Razvoj heurističkih algoritama za igranje igrica

Seminarski rad u okviru kursa
Računarska Inteligencija
Matematički fakultet

Aleksnadar Ranković, Jelisaveta Smiljanić
mi16111@alas.matf.bg.ac.rs, mi16138@alas.matf.bg.ac.rs

15. april 2020.

Sažetak

Igranje igrica zna da bude jako težak i zahtevan hobi, ako imate nespretne prste i nisku inteligenciju može se pokazati kao nemoguć. Zbog toga na scenu stupa ovaj seminarski rad. Koliko god Vi bili očajni u igranju igrica, naš robot ne može da bude gori. Isprobane su dve različite metode učenja uz pomoć genetskog algoritma, jedna se oslanja na trenutno stanje u kome se robot nalazi, a druga na metodu kreiranu za rešavanje igrice Infinite Mario Bros. Posle testiranja metoda upoređeni su dobijeni rezultati.

Sadržaj

1 Uvod

U cilju razumevanja ovog rada u potpunosti, bitno je razumeti kako funkcionišu genetski algoritmi. Genetski algoritmi su algoritmi koji oponašaju proces evolucije. Koriste se kod problema optimizacije ili pretrage, s ciljem pronalaženja približnog ili tačnog rešenja. Jedno rešenje naziva se jedinka i predstavlja se pomoću hromozoma ili genotipa. Svaka jedinka sadrži skup gena koji određuju njene osobine. Korišćenjem pojmova kao što su funkcija cilja, selekcija, mutacija, ukrštanje i nasleđivanje kreiraju se generacije jedinki. Pomoću funkcije prilagođenosti računa se kvalitet, tj. prilagođenost okolini, svake od jedinki. Početna generacija obično se sastoji od slučajno generisanih jedinki. Kroz proces selekcije biraju se jedinke koje će učestvovati u procesu ukrštanja (stvaranja novih jedinki). Verovatnoća da će jedinka biti izabrana procesom selekcije, proporcionalna je njenom kvalitetu. Kombinacijom gena dve izabrane jedinke, procesom ukrštanja, dobijaju se nove jedinke (obično jedna ili dve). Kao što u prirodi vremenom dolazi do mutacija određenih gena, tako se u genetskim algoritmima korišćenjem operatora mutacije može modifikovati deo polazne jedinke. Nova generacija može se kreirati od postojećih jedinki i njihovog potomstva na različite načine, u zavisnosti od politike zamene generacija. Bez obzira na način, očekuje se da prilagođenost nove generacije bude bolja od prilagođenosti prethodne. Genetski algoritam, u zavisnosti od određenih parametara, može se zaustaviti iz više razloga. Ukoliko je uzrok zaustavljanja dostignut zadati broj generacija, ne možemo garantovati da je pronađeno najbolje rešenje, dok to možemo ukoliko je uzrok zaustavljanja dostignut željeni kvalitet generacije [2].

Kako bi bilo jasno na šta je genetski algoritam primenjen, bitno je upoznati se sa igricom koja je kreirana specijalno za ovaj projekat. Construct Crusade je jednostavan platformer relativno nalik igrici Super Mario. Naš junak je robot koji ima pet pokreta: hodaj levo, hodaj desno, skoči u mestu, pucaj, ispali laser. Neki od tih pokreta mogu se izvoditi simultano: skoči levo, skoči desno, skači i pucaj. Osim robota u igri se nalaze platforme, rupe između njih i neprijatelji koje robot mora da pobedi. Dužine platforma i neprijatelji su nasumično generisani, tako da je nivo svaki put malo drugačiji. Robot pobeđuje kada stigne do poslednje platforme. Na slici 3 je prikazana napravljena igra. U narednim sekcijama opisano je prilagođavanje dva različita genetska algoritma pri rešavanju igre i dati su rezultati njihovog testiranja.

Slika 1: Prikaz igre Construct Crusade

2 Implementacija genetskog algoritma korišćenjem prve metode

Jedan hromozom u našem genetskom algoritmu predstavljen je kao vektor celih brojeva (izmedju 0 i 11)dužine sedam. Indeks vektora (od 0 do 6)predstavlja situaciju u kojoj robot može da se nađe, a celi brojevi predstavljaju različite odgovore na datu situaciju (pokret ili niz pokreta koje robot treba da izvede da bi napustio/razrešio situaciju). Robot se može naći u sedam različitih stanja: platforma na kojoj se nalazi je prazna, na platformi levo od njega je šišmiš (neprijatelj), na platformi desno od njega je šišmiš, na platformi se nalazi zlato, na platformi levo od njega je imp (drugi neprijatelj), na platformi desno od njega je imp, došlo je do kolizije robota i impa. Navedene situacije predstavljaju aktivno stanje robota i konstantno se ažuriraju. Naš hromozom predstavlja šablon za rešavanje nivoa, koji robot koristi kako bi znao šta treba da uradi u datoj situaciji. Neki od odgovora na situacije su namerno loši da bismo utvrdili da li će ih genetski algoritam odbaciti. Postoje stanja koja imaju više naizgled dobrih solucija, tad imamo za cilj da utvrdimo koja opcija je najbolja (koja će odvesti robota najdalje).

Fitnes je izračunavan na osnovu poena koje robot stekne igrajući igru korišćenjem prosleđenog vektora jednog hromozoma. Što je robot bliži kraju nivoa fitnes je veći. Može osvojiti bonus poene ako je prešao igru ili skupio zlato (koje ispuštaju neprijatelji kada umru)i negativne poene ako je izgubio. Računanje fitnesa je najkompleksniji deo programa koji zahteva najviše vremena (pokušali smo na par načina da ubrzamo, ali su svi završeni neuspehom).

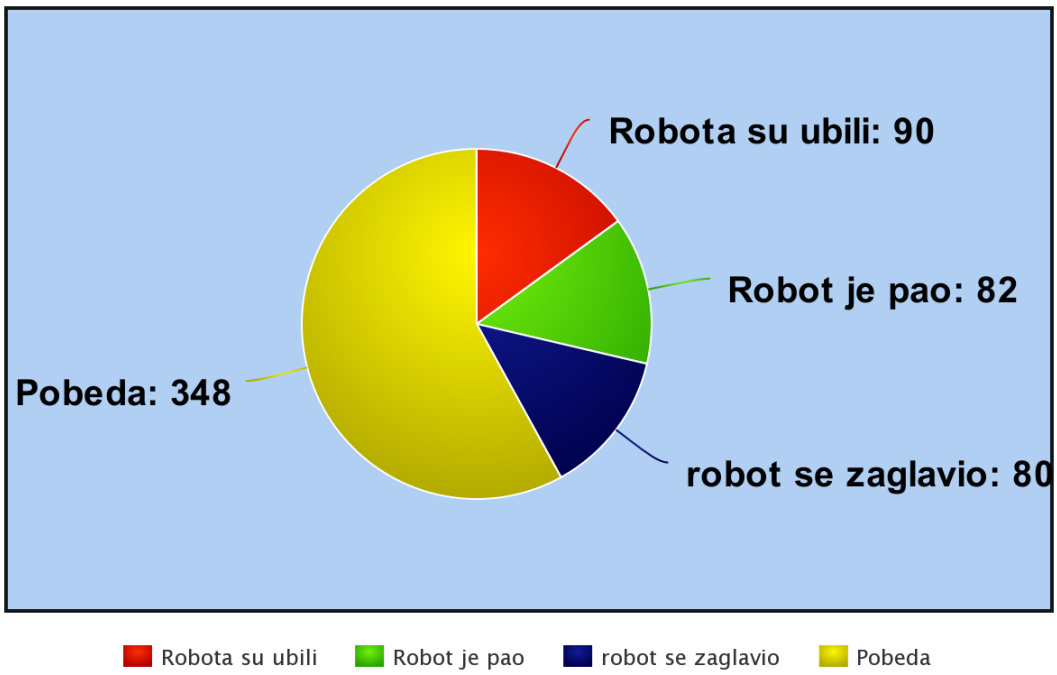
Inicijalna generacija kreirana je stvaranjem vektora sa nasumičnim vrednostima (u željenim opsezima). Nakon izgradnje vektora izračunavan je njegov fitnes. Što se tiče odabira jedinki koje će ostaviti potomstvo (selekcije), koriste se dve metode: turnirska i ruletska selekcija [3]. Nakon selekcije naredna generacija nastaje primenom operacija ukrštanja i mutacije, nad selektovanim jedinkama, dok se ne dobije generacija željene veličine. Ukrštanjem dva nasumična roditelja dobijaju su dve nove jedinke nad kojima je potom primenjena operacija mutacije. Ukrštanje može biti jednopoziciono, dvopoziciono i nasumično/potpuno, dok je mutacija jednopoziciona na nasumičnom indeksu. Ovaj proces ponavlja se zadati broj puta. Ne postoji neka politika zaustavljanja osim dostizanja maksimalnog broja iteracija.

2.1 Testiranje i rezultati

Algoritam je testiran menjanjem ulaznih parametara: veličine generacije, maksimalnog broja iteracija, verovatnoće mutacije, tipa ukrštanja, tipa selekcije i još nekih specifičnih parametara. U tabeli 1 prikazani su rezultati testiranja. Od testa osam uvedeno je simulirano kaljenje, zbog problema sa preuranjenom konvergencijom. Preuranjena konvergencija se dešava češće pri ruletskoj selekciji, zato što je fitnes jednog rešenja dosta veći u odnosu na ostale. Ovime problem nije rešen u potpunosti ali je napravljeno blago poboljšanje. Na slici 3 prikazane su statistike o načinu na koji je partija završena pri korišćenju parametara iz testa devet, koji su se pokazali kao najbolji.

Tabela 1: Rezultati učenja robota genetskim algoritmom, korišćenjem prve metode

test_num	gen_size	max_iter	mut_rate	cross_type	sel_type	best fitness
1	10	5	0.2	dvopoziciona	ruletska	740
2	20	10	0.1	potpuna	ruletska	2120
3	50	10	0.15	dvopoziciona	ruletska	2020
4	10	10	0.15	potpuna	ruletska	2070
5	10	10	0.1	jednopoziciona	turnirska 2	840
6	20	10	0.1	jednopoziciona	turnirska 3	2170
7	20	10	0.2	dvopoziciona	turnirska 4	2070
8	20	10	0.2	dvopoziciona	turnirska 4	2120
9	50	10	0.1	jednopoziciona	turnirska 5	2220
10	50	10	0.15	potpuna	turnirska 3	2070
11	50	10	0.2	dvopoziciona	ruletska	2170
12	100	15	0.25	potpuna	turnirska 3	2170



Slika 2: Statistike o načinu na koji je partija završena (test 9)

3 Implementacija genetskog algoritma korišćenjem druge metode (Super Mario pristup)

Ova metoda je nastala inspirisana igricom Infinite Mario Bros. Konkretno, održano je takmičenje na temu ko može da napravi boljeg "bota"koji će prelaziti nivoe i igrati slično kao čovek [1]. Ovaj algoritam, za razliku od našeg, ne oslanja se na stanja u kojima se igrač (bot koga podučavamo)nalazi. Jedan hromozom je vektor celih brojeva (izmedju 0 i 7), dužine 60. Dužina vektora predstavlja koliko akcija će biti izvršeno, dok celi brojevi predstavljaju moguće akcije. Robot ima 8 različitih akcija: osnovne - hodaj levo, hodaj desno, skoči u mestu, pucaj, ispali laser i kombinacije - skoči desno, skoči levo, skači i pucaj. Jedna akcija traje jednu sekundu, koliko otprilike traje skok na osnovu kog smo postavili taj uslov. U prvoj metodi uspešan prelazak jednog nivoa u proseku traje 30 sekundi. Taj podatak smo pokušali da primenimo ovde korišćenjem vektora dužine 30. To se pokazalo kao premalo pa je vrednost podignuta na 60. Dakle, pri prelasku nivoa nije uzeto u obzir okruženje robota, on samo prati instrukcije zadate vektorom i odrađuje akciju koja je tada na redu.

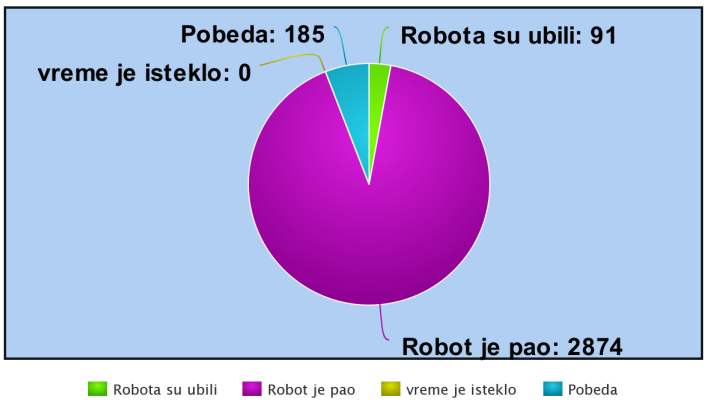
Fitnes je izračunavan potpuno isto kao i korišćenjem prethodne metode. Pri prvom kreiranju populacije napravljeni su vektori nasumičnih celih brojeva, dužine 60. Selekcija i ukrštanje su isti kao u prethodnoj metodi. Mutacija se vrši na svakom polju vektora sa određenom verovatnoćom. U prvoj metodi vektor je bio mali, pa nije bilo potrebe za mutacijom na svakom elementu, ali u ovoj metodi promena na jednoj poziciji nema značajan uticaj na fitnes jedinke.

Tabela 2: Rezultati učenja robota genetskim algoritmom, korišćenjem druge metode

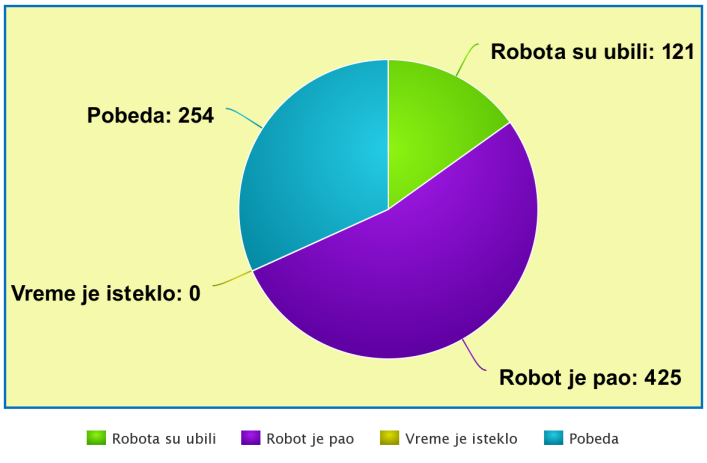
test_num	gen_size	max_iter	mut_rate	cross_type	sel_type	best fitness
1	10	5	0.1	jednopoziciona	turnirska 3	-550
2	20	5	0.2	jednopoziciona	turnirska 4	-380
3	50	5	0.2	jednopoziciona	turnirska 5	-380
4	50	15	0.2	potpuna	turnirska 5	330
5	150	25	0.2	potpuna	turnirska 20	-400
6	150	25	0.2	potpuna	turnirska 20	1670
7	150	25	0.1	potpuna	ruletska	1770
8	150	25	0.2	dvopoziciona	ruletska	1870
9	100	15	0.2	dvopoziciona	ruletska	1770
10	20	15	0.15	specijalna	turnirska 3	1670
11	50	15	0.2	specijalna	turnirska 5	1770
12	150	25	0.2	specijalna	turnirska 5	1770

3.1 Testiranje i rezultati druge metode

Kao i u prvoj metodi testiramo algoritam menjanjem ulaznih parametara. U prvih pet pokušaja, prikazanih tabelom 2, nisu korišćeni ni elitizam ni usmerena kreacija. Pošto su rezultati bili relativno loši promenjen je način pravljenja inicijalne populacije. Skoči desno je najznačajnija akcija zato što je neophodna da bi robot stigao do kraja nivoa. Pri usmerenoj



Slika 3: Statistike o načinu na koji je partija završena (test 8)



Slika 4: Statistike o načinu na koji je partija završena (test 11)

kreaciji povećana je verovatnoća da se baš ta akcija nađe u vektoru hromozoma. Ovo je dovelo do znatnog poboljšanja rezultata. Posle standardnih metoda pokušali smo jednu naprednu, koja koristi indeks (u vektoru hromozoma) akcije koja je dovela do smrti robota. Ukrštanje dve jedinke je slično kao jednopoziciono ali izabrani indeks ukrštanja nije nasumičan već je to prethodno pomenuti indeks. Ideja je sledeća: iz prvog roditelja uzimamo sve akcije koje prethode onoj koja je dovela do smrti, dok iz drugog roditelja uzimamo sve akcije koje se nalaze na pozicijama većim od pozicije smrti prvog roditelja. Za drugo dete analogno suprotno. Što se tiče mutacije, sada se vrši od pozicije smrti pa do kraja vektora, zato što je robot sigurno živ pre toga tako da taj deo ne želimo da menjamo. Na slici 3 predstavljena je statistika o načinu na koji je partija završena korišćenjem običnih parametara. Ovi izveštaji korišćeni su kako bismo znali

šta najčešće prouzrokuje smrt robota i kako bismo mogli da unapredimo algoritam. Uzrok najvećeg broja smrti je padanje sa platforme, na osnovu te činjenice je implementirana usmerena kreacija. Korišćenjem specijalne metode nije dobijen bolji fitnes, ali kao što se može videti na slici 4 broj pobeda se znatno povećao.

4 Zaključak

Primenom obe metode dobijena su relativno dobra rešenja. Jedan od predloga za projekat je bio da izgradimo deterministički algoritam koji proverava rešenje, ali u ovom radu bi to bilo redudantno. Budući da smo implementirali igricu i načine na koje može da se pređe nivo, znamo koji je najbolji način za njeno rešavanje barem kada je prva metoda u pitanju. Što se tiče druge metode, kao provera korišćena je prva metoda zato što se ona pokazala kao bolja. Na osnovu rezultata primećeno je da korišćenjem prve metode robot češće stiže do kraja nivoa. Pre implementacije napredne tehnike korišćene pri drugoj metodi, robot je najčešće završavao nivo padom sa platforme. Njenim korišćenjem došlo je do znatnog poboljšanja rezultata, većina jedinki završava nivo bliže cilju. Nažalost ni korišćenjem specijalne metode nije pronađen način za treniranje robota tako da ubija neprijatelje i time poveća svoj fitnes. Iako su primenom prve metode dobijeni bolji rezultati, njena implementacija je dosta komplikovana. Bilo kakve izmene nivoa ili robota bi drastično promenile implementaciju prve metod. Druga metoda je robusnija u tom pogledu, izmene nivoa nemaju uticaj na nju, ali izmene u akcijama koje robot može da preduzme imaju. Najbolja rešenja druge metode prouzrokuju nasumične i bespotrebe pokrete, dok je kod prve metode kretanje maksimalno efikasno. Ako povećamo broj jedinki i broj iteracija algoritma druga metoda ima potencijal da se poboljša, dok prva metoda ne pokazuje bolje rezultate. U zavisnosti od ciljeva i resursa obe metode mogu biti korisne, teško je reći koja je definitivno bolja.

Literatura

[1] Alejandro Baldominos, Yago Sáez, Gustavo Recio, and Francisco Calle. *Learning Levels of Mario AI Using Genetic Algorithms*, volume 9422, pages 267–277. 11 2015.

[2] Andries P. Engelbrecht. *Computational Intelligence An Introduction – 2nd ed.* John Wiley and Sons Ltd, University of Pretoria South Africa, 2007.

[3] Mladen Nikolić Predrag Janičić. Veštačka inteligencija, 2020. on-line at: http://poincare.matf.bg.ac.rs/~janicic//courses/vi.pdf.