Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK (Upisuje biblioteka, ostaviti jedan red)

UPOTREBA EVOLUTIVNIH ALGORITAMA ZA PRONALAŽENJE REŠENJA U IGRAMA DEKODIRANJA

USE OF EVOLUTIVE ALGORITHMS FOR FINDING SOLUTIONS IN DECODING GAMES

Goran Rajn, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu je opisana implementacija genetskog algoritma koji je primenjen na pronalaženje rešenja u igri Mastermind. Kvalitet rada aplikacije je meren na osnovu ukupnog broja pokušaja koji je algoritmu bio potreban za nalaženje rešenja, zavisno od veličine prostora pretrage.*

Abstract – *This paper describes implementation of genetic algorithm which is used to find solution in game Mastermind. Quality of application is measured as total number of attempts that algorithm requires before finding solution, depending on size of search space.*

Ključne reči: *Genetski algoritam, Mastermind, pretraga, Skočko.*

1. UVOD

Svet koji vidimo danas, sadržan je od velikog broja različitih vrsta organizama koji su izuzetno dobro prilagođeni u okruženju u kojem borave. To je rezultat evolucije, tj. procesa koji se u prirodi odvija već tri milijarde godina. Složenost i visoka prilagođenost današnjih životnih formi postignuta je rafinisanjem i kombinacijom genetskog materijala u tom vremenskom periodu [1].

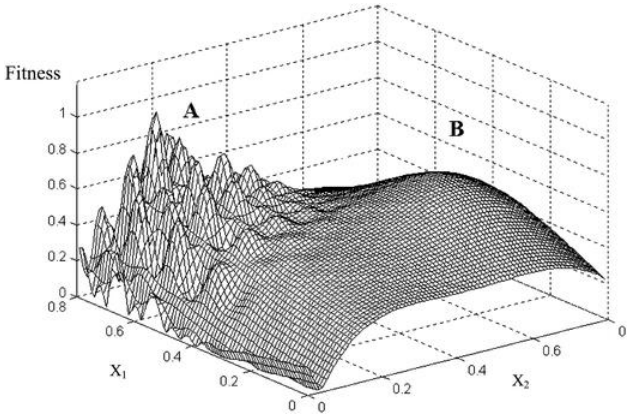
Danas, područje primene genetskih algoritama je veliko, pre svega zato što efikasno nalaze zadovoljavajuća rešenja na velikim prostorima pretrage, za šta bi tradicionalnim metodama trebalo mnogo više vremena. Neke od oblasti u kojima se koriste su : Auto-moto industrija, robotika, evolucija hardvera, pravljenje rasporeda, izračunavanje putnih ruta i regulisanje saobraćaja, industrija igara, bankarstvo, razvoj strategija investiranja, marketing, itd.

Kroz ovaj rad je opisana implementacija aplikacije koja koristi ovu vrstu algoritama kako bi došla do rešenja u igri Mastermind u konačnom broju pokušaja. Rešenje predstavlja kombinaciju nekoliko znakova, gde mogu postojati i duplikati, iz konačnog skupa mogućih znakova. Od veličine tražene kombinacije zavisi i veličina prostora pretraživanja nad kojim će genetski algoritam da vrši pretragu.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Đorđe Obradović, red.prof.

1.1 Prostor pretrage

Populacija individua se održava unutar prostora pretrage, pri čemu svaka jedinka predstavlja potencijalno rešenje problema. Jedinke su kodirane kao konačan vektor bita, realnih brojeva, komponenti, promenljivih, ili nekih drugih struktura podataka. U svetlu analogije sa genetikom, te životne forme su uporedive sa hromozomima, dok su strukture podataka korištene u kodiranju jednake genima. Hromozom (potencijalno rešenje) je sačinjen od nekolicine gena (struktura podataka). Ukoliko bi se populacija individua prikazala kao "oblak tačaka" u prostoru, on bi se kretao njime sa svakom novom iteracijom, vođen procesom evolucije, u pravcu rešenja. Na slici 1 prikazana je jedna generacija kandidata. Vertikalna osa označava vrednost prilagođenosti individua, dok ostale dve predstavljaju vrednosti drugih osobina jedinki.



Slika 1. *Prstor pretraživanja u kojem moguća rešenja poseduju dve osobine*

Genetski algoritam održava populaciju konstantnog broja jedinki sa dodeljenim vrednostima prilagođenosti. Smena generacija sa sobom donosi i nova moguća rešenja, koja u proseku sadrže više dobrih gena od tipične jedinke iz skupa roditelja.

1.2 Evaluacija rešenja

Funkcija prilagođenosti zavisi od problema koji se posmatra, ali u svakom slučaju, to je funkcija koja kao ulaz uzima jedinku a kao izlaz vraća realan broj [2].

Računanje vrednosti prilagođenosti članova populacije se ponavlja često u toku izvršavanja algoritma tako da bi ono trebalo da bude što brže. Sporo izračunavanje fitness-a bi moglo negativno da se odrazi na genetski algoritam i učini ga nedopustivo sporim.

Evaluaciona funkcija treba da ispunjava dva osnovna uslova:

1) Njeno izračunavanje bi trebalo biti dovoljno brzo.
2) Mora kvantitativno da izrazi koliko su data rešenja kvalitetna.

Rešenje problema je dobro onoliko koliko je dobra i funkcija prilagođenosti. Izbor dobre evaluacione funkcije često predstavlja najteži zadatak, pre svega zbog toga što je izuzetno zavisna od problema koji se rešava i ne postoji generalno pravilo prilikom njenog izbora.

1.3 Genetski operatori

Varijacije osobina životnih formi su neophodne u procesu evolucije. Genetski operatori korišteni u ovim algoritmima omogućavaju te promene i analogni su onima u prirodi:

1) Selekcija ili preživljavanje prilagođenijih
2) Reprodukcija ili ukrštanje
3) Mutacija

Ove tri vrste operatora moraju da rade u međusobnoj sprezi kako bi algoritam bio uspešan. Mutacija obezbeđuje raznovrsnost u populaciji slučajnim promenama gena unutar hromozoma, dok se selekcijom i ukršatanjem postojećih rešenja omogućava vođenje pretrage u novim pravcima.

2. SPECIFIKACIJA I IMPLEMENTACIJA SOFTVERA

Arhitektura softvera pretstavlja opis podsistema i komponenti softverskog sistema zajedno sa njihovim međusobnim vezama. Podsistemi i komponente mogu biti specificirane iz više uglova sa ciljem ilustrovanja funkcionalnih i nefunkcionalnih osobina softverskog sistema. Softverska arhitektura sistema pretstavlja proizvod koji je posledica aktivnosti projektovanja softvera [3]. Namena ovog rešenja je da demonstrira upotrebu genetskog algoritma u što bržem rešavanju igre Mastermind, kao i da prikaže napredak u procesu pretrage kroz svaki korak. Na slici 2 je predstavljen dijagram slučajeva korišćenja aplikacije sa osnovnim funkcionalnostima.

2.1 Model

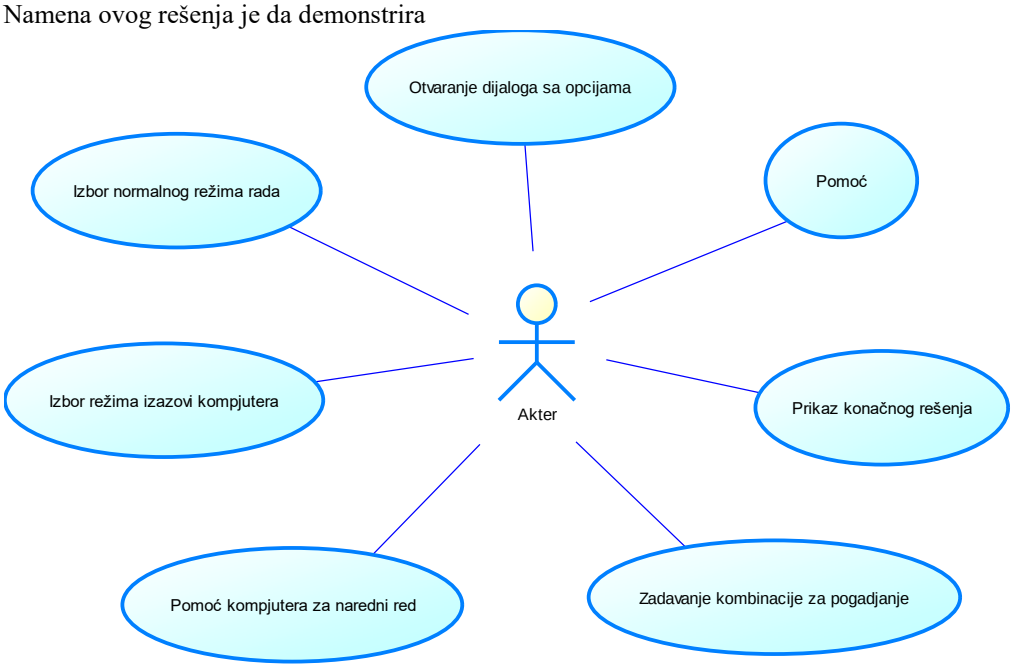
Model predstavlja prezentaciju ili apstrakciju realnosti. On je najčešće pojednostavljeni prikaz realnosti, s obzirom da je ona suviše kompleksna da bi se tačno opisala. Pri rešavanju određenog problema, veći deo kompleksnosti u suštini je irelevantan. Pri izradi modela, stepen pojednostavljena i apstrakcije zavisiće od ugla iz koga se realnost posmatra i potrebnog nivoa detaljnosti [3]. Elementi modela predstavljeni su klasama *Individual, Population, Generation i Utility*. Na slici 3 prikazan je dijagram klasa koje predstavljaju elemente modela.

Klasa *Individual* predstavlja jednog člana populacije, odnosno potencijalno rešenje problema. Sadrži informacije o individui kao i metode za njihovo podešavanje i inicijalizaciju.

Element modela *Population* opisuje populaciju kandidata. Sadži informacije o opštoj prilagođenosti, kao i o prilagođenosti najboljeg pripadnika. Ovde su sadržani genetski operatori (selekcija, mutacija i rekombinacija) koji se koriste pri vođenju pretrage u vidu metoda, kao i metode za dobavljanje i podešavanje indormacija o populaciji.

Klasa *Generation* u sebi sadrži parametre potrebne za vođenje genetskog algoritma, informacije o populacijama roditelja i naslednika, kao i metode za manipulisanje tim podacima.

Utility implementira interfejs *IUtility* i predstavlja poveznicu između ostalih elemenata modela i kontrolera. Pozivom njegovih metoda dobavljaju se i podešavaju podaci sadržani u prethodno opisanim klasama. Sadrži informacije koje se tiču same igre i po potrebi ih prosleđuje ostalim elementima modela u kojima je

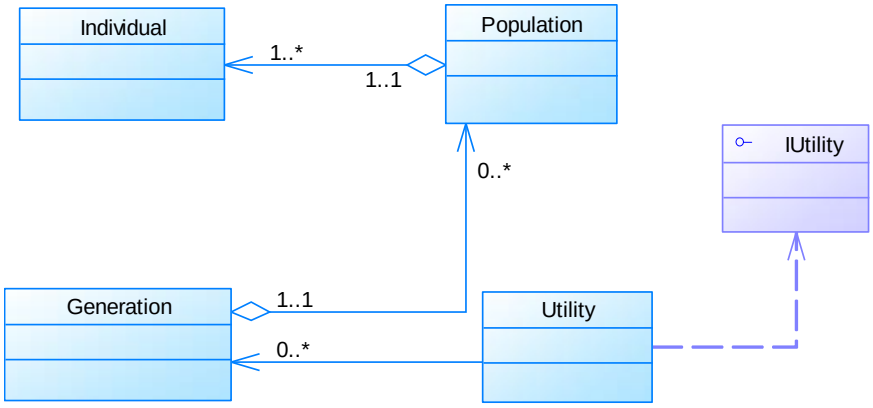


Slika 2. *Dijagram slučajeva korišćenja aplikacije*

sadržana sva funkcionalnost genetskog algoritma. Operacije ove klase su navedene u interfejsu *IUtility*. Kontroler u svom konstruktoru uzima instancu klase koja implementira taj interfejs i time se povećava otpornost na eventualne izmene.

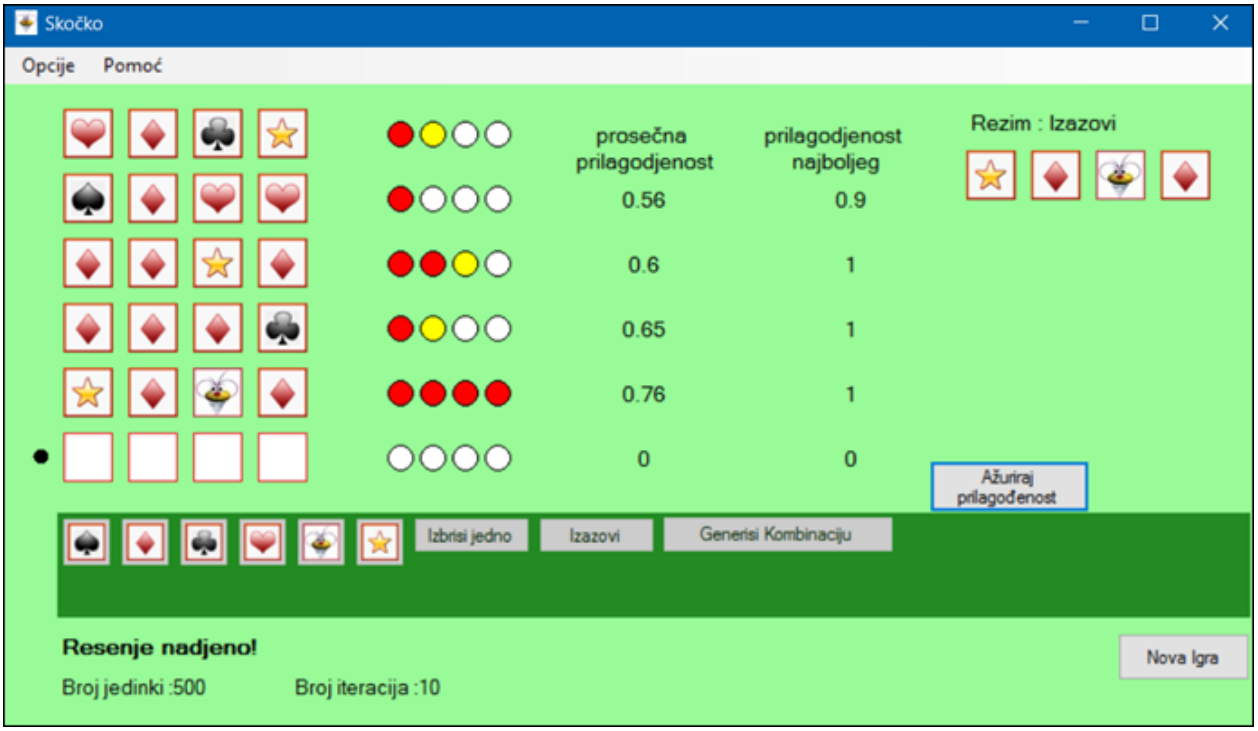
2.2 Korisnički interfejs

Prostor u kojem se dešava interakcija između čoveka i računara naziva se korisnički interfejs. Cilj njegovog dizajniranja je da omogući lako, efikasno i prijateljski naklonjeno okruženje iz kojeg bi se upravljalo računarom



Slika 3. *Dijagram klasa modela podataka*

i postigli željeni rezultati uz minimum napora [4].
Osnovni elementi ovoga interfejsa su implementirani u vidu korisničkih kontrola (user controls), koje se nalaze na glavnom prozoru aplikacije. Korisničke kontrole predstavljaju podesivu grupu kontrola i ponašanja koje se grupišu kako bi se naknadno koristile u istom sastavu.
Klase koje prestavljaju implementaciju ovih elemenata su: *TablePanel, CombinationPanel, FeedbackPanel, FitnessPanel i ChoicePanel.* Na slici 4 prikazan je glavni prozor aplikacije zajedno sa navedenim komponentama.



Slika 4. *Glavna forma aplikacije*

3. VERIFIKACIJA REŠENJA

Merenje uspešnosti rada aplikacije vršeno je eksperimentalno na osnovu ukupnog broja pokušaja koji je potreban da se pronađe rešenje. Pri kombinaciji 4 znaka od 6 mogućih (P=4,N=6), algoritmu demonstriranom u ovom radu bilo je potrebno u proseku 4,51 pokušaja da bi pronašao rešenje. Ovaj rezultat ne odstupa mnogo od rezultata koje postižu algoritmi implementirani u prošlosti na istu temu (Knuth, Irving, Norvig, Rosu) pri čemu je vreme izvršenja prezentovanog rešenja kraće nego kod navedenih agoritama, naročito pri kombinaciji više znakova. Napredak između svakog ciklusa iteracija može se pratiti i kroz prosečnu prilagođenost najboljih jedinki iz svakog pokušaja kao što je prikazano na slici 5. U proseku povećanje ovih vrednosti za kombinacije od 8 znakova iznosi oko 4%, dok je za kombinacije sa manje znakova ovaj raspon veći i iznosi 8% u proseku.

Slika 5 *Primer povećanja prosečne prilagođenosti individua*

4. ZAKLJUČAK

Genetski algoritmi, kao jedan od pravaca evolutivnog računarstva, predstavljaju metode za rešavanje optimizacionih problema, bazirane na prirodnoj selekciji, tj. procesu kojim je vođena biološka evolucijia. Konstantne izmene na populaciji individua upotrebom genetskih operatora dovode do njenog „evoluiranja“ u pravcu optimalnog rešenja. U ovom radu je prezentovana jedna implementacija takvog algoritma čija je namena rešavanje igre Mastermind koja zahteva kratko vreme izvršavanja i što manji broj pokušaja pronalaska tražene kombinacije. Kvalitet opisanog rešenja je meren u odnosu na algoritme koji su nastali ranije a primenjivani su na isti problem. Iz tog posmatranja je zaključeno da su performane opisanog algoritma uporedive sa ranijim rešenjima pri standardnim podešavanjima. Takođe, primetan je uspon prosečne prilagođenosti svakog pokušaja, naročito za veće kombinacije znakova,a samim tim i većim prostorom pretraživanja.

5. LITERATURA

[1] Ulrich Bodenhofer, Genetic Algorithms : Theory and Applications
[2] Stuart J. Russell and Peter Norvig,
Artificial Intelligence: A Modern Approach
[3] Branko Perišić, Projektovanje softvera, materijali sa predavanja, Univerzitet u Novim Sadu
[4] User Interface,
https://en.wikipedia.org/wiki/User_interface

Kratka biografija:

Goran Rajn je rođen 04.10.1990. godine u Somboru. Diplomirao je na Fakultetu tehnikih nauka 2013. godine na odseku Računarstvo i automatika. Iste godine je upisao Master akademske studije na istoimenom odseku. Položio je sve ispite predviđene planom i programom.