kocke, točnije, moći će odrediti najbolju rotaciju (ili niz rotacija) za neko stanje kocke.

Potrebno je definirati dodatne funkcije, koje će za svaku rubnu ili kutnu kockicu odrediti je li na pravom mjestu ili ne. Ovisno o tome, funkcija će izvesti lijevo ili desno podstablo.

Završni čvorovi su cijeli brojevi koji predstavljaju operaciju koja se treba izvršiti.

Operatori križanja i mutacije koji se koriste su uobičajeni. Za križanje koristi se: zamjena čvorova, uniformno križanje . . . Za mutaciju koriste se: mutacija cijelog podstabla, mutacija čvora . . .

Svaki program se izvodi dok ne dođe do rješenja rubikove kocke ili dok dosegne dozvoljeni broj rotacija. Korištene dobrote su iste kao za genetski algoritam, uz to što se dodatno nagradi algoritam koji je prije došao do rješenja.

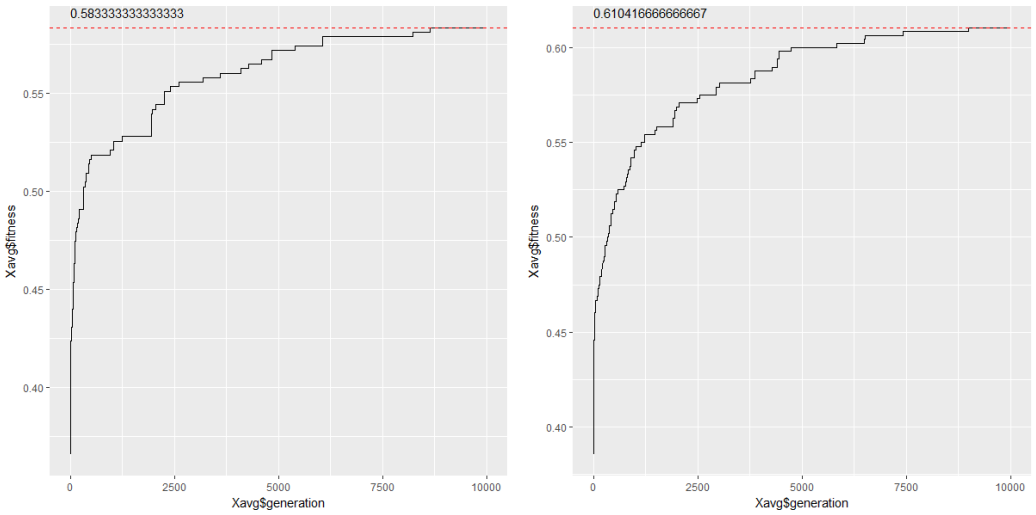
5. Rezultati

Za implementaciju rješenja problema korištena je programska biblioteka JGAP[4] napisana u programskom jeziku Java. Parametri korišteni u dobivanju rezultata prikazani su u tablici 5.1. Svi rezultati dobiveni su uprosječivanjem 10 pokretanja algoritma na različito izmiješanim kockama.

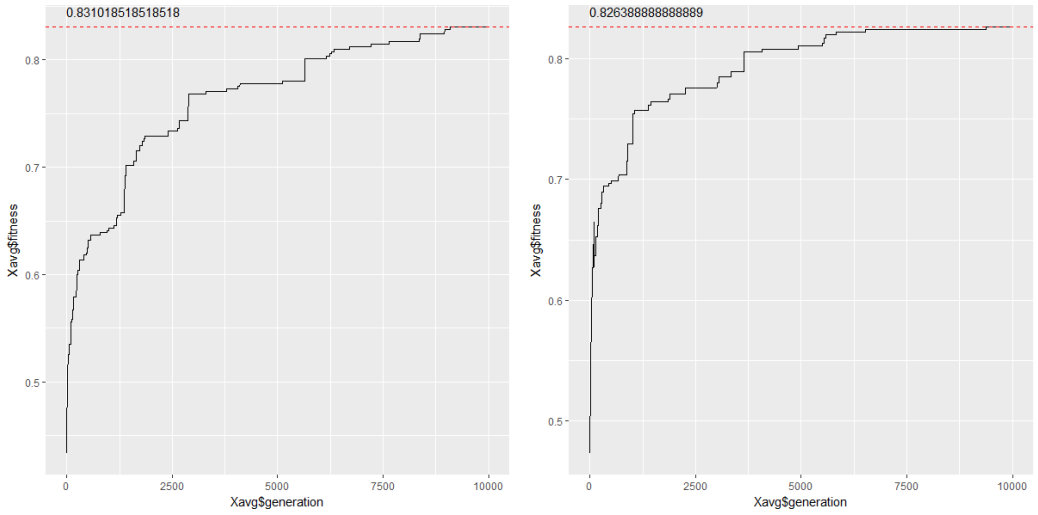
Veličina populacije	50
Vjerojatnost križanja	0.5
Vjerojatnost pohlepne mutacije	0.1
Vjerojatnost nasumične mutacije	0.05

Slika 5.1: Parametri genetskog algoritma

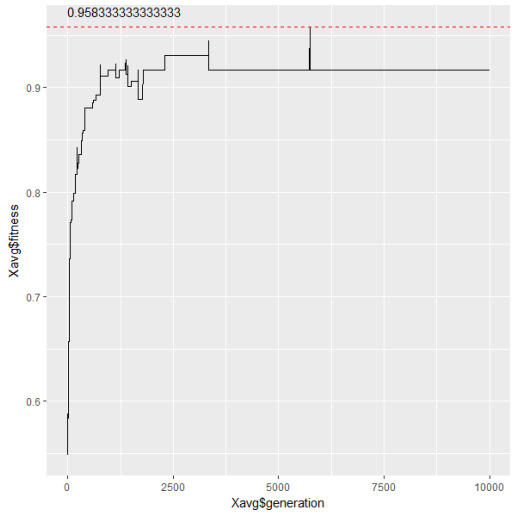
Na sljedećim grafovima prikazana je prosječna vrijednost funkcija dobrote po generacijama.



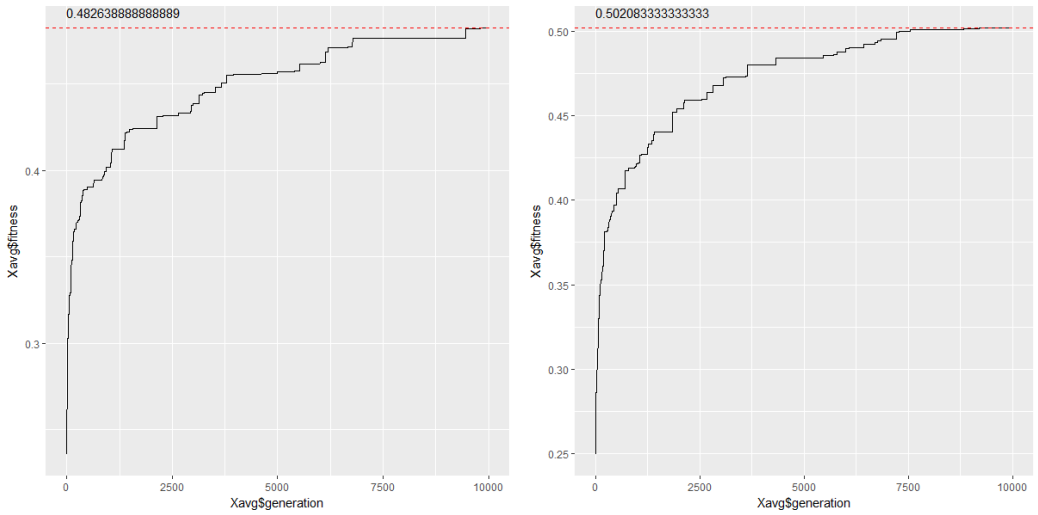
Slika 5.2: Prosječna vrijednost prve dobrote po generacijama za 50 (lijevo) i 20 (desno) miješanja kocke



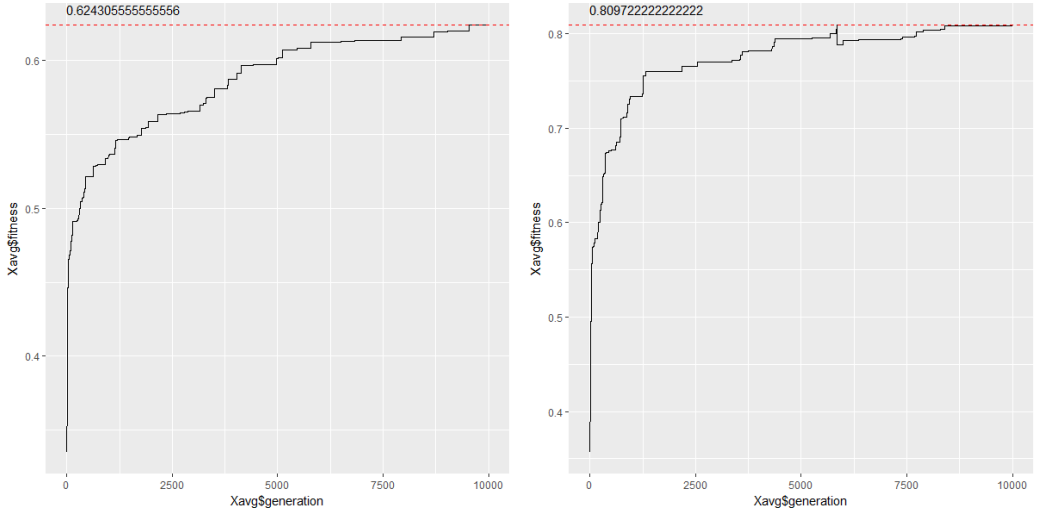
Slika 5.3: Prosječna vrijednost prve dobrote po generacijama za 10 (lijevo) i 7 (desno) miješanja kocke



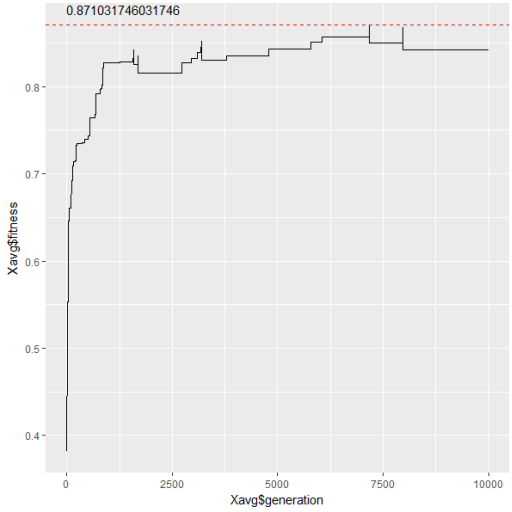
Slika 5.4: Prosječna vrijednost prve dobrote po generacijama za 5 miješanja kocke



Slika 5.5: Prosječna vrijednost druge dobrote po generacijama za 50 (lijevo) i 20 (desno) miješanja kocke

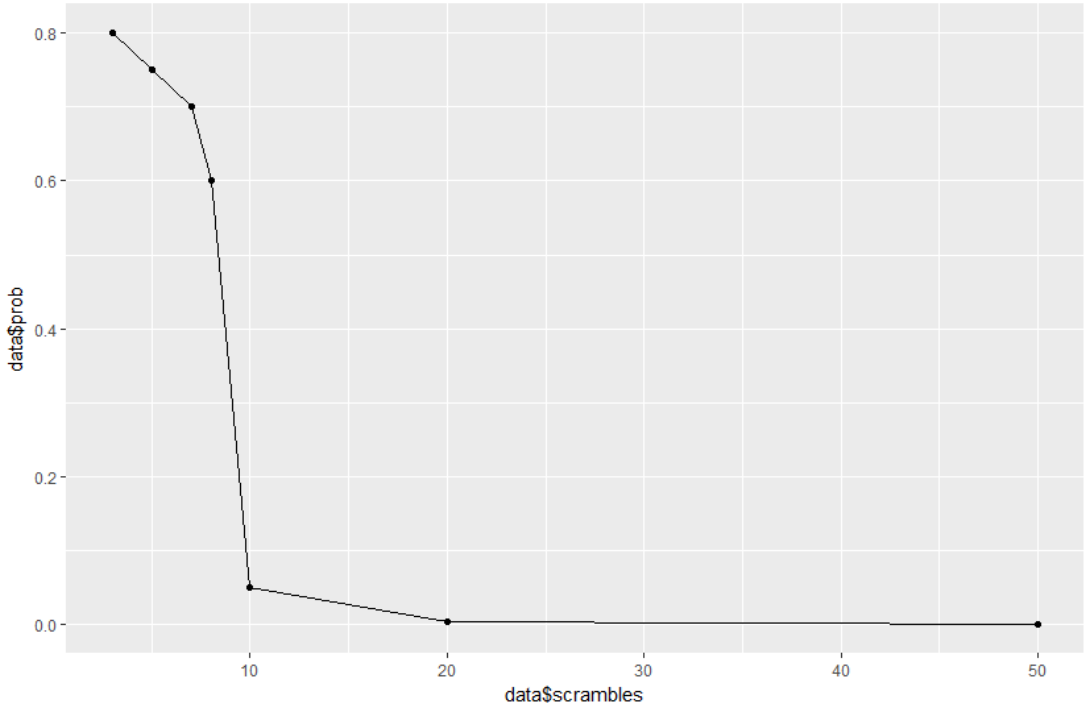


Slika 5.6: Prosječna vrijednost druge dobrote po generacijama za 10 (lijevo) i 7 (desno) miješanja kocke



Slika 5.7: Prosječna vrijednost druge dobrote po generacijama za 5 miješanja kocke

Iz grafova se može vidjeti da za mali broj miješanja kocke, brzo dobijemo rješenja koja su blizu ciljnog stanja (po dobroti, ne nužno i po broju koraka potrebnih do ciljnog stanja). Također za mali broj miješanja često dođemo i do konačnog rješenja. S druge strane, broj miješanja veći ili jednak 10, uzrokuje nemogućnost pronalaska ispravnog rješenja u većini slučajeva. Navedena ovisnost prikazana je na slici 5.8



Slika 5.8: Vjerojatnost pronalaska rješenja u ovisnosti o broju miješanja kocke (udaljenosti od ciljnog stanja)

Uspoređujući dvije mjere dobrote vidimo da nema neke razlike u uspješnosti rješenja u ovisnosti o mjeri. Jedina razlika je iznos prosječne dobrote, ali ta razlika nastaje zbog dodavanja težina rubnim i kutnim kockicama što uzrokuje veće skokove u iznosu dobrote.

6. Zaključak

Cilj ovog rada je bio implementirati rješavanje rubikove kocke genetskim algoritmom. Od zamišljenog, uspješno je napravljena programska reprezentacija rubikove kocke i jednostavno sučelje za rad s kockom. Međutim, rezultati pokazuju da ovakav način rješavanja nije sasvim primjenjiv na problem.

Glavni problem ovakvog rješenja leži u tome što se rubikova kocka ne može riješiti ako se prije toga, makar na trenutak, ne pokvari već djelomično složeno stanje. Genetski algoritam će uglavnom tražiti rješenja s većom dobrotom, ali ta veća dobrota ne mora nužno voditi do ispravnog niza koraka do ciljnog stanja.

Taj problem je moguće riješiti dodavanjem još više znanja iz područja rješavanje rubikove kocke, korištenjem teorije grupa, dodavanjem dodatnih složenih operacija rotacije koji izvršavaju više rotacija odjednom.

Dakle, može se zaključiti kako problem s ovakvom mjerom dobrote i bez korištenja specijaliziranijih genetskih operatora nije prikladan za rješavanje prave rubikove kocke. Ali, daje osnovni uvid u dobre i loše strane genetskog algoritma, brzo može doći do nekakvog rješenja bez prevelikog znanja o stvarnom problemu u pozadini.

7. Literatura

[1] Dénes Ferenc. Rubik's cube algorithms, 2017. URL https://ruwix.com/the-rubiks-cube/algorithm/.

[2] Karl Hörnell. Progressive improvements in solving algorithms, 1996. URL http://cubeman.org/dotcs.txt.

[3] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992. ISBN 0-262-11170-5. URL http://mitpress.mit.edu/books/genetic-programming.

[4] Klaus Meffert i Neil Rotstan. Java genetic algorithms package, 2017. URL http://jgap.sourceforge.net/.

[5] Walter Randelshofer. Virtual cubes, 2017. URL http://www.randelshofer.ch/rubik/virtual_cubes/rubik/instructions/instructions.html.

[6] Morley Davidson Tomas Rokicki, Herbert Kociemba i John Dethridge. God's number is 20, 2010. URL http://www.cube20.org/.

[7] Ashutosh Tyagi i Poonam Tyagi. Gen-r: Genetic algorithm based model for rubik's cube solution generator. *International Journal of Science and Advanced Technology (ISSN 2221-8386)*, 2011.

8. Sažetak

Rubikova kocka je logička igra u kojoj se nizom rotacija pokušava doći do ciljnog stanja. Zbog brojnosti stanja, naivni algoritmi su neefikasni i potrebno je koristiti pametnija rješenja.

Uobičajeni algoritmi za rješavanje rubikove kocke koriste teoriju grupa i činjenicu da kada se kocka nađe u nekom stanju, korištenjem nekog podskupa rotacija, ostat će u istoj grupi stanja i može samo prijeći u grupu bližu ciljnom rješenju.

Ovaj seminarski rad bio je usmjeren na pokušaj rješavanja rubikove kocke evolucijskim algoritmima, genetskim algoritmom i genetskim programiranjem bez korištenja teorije grupa i sličnoga. Genetski algoritam se izvodi na samo jednoj konfiguraciji kocke i pokušava nju riješiti, dok se genetskim programiranjem pokušava doći do općenitog rješenja za svaku konfiguraciju rubikove kocke.