

Genetski algoritam za rešavanje Sudoku-a

Seminarski rad u okviru kursa
Računarska inteligencija
Matematički fakultet

Ana Miloradović, Stefan Jačović

ana.miloradovic7@gmail.com, stefanjacovic25@gmail.com

16. april 2020.

Sažetak

U okviru ovog rada predstavljen je način implementacije genetskog algoritma za rešavanje popularne logičke igre (slagalice) Sudoku. Takođe, opisana je i implementacija rešenja tehnikom brute-force, njihovo međusobno poređenje ali i poređenje sa drugim, sličnim implementacijama iz literature.

Sadržaj

1	Uvod	2
1.1	Sudoku	2
1.1.1	Nastanak i poreklo igre Sudoku	2
1.1.2	Pravila igre	2
1.2	Genetski algoritam	3
1.2.1	Način izvršavanja algoritma	3
1.2.2	Inicijalizacija populacije	4
1.2.3	Funkcija prilagođenosti	4
1.2.4	Selekcija	4
1.2.5	Ukrštanje	5
1.2.6	Mutacija	5
1.3	Brute-force	6
2	Opis rešenja zadatog problema	6
2.1	Implementacija rešenja Genetskim Algoritmom	6
2.1.1	Generisanje inicijalne populacije	6
2.1.2	Funkcija prilagođenosti	6
2.1.3	Selekcija	7
2.1.4	Ukrštanje	7
2.1.5	Mutacija	7
2.1.6	Ostali aspekti algoritma	7
2.2	Implementacija rešenja Brute-force metodom	7
3	Eksperimentalni rezultati	8
3.1	Eksperimentalno okruženje	9
4	Zaključak	10
	Literatura	11

1 Uvod

1.1 Sudoku

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 sudoku 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 se Sudoku rešava širom sveta- bez obzira na jezik(jer je na svim jezicima ista), donela je ovoj igri svetsku popularnost. Danas, gotovo da ne postoji osoba na zemlji koja nije čula za Sudoku.

1.1.1 Nastanak i poreklo igre Sudoku

Čuveni švajcarski matematičar **Leonhard Euler** sačionio je tzv. Latinski kvadrat, koji se sastojao od $N \times N$ 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 9×9 , koja se sastoji od 9 manjih kvadrata dimenzija 3×3 , 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 potiče iz **Evrope i Amerike**.

Sredinom osamdesetih godina prošlog veka Sudoku doživljava pravi pro-dor u **Japanu**, kada je prvi put objavljena u njihovom poznatom časopisu "Nikoli".

		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		

Slika 1: Primer slagalice Sudoku

1.1.2 Pravila igre

Pravila igre su vrlo jednostavna, iako rešenje često nije. Slagalicu čini kvadrat. Broj polja je najčešće $81 (9 \times 9)$, 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 sadrže cifre od 1 do 9 samo jedanput[5].

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 slagalice 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.

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

Slika 2: Rešeni primer slagalice Sudoku

1.2 Genetski algoritam

Genetski algoritam je pretraživačka optimizaciona tehnika koja teži da imitira biološku evoluciju koju su definisali Čarls Darwin i Žan-Batist Lamark. Ideja algoritma je da tretira svoje ulazne podatke problema kao hromosome.

Tokom vremena, populacija evoluiru i prilagođava se okruženju. Bolje prilagođene jedinke imaju veću šansu da prežive i učestvuju u razmnožavanju, a time i da prenesu svoj genetski materijal u narednu generaciju. Tako slabije jedinke i njihov genetski materijal postepeno nestaju iz populacije.

Brojni su primeri primene genetskog algoritma, među kojima se najpre svrstava rešavanje NP-teških optimizacionih problema.

Kriterijum zaustavljanja GA može biti dostignut maksimalan broj generacija, maksimalan broj generacija bez poboljšanja najbolje jedinke, isteklo unapred zadato maksimalno vreme izvršavanja algoritma, pronađeno optimalno rešenje i slično[6].

1.2.1 Način izvršavanja algoritma

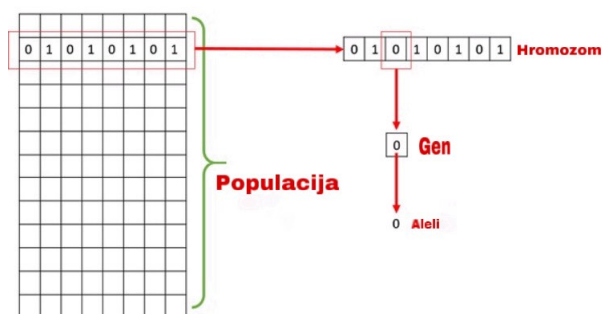
Genetski algoritam polazi od inicijalne populacije jedinki, gde svaka jedinka odgovara jednom rešenju u prostoru pretrage. Potom se iterativno primenjuju genetski operatori na jedinke populacije.

Prilagođenost (engl. fitness function) jedinke se računa na osnovu funkcije cilja. Dalje, jedinke izabrane na osnovu prilagođenosti se **ukrštaju** (engl. crossover) i stvaraju nove jedinke, koje imaju karakteristike oba

roditelje. Dodatno, vrši se **mutacija** (engl. mutation) nekih gena jedinki, kako bi se očuvala raznovrsnost genetskog materijala. Tako dobijene jedinke zamenjuju celu prethodnu populaciju ili njen manje prilagođeni deo.

1.2.2 Inicijalizacija populacije

Populacija je podskup rešenja u trenutnoj generaciji tj. to je skup **hromozoma**. U većini slučajeva, inicijalna populacija se generiše na slučajan način. Međutim, treba imati na umu par bitnih stvari: važno je održavati **raznolikost populacije**, u suprotnom, može doći do prerane konvergencije kao i voditi računa o **veličini populacije** - ne bi trebalo da bude prevelika, jer to može dovesti do usporenja GA ali isto tako ne sme biti ni premala jer nam ne daje dovoljan prostor za raznoliko mešanje hromozoma. Skup **gena** (parametri u okviru rešenja) predstavlja hromozom. Geni su zapravo odgovarajuća svojstva organizma, dok se različite mogućnosti za svako svojstvo naziva genski **alel**[1].



Slika 3: Osnovni pojmovi GA

1.2.3 Funkcija prilagođenosti

Funkcija prilagođenosti (eng. fitness function) nam daje ocenu kvaliteta jedinke. Prilagođenost jedinke se računa na osnovu funkcije cilja. Najčešće se za prilagođenost koristi direktno funkcija cilja, ali pod uslovom da njeno računanje nije vremenski zahtevno. U zavisnosti od konkretnog problema, funkcija cilja se transformiše na različite načine. U nekim slučajevima, teško je ili gotovo nemoguće odrediti ovu vrednost. Funkcija prilagođenosti nam pomaže u odabiru pojedinaca koji ce se koristiti dalje za reprodukciju.

1.2.4 Selekcija

Kao i u prirodi, uloga selekcije kod genetskog algoritma je odabrati jedinke koje će učestvovati u stvaranju nove populacije. Cilj je stvoriti populaciju koja je bolja od prethodne, odnosno sastoji se od jedinki koje imaju bolju fitness funkciju. Zbog toga, genetski materijal za stvaranje nove generacije mora biti kvalitetan i zato se veća verovatnoća pridružuje jedinkama sa boljom prilagođenošću. Međutim, ne treba u potpunosti zanemariti ni one sa lošijom vrednošću, jer svakako postoji verovatnoća da

i one mogu učestvovati u stvaranju deteta sa dobrom prilagođenošću. Iako je pretpostavka da će kvalitetni geni sa velikom verovatnoćom preći u narednu generaciju, to ne mora uvek biti slučaj.

Postoji više tipova operatora selekcije[2].

Kod **rulet selekcije** (engl. roulette wheel selection), verovatnoća izbora neke jedinke je proporcionalna njenoj prilagođenosti. Nedostatak ovakve selekcije jeste mogućnost čestog izbora visoko prilagođenih jedinki, što može dovesti do preuranjene konvergencije.

Turnirska selekcija (engl. tournament selection) predstavlja alternativu rulet selekciji. To je popularna tehnika, gde se populacije dele na grupe od po k jedinki (broj k je manji od ukupne veličine populacije) koje se takmiče za prelazak u narednu generaciju, odigravajući turnire. Turnirska selekcija se takođe naziva i eliminacijska selekcija, upravo zbog svog načina rada[4].

1.2.5 Ukrštanje

Kao što smo naveli, selekcijom izabrane jedinke ulaze u proces reprodukcije, gde se primenom operatora ukrštanja i/ili mutacije proizvodi potomstvo. Ukrštanjem se kombinuju roditeljski hromozomi a rešenje se optimizuje samo u okviru postojeće oblasti. Dakle, to je proces kojim se kombinovanjem proizvoljnog izabranog genetskog materijala dva ili više roditelja formira jedna ili više jedinki neposrednog potomstva tj. dece. Jedinke roditelja izabrane selekcijom ne moraju nužno da se ukrštaju. Svaki od parova hromozoma se ukršta sa nekom verovatnoćom p_c . Verovatnoća (stopa) ukrštanja je uglavnom velika, uzima vrednosti iz intervala $[0.6, 0.9]$. Prilikom selekcije roditelja za ukrštanje potrebno je obezbediti da jedna ista jedinka ne bude odabrana za oba roditelja (u tom slučaju je generisani potomak kopija roditelja), i da ista jedinka ne učestvuje u više od jednog procesa ukrštanja ukoliko se koriste fitnes-proporcionalne sheme selekcije.

Postoje 3 glavne vrste ukrštanja:

1. **Jednopoloziciono ukrštanje:** Tačka na hromozomima oba roditelja bira se nasumično i označava se kao tzv. crossover tačka. Bitovi desno od te tačke razmenjuju se između dva matična hromozoma.
2. **Dvopoloziciono ukrštanje:** Dve tačke ukrštanja izabrane su nasumično iz roditeljskih hromozoma. Bitovi između dve tačke se menjaju između roditelja
3. **Uniformno ukrštanje:** U uniformnom ukrštanju, obično se svaki bit bira između bilo kog roditelja s jednakom verovatnoćom.

1.2.6 Mutacija

Mutacija je takođe operator reprodukcije čijim se delovanjem vrši izmena slučajno odabranih gena jedinke. Kako deluje nad samo jednom jedinkom, mutacija je unarni operator koji kao rezultat daje izmenjenu jedinku. Ovaj operator omogućava vraćanje korisnog genetskog materijala koji je izgubljen u selekciji i ukrštanju. Može se reći i da mutacija predstavlja odličan mehanizam za izbegavanje lokalnih ekstremuma pretragom okoline rešenja, zbog činjenice da proširuje prostor pretrage koji se razmatra uvođenjem nove genetske informacije. Time se povećava šansa za nalazjenje globalnog ekstremuma. Tačnije, ako bi cela populacija završila u nekom od lokalnih ekstremuma, jedino se slučajnom pretragom okoline rešenja pronalazi bolje, te je dovoljno da jedna jedinka nastala mutacijom bude bolja, pa da se cela populacija u nekoliko narednih generacija

“preseli” u okolinu boljeg rešenja. Parametar koji određuje verovatnoću mutacije jednog gena predstavlja nivo mutacije i zadaje se na početku izvršavanja genetskog algoritma. Ako nivo mutacije teži jedinici, algoritam se pretvara u algoritam slučajne pretrage rešenja, a ako teži nuli, može se očekivati zaustavljanje procesa već na samom početku procesa optimizacije u nekom lokalnom ekstremumu. Kod genetskih algoritama, mutacija se obično primenjuje sa niskom verovatnoćom, od 0.001 do 0.01, kako se ne bi narušio genetski materijal jedinki sa dobrom funkcijom prilagođenosti.

1.3 Brute-force

U računarstvu jedna od poznatijih algoritamskih strategija **brute-force** (gruba sila) ili iscrpljujuća pretraga (takode poznata i kao generiši i testiraj), je opšta tehnika rešavanja problema koja se sastoji od sistematičnog nabiranja svih mogućih kandidata za rešavanje i provere da li svaki kandidat zadovoljava problem. Dok je brute-force pretraga jednostavna za primenu i uvek će pronaći rešenje ako postoji, njegova cena srazmerna je broju kandidata rešenja – što u mnogim praktičnim problemima pretenduje veoma brzim rastom kako se veličina problema povećava. Dakle, brute-force pretraga se obično koristi kada je veličina problema ograničena i kada je jednostavnost implementacije važnija od brzine[8].

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 takode biti reći u ovom poglavlju[7].

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 popunjavanje kolona, vrsta i 3x3 blokova koje Sudoku nalaže, prazna polja hromozoma početne populacije se popunjavaju (random). Ažurira se prilagođenost (engl. fitness function) svake jedinke i populacija se sortira na osnovu te vrednosti.

2.1.2 Funkcija prilagođenosti

U skladu sa pravilima Sudoku, da se u svakoj koloni, vrsti i bloku pojavljuju svi brojevi iz [1, 9] tačno jednom, formirali smo tzv. fitness funkciju koja nam računa prilagođenost jedinke. Prolazeći kroz sve kolone, računali smo frekvenciju pojavljivanja svake cifre u koloni a potom izdvojili ukupan broj različitih cifara. Ta vrednost nam govori koliko se cifara od mogućih 9 već nalazi u toj koloni. Izračunata je verovatnoća iskorišćenih cifara u toj koloni, a potom se sumiranjem svih kolona, računa verovatnoća za sve kolone. Analogno se vrši računanje i za vrste i blokove.

2.1.3 Selekcija

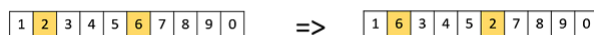
Implementiran je operator **turnirske selekcije**. Iz populacije se na slučajan način biraju 2 jedinke, zatim se od izabranih uzima najbolja u smislu vrednosti funkcije prilagođenosti. Za parametar ‘selection rate’ koristili smo vrednost 0.85, čime smo obezbedili da bolje prilagođena jedinka ima verovatnoću da pobedi 0.85, dok manje prilagođena ima 0.15. Pobjednik 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. Primenujemo ga na deo ograden 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 60% najboljih jedinki 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 početku popunjavamo sva očigledna rešenja, odnosno u svako polje u koje se može postaviti samo jedna vrednost, tu vrednost i postavljamo. Kada imamo više mogućih vrednosti za jedno polje, onda uzimamo jednu od vrednosti i nastavljamo sa rešavanjem, a ako dođe do kontradikcije, koristimo backtracking i idemo

unazad. Tu vrednost brišemo a postavljamo drugu vrednost i ponavljamo proces dok ne dodemo do konačnog rešenja.

3 Eksperimentalni rezultati

Podsetimo se najpre parametara i činjenica koje koristimo u algoritmi-ma. Dakle, u skladu sa pravilima Sudoku, imamo da je dužina hromozoma GA niz celobrojnih vrednosti kojih ima 81, podeljenih u 9 blokova od po 9 brojeva. Takođe, imamo 9 kolona i 9 vrsti od po 9 brojeva. Parametri koje smo koristili prilikom ovog eksperimenta su: veličina populacije 200, veličina generacije 1000, elitizam 120. Verovatnoća mutacije je 0.5. U na-rednoj tabeli prikazaćemo rezultate primenom genetskog algoritma na 4 slagalice različitih težina.

Početni broj polja	25	26	29	31
Procenat uspešnosti	20	33	100	83
Prosečan br. generacija	451	212	22	126
Najbolja generacija	45	50	14	25
Prosečno vreme nalaženja	93.71	71.42	7.41	24.19

Slika 5: Rezultati dobijeni primenom GA

Vršimo poređenje našeg GA sa principom koji je opisan u [akadems-kom radu \[3\]](#).

U ovom radu korišćeni su sledeći parametri: veličina populacije je 500, elitizam 40. Korišćen je višepozicioni i ciklični crossover sa verovatnoćom 0.7, swap mutacija sa verovatnoćom 0.4. Korišćeni kriterijum zaustavljanja je kao i u našem slučaju- pronađeno rešenje ili nakon obrađenih 1000 generacija. Obe implementacije su pokušale da nađu rešenje za svaku od 4 ponuđenih slagalice 100 puta. Testirane slagalice su različitih težina, a broj poznatih polja je između 25 i 36.

U narednoj tabeli prikazani su dobijeni rezultati korišćenjem obe metode. Razmatrana je prosečna generacija nalaženja rešenja, najbolja generacija, procenat uspešnosti, najbolje vreme nalaženja. Primećujemo da za slagalice istog nivoa težine, GA iz ovog rada prikazuje bolje performanse i treba mu, prosečno, manje generacija da bi našao rešenje. Takođe, veći je i procenat uspešnosti kod GA iz rada.

Uporedna analiza genetskog algoritma(GA) i algoritma Brute-force koji smo takođe implementirali, ukazuje da se Brute-force izvršava znatno brže, što se može videti u tabeli koja je u nastavku. Iako je genetski algoritam možda našao široku primenu u rešavanje drugih problema optimizacije i NP kompletnih, ne daje zadovoljavajuće rezultate u rešavanju Sudoku-a. Brute-force nam garantuje rešenje, dok to ne važi za genetski

Table 3. Performance of multistage GA for different clue sizes of Sudoku (Out of total 15 runs)											
Clue Size	15	17	19	21	23	25	27	29	31	33	35
Success Rate	15	15	14	14	12	14	14	15	15	15	15
Success percentage	100	100	93	93	80	93	93	100	100	100	100
Average cycle	5.2	5.3	10.4	12.6	18.5	10.4	10.8	8.2	5.6	5.6	3.0
Average generation (rounded)	55	55	142	128	188	108	102	85	52	55	32

Slika 6: Rezultati iz literature

algoritam koji nas je ostavljao bez rešenja kod puzli težine Hard.

<i>Broj početnih polja</i>	25	26	29	31
<i>Prosečno vreme izvršavanja GA</i>	93.71	71.42	7.41	24.19
<i>Prosečno vreme izvršavanja Brute-force</i>	0.005	0.003	0.0009	0.001

Slika 7: GA vs Brute-force

3.1 Eksperimentalno okruženje

Rezultati dobijeni korišćenjem gore opisanim algoritmima testirani su u sledećem okruženju: Računar: Intel® Core™ i5 Dual Core Processor 4300M, Ram: 8 GB, Operativni sistem: Windows 10

4 Zaključak

Ovaj rad završili bismo čuvenom rečenicom Čarlsa Darvina,

“Ne opstaju najjači pripadnici jedne vrste, niti najinteligentniji, već oni koji se najbolje prilagođavaju promenama.”

Upravo njom, možemo opisati genetski algoritam, koji smo u ovom radu koristili za rešavanje popularne slagalice Sudoku, veličine 9×9 , različitih težina. Utvrdili smo da je od izuzetne važnosti što bolji odabir funkcije prilagođenosti kao i odabir parametra za ukrštanje, mutaciju, potom veličina populacije, elitizma kako bi on bio što optimalniji. Međutim, kroz više primera u radu, pokazali smo, da iako je primena genetskog algoritma vrlo raznovrsna i značajna za rešavanje mnogobrojnih optimizacionih problema, u slučaju Sudoku to nije baš najbolje moguće rešenje. Kako je gotovo nemoguće bilo naći rešenja za slagalice teškog nivoa ali i pojedinih primera srednjeg nivoa, želeli smo da primenimo još neku tehniku za rešavanje ovog problema kako bismo videli i druge rezultate. Opredelili smo se za brute-force algoritam, poznatu strategiju koja se pokazala kao vrlo jednostavna za primenu ali i kao algoritam pomoću kojeg uvek dolazimo do rešenja. U našem slučaju, rešavanju Sudoku puzzle, brute-force je izuzetno brzo dolazio do tačnih rešenja.

Literatura

- [1] Kartelj Aleksandar. Rešavanje problema minimalne energetske povezanosti u težinskom grafu primenom genetskog algoritma. 2010.
- [2] Andries P. Engelbrecht. *Computational Intelligence : An Introduction*. John Wiley & Sons, Ltd, Chichester, England, 1992.
- [3] Deepak Mylavarapu Haradhan Chel and Deepak Sharma. A Novel Multistage Genetic Algorithm Approach for Solving Sudoku Puzzle. *International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, 2006. on-line at: https://www.researchgate.net/publication/311250094_A_novel_multistage_genetic_algorithm_approach_for_solving_Sudoku_puzzle.
- [4] Noraini Mohd Razali John Geraghty. *Genetic algorithm performance with different selection strategies in solving tsp*. Matematički fakultet, Univerzitet u Beogradu, 2020.
- [5] Timo Mantere and Janne Koljonen. Solving and Rating Sudoku Puzzles with Genetic Algorithms. *Department of Electrical Engineering and Automation University of Vaasa, Vaasa*, 2007. on-line at: https://www.researchgate.net/publication/224301656_Solving_rating_and_generating_Sudoku_puzzles_with_GA.
- [6] Predrag Janičić Mladen Nikolić. *Veštačka inteligencija*. Matematički fakultet, Univerzitet u Beogradu, Srbija, 2020.
- [7] Raktim Chakraborty Sankhadeep Chatterjee, Saubhik Paladhi. Comparative Study On The Performance Characteristics Of Sudoku Solving Algorithms. 2007. on-line at: https://www.researchgate.net/publication/224301656_Solving_rating_and_generating_Sudoku_puzzles_with_GA.
- [8] Miodrag Živković. *Algoritmi*. Matematički fakultet, Univerzitet u Beogradu, Srbija.