Genetski algoritam za rešavanje Sudoku-a

Seminarski rad u okviru kursa
Računarska inteligencija
Matematički fakultet

Ana Miloradović, Stefan Jaćović
ana.miloradovic7@gmail.com, stefanjacovic25@gmail.com

14. april 2020.

Sažetak

U okviru ovog rada predstavljen je primer implementacije genetskog algoritma za rešavanje popularne logičke igre Sudoku. Takođe, dat je i primer implementacije rešenja tehnikom backtracking i napravljena je analiza učinka obe tehnike. Obe tehnike implementirane su korišćenjem Python3.

Sadržaj

1 Uvod

Sudoku je popularna logička slagalica čija je pojava prvi put zabeležena 1892. godine i njen izgled je bio dosta drugačiji nego danas. Naime, današnji izgled sudokua je zasluga mnogobrojnih modifikacija iz prošlosti, pa je tako ova igra ranije podrazumevala polje koje je imalo samo četiri kvadrata, a ne devet kao što je slučaj danas.
Činjenica da igru mogu igrati ljudi širom sveta- bez obzira na jezik, jer su brojevi, slova i pravila svuda ista, donela je ovoj igri svetsku popularnost. Danas, gotovo da ne postoji osoba na svetu koja nije čula za ovu igru.

1.1 Nastanak i poreklo igre Sudoku

Čuveni švajcarski matematičar **Leonhard Euler** sačionio je tzv. Latinski kvadrat , koji se sastojao od NxN polja obeleženih sa N različitih simbola tako da se svaki simbol u svakom redu i svakoj koloni samo jedanput ponavlja. Koristeći ovaj koncept, 1979.godine na Manhattan-u (New York), američki časopis **"Dell Math Puzzles & Logic Problems"**, koji je inače objavljivao razne ukrštenice i zagonetke, je po prvi put objavio kvadratnu slagalicu dimenzija 9x9, koja se sastoji od 9 manjih kvadrata dimenzija 3x3, tj. upravo ono što danas nazivamo Sudoku.
Popularno ime koje ova slagalica nosi, osmislili su Japanci kao skraćenicu fraze "Suuji wa dokushin ni kagiru", što grubo prevedeno znači "brojevi moraju ostati jedinstveni". Iako ime potiče iz Japana, ova logička igra ima potiče iz **Evrope i Amerike**.
Sredinom osamdesetih godina prošlog veka Sudoku doživljava pravi prodor u **Japanu**, kada je prvi put objavljena u njihovom poznatom časopisu "Nikoli". Novozelanđanin Wayne Gould je na svom putovanju po Japanu naučio da igra Sudoku, oduševio se njom, i narednih nekoliko godina razvijao kompjuterski program za generisanje ovih slagalica. Krajem 2004. godine uspeo je da ubedi londonske novine "The Times" da koristeći njegov program štampa dnevne Sudoku slagalice. Ubrzo su i druge britanske novine počele da štampaju svoje Sudoku slagalice. Igra je postala jako popularna, ne samo u Engleskoj, vec i u drugim zemljama: Nemačkoj, Austriji, SAD-u,...

1.2 Pravila igre

Pravila igre su vrlo jednostavna, iako rešenje često nije. Slagalicu čini kvadrat. Broj polja je najčešće 81 (9×9), koji je podeljen na manje kvadrate (podkvadrate) dimenzija 3×3 polja. Kao početni „tragovi", u nekoliko polja su unešene cifre, a cilj igre je da se ostala prazna polja popune tako da na kraju svaki red, kolona i podkvadrat sadrze cifre od 1 do 9 samo jedanput.
Sudoku slagalice su u zavisnosti od težine uglavnom podeljene na **„lake"**, **„srednje"** i **„teške"** . Treba naglasiti da težinu slagalice ne određuje samo broj unapred datih cifara, već i njihov razmeštaj u kvadratu.
Sudoku slagalica po pravilu ima samo **jedno moguće rešenje** (može ih imati više u slučaju da je na početku otkriven mali broj polja). U međuvremenu su se pojavile i slagalice sa 16×16 polja kao i druge još komplikovanije varijante, na primer kombinacija igara Sudoku i Kakuro nazvana "Killer Sudoku". Postoji i Sudoku za decu, koji se sastoji od malog broja polja.

		4	6			1		
3	6			9	2	4		7
7					4		3	6
	8		9	2		5		3
6		3				7		8
4		2		3	8		9	
8	3		2					4
9		6	3	4			1	2
		1			9	3		

(a) Početna Sudoku postavka

5	2	4	6	7	3	1	8	9
3	6	8	1	9	2	4	5	7
7	1	9	5	8	4	2	3	6
1	8	7	9	2	6	5	4	3
6	9	3	4	5	1	7	2	8
4	5	2	7	3	8	6	9	1
8	3	5	2	1	7	9	6	4
9	7	6	3	4	5	8	1	2
2	4	1	8	6	9	3	7	5

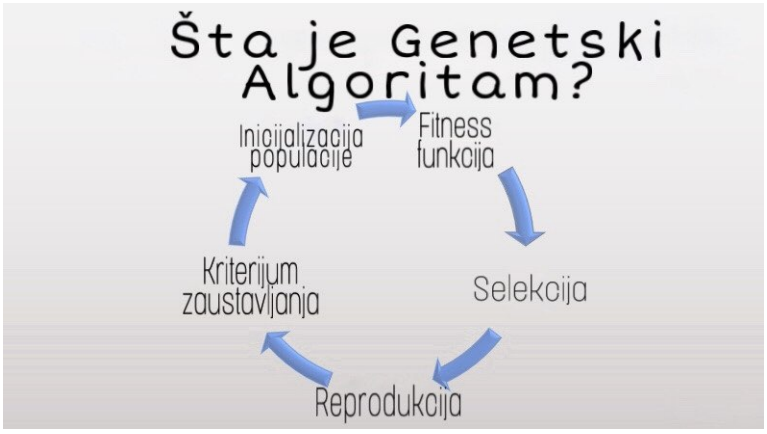
(b) Rešena Sudoku slagalica

Slika 1: Primer slagalice Sudoku

1.3 Sudoku i algoritmi

Činjenica da niko ne zna algoritam kojim se uvek dolazi do rešenja bez prethodnog isprobavanja ogromnog broja kombinacija stavlja Sudoku u klasu tzv. NP-potpunih problema. Za rešavanje ove igre mogu se koristiti razni algoritmi, među kojima prednjači **backtracking** posebno za rešavanje "težih" slagalica kao i onih dimenzija 9×9. Takođe, koriste se i prirodom inspirisani optimizacijski algoritmi- genetski algoritam, simulirano kaljenje, optimizacija rojem čestica itd. **Genetski algoritmi** su se pokazali kao vrlo efikasni u rešavanju NP problema, međutim za rešavanje Sudoku i nisu baš efikasni (spora konvergencija, nemogućnost izlaska iz lokalnih minimuma ...)

1.4 Genetski algoritam



Slika 2: Faze genetskog algoritma

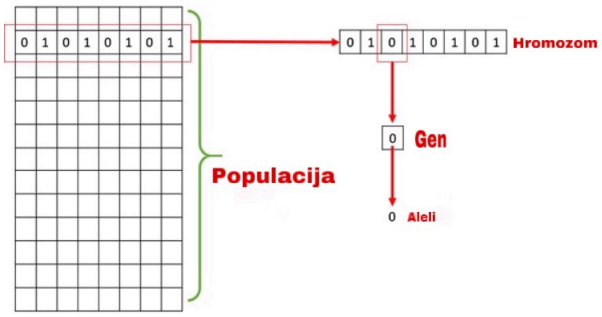
Genetski algoritam je pretraživačka optimimizaciona tehnika, zasnovana na principima genetike i prirodne selekcije. Pripada većoj kla-

si evolucionih algoritama. Koristi se da se nađu optimalna ili približno-optimalna rešenja za teške probleme. Često se koristi za rešavanje optimizacionih problema, u istraživanjima i u mašinskom učenju.

1.5 Faze genetskog algoritma

Na prethodnoj fotografiji imamo slikoviti prikaz faza genetskog algoritma. One su:

1. **Inicijalizacija populacije (kodiranje):**
   - Svaki **gen** predstavlja parametar u okviru rešenja.
   - Ovaj skup parametara koji formira rešenje naziva se **hromozom**.
   - **Populacija** je kolekcija hromozoma.
   - Redosled gena u hromozomu je bitan.
   - Uglavnom, hromozomi su prikazani u binarnim formatima kao 0 i 1, ali moguca su i drugačija kodiranja.



Slika 3: Osnovni pojmovi GA

2. **Funkcija fitnesa:**
   - Od raspoloživih hromozoma moramo odabrati one koji su najbolji za reprodukciju potomka, tako da se svakom hromozomu daje fitnes vrednost .
   - Fitness vrednost pomaže u odabiru pojedinaca koji ce se koristiti za reprodukciju.
   - Funkcija za fitnes je definisana u genetskoj reprezentaciji i meri kvalitet predstavljenog rešenja i ona uvek zavisi od problema.
   - U nekim slučajevima, teško je ili nemoguće odredititi fitnes.
3. **Selekcija:**
   - Tokom svake sledeće generacije deo postojeće populacije je izabran da odgoji novu generaciju.
   - Individualna rešenja su izabrana pomoću procesa baziranog na fitnesu(zdravlju), gde više fit(zdravija, sa boljom kondicijom) rešenja(merena fitnes funkcijom) imaju veće šanse da budu izabrana. Određene metode selekcije ocenjuju kondiciju svakog rešenja i prvenstveno će izabrati najbolja rešenja.

- U procesu selekcije, bira se hromozom za reprodukciju.

4. **Reprodukcija:** Pravljenje potomstva se dešava na dva načina: **Ukrštanjem** i **Mutacijom**
   (a) Ukrštanje je najbitnija faza u genetskom algoritmu. Tokom crossovera, bira se slučajna tačka i onda se ukršta par roditelja kako bi nastali potomci. Postoje 3 glavne vrste ukrštanja:
      i. **Jednopoziciono ukrštanje:** Tačka na hromozomima oba roditelja bira se nasumično i označava se kao „crossover tačka“. Bitovi desno od te tačke razmenjuju se između dva matična hromozoma.
      ii. **Dvopoziciono ukrštanje:**Dve tačke ukrštanja izabrane su nasumično iz roditeljskih hromozoma. Bitovi između dve tačke se menjaju između roditelja
      iii. **Uniformno ukrštanje:** U uniformnom ukrštanju, obično se svaki bit bira između bilo kog roditelja s jednakom verovatnocom.

   Na kraju ukrštanja novo potomstvo se dodaje populaciji.
   (b) **Mutacija:** U nekoliko formiranih novih potomaka, neki od njihovih gena mogu se podvrgnuti mutacijama uz malu slučajnu verovatnocu. Ovo ukazuje da se neki bitovi u hromozomu mogu okrenuti. Mutacije se rade da bi se održala raznolikosti populacije i da bi se sprečila preuranjena konvergencija.
5. **Konvergencija (kriterijum zaustavljanja):**
   Nekoliko pravila koja slede govore o tome kada se treba zaustaviti:
   - Kada ne dođe do poboljšanja kvaliteta rešenja nakon završetka određenog broja generacija zadatih unapred.
   - Broj generacija je dostigao maksimalan broj generacija.
   - Sličnost između jedinki u okviru jedne generacije je prevelika.
   - Kada se dobije prihvatljivo rešenje.

1.6 Prednosti i mane genetskog algoritma

- **Prednosti GA:**
  - ☐ Ne zahteva nikakve derivatne informacije (koje možda nisu dostupne za mnoge probleme u stvarnom svetu).
  - ☐ Brži je i efikasniji u poređenju sa tradicionalnim metodama.
  - ☐ Ima vrlo dobre paralelne mogucnosti.
  - ☐ Optimizira i kontinuirane i diskretne funkcije, kao i više objektivne probleme.
  - ☐ Pruža listu „dobrih“ rešenja, a ne samo jedno rešenje.
  - ☐ Uvek dobije odgovor na problem koji se vremenom popravlja.
  - ☐ Korisno kada je prostor za pretragu veoma velik i ako je uključen veliki broj (proveri sta) parametara.
- **Mane GA:**
  - ☐ Nisu pogodni za sve probleme, posebno probleme koji su jednostavni i za koje su izvedene informacije o derivatima.
  - ☐ Vrednost fitnesa izračunava se više puta, što može biti skupo.
  - ☐ Buduci da je stohastičan, nema garancija za optimalnost ili kvalitet rešenja.
  - ☐ Ako se ne sprovede pravilno, GA se možda nece približiti optimalnom rešenju.

2 Opis rešenja zadatog problema

U ovom poglavlju, opisaćemo način na koji smo implementirali algoritme za rešavanje Sudoku. Prvo rešenje implementirano je primenom **GA**, gde ćemo opisati sve njegove faze: funkciju cilja koju smo koristili, odabir selekcije, ukrštanje, mutaciju, kriterijum zaustavljanja kao i nešto više reći o svakoj primenjenoj metodi. Drugo rešenje implementirano je **brute-force** metodom, o kojoj će takođe biti reči u ovom poglavlju.

2.1 Implementacija rešenja Genetskim Algoritmom

2.1.1 Generisanje inicijalne populacije

Inicijalna populacija se generiše slučajno i sadrži N_c jedinki. Prateći pravila za popunjavanja kolona, vrsta i 3x3 blokova koje Sudoku nalaže, polja početne populacije se popunjavaju, vrši se tzv. sejanje već korišćenih (poznatih) hromozoma. Ažurira se prilagođenost(engl. fitness function) svake jedinke i populacija se sortira na osnovu te vrednosti.

2.1.2 Funkcija prilagodjenosti

2.1.3 Selekcija

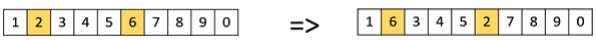
Implementiran je operator **turnirske selekcije**. Iz populacije se na slučajan način biraju 2 jedinke, zatim se od izabranih se uzima najbolja u smislu vrednosti funkcije prilagođenosti. Za parametar 'selection rate' koristili smo vrednost 0.85, čime smo obezbedili da bolje prilagodjena jedinka ima verovatnoću da pobedi 0.85, dok manje prilagodjena ima 0.15. Pobednik turnira prolazi dalje i učestvuje u stvaranju naredne populacije jedinki.

2.1.4 Ukrštanje

Ukrštanje se odvija kombinovanjem dvopozicionog i cikličnog ukrštanja. Najpre, na slučajan način se biraju dve tačke ukrštanja, pri čemu ukrštanje mora da sadrži najmanje jedan red a najviše osam. Prva jedinka potomak uzima deo koda rešenja prvog roditelja do tačke ukrštanja i nakon tačke ukrštanja. Analogno, druga jedinka potomak uzima na isti način deo koda drugog roditelja. Zbog činjenice da su kolone, vrste i blokovi slagalice Sudoku permutacije, koristimo ciklično ukrštanje. Primenjujemo ga na deo ograđen tačkama ukrštanja i u njemu učestvuju delovi između tačaka ukrštanja oba roditelja.

2.1.5 Mutacija

Do mutacije nekog gena dolazi sa verovatnoćom mutation_rate. Kod našeg problema pogodno je bilo koristiti tzv. **swap mutation**, odnosno mutaciju zasnovanu na zameni, jer se ona obično koristi u kodiranjima baziranim na permutaciji. Kod swap mutacije izaberemo nasumično dva gena na hromozomu i zamenimo im vrednosti.



Slika 4: Mutacija zasnovana na zameni

2.1.6 Ostali aspekti algoritma

Da se ne bi dogodilo da pojedini potomoci budu lošiji od roditelja ili da najbolja jedinka ne pređe u narednu generaciju koristimo **princip elitizma** i uzimamo 5% najboljih jediniki koje sigurno idu u sledeću populaciju. N_e najboljih jedinki se prenosi u narednu generaciju bez promena Za ostale jedinke se primenjuje gore opisana turnirska selekcija.

2.2 Implementacija rešenja Brute-force metodom

Implementiran je algoritam koji rešava isti problem, problem igre Sudoku, primenom metode **brute-force**. Koristeći pravila za sudoku smanjujemo broj kandidata za rešenje i time ubrzavamo rad brute-force algoritma. Zbog toga, brojeve koji se već nalaze u nekoj vrsti, koloni ili 3x3 kvadratu ne razmatramo kao rešenje. Na pocetku popunjavamo sva ocigledna resenja, odnosno u svako polje u koje se može postaviti samo jedna vrednost, tu vrednost i postavljamo. Kada imamo vise mogućih vrednosti za jedno polje onda uzimamo jednu od vrednosti i rekurzivno pozivamo funkciju resi(), ako dođe do kontradikcije koristimo backtracking i idemo unazad i uzimamo neku drugu vrednost i opet radimo sve isto dok ne dođemo do konačnog rešenja.