Факултет инжењерских наука Универзитета у Крагујевцу

Александар С. Вукомановић

Утицај избора параметара на перформансе генетског алгоритма

ДИПЛОМСКИ РАД

Комисија за преглед и одбрану:

1. **Проф. Др Весна Ранковић – ментор**
2. **_____**
3. **_____**

Датум одбране: _____

Оцена: _____

У оквиру овог дипломског рада кандидат треба да покаже како избор параметара утиче на понашање и рад генетског алгоритма. Потребно је да опише функционисање и дефинише све делове генетског алгоритма. Кроз практичан пример је потребно да покаже значај одабира и подешавања параметара и њихово дејство на перформансе генетског алгоритма.

Препоручена литература:

[1] Bäck T., Fogel D. B., Michalewicz Z., "Basic Algorithms and Operators", in: Evolutionary Computation 1, Institute of Physics Publishing, Bristol-Philadelphia, (2000).

[2] Beasley D., Bull D.R., Martin R.R., "An Overview of Genetic Algorithms, Part 1, Fundamentals", University Computing, Vol. 15, No. 2, (1993).

[3] Kratica J., "Paralelizacija genetskih algoritama za rešavanje nekih Np-kompletnih problema", Doktorska disertacija, Matematički fakultet, Beograd, (2000).

[4] Kratica J., Tošić D., "Introduction to Genetic Algorithms and Some Applications", Proceedings of a Workshop on Computational Intelligence - Theory and Applications, Niš, Yugoslavia, February 27, (2001).

Крагујевац, 2015.

Ментор:
Проф. Др Весна Ранковић

САДРЖАЈ:

Утицај избора параметара на перформансе генетског алгоритма

Резиме

У овом раду је описан утицај при избору параметара на перформансе генетског алгоритма (ГА). Како би се дошло до самих параметара и њиховог утицаја, у првом делу рада је изложена презентација генетског алгоритма као модела за решавање низа проблема. Дат је осврт на еволуцијски процес у природи јер је на њему као пресликана метода настао генетски алгоритам. У даљем току рада кроз дефинисање генетског алгоритма, описани су генетски оператори, бинарно приказивање решења са одрађеним примером за једноставније схватање овог појма. Такође као главни аспект алгоритма представљена је функција доброте. Како би на што лакши начин презентован генетски алгоритам, дат је пример и кроз пример је праћен рад алгоритма кроз неколико итерација са најситнијим детаљима његовог функционисања. Пример који је одрађен у радном окружењу MATLAB-а, је коришћен у излагању о утицају параметара на рад генетског алгоритма. Најпре су описане битне поставке у MATLAB-у које су везане за параметре. Сваки параметар је представљен понаособ са датим графицима из одрађеног примера и њихово дејство на генетски алгоритам.

Кључне речи: Генетски алгоритми, Избор параметара, Перформансе ГА

Abstract:

In this paper describes influence in the selection of the parameter on the performance genetic algorithm (GA). To reach the very parameters and their impact in the first part of the paper is exposed to the presentation of the genetic algorithm as a model for solving many problems. It also discusses the evolutionary process in nature because it is on him as mapped method developed genetic algorithm. In the further course work through the definition of a genetic algorithm, written in the genetic operators, binary display solutions with well realized example for easier understanding of this term. Also as a major aspect of the algorithm is presented, the fitness function. In order to easier way presented a genetic algorithm, an example is given and the example was followed by the work of the algorithm through several iterations with the smallest details of its functioning. An example of which is defined in the MATLAB workspace and is used in the presentation on the impact of parameters on the operation of the genetic algorithm. First, they described the essential settings in MATLAB related to the parameters. Each parameter is represented individually with the given limits of the specific examples and their effects on genetic algorithm.

Key words: Genetic algorithms, Selection of parameters, Performance GA

1. УВОД

У последњих неколико година у инжењерској пракси, генетски алгоритми као метод решавања низа проблема бележе изузетан напредак и развој. Генетски алгоритам је пресликан природни генетски процес приказан у информатичкој форми.

Дакле генетски алгоритми су хеуристичка метода оптимизације која је заснована на принципу природног еволуцијског процеса. Идеја о генетском алгоритму се јавила 60-их и раних 70-их година и то на почетку као еволуција рачунарства од стране Инга Риченберга (енг. *Ingo Rechenberg*) који је измислио веома утицајан скуп метода оптимизације познатих као "Еволуцијске стратегије"[1]. Ипак главни темељ самој теми генетских алгоритама је поставио Џон Холанд (енг. *John Holland*)[2].

Када се говори о еволуцији треба рећи да је то веома снажан процес претраживања великог простора могућности како би се дошло до одговарајућег решења. Сам пример овога су жива бића која се током еволуције прилагођавају различитим условима у природи. Сличност у особинама еволуције и генетског алгоритма огледају се у процесу селекције и генетских оператора. Како би се упоредиле сличности може се констатовати да одабир над неком врстом организма у еволуцијском процесу чине околина и услови у природи а у генетским алгоритмима кључ селекције је функција циља или позната и као функција доброте, она је како и име каже циљано решење проблема. Како услови у природи и околина имају највећи утицај на селекцију неке врсте организма, тако је и функција циља кључ при селекцији одређене популације у генетском алгоритму.

Под утицајем тих услова у природи и околине највећу шансу за преживљавање имају најбоље јединке, односно оне које могу најбоље да се прилагоде условима, самим тим имају највећу шансу да преживе и да даљим размножавањем пренесу своје добре гене на потомке. Готово исти случај је и код генетском алгоритма где се селекцијом одабирају добре јединке које се преносе у следећу популацију и тако из генерације у генерацију уз манипилацију и репродукцију све док се не дође до најбоље јединке односно до задовољавајућег решења.

Генетски алгоритми нарочито у последњих неколико година се показују као моћна метода која има своје јако добре стране приликом решавања великог броја проблема у инжењерству. То је подстакнуто њиховом листом предности:

- једноставност идеје на којој су засновани као и једноставност њихове примене
- могућа је оптимизација функције са великим бројем променљивих
- могућа је опитимизација сложених функција

- као резултат оптимизације имамо већи број решења

Генетски алгоритми спадају у групу методе усмереног случајног претраживања простора решења. Овој групи припадају још неке методе које су сличне генетским алгоритмима а то су:

- генетско програмирање (енг. *genetic programming*)
- еволуцијске стратегије (енг. *evolutionary strategy*)
- симулирано каљење (енг. *simulated annealing*)

Како је повећана њихова примена, тако се повећао опсег у истраживању рада генетских алгоритама и они су и данас основна хеуристичка метода. Примену су пронашли у различитим областима и описан је поприличан број решених проблема оптимизације. Неки од њих су:

- Локацијски проблем снабдевача неограниченог капацитета (енг. *The uncapacitated facility location problem – UFLP*) . Овај проблем се у литератури јавља и под називом *Uncapacitated warehouse location problem* или *The simple plant location problem* и он је један од основних и највише решаваних проблема у теорији локацијских проблема. Проблем ове природе представља посебну класу проблема оптимизације. Као што је и у већини случајева код многих проблема и овде је циљ пронаћи што краће растојање, најмањи могући утрошак времена укупног путовања и осталих параметара. Поред ове минимизације трошкова приликом транспорта, мора се водити рачуна да се испоштују сви захтеви купаца. Све је ово могуће објединити и решити правилним одабиром потенцијалних локација из скупа могућих локација. Објекти за које се тражи лоцирање су обично центри који пружају услуге, па је њихов назив снабдевачи, корисници услуга су клијенти. Овај проблем се примењује у многим областима и има широку распрострањеност због чега је популаран и као такав предмет великог броја истраживања. Проблем овакве врсте се може пронаћи у радовима [3-5].

- Хијерархијско распоређивање радника (енг. *Hierarhical workforce sheduling problem -HCLP*). У данашњем времену када се фирме јако брзо развијају (број послова и број радника који те послове треба да обаве расте) распоређивање радника у фирмама представља озбиљан проблем. Ефикасна расподела радне снаге је веома важан део слагалице у успешном функционосању предузећа. Што је мањи број радника смањују се и трошкови фирме али се уједно погоршава и ниво услуге. Обрнут случај ако је већи број запослених, доноси веће трошкове и проблем радног простора али побољшава се ниво услуге. Овај проблем покушава да направи идеалан баланс између свих прохтева и да се максимално задовоље постављени

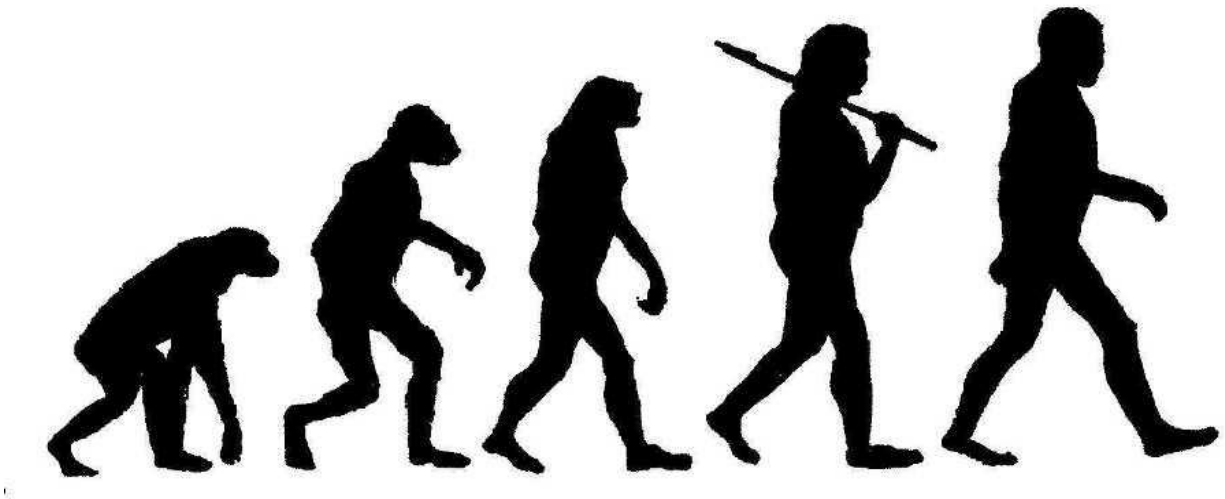
задаци. У HCLP полазне параметре проблема представљају хијерархија стручне квалификације радника и различите потребе послодавца. Овај проблем је презентован у многим радовима и презентован је у литератури[6-10].

Циљ овог рада је да на што једноставнији начин приближи функционисање процеса генетског алгоритма и покаже колики утицај имају параметри на његов рад. Описани су сви делови генетског алгоритма и приказана је симулација рада алгоритма у неколико итерација како би на што вернији начин свако могао да види његово функционисање. Понашање генетског алгоритма највише зависи од дефинисања и подешавања параметара, па је од настанка ове методе ово предмет многих радова. Многи су покушали да пронађу најбоље поставке параметара алгоритма за велики низ проблема али се показало да је то веома комплексан задатак и тешко изводљив. И данас се још увек ради на дефинисању скупа параметара који би били оптимални за што већи број проблема. Овим радом је показана колика је међусобна зависност параметара, да су они веома зависни једни од других и колико промена једног параметра може да утиче на понашање генетског алгоритма. Кроз практичан пример са функцијама за оптимизацију су дате препоруке при подешавању параметара и како се то одражава на добијање решења функција.

2. ЕВОЛУЦИЈСКИ ПРОЦЕС У ПРИРОДИ

Еволуција представља промену наследних особина популације кроз генерације и подразумева непрекидан процес прилагођавања живих бића на околину и услове у којима живе. Ове особине су експресија гена који се приликом размножавања преносе на потомство.

Чарлс Дарвин (енг. *Charles Darwin*) се у својој теорији еволуције позива најпре на природан одабир и селекцију која уништава јединке слабије прилагођене условима живота, а поспешује оне које се боље прилагођавају[11]. Он је дакле приметио различитост међу јединкама исте врсте и да природа селекцијом јединки регулише величину популације. Свака нова генерација мора да наслеђује добре особине претходне генерације како би опстала.



Слика 1. Еволуција човека

У природи траје константна борба за преживљавањем у којој побеђују најбољи а лоши изумиру, јер се услови и околина мењају током времена и врсте морају да се прилагођавају тим новим условима како би опстале током еволуције. Свака следећа генерација мора поприматидобра својства, па самим тим лоша изумиру. Јединке су окарактерисане тим својствима а као примери се могу узети боја очију, способност прилагођавања на високе и ниске температуре, боја коже, оштрина слуха и вида итд. Дакле добра својства имају већу вероватноћу преживљавања, односно преношења на следећу генерацију.

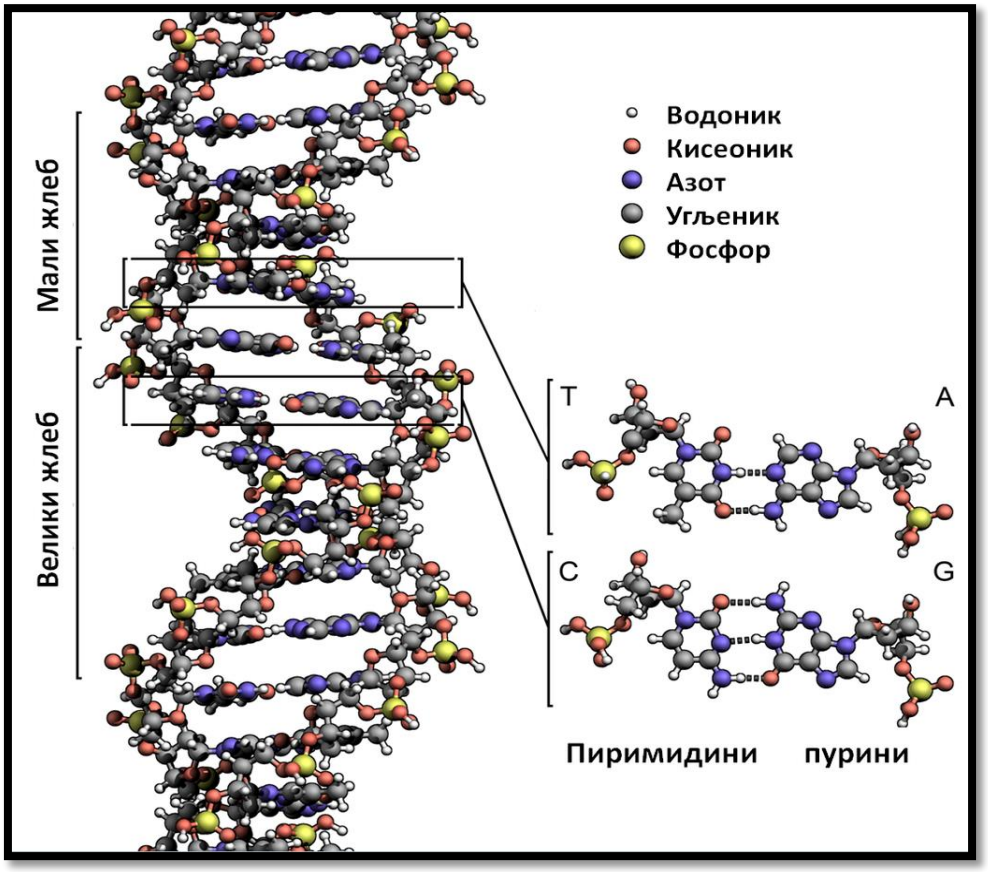
Сва својства су записана у хромозому. Хромозоми су дуге и танке завојнице у којима су груписани гени. Они су ланчастог облика и носе нашу генетичку информацију. Хромозоми увек долазе у паровима један од мајке, други од оца, што значи да за свако својство постоје два гена. У генетском пару гени могу бити равноправни и неравноправни, или другим називом код неравноправних гена један ген може бити доминантан а други рецесиван. Новонастала јединка која поседује равноправан пар гена, у њој преовлађују својства и оца и мајке, док код неравноправног пара својство представља резултат доминантног гена. У једном пару хромозома нису записана сва својства јединке, већ у више парова хромозома, као на пример човек поседује 46 хромозома или 23 пара хромозома.

Сва жива бића свој генетички материјал носе у облику ДНК, са изузетком неких вируса. Дезоксирибонуклеинска киселина (ДНК) је нуклеинска киселина која садржи упутства за развој и правилно функционисање свих живих организама. ДНК има веома важну улогу не само у преносу генетичких информација са једне на другу генерацију, већ садржи и упутства за грађење неопходних ћелијских органела, протеина.

Вотсон (енг. *James Watson*) и Кирк (енг. *Francis Crick*) су 1953. године[12] показали да је у живим организмима ДНК молекул састављен од два полинуклеотидна ланца који су спирално увијени један око другог. Спирални ланац који чини ДНК се одржава у том облику помоћу водоничних веза између парова база. У ДНК молекулу постоје четири базе, структура ДНК молекула је приказана на слици 2 :

- Аденин (А)
- Цитозин (Ц)
- Гуанин (Г)
- Тимин (Т)

Четири базе су међусобно коплементарне као што се види са слике 2: аденин (А) једног ланца је увек у пару са тимином (Т) наспрамног ланца и повезани су двема водоничним везама. Гуанин (Г) једног ланца је увек у пару са цитозином (Ц) наспрамног ланца и повезани су трима водоничним везама. Управо је у овим базама записана информација, па ако се направи поређење са рачунаром код кога је најмања јединица информације један бит, тако је у природи најмања јединица информације једна база.



Слика 2. Структура ДНК молекула

Генетски материјал у природи се са генерације на генерацију преноси укрштањем и то особине са родитеља на децу. Свака јединка поседује генетски материјал оба родитеља. Приликом укрштања неки гени мутирају односно настаје неки други ген. Такве особине које стекне јединка у животу (на пример ожиљак) не преносе се на потомке. Дакле мутација је квалитативна и/или квантитативна промена у генетичком материјалу која није наслеђена од родитеља. Мутацијама настају нове генетичке варијанте чиме се остварује разноврсност и основа за деловање природне селекције. Сада се може прећи на генетске алгоритме јер је ово био добар увод за то.

3. ГЕНЕТСКИ АЛГОРИТМИ

Генетски алгоритам је оптимизациони алгоритам који је настао опонашањем еволуције у природи. Како је описано у претходном поглављу у процесу еволуције преживљавају најбоље јединке, тако и код генетских алгоритама ће опстати скуп најбољих могућих решења. Дакле природни еволуцијски процес је метод оптимизације. Сам генетски алгоритам се може наћи у разним механизмима природног одабира: у природи, преживљавање у околини која је сиромашна средствима за живот (дешавања у данашњој економији). Ако се изједначи способност појединца да опстане са наследним особинама које поседује, генима, онда се може рећи да гени доминантних појединаца преживљавају а гени слабијих изумиру јер немају потомство.

Речено је у уводном делу да је појам генетског алгоритма као модела постављен од стране Џона Холанда са циљем да се проучи адаптивно понашање. У свом раду је предочио генетски алгоритам као рачунарски процес који имитира еволуитивни природни процес. Ова врста алгоритма се у доста случајева може пронаћи и под називом канонски генетски алгоритам или прост генетски алгоритам. Концепт генетског алгоритма је описан у многим радовима, и може се пронаћи мноштво информација о њему у литератури[13-18].

Како је већ споменуто, генетски алгоритми симулирају еволуитивни процес у природи, па се као њихове заједничке особине могу навести:

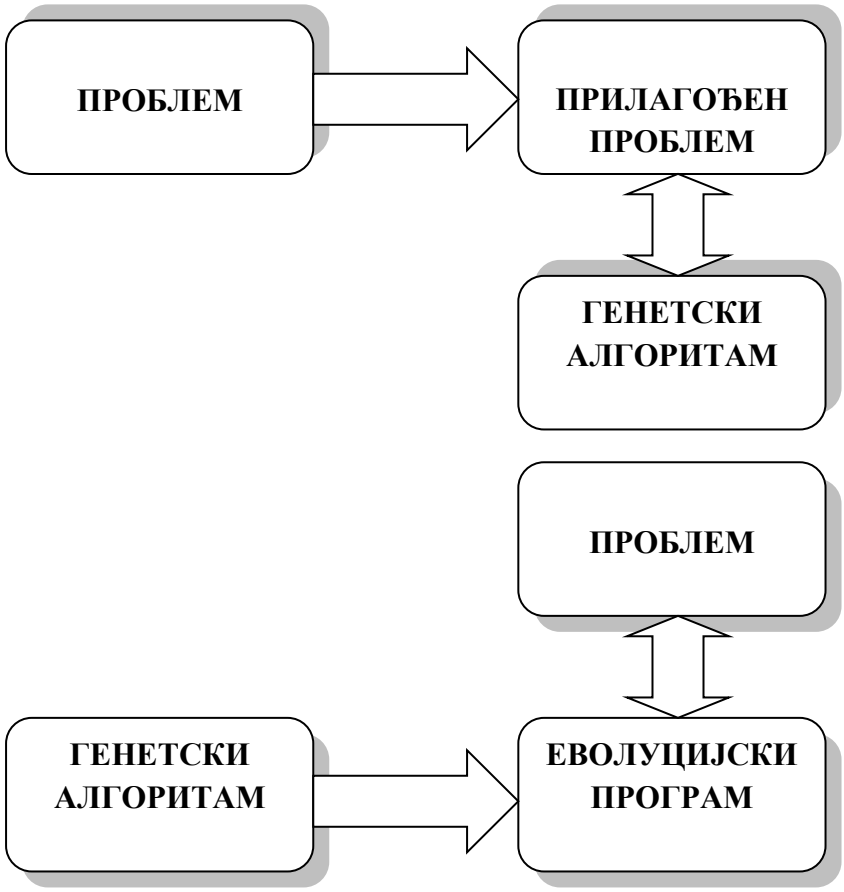
- постоји популација јединки
- особине јединки записане су помоћу генетског кода
- неке јединке су боље прилагођене околини
- боље јединке имају већу шансу преживљавања и репродукције
- јединке могу да мутирају
- деца наслеђују особине од родитеља

Као што и у еволуцији долази до промене генетског материјала тако у процесу генетског алгоритма током репродукције долази до промене гена, и ова појава се назива укрштање. Такође може доћи до случајне промене генетског материјала под спољашњим утицајима, треба нагласити да се ове промене јављају у мањим процентима, и оне се називају као и у еволуцији мутације. Ови процеси се у генетским алгоритмима називају генетским операцијама, а одабир најбољих јединки се назива селекцијом.

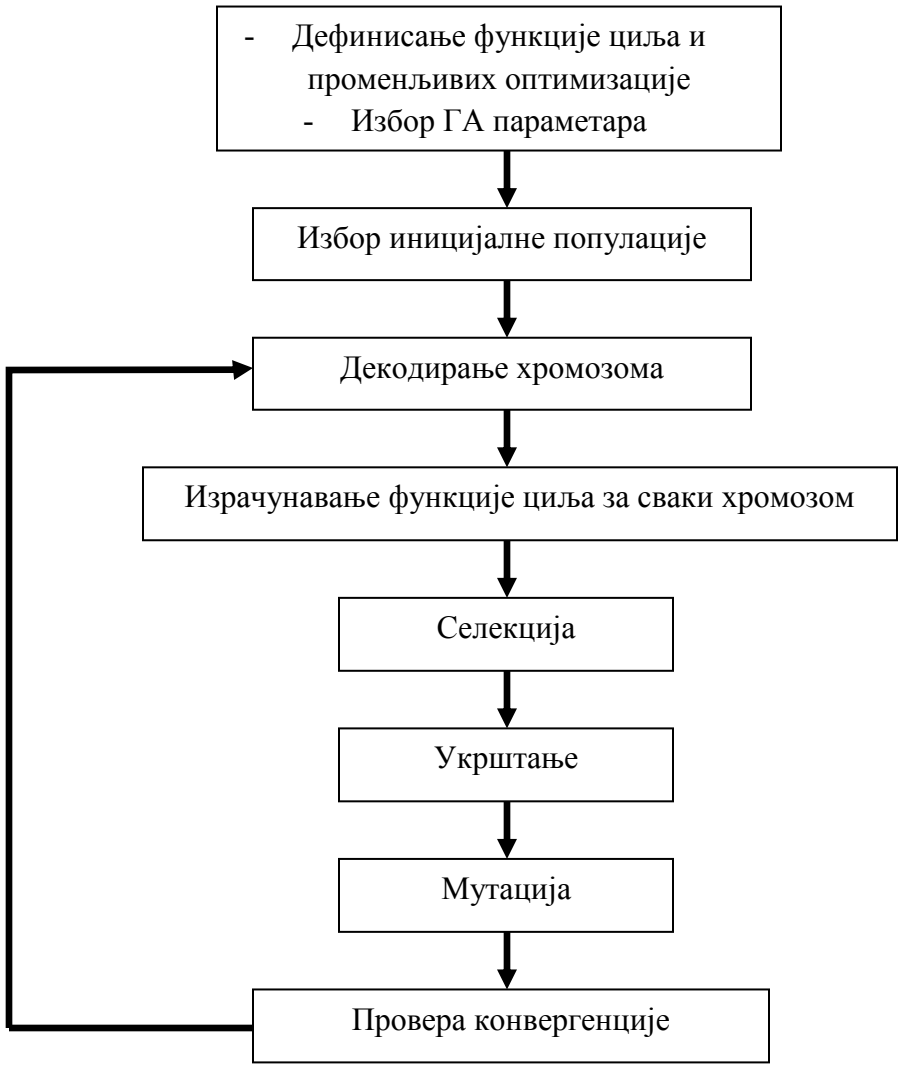
На почетку генетског алгоритма прво се обавља случајан одабир јединки из домена тражења, чиме се врши иницијализација популације. Ове јединке се називају хромозоми и представњају потенцијална решења проблема. Хромозоми чине скуп који се назива популација. Свако потенцијално решење поседује одређени квалитет који је познат у генетским алгоритмима као доброта. Функција која одређује тај квалитет се назива функција доброте. Из почетне популације се одабиром бољих јединки формира нова популација уз утицај генетских операција код неких јединки, а овим је добијена нова генерација. Дакле генерација је скуп свих хромозома у одређеној итерацији. А читав цилкус добијања нових генерација се завршава када се задовољи услов и постигне критеријум за заустављање, а најбољи члан из завршне популације се узима као решење проблема. О раду генетског алгоритма се може пронаћи у домаћој лиетратури [19-23].

Решењу проблема можемо приступити на два начина:

- проблем прилагодити генетском алгоритму
- генетски алгоритам прилагодити специфичности проблема



Слика 3. Приступи решавању проблема



Слика 4. Структура генетског алгоритма

3.1 Популација

Популација је група јединки исте врсте која насељава одређени простор и међусобно се размножавају и дају потомство. Као што је у природи ретко да постоји изолована јединка, тако и у генетском алгоритму популацију чине више јединки. Како део популације одумире тако се стварају нови потомци и величина популације у свакој генерацији се одржава на константном нивоу. Као што је већ речено сваку популацију чини скуп потенцијалних решења.

3.2 Бинарни приказ решења

Теоријом из генетских алгоритама је превасходно описан приказ решења уз помоћ природног бинарног кода. И у пракси као и у примерима је коришћен најчешће бинарни приказ, а свакако даје и најбоље резултате. Због своје једноставности бинарни приказ је погодан за имплементацију, ипак у одређеним проблемима он није погодан за коришћење при чему је неопходно извршити извесне корекције (након укрштања и/или мутације).

Како су у бинарном коду промењљиве представљене бинарно постоји метод за конвертовање реалне вредности у бинарне бројеве. Хромозом је приказан као бинарни вектор и има кодирану вредност $x=[dg,\ \ gg\ ]$. Прецизност је условљена дужином n бинарног броја (број битова), односно то је број јединица или нула у хромозому. У вектор може се записати 2^n различитих комбинација јединица и нула, односно могуће је записати сваки број у интервалу $[0,\ 2^n-1]$. Вектор $v(0)=[0\ 0\ 0\ldots]$ представља вредност $x=dg$ а вектор $v(2^n-1)=[1\ 1\ 1\ldots1]$ представља вредност $x=gg$. Ако је неки бинарни број b у интервалу $[0,\ 2^n-1]$ записан у облику вектора $v(b)=[B_{n-1}\ B_{n-1}\ B_{n-3}\ \ldots\ B_0]$ где је $B_i=0$ или $B_i=1$ тада се може записати[24]:

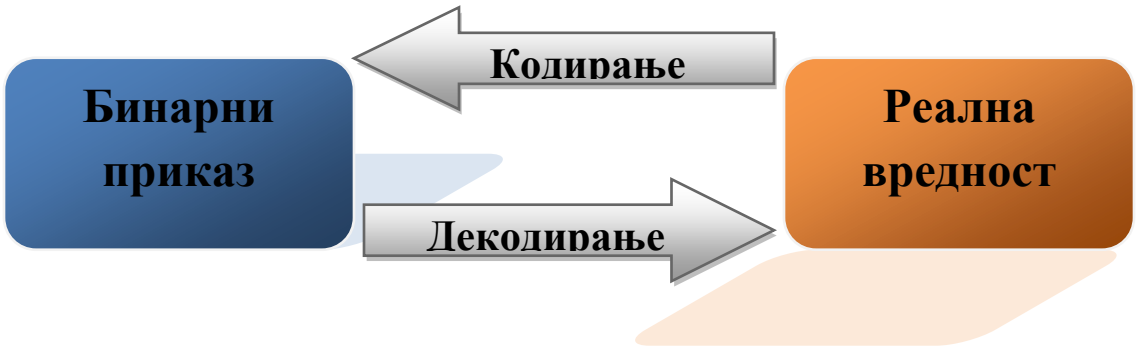
бинарни код:

$$b=\sum_{i=0}^{n-1} B_i * 2^i$$

а еквивалентан реалан број:

$$x=dg+\frac{b}{2^n-1}(gg-dg)$$

Кодирање је процес одређивања бинарног броја b за задати реалан број x а декодирање је обрнут процес где се претвара бинарни код у потенцијално решење (реалну вредност).



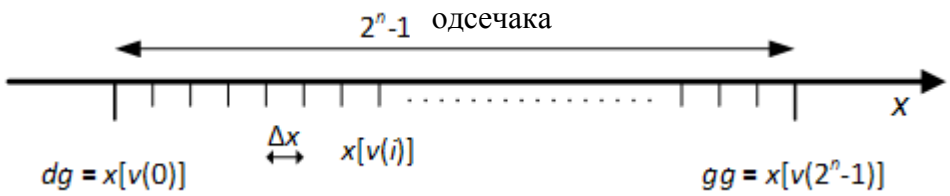
Слика 5. Кодирање и декодирање у генетском алгоритму

Кодирање бинарног броја *b* одређује се према обрасцу:

$$b = \frac{x - dg}{gg - dg}(2^n - 1)$$

За решење *x* прецизности *p* и ако је *pєN* хромозом мора бити дужине:

$$n \geq \frac{\log[(gg - dg) * 10^p + 1]}{log2}$$



Слика 6. Хромозом *v(i)* као једна од 2^n тачака распоређених у интервалу *[dg, gg]*

где су:

dg - доња граница (најмања) вредности *i*–те променљиве а
gg - горња граница (највећа) вредности *i* –те променљиве

Пример:

Ако је *dg=-50* и *gg=50* и ако је прецизност у датом примеру *p=6* онда узимајући из претходно дефинисаног обрасца, дужина хромозома треба да буде:

$$n \geq \frac{\log[(50 + 50) * 10^6 + 1]}{log2}$$

$$n \geq 26.575$$

Усваја се да је *n=27*.
Бинарни вектор 000000000000000000000000000 одговара броју -50 а бинарни вектор 111111111111111111111111111 одговара броју 50. Па тако се прорачунава даље за неки бинарни вектор 110000101100001101111001100 да одговара реалном броју 26.0795 што се да видети из прорачуна:

$$b = \sum_{i=0}^{26} B_i 2^i = 2^2 + 2^3 + 2^6 + 2^7 + 2^8 + 2^9 + 2^{11} + 2^{12} + 2^{17} + 2^{18} + 2^{20} + 2^{25} + 2^{26}$$
$$= 102112204$$

$$x = dg + \frac{b}{2^n - 1}(gg - dg) = -50 + \frac{102112204}{134217727} * 100 = 26.0795$$

3.3 Функција доброте

После процеса иницијализације популације односно генерисањем величине популације случајним одабиром бројева у интервалу *[0, 2^n-1]*, врши се процена функције циља.

Функција циља или функција доброте ако се дефинише на еволуцијском нивоу она представља природну околину која врши селекцију над јединкама. Дакле помоћу ње се оцењује квалитет јединке. Ако се јединка боље прилагођава околини и условима има већу шансу да преживи, у преводу што је решење боље, боље испуњава функцију циља (поседује већу доброту) има већу вероватноћу да пређе у наредну генерацију. У литератури је позната и под другим називима, *fitness* функција, функција способности. Она је једнака функцији *f* коју треба оптимизовати:

$$dobrota(v_i) = f(x_i)$$

Па је укупна доброта популације је дефинисана формулом:

$$D = \sum_{i=1}^{vel\_pop} dobrota(v_i)$$

А просечна доброта популације је:

$$\overline{D} = \frac{D}{vel\_pop}$$

3.4 Селекција

Циљ селекције је да се сачувају квалитетна својства из једне генерације и да се пренесу у следећу генерацију а да се отклоне лоша својства. Њом се врши селектирање или одабир добрих јединки које ће се пренети у следећу популацију, и које ће даље учествовати у репродукцији. Дакле добри гени опстају а лоши изумиру.

Селекција би се могла свести на веома једноставан начин примене ако би се јединке одвајале чисто по доброти, а већ одређеним бројем уклониле најлошије. Оваквим поступком који би се релативно брзо завршио, одбацили би такође битну ставку да и најлошије јединке имају нека добра својства која треба искористити. Ако би се одбациле све најлошије јединке, остале би само оне најбоље али то не значи да оне поседују и најбоља својства. Како би дали могућност да лоше јединке преживе али да постоји нека равнотежа додељује се вероватноћа преживљавања која је код лоших јединки ниска али већа од 0 а код добрих јединки висока али мања од 1. Постоје два типа поступака селекције:

- генерацијске селекције
- елиминацијске селекције

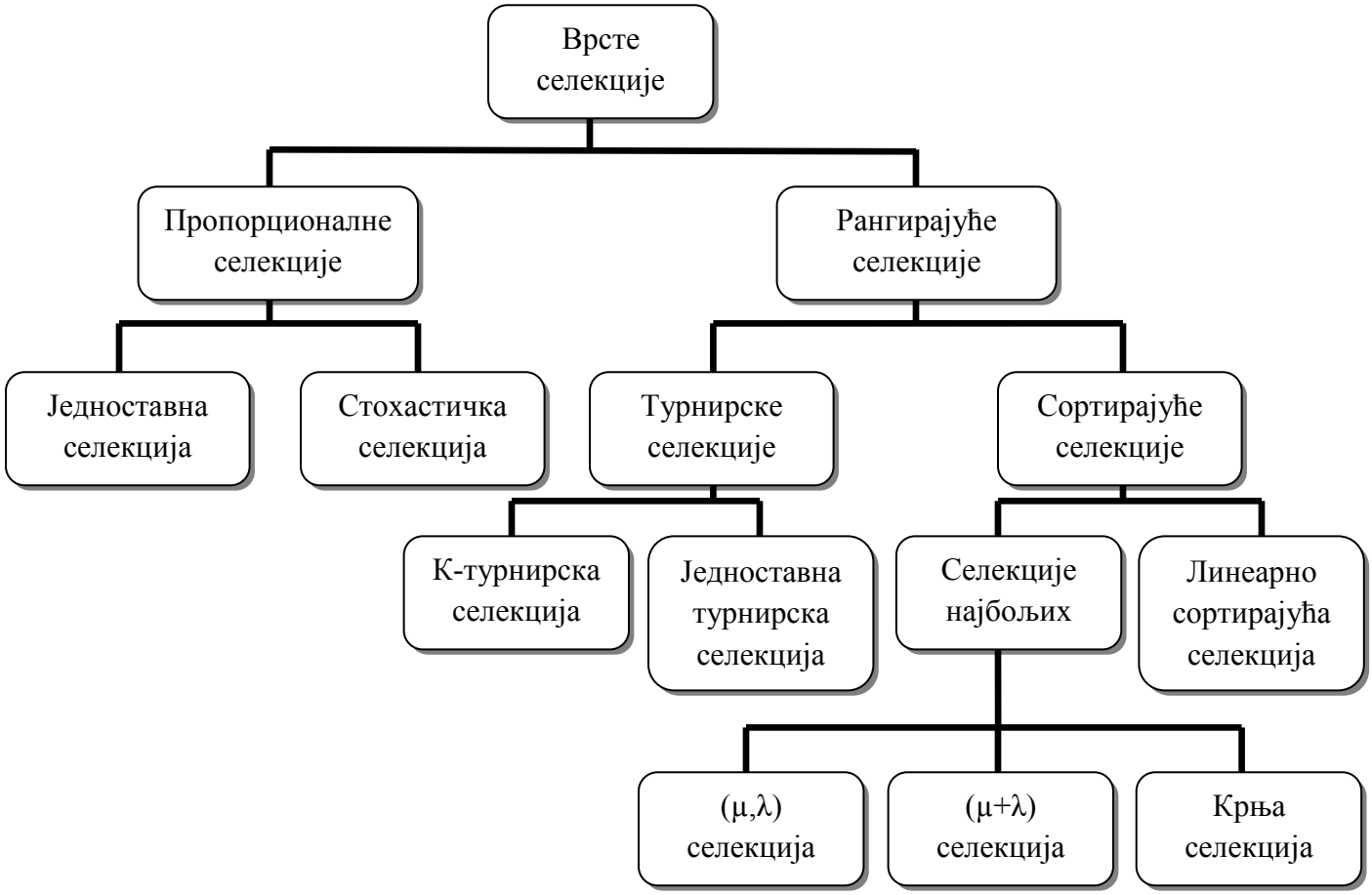
Генерацијске селекције врше селекцију добрих јединки из старе популације које касније чине нову популацију и учествују у даљој репродукцији, док елиминацијске обављају супротан процес тј. одабирају лоше јединке за елиминацију.

Такође се селекције могу груписати и према начину одабира јединки и то у (слика 7):

Пропорционалне селекције:
- једноставна (рулет) селкција
- стохастичка селекција

Рангирајуће селекције:
Турнирске селекције:
- к-турнирска селекција
- једноставна турнирска селекција

Сортирајуће селекције:
- (μλ) селекција
- (μ+λ) селекција
- крња селекција
- линеарно сортирајућа селекција



Слика 7. Врсте селекције

3.4.1 Једноставна (рулет) селекција

Једноставна селекција је позната и под називом рулет или пропорционална селекција. Вероватноћа избора јединке при рулет селекцији је пропорционална њеној доброти. Дакле вредност функције циља јединке директно одређује вероватноћу њеног одабира. Процес селекције је сачињен од неколико фаза:

- За сваку јединку се рачуна доброта а за оне код којих је функција $f(x_i)$ поприма негативне вредности, у том случају се додаје позитивна константа C тј. да све јединке имају доброту већу од нуле или једнаке нули:

$$dobrota(v_i) = f(x_i) + C, \qquad dobrota(x_i) \geq 0, \forall x \in [dg, gg]$$

- Рачуна се доброта комплетне популације уз помоћ обрасца који смо већ користили код функције доброте

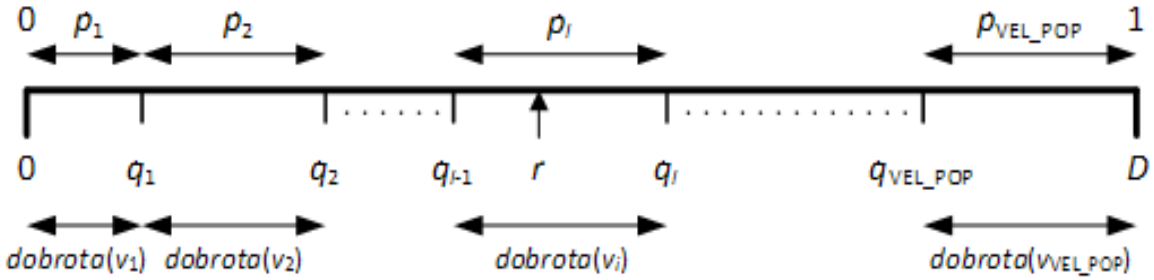
$$D = \sum_{i=1}^{vel\_pop} dobrota(v_i)$$

- Рачуна се кумулативна доброта за сваки хромозом и то према обрасцу:

$$q_k = \sum_{i=1}^{k} dobrota(v_i), \quad k = 1,2,\dots,vel\_pop$$

- И на крају се рачуна вероватноћа селекције за сваку јединку (хромозом):

$$p_k = \frac{dobrota(v_k)}{D}$$



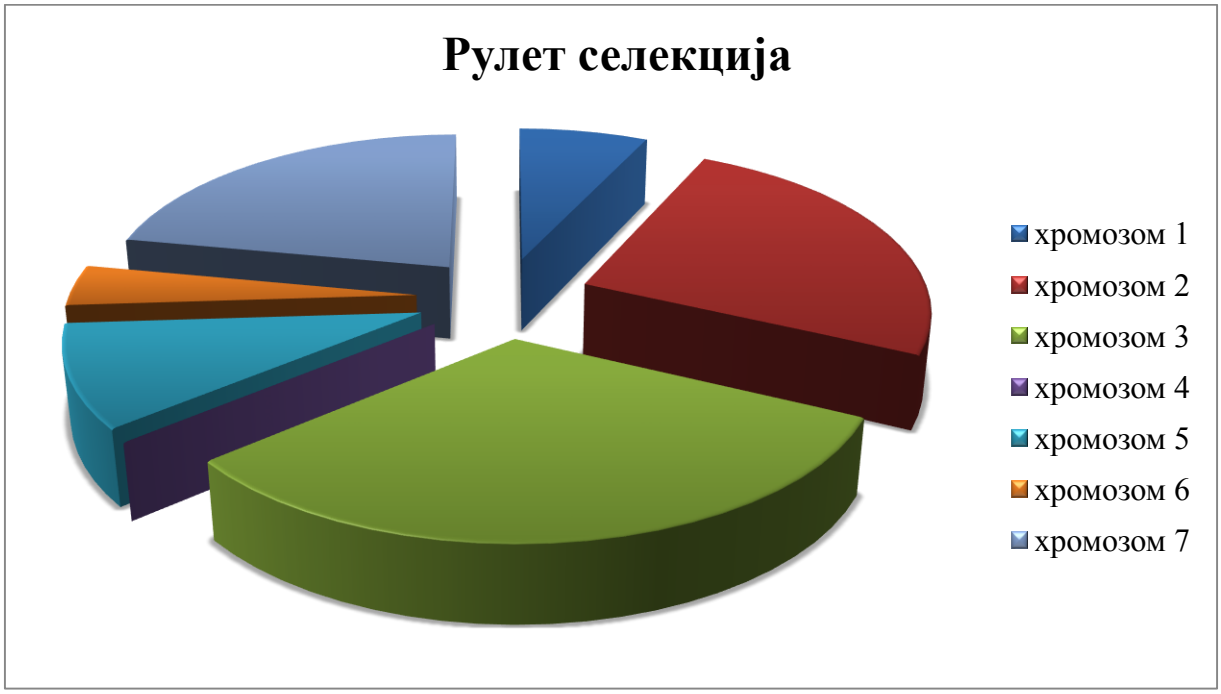
Слика 8. Вероватноћа селекције *(p_i)* и кумулативна доброта хромозома *(q_i)*

Пример:

У датом примеру испод (табела 1), величина популације је 7, дакле њу чини седам јединки или хромозома. За сваки хромозом је у табели дата његова доброта (друга колона) и вероватноћа селекције (четврта колона). Може се приметити да трећи хромозом има највећу доброту, па самим тим и његова вероватноћа селекције је највећа и може бити селектован за даљи ток репродукције у генетском алгоритму. Док је доброта четвртог хромозома једнака 0 и он нема никакву шансу у процесу селекције.

k	$dobrota(v_k)$	q_k	p_k
1	0.5	0.5	0.07
2	1.8	2.3	0.25
3	2.3	4.6	0.32
4	0	4.6	0
5	0.7	5.3	0.10
6	0.3	5.6	0.04
7	1.6	7.2	0.22

Табела 1. Пример рулет селекције са добротама и вероватноћама селекције



Слика 9. Пример једноставне (рулет) селекције

Овакав тип селекције има недостатак у виду фаворизовања добрих јединки и више него што је пожељно, па због тога долази до врло брзог губитка генетског материјала. Такође као велика мана ове селекције је појава дупликата у следећој генерацији, што доводи до успоравања алгоритма.

3.4.2 Линеарно сортирајућа селекција

Код линеарно сортирајуће селекције вероватноћа одабира јединке је пропорционална њеној позицији у сортираном поретку свих јединки популације по доброти. Када је у питању генерацијска линеарно сортирајућа селекција једнике се

сортирају тако да најбоља јединка тј. са највећом добротом има индекс величине популације а најгора се обележаба индексом 1. Или записано формулом:

$$dobrota(v_1) \leq dobrota(v_2) \leq \cdots \leq dobrota(v_{vel\_pop})$$

Вероватноћа селекције јединке се рачуна изразом:

$$p_k = \frac{k}{\sum_{k=1}^{vel\_pop} k}$$

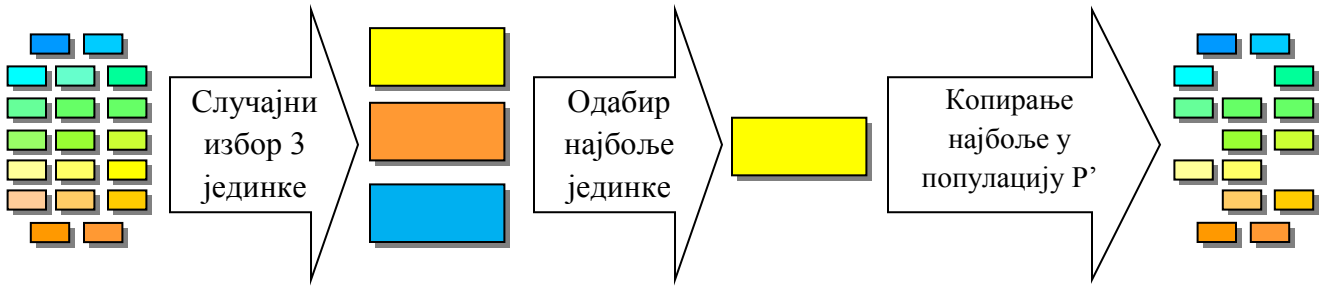
За елиминацијску линеарно сортирајућу селекцију користи се такође исти израз за одређивање вероватноће само овде најбоља јединка имам индекс 1 а најгорој се додељује индекс величине популације тј:

$$dobrota(v_1) \geq dobrota(v_2) \geq \cdots \geq dobrota(v_{vel\_pop})$$

Код овог типа селекције се не јавља проблем фаворизовања неких јединки.

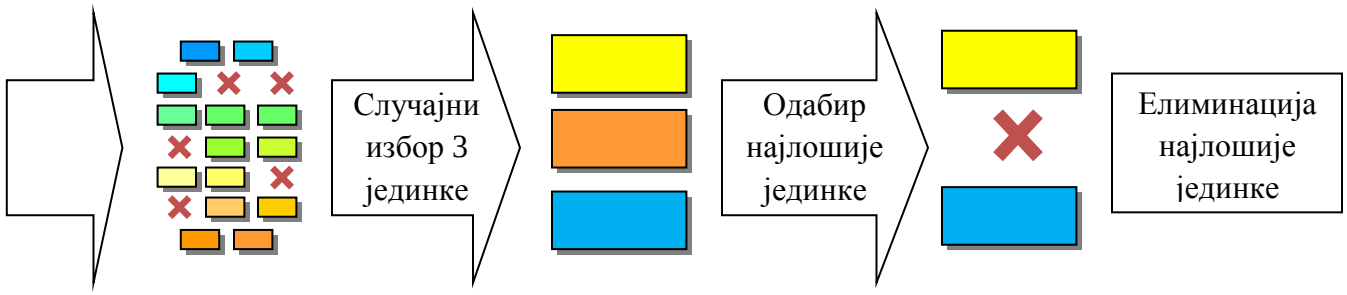
3.4.3 К-турнирска селекција

Генерацијска к-турнирска селекција је врста рангирајуће селекције код које се у сваком кораку за генерисање нове популације случајним одабиром селектује k јединки (три у примеру испод, слика 10) из старе популације, затим их рангира по доброти и најбољу међу њима пребацује у даљи ток репродукције.



Слика 10. Пример једног корака генерацијске 3-турнирске селекције

Код елиминацијске к-турнирске селекције се такође врши случајан одабир k јединки, и најлошија међу њима се брише. Избрисана јединка се може надокнадити потомком преосталих јединки (у примеру су две, слика 11).



Слика 11. Пример једног корака елиминацијске 3-турнирске селекције

3.4.4 Елиминацијска селекција

Док се код једноставне селекције врши одабир добрих хромозома за следећу популацију, код елиминацијске селекције се бирају лоши хромозоми и елиминишу се како би на њихово место дошли нови. Пошто лоши хромозоми изумиру, на њихово место укрштањем долазе деца преживелих хромозома, тј. родитеља. Могуће је у том процесу репродукције да дође до стварања дупликата, а већ је напоменуто да се у том случају успорава рад генетског алгоритма. Како би се избегло успоравање направљен је генетски алгоритам без дупликата. А то значи да се новонастала јединка проверава да ли постоји, а процес се понавља све док се не развије потребан број јединки.

Као што је случај да се код једноставне селекције бирају хромозоми са што већом добротом, код елиминацијске селекције се врши одабир хромозома са што мањом добротом, па је због те чињенице уведена фунција казне:

$$p_k = kazna(v_k), \quad k = 1,2, \dots, vel\_pop$$

$$kazna(v_k) = \max_i[dobrota(v_i)] - dobrota(v_k)$$

Код овог типа селекције постоји кумулативна казна:

$$k_k = \sum_{i=1}^{k} kazna(v_i), \quad k = 1,2, \dots, vel\_pop$$

Неке од предности ове селекције је бржи рад алгоритма, даје врло добре резултате. Алгоритам има једноставану примену, као и наведене предности алгоритма без дупликата.

3.4.5 Елитизам

Током великог броја итерација постоји могућност да се под дејством генетских оператора (укрштања, селекције) најбољи хромозоми измене. Како не би дошло до губитка или измене најбољих јединки током еволуцијског процеса потребно је сачувати и заштити најбоље јединке. Таква заштита одабраних најбољих јединки назива се елитизам. Овим начином се у току генетског алгоритма осигурава да свака генерација тежи глобалном оптимуму тј. траженом решењу проблема. Пошто овај поступак треба извршити у свакој генерацији, овај механизам (елитизам) захтева много више времена па долази до успоравања алгоритма.

3.5 Укрштање

Укрштање је део генетског алгоритма где од две јединке које се називају родитељима, настају једна или две нове јединке које се називају деца. Деца попримају карактеристике својих родитеља, пошто су у добром броју случајева после процеса селекције остале добре јединке (родитељи), деца ће бити са подједнако добрим својствима ако не и бољим.

Поступци укрштања су следећи:

- једнопозиционо укрштање
- двопозиционо укрштање
- униформно укрштање
- мешајуће укрштање

О свим овим начинима укрштања може се доста тога пронаћи у радовима[14][25].

3.5.1 Једнопозиционо укрштање

Код једнопозиционог укрштања (енг. *One-point crossover*) се случајним избором одабере позиција укрштања у бинарном коду хромозома. Затим се размене гени тако што се преносе гени првог родитеља лево од места укрштања и гени десно од места укрштања код другог родитеља, на први добијени хромозом односно првог потомка. На исти начин само супротним редоследом настаје други потомак.

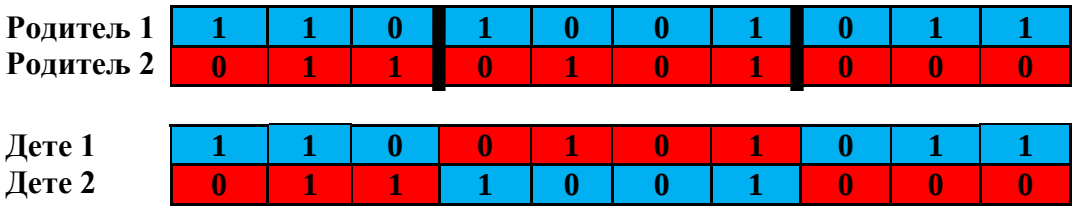
Може се приметити да овај принцип укрштања има недостатке у виду саме размене генетског материјала, јер се гени испред позиције укрштања неће готово никад разменити, а други део гена учествује у свакој размени.

Родитељ 1	1	1	0	1	0	0	1	0	1	1
Родитељ 2	0	1	1	0	1	0	1	0	0	0
Дете 1	1	1	0	1	1	0	1	0	0	0
Дете 2	0	1	1	0	0	0	1	0	1	1

Слика 12. Једнопозиционо укрштање

3.5.2 Двопозиционо укрштање

У поступку двопозиционог укрштања (енг. *Two-point crossover*) се случајним избором изаберу две позиције укрштања, и потом се наизменично размењују гени родитеља при чему настају потомци.

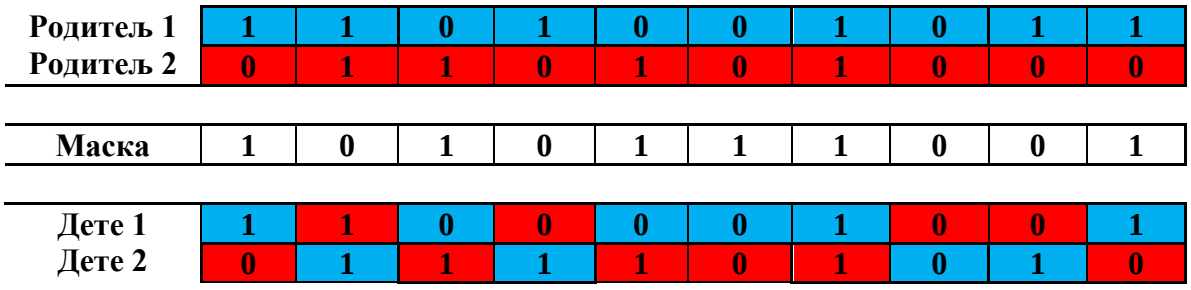


Слика 13. Двопозиционо укрштање

3.5.3 Униформно укрштање

Док се код претходних случајева укрштања прецизно дефинишу позиције укрштања, у поступку униформног укрштања свако место између два гена постаје потенцијална тачка укрштања. Код униформног укрштања (енг. *Uniform crossover*) је битно истаћи вероватноћу наслеђивања, где дете има 0.5 вероватноћу наслеђивања сваког гена и од једног и од другог родитеља, дакле иста је вероватноћа неслеђивања генетског материјала од оба родитеља. Уколико се вероватноћа наслеђивања разликује за поједине гене, онда је у питању поступак *p*-униформног укрштања. У овом случају се на пример задаје вероватноћа неслеђивања $p=0.65$ за одређени ген, што значи да је могуће да ће потомак наследити ген од првог родитеља 65% а вероватноћа наслеђивања гена од другог родитеља 35%.

Битно је такође истаћи појам маске укрштања, која је исте дужине као и родитељски хромозоми. Маска се састоји из случајно генерисаних нула и јединица. Наравно она се користи и код претходно наведеног случаја код различитих процената вероватноће наслеђивања.



Слика 14. Униформно укрштање са маском као бинарним низом

Маска одређује који ће ген код потомака бити наслеђен од којег родитеља. У примеру је дата маска са бинарним низом. Када је бит маске јединица први потомак преузима ген првог родитеља тј. други потомак преузима ген другог родитеља, а када је бит маске нула први потомак преузима ген другог родитеља односно другом потомаку следује ген првог родитеља.

3.5.4 Мешајуће укрштање

У процесу мешајућег укрштања (енг. *Shuffle crossover*) прво се промешају битови код родитеља тј. замене места, а затим се изврши укрштање као и код једнопозиционог или двопозиционог укрштања. Након рекомбинације, битови родитеља се враћају на стара места.

3.6 Мутација

Мутација је генетски оператор који врши случајну промену одређеног броја бита у хромозому. Мутација делује само на једну јединку и веома битан параметар који се користи у процесу мутације је вероватноћа или степен мутације. Ако вероватноћа мутације тежи јединици, алгоритам се претвара у алгоритам случајне претраге решења, а ако тежи нули постоји могућност да се заустављање процеса деси на самом почетку процеса оптимизације у неком од локалних екстремума. Код генетских алгоритама мутација се примењује случајно често са веома ниском вероватноћом (степеном).

После процеса селекције и укрштања може доћи до губитка корисног генетског материјала, мутацијом се омогућава његово враћање. Претрагом околине решења, мутација представља веома добар механизам за избегавање локалних екстремума.

Неки од типова мутације су:

- једноставна мутација
- потпуна мутација
- мешајућа мутација

Код једноставне мутације за сваки бит хромозома генерише се насумичан број *s* из интервала [0,1] а затим ако је број *s* мањи од вероватноће мутације промени вредност бита.

Хромозом пре мутације	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**
Мутирани хромозом	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**0**

Слика 15. Једноставна мутација

Потпуна мутација случајним поступком одабере хромозом (а не ген) и врши се мешање свих битова. Код мешајуће мутације се такође случајно изабере хромозом за мутацију, па затим прву и другу границу и онда случајно генерише гене или их инвертује.

Као што је речено у генетском алгоритму мутација има улогу у спречавању од преурањене оптимизације и никако се не извршава у последњој итерацији алгоритма.

3.7 Критеријум заустављања

Након генерисања популације, над њом се извршавају сви наведени и описани генетски оператори селекција, укрштање, мутација и овај процес се изводи све док се не задовољи услов за заустављање. Овим критеријумом се дефинише и крај генетског алгоритма. Пошто су генетски алгоритми строхастишка метода претраге, веома је тешко дефинисати услов за заустављање.

Када се задовољи критеријум заустављања, крајње решење генетског алгоритма не мора бити и оптимално решење проблема који се решава. Постоје неки услови заустављања који се чешће користе, а дати су неки од критеријума заустављања:

- достигнут максимални број генерација (најчешће примењивани услов)
- најбоља јединка поновљена одређен број пута
- ограничено време извршавања генетског алгоритма
- сличност јединки у популацији
- достигнуто унапред задато оптимално решење
- доказана оптималност најбоље јединке (ако је могуће)
- прекид од стране корисника

3.8 Пример рада генетског алгоритма

Рад генетског алгоритма може да се покаже на примеру оптимизације функције циља једне реалне непознате. Биће објашњен корак по корак рада прве две итерације. У задатку се тражи одређивање максимума функције:

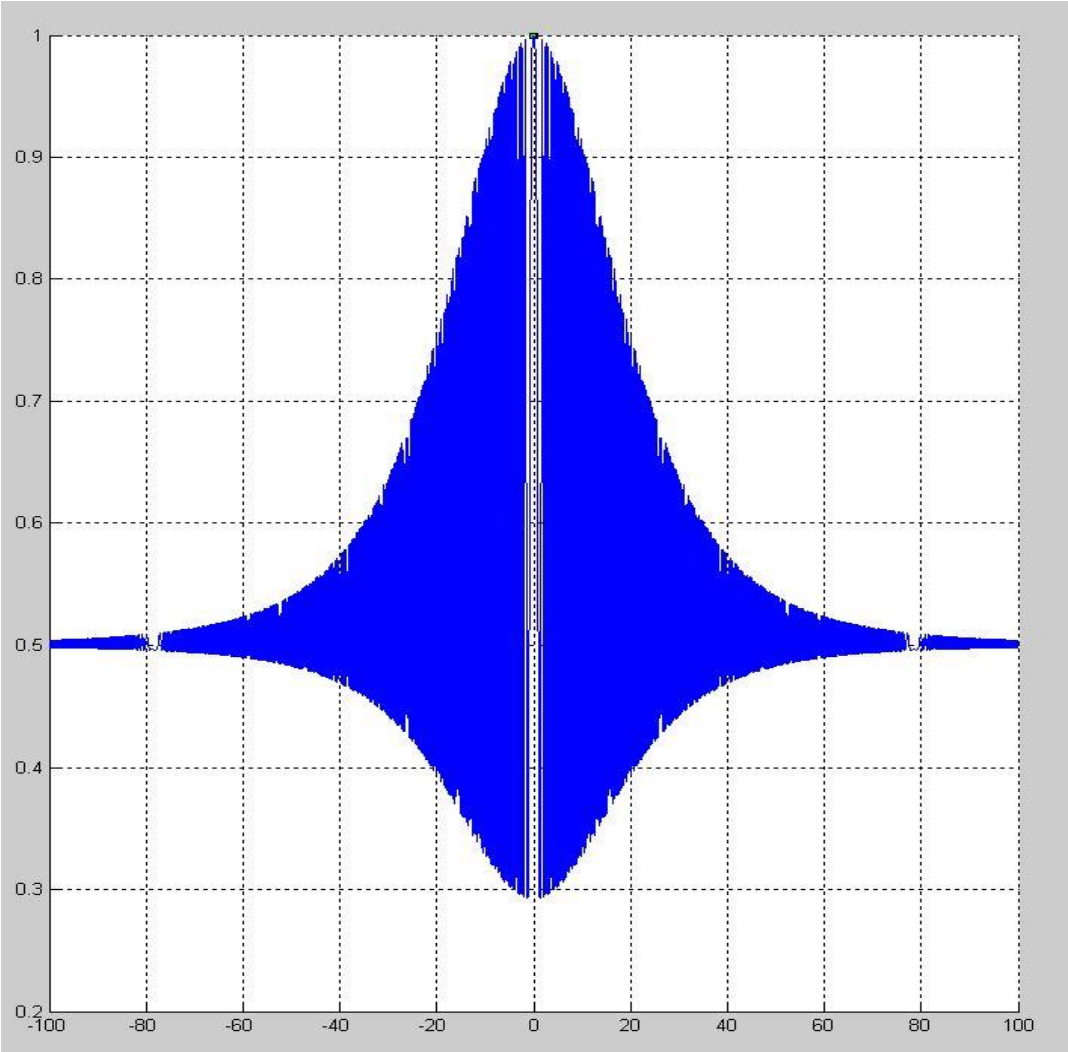
$$f(x) = 0.5 + \frac{\cos^2(\sin(x^2)) - 0.5}{(1 + 0.001 * x^2)^2}$$

која је приказана на слици 14.

Глобални максимум функције се постиже у тачки (0, 1). У овом примеру се користи бинарни приказ од 32 бита како би се постигла прецизност на седам децимала. Узето је да величину популације чини 16 јединки. Број јединки за елиминацију је постављен на 8, а број хромозома за мутацију је постављен на 5. Када се већ одреди хромозом за мутацију, мутирају случајним одабиром његова три до пет гена.

Како се у примеру тражи глобални оптимум у виду максимума функције, тако онда што је вредност функције већа и доброта је већа, што се може представити и формулом:

$$dobrota(v_i) = f(x_i) - \min_i f(x_i)$$



Слика 16. Функција $f(x) = 0.5 + \frac{\cos^2(\sin(x^2)) - 0.5}{(1+0.001*x^2)^2}$

Почетна иницијална популација је добијена генерисањем случајних бројева, у овом случају генерисано је 16 јединки и дата је у табели 2.

Прва итерација.

1. корак: Евалуација. У првом кораку се рачунају вредности хромозома према обрасцу за претварање бинарног кода у еквивалентну реалну вредност. А затим се добијени реални бројеви пропуштају кроз функцију а вредности су уписане у последњој колони.

2. корак: Процена. Врши се процена најбоље и најлошије јединке, у овом случају најбоља јединка је хромозом под редним бројем 10 а најлошија под бројем 6. Након тога се рачуна доброта $d(v_i)$ за сваку јединку према формули која је већ изнад одређена као функција циља. Може се приметити да најлошија јединка има доброту 0. И како је у овом случају у питању елиминацијска селекција, потребно је одредити казне.

3. корак: Селекција. Како је и објашњено у процесу селекције генерешемо у овом примеру осам случајних бројева у интервалу $[0,max\{k_k\}]$, где је $max\{k_k\}$ максимална вредност кумулативне казне, у овом случају је 3.4427.

Редни број	Хромозом	x_i	$f(x_i)$
1	00101110001101100010010000000110	-63.8973	0.4943
2	10000101011001001000111101110010	4.2131	0.3989
3	11001000110000000110001011010000	56.8371	0.5002
4	10011110101110000110010000001010	24.0002	0.4580
5	10001010011100000111111010011100	8.1558	0.7246
6	10001101111001001000111100110010	10.8538	0.3335
7	10001110101010111000101100000000	11.4610	0.6714
8	00100011011101011111100000101000	-72.2962	0.5005
9	10101100101110100000010010110100	34.9427	0.4799
10	01110010101111011111001011111000	-10.3578	0.7517
11	00100001110001110011100101111010	-73.6108	0.5034
12	11000110001011101110110011001110	54.8307	0.5305
13	10000011101100110000010101100100	2.8901	0.4092
14	10001111110110110011100100010110	12.3878	0.7264
15	11100001101100011101010001001100	76.3239	0.5011
16	01010100111001011101001100000010	-33.6736	0.6009

Табела 2. Иницијална популација са 16 јединки

Ако се на пример генерише случајни број 1.2584 који се налази у интервалу [11825, 1.6007] брише се јединка под редним бројем 6, дакле јединка која представља другу границу интервала.

Р. бр.	$d(v_i)$	Казна	k_k	Хромозом	x_i	Укрштање
1	0.1608	0.2574	0.2574	00101110001101100010010000000110	-63.8973	
2	0.0654	0.3528	0.6102	1000010101100100 1000111101110010	4.2131	4+13
3	0.1667	0.2515	0.8617	11001000110000000110001011010000	56.8371	13+10
4	0.1245	0.2937	1.1554	10011110101110000110010000001010	24.0002	
5	0.3911	0.0271	1.1825	10001010011100000111111010011100	8.1558	8+10
6	0	0.4182	1.6007	10001101111001001000111100110010	10.8538	4+16
7	0.3379	0.0803	1.6810	10001110101010111000101100000000	11.4610	
8	0.1670	0.2512	1.9322	00100011011101011111100000101000	-72.2962	
9	0.1464	0.2718	2.2040	10101100101110100000010010110100	34.9427	1+13
10	0.4182	0	2.2040	01110010101111011111001011111000	-10.3578	
11	0.1699	0.2483	2.4523	00100001110001110011100101111010	-73.6108	4+13
12	0.1970	0.2212	2.6735	11000110001011101110110011001110	54.8307	1+8
13	0.0757	0.3425	3.0160	10000011101100110000010101100100	2.8901	
14	0.3929	0.0253	3.0413	10001111110110110011100100010110	12.3878	
15	0.1676	0.2506	3.2919	11100001101100011101010001001100	76.3239	7+14
16	0.2674	0.1508	3.4427	01010100111001011101001100000010	-33.6736	

Табела 3. Стање популације после селекције а пре укрштања

Редни број	Хромозом	x_i	$f(x_i)$
1	00101110001101100010010000000110	-63.8973	0.4943
2	00100110**0**0110110**0**011010000100100	-70.1471	0.5004
3	10100011101111110010010101110000	27.9271	0.5197
4	10011110101110000110010000001010	24.0002	0.4580
5	01100010011**1**1101111110**0**010101000	-23.0531	0.6175
6	10011110101100000110010100001010	23.9758	0.6995
7	10001110101010111000101100000000	11.4610	0.6714
8	00100011011101011111100000101000	-72.2962	0.5005
9	10101111001100100**0**010010100100100	36.8718	0.5147
10	01110010101111011111001011111000	-10.3578	0.7517
11	1001101**0**10110010000**0**0010**1**01001010	20.8558	0.4037
12	00101111001101000011110000101010	-63.1218	0.5021
13	10000011101100110000010101100100	2.8901	0.4092
14	10001111110110110011100100010110	12.3878	0.7264
15	10000111**1**11100110001000100000110	6.2105	0.5058
16	01010100111001011101001100000010	-33.6736	0.6009

Табела 4. Популација након прве итерације

Након сваке елиминације јединке потребно је поново прорачунати кумулативну казну али за јединке које следе. У овом случају се рачуна поновљена кумулативна казна за јединке под редним бројевима 7 до 16. Тек после израчунате нове кумулативне казне се приступа генерисању случајног броја сада за нови интервал. Вредност нове $max\{k_k\}$ је мања од оне у претходном кораку за износ казне елиминисане јединке. Овакав поступак се понавља све док се не елиминише осам јединки. У овом случају, у првој генерацији елиминисане су јединке: 2,3,5,6,9,11,12,15. У табели 3 су елиминисане јединке означене затамњеним колонама.

4. корак: Укрштање. Елиминисане јединке се попуњавају укрштањем преосталих јединки, јер је величина популације током целог алгоритма једнака. Све преостале јединке имају исту вероватноћу укрштања, па је тако новонастала јединка под редним бројем 2 добијена укрштањем јединки под редним бројевима 4 и 13, и тако даље за преостале елиминисане хромозоме .

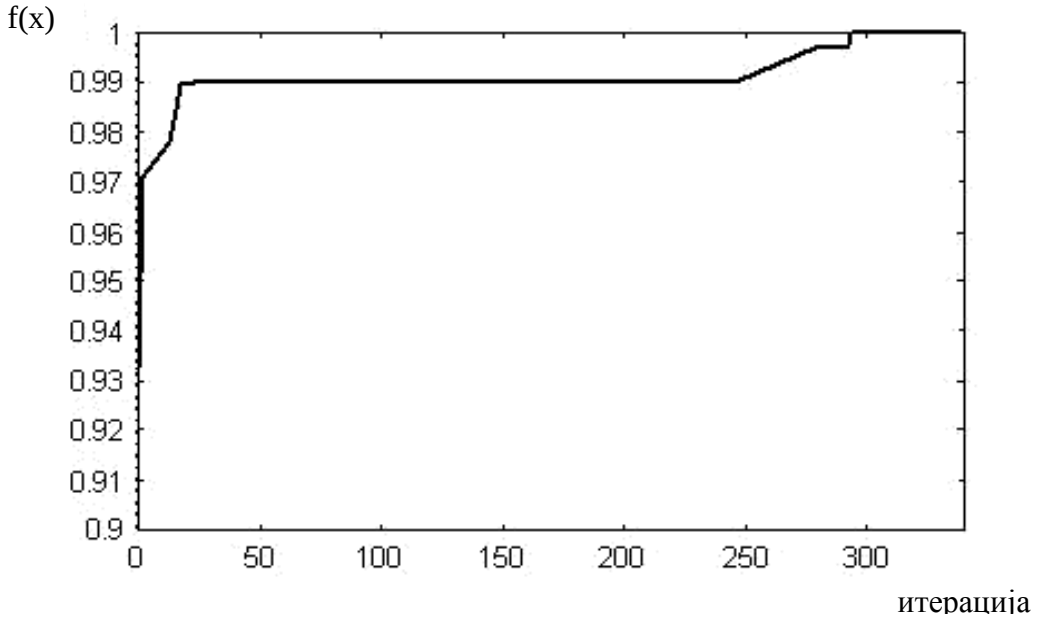
5. корак: Мутација. За мутацију се случајно селектира пет јединки с'тим што једна јединка може бити више пута одабрана за селекцију. Најбоља јединка је заштићена од мутације механизмом елитизма, а све преостале јединке имају исту вероватноћу мутације. Гени на које је деловала мутација су у табели 4 приказани црвеном бојом.

Р. бр.	$d(v_i)$	Казна	k_k	Хромозом	x_i	Укрштање
1	0.0906	0.2574	0.2574	00101110001101100010010000000110	-63.8973	
2	0.0967	0.2513	0.5087	00100110001101100011010000100100	-70.1471	
3	0.1160	0.2320	0.7407	10100011101111110010010101110000	27.9271	
4	0.0543	0.2937	1.0344	10011110101110000110010000001010	24.0002	
5	0.2138	0.1342	1.1686	01100010011111011111100010101000	-23.0531	
6	0.2958	0.0522	1.2208	10011110101100000110010100001010	23.9758	
7	0.2677	0.0803	1.3011	10001110101010111000101100000000	11.4610	
8	0.0968	0.2512	1.5523	00100011011101011111100000101000	-72.2962	
9	0.1110	0.2370	1.7893	10101111001100100010010100100100	36.8718	
10	0.3480	0	1.7893	01110010101111011111001011111000	-10.3578	
11	0	0.3480	2.1373	10011010101100100000010101001010	20.8558	
12	0.0984	0.2496	2.3869	00101111001101000011110000101010	-63.1218	
13	0.0055	0.3425	2.7294	10000011101100110000010101100100	2.8901	
14	0.3227	0.0253	2.7547	10001111110110110011100100010110	12.3878	
15	0.1021	0.2459	3.0006	10000111111100110001000100000110	6.2105	
16	0.1972	0.1508	3.1514	01010100111001011101001100000010	-33.6736	

Табела 5. Популација у другој итерацији пре укрштања

Друга итерација.

Најбоља јединка је у другој иритацији остала иста, и већ у другој иритацији се бележи побољшање, сад је кумулативна казна 3.1514. Понавља се потпуно иста процедура као и у првој итерацији и тако из генерације у генерацију док се не задовољи обично унапред задат услов за завршетак еволутивног процеса. Критеријум за заустављање процеса је најчешће задат у виду броја итерација (генерација), мада могу бити задати и други критеријуми као на пример минимални напредак најбољег решења за одређени број генерација (побољшање функције циља најбољег хромозома). Како је у овом примеру унапред задат број итерација тј. 350 генерација, глобални максимум функције се добија тако што се у 350-тој итерацији одабере јединка која има највећу доброту а то је хромозом са бинарним кодом 10000000000000000000000000000000.



Слика 17. Графички приказ рада генетског алгоритма који из генерације у генерацију чува најбољу јединку и тежи ка глобалном оптимуму

На слици 17 је графички приказан генетски алгоритам примера који је објашњен, постепено за сваку генерацију. На x оси је дат број итерација (генерација) а на y оси је дата функција циља која се добија евалуацијом најбоље јединке у тој генерацији.

4. ИЗБОР ПАРАМЕТАРА И ЊИХОВ УТИЦАЈ НА РАД ГЕНЕТСКОГ АЛГОРИТМА

4.1 Генетски алгоритми у MATLAB-у

MATLAB као програмски језик поред поседовања многих својих алата садржи и алат за генетске алгоритме. У наставку ће бити описане битне наредбе и синтаксе везане за генетске алгоритме и коришћење самог алата. Све ове наредбе су значајне јер учествују у подешавању параметара код генетског алгоритма.

Како би покренули генетски алгоритам из линије наредби треба позвати функцију за генетске алгоритме са синтаксом[26]:

[x fval] = ga(@fitnessfun, nvars, options)

где су:

@fitnessfun – handle за функцију циља
nvars – број независних улазних променљивих у функцију циља
options – структура у којој су садржане поставке параметара за ГА, и ако се овај аргумент изостави, постављају се претпостављене поставке

Како би се добиле детаљне информације може се користити следећа синтакса:

[x fval exitflag output population scores] = ga(@fitnessfcn, nvars, options)

где су:

x – моменат када је достигнута коначна вредност функције циља
fval – коначна вредност функције циља
exitflag – аргумент који описује разлоге прекида ГА
output – структура садржи информације о обављању ГА у свакој генерацији
population – коначна популација
scores – коначна решења

Генетски алгоритми са претпостављеним поставкама се позивају функцијом у нешто краћем облику:

[x fval] = ga(@fitnessfun, nvars)

У самој структури *options* која је део функције генетског алгоритма су дате све поставке везане за параметре. За преглед свих параметара структуре *options* и вредности које оне могу имати у линији наредби треба уписати **gaoptimset** без аргумената (табела 6).

Потребно је структуру *options* створити, пре њеног коришћења као аргумента функције генетског алгоритма[27]:

options = gaoptimset(@ga)

Када се упути овакав позив, врши се генерисање структуре *options* са претпостављеним вредностима, а оне су:

options =

PopulationType: 'doubleVector'
PopInitRange: []
PopulationSize: '50 when numberOfVariables <= 5, else 200'
EliteCount: '0.05*PopulationSize'
CrossoverFraction: 0.8000
ParetoFraction: []
MigrationDirection: 'forward'
MigrationInterval: 20
MigrationFraction: 0.2000
Generations: '100*numberOfVariables'
TimeLimit: Inf
FitnessLimit: -Inf
StallGenLimit: 50
StallTest: 'averageChange'
StallTimeLimit: Inf
TolFun: 1.0000e-06
TolCon: 1.0000e-03
InitialPopulation: []
InitialScores: []
NonlinConAlgorithm: 'auglag'
InitialPenalty: 10
PenaltyFactor: 100
PlotInterval: 1
CreationFcn: @gacreationuniform
FitnessScalingFcn: @fitscalingrank
SelectionFcn: @selectionstochunif
CrossoverFcn: @crossoverscattered
MutationFcn: {[@mutationgaussian] [1] [1]}

DistanceMeasureFcn: []
HybridFcn: []
Display: 'final'
PlotFcns: []
OutputFcns: []
Vectorized: 'off'
UseParallel: 0

Обично се већина параметара у генетским алгоритмима користе са претпостављеним поставкама, ипак током рада са генетским алгоритмима потребно је неке мењати у жељи за одређеним резултатима, како би се деловало на његове радне карактеристике. То је могуће извршити позивањем функције:

options = gaoptimset('parametar1', vrednost1, 'parametar2', vrednost2,…)

Параметри	Опис	Могућа вредност
CreationFcn	Handle за функцију која креира почетну популацију	{@gacreationuniform}
CrossoverFcn	Handle за функцију чији се алгоритам користи за креирање деце при укрштању	@crossoverheuristic {@crossoverscattered} @crossoverintermediate @crossoversinglepoint @crossovertwopoint @crossoverarithmetic
CrossoverFraction	Део популације следеће генерације, не укључује децу елитних јединки, који је креиран функцијом укрштања	Positive scalar | {0.8}
Display	Ниво приказа	'off' | 'iter' | 'diagnose' | {'final'}
DistanceMeasureFcn	Handle за функцију која рачуна удаљеност међу јединкама	{@distancecrowding, 'phenotype'}
EliteCount	Позитиван цели број који говори колико јединки из тренутне генерације сигурно прелази у наредну генерацију, тј. колико је елитних јединки	Pozitivan celi broj | {2}
FitnessLimit	Скалар. Када функција циља досегне ову вредност, алгоритам стаје	Skalar | {-Inf}
FitnessScalingFcn	Handle за функцију која мери вредност функције циља	@fitscalingshiftlinear @fitscalingprop @fitscalingtop

		{@fitscalingrank}
Generations	Позитиван цели број који одређује највећи могући број понављања алгоритма пре његовог прекидања	Pozitivan celi broj | {100}
HybridFcn	Handle за функцију која наставља оптимизацију након што ГА бива прекинут или поље ћелија које одређује хибридну функцију и могућности њене структуре	Function handle | @fminsearch @patternsearch @fminunc @fmincon {[]} ili 1-2 polje ćelija| {@solver, hybridoptions}, gde je solver = fminsearch, patternsearch, fminunc, or fmincon {[]}
InitialPenalty	Почетна вредност параметра казне	Pozitivan skalar | {10}
InitialPopulation	Почетна популација која се користи као семе за генетски алгоритам	Matrica | {[]}
InitialScores	Почетна вредност која одређује доброту; може бити делимична	Jednostepeni vektor | {[ ]}
MigrationDirection	Смер миграције	'both' | {'forward'}
MigrationFraction	Скалар између 0 и 1 који одређује проценат јединки у свакој подпопулацији која мигрира у различиту подпопулацију	Skalar | {0.2}
MigrationInterval	Позитиван цели број који одређује сваких колико генерација долази до миграције	Pozitivan celi broj | {20}
MutationFcn	Handle за функцију која производи мутирану децу	@mutationuniform @mutationadaptfeasible {@mutationgaussian}
OutputFcns	Поље handle-ова за функције које цртају податке израчунате алгоритмом	@gaplotbestf @gaplotbestindiv @gaplotdistance @gaplotexpectation @gaplotgeneology @gaplotselection @gaplotrange @gaplotscorediversity @gaplotscores @gaplotstopping | {[]}
PlotInterval	Позитиван цели број који одређује сваких колико генерација се врши испис	Pozitivan celi broj | {1}

PopInitRange	Матрица или вектор који одређује распон јединки у почетној популацији	Matrica ili vektor | [0;1]
PopulationSize	Величина популације	Pozitivan celi broj | {20}
PopulationType	String описује тип података функције	'bitstring' | 'custom' | {'doubleVector'}
SelectionFcn	Handle за функцију која бира родитеље деце која настају укрштањем и мутацијом	@selectionremainder @selectionuniform {@selectionstochunif} @selectionroulette @selectiontournament
StallGenLimit	Позитиван цели број. Алгоритам стаје ако у објектној функцији нема напретка у наредних StallGenLimit генерација	Pozitivan cijeli broj |{50}
StallTimeLimit	Позитиван скалар. Алгоритам стаје ако нема напретка у објектној функцији StallTimeLimit секунди	Pozitivan skalar | {Inf}
TimeLimit	Позитиван скалар. Алгоритам завршава након што је покренут TimeLimit секунди	Pozitivan skalar | {inf}
TolCon	Позитиван скалар. TolCon се користи за одређивање изводљивости с' обзиром на нелинеарна ограничења	Pozitivni skalar | {1e-6}
TolFun	Позитиван скалар. Алгоритам се изводи све док је укупна промена у функцији циља кроз StallGenLimit мања од TolFun	Pozitivni skalar | {1e-6}
UseParallel	Рачуна функцију циља популације паралелно	'always' | {'never'}
Vectorized	String одређује да ли је рачунање функције прелаза векторизовано	'on' | {'off'}

Табела 6. Опис структуре *options* и свих њених параметара

4.2 Параметри генетског алгоритма

Улога параметара је да управљају радом генетског алгоритма. Као што се види и из табеле 6 постоји мноштво параметара у генетском алгоритму, ипак треба издвојити неке које спадају у групу стандардних, који имају највећи утицај на перформансе генетског алгоритма. То су следећи параметри:

- величина популације
- број итерација или генерација
- вероватноћа мутације

У генетским алгоритмима поред ових битну улогу на њихов рад и понашање имају и остали параметри а као што је речено њихов број је велики, неки од њих су:

- вероватноћа укрштања
- број јединки за елиминацију код елиминацијског алгоритма
- број елитних јединки (елитизам)
- иницијална популација
- дужина бинарног кода

Неки генетски алгоритми могу користити све наведене параметре, а неки могу користити само одређене, ипак сви примењују прва три у свом раду. Поред ових наведених параметара овде, постоје посебни које користе генетски алгоритми сложене структуре.

Уз помоћ параметара могуће је утицати на брзину којом ће алгоритам да конвергира ка оптимуму, на време извођења генетског алгоритма, на начин који ће алгоритам претраживати простор решења.

Параметри представљају важну компоненту рада алгоритма. При извршавању генетског алгоритма примећено је да фиксни избор параметара, а нарочито вероватноћа мутације, није увек најпогоднија[28]. Такође у свим фазама извршавања генетског алгоритма, разноврсност генетског алгоритма није увек једнака, па се веома често дешавају случајеви да се оптимална вредност нивоа укрштања, као и мутације мењају током извршавања генетског алгоритма. Многи су покушавали да одреде скупове параметара који би били оптимални за велики број проблема, међутим показало се да сваки проблем има свој скуп оптималних параметара, тако да није могуће одредити параметре који би одговарали за шири спектар проблема. На основу овога може се рећи да је оптимизација параметара генетског алгоритма веома сложен задатак.

Може се рећи да ипак постоје уобичајене вредности параметра за генетске алгоритме са малим и великим популацијама, препоруке су дате у табели *7*.

Параметри	Мала популација	Велика популација
Величина популације	30	100
Вероватноћа укрштања	0.9	0.6
Вероватноћа мутације	0.01	0.001
Број јединки за елиминацију	Vel_pop/2	Vel_pop/4

Табела 7. Препорука параметара за мале и велике популације

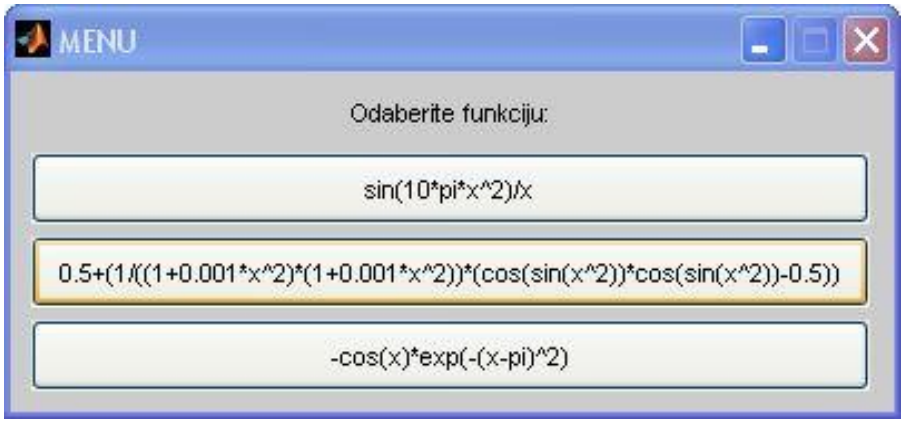
У току извршавања генетског алгоритма постоје два начина промене параметара:

- фиксна промена параметара – овим начином се унапред задаје смер промене (током генерација се повећавају или смањују вредности), и формула по којој се промена врши. Код неких случајева је постепено смањивање процента укрштања и истовремено постепено повећавање процента мутације давало добре резултате[29]. Под другим околностима се ефикаснијим показало експоненцијално смањивање процента мутације [30][31].

- адаптивно одређивање параметара – начин код кога се посматра како се дати оператор понаша и какве је до тада резулатате постигао тако се и одређени параметри прилагођавају. Случајним одабиром на основу своје тежине, тј. дотадашње успешности бира се оператор који ће бити примењен [32-35][25]. Током рада генетског алгоритма, он се може тако поставити да се параметри могу сами подешавати, без деловања корисника, како би се променило понашање алгоритма. На пример, ако се најбоља јединка није променила кроз сто генерација , алгоритам тада мења величину популације и вероватноћу мутације како не би заглавио у локалном оптимуму.

4.2.1 Пример

Пример који је коришћен како би се показао утицај параметара на рад генетског алгоритма, је одрађен у MATLAB-у. Како би се лакше приказао и пратио пример, а и како би се лакше управљало параметрима као и за бољи преглед резултата, направљен је графички интерфејс (слика 18).



Слика 18. Интерфејс за одабир функције за оптимизацију

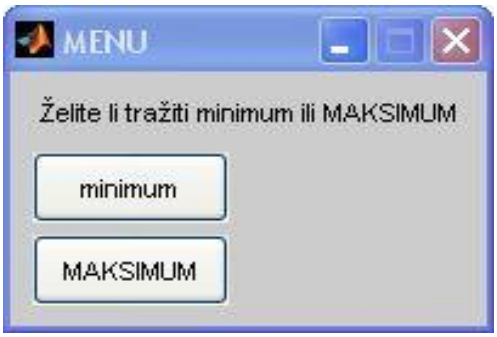
Као што се види на мени интерфејсу су дате за одабир три функције:

$$f_1(x) = \frac{\sin(10 * \pi * x^2)}{x}$$

$$f_2(x) = 0.5 + \frac{cos^2(\sin(x^2)) - 0.5}{(1 + 0.001 * x^2)^2}$$

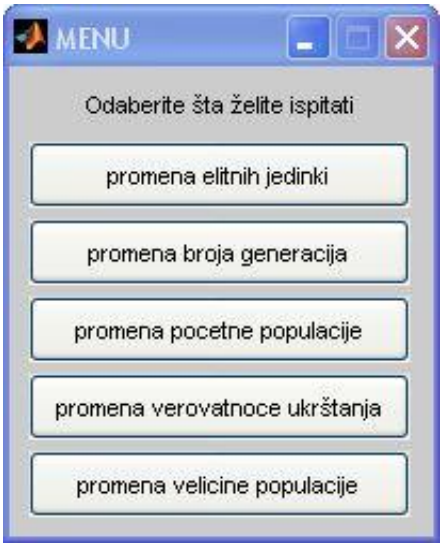
$$f_3(x) = -\cos(x) * e^{-(x-\pi)^2}$$

Одабиром једне од њих отвара се следећи прозор (у овом примеру је одабрана прва функција):



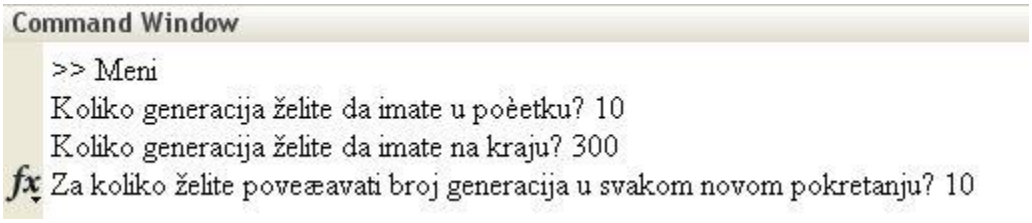
Слика 19. Интерфејс за одабир минимума или максимума

На овом интерфејсу (слика 19) се може бирати да ли се циља минимум или максимум функције. У овом случају је одабран као глобални оптимум максимум функције. Након тога се отвара следећи прозор:

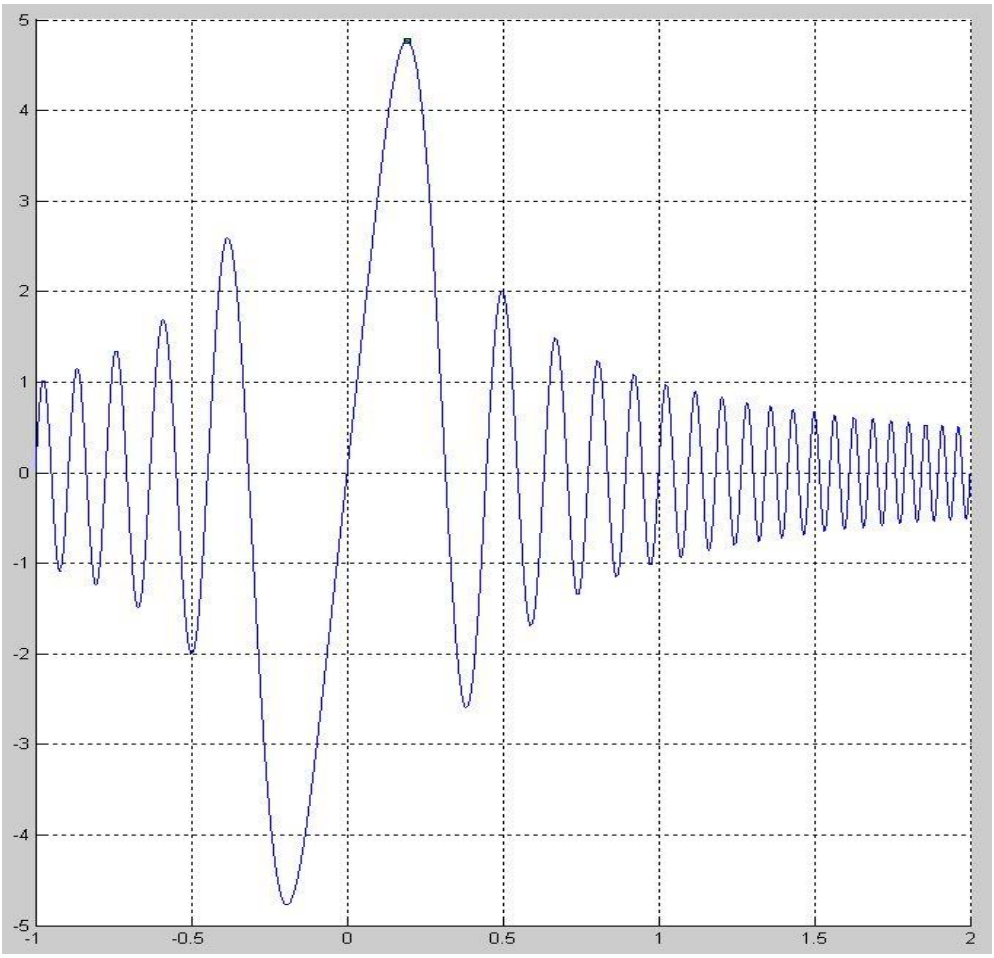


Слика 20. Интерфејс за одабир параметра

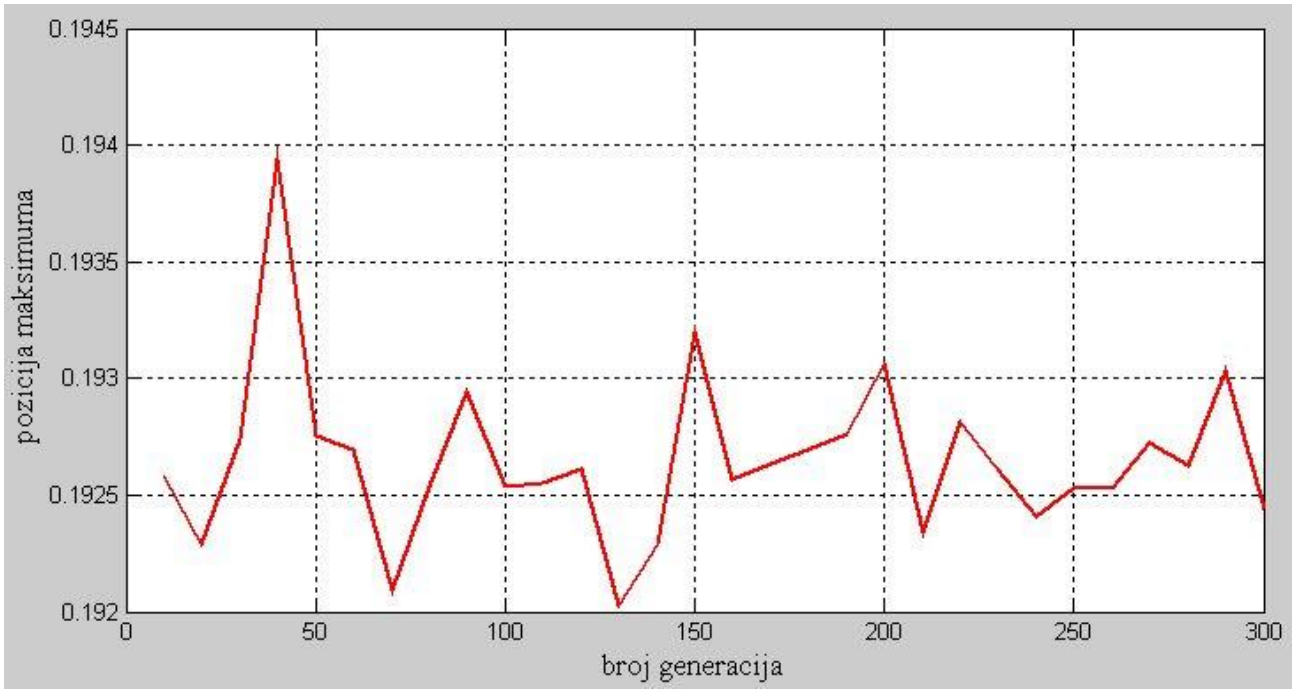
У овом менију (слика 20) су дати различити параметри кроз које се може посматрати промена рада генетског алгоритма, с' тим што се у самом коду овог програма у MATLAB-у може у истом случају задати више параметара како би се добило што боље оптимално решење. Нека је у овом примеру изабрана опција са променом броја генерација. По клику на то дугме прелази се на рад у главном прозору MATLAB-а:



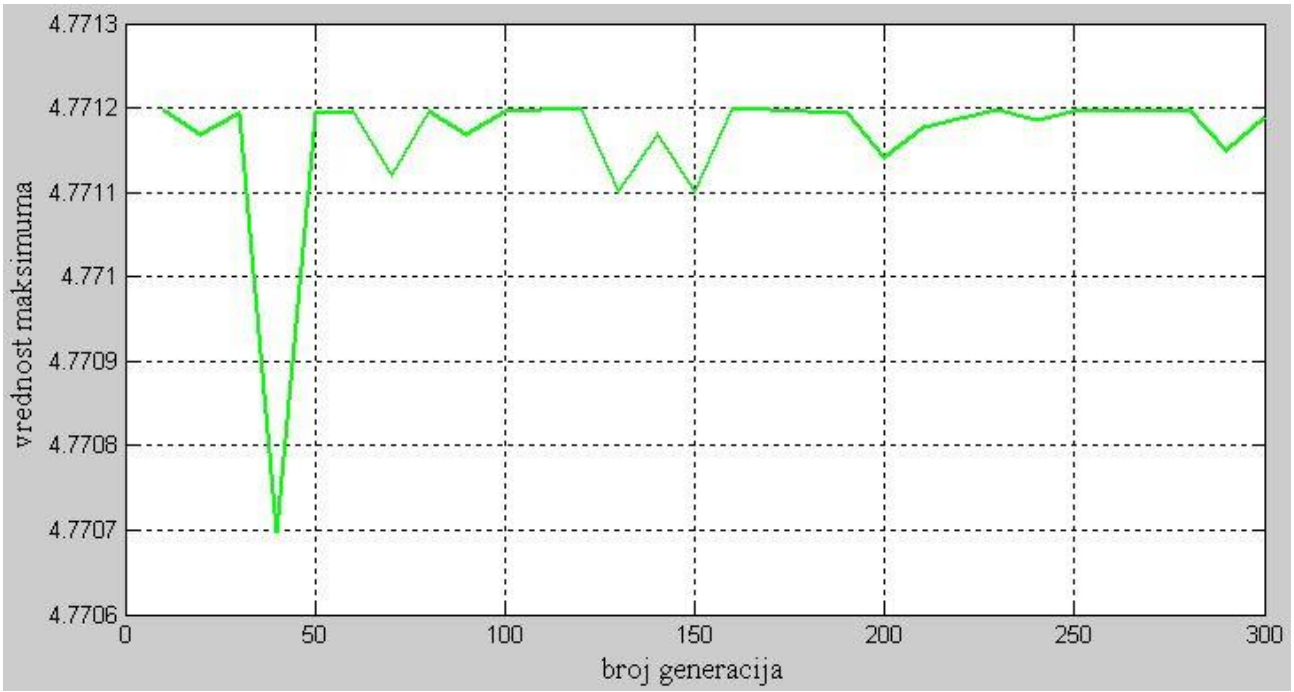
Слика 21. Командни прозор MATLAB-а



Слика 22. График функције са максимумом



Слика 23. Позиција максимума функције у односу на број генерација



Слика 24. Вредност максимума функције у односу на број генерација

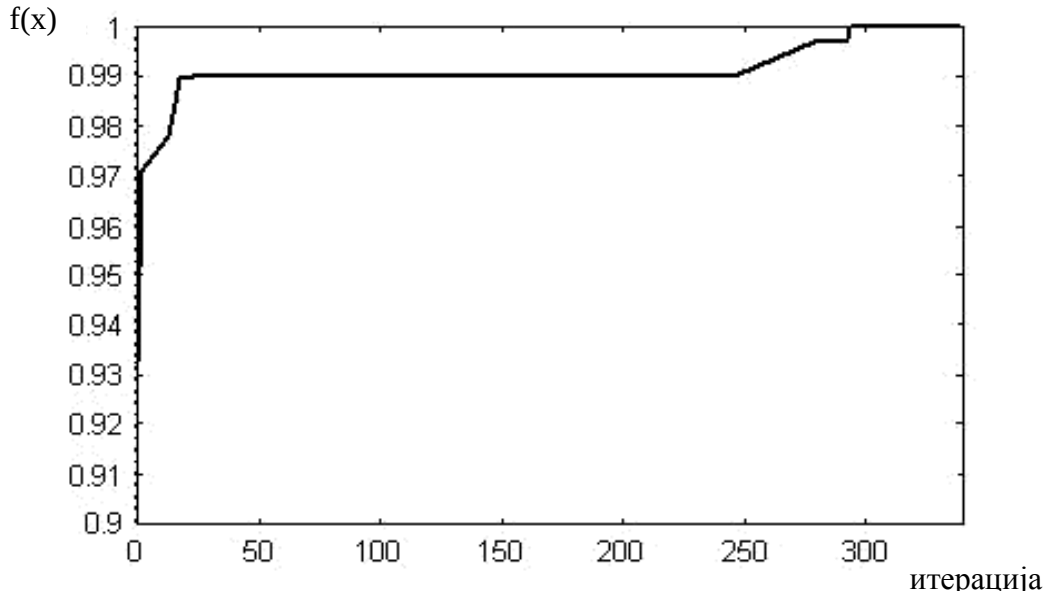
У главном прозору MATLAB-а су постављена питања са колико генерација се почиње а са колико завршава рад генетског алгоритма (слика 21). Такође се пита за колико се број генерација повећава у сваком покретању, то значи да се резултат исписује на толико итерација. Дакле дат је критеријум за заустављање, у овом примеру је задато 300 генерација.

Након одрађеног читавог процеса задавања параметара, алгоритам ради на принципу који је приказан у одељку 3.8 и након тога се појављује интерфејс са графиком функције на којем су обележени сви максимуми који су резултат рада алгоритма из генерације у генерацију (унапред већ задат критеријум, за колико се повећава број генерација у сваком новом покретању). Са десне стране су графици са позицијом и вредношћу максимума функције у односу на број генерација, где се види њихова промена кроз раст генерације. Сви графици су приказани на сликама 22,23,24.

Слика 25. Интерфејс за наставак или завршетак генетског алгоритма

4.2.2 Утицај броја генерација

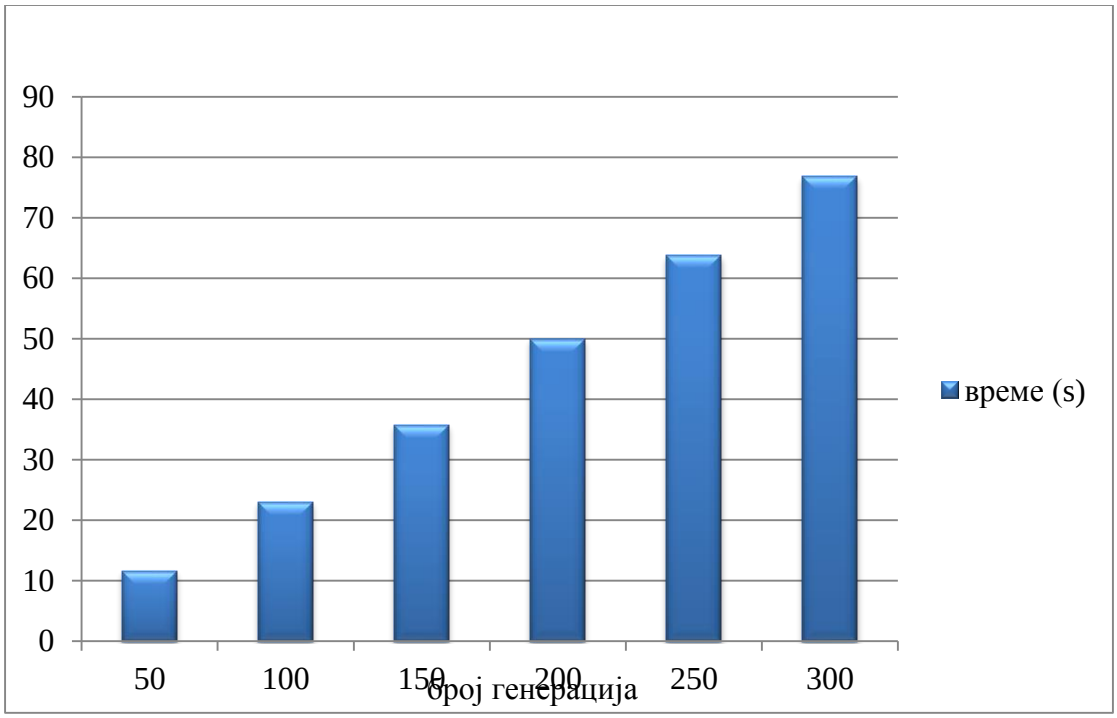
Број генерација или итерација је параметар који директно утиче на време извођења генетског алгоритма као и на квалитет добијеног решења. Овај параметар има велики утицај на перформансе алгоритма. Може да се искористи као показатељ дејства броја итерација, график из одељка о примеру рада генетског алгоритма 3.8.



Слика 26. Графички приказ рада генетског алгоритма и зависност функције циља и броја генерација

Може се констатовати са примера да што је већи број итерација решење ће бити боље. Како се тражи у примеру максимум функције а он се налази у 1, види се да из генерације у генерацију се постижу све бољи резултати. Тако се прелаз из локалног оптимума *f(x)=0.99* на глобални *f(x)=1* догађа око 250-те итерације.

Нормално је заочекивати да што је већи број итерација то ће се и више времена потрошити на извођење генетског алгоритма. Већи број генерација значи и дужи рад алгоритма.

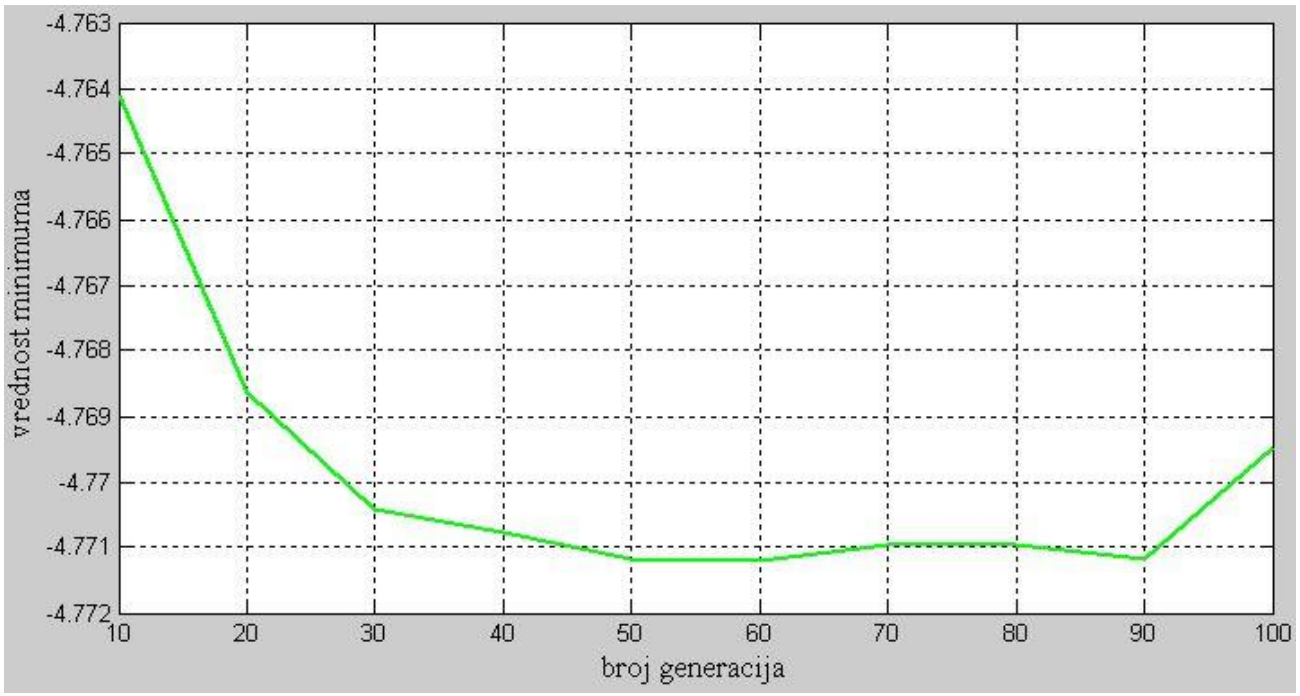


Слика 27. Однос броја генерација и времена рада генетског алгоритма

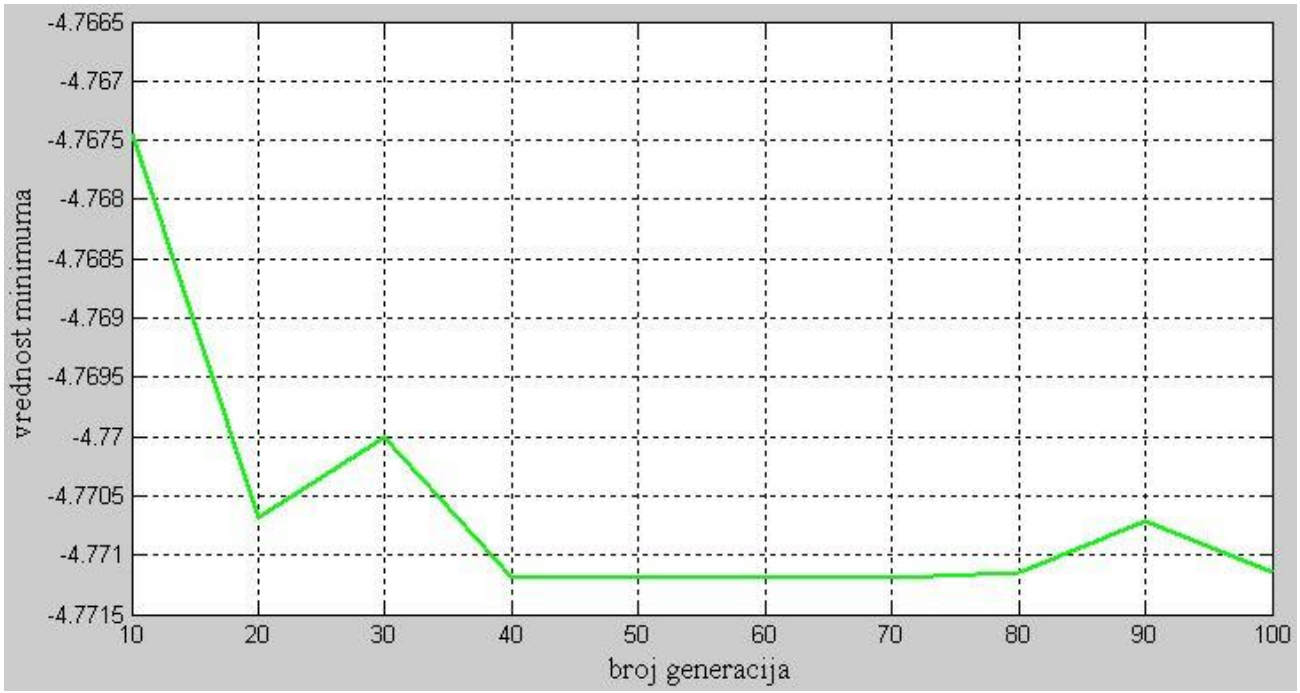
На слици 27 је дат график са односом броја генерација и потрошеног времена при раду алгоритма за пример прве функције. Свакако да на време рада имају утицај и други параметри али су они у овом случају били фиксни како би се видео само утицај броја генерација. Како је циљ да се постигне што боље решење у што краћем времену, потребно је пронаћи број итерација у којем ће алгоритам дати задовољавајуће решење.

4.2.3 Утицај величине популације

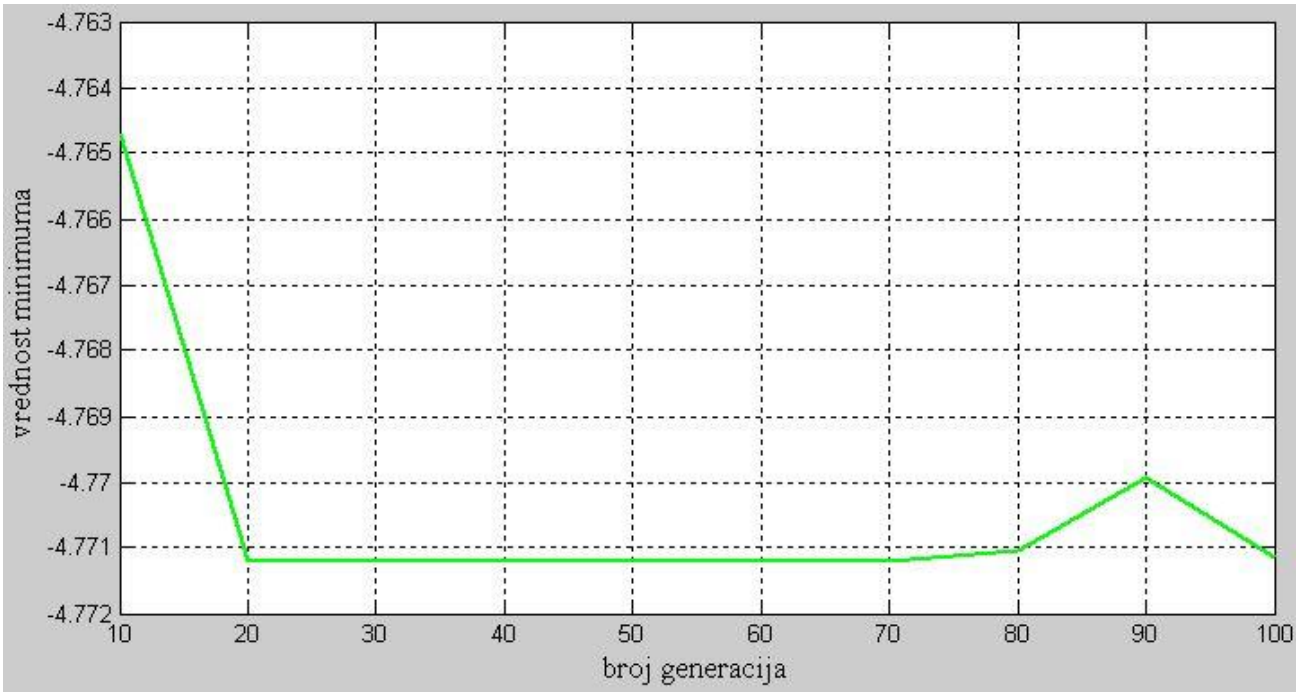
Утицај величине популације на перформансе генетског алгоритма се огледа у квалитету добијеног решења. Величина популације је параметар који одређује колико ће јединки бити у појединој генерацији. Што је већи број величине популације, генетски алгоритам решење тражи темељније па се смањује могућност да алгоритам врати локални минимум који уједно није и глобални. Као што је речено и код броја генерација, и овде је исти случај са временом. Како се повећа број величине популације, решење ће несумњиво бити боље, тако се повећава и време рада алгоритма.



Слика 28. График добијеног решења када је величина популације 50



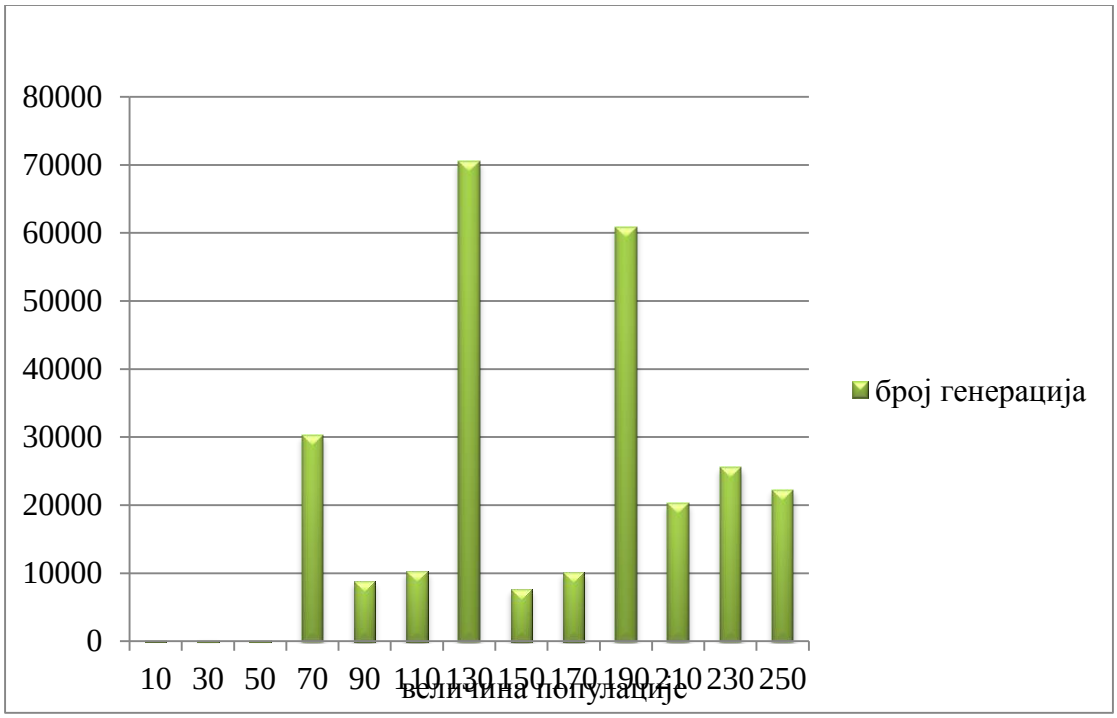
Слика 29. График добијеног решења када је величина популације 75



Слика 30. График добијеног решења када је величина популације 100

У примеру са три функције (4.2.1) одрађен је процес генетског алгоритма у овом случају је изабрана прва функција у којој је тражен минимум. У првом случају је избор величине популације био 50, а у другом је изабрана популација од 75 јединки, а у трећем случају је та бројка била 100. Рад алгоритма је посматран у броју генерација 100. На првом графику се види да је минимум функције добијен тек око 50 генерације, а на другом графику где је величина популације 75 решење је пронађено око 40 генерације. Ипак се до глобалног минимума најпре дошло у трећем случају када је он пронађен око 20 генерације. Може се приметити да су у трећем случају и решења константнија и боља из итерације у итерацију. Овим је потврђена претходна тврдња да што је већа величина популације, алгоритам тражи темељније решење уједно даје и боље резултате, и да алгоритам стиже пре до глобалног оптимума.

Такође за различите величине популација је потребно избалансирати број генерација како би се дошло до што бољег решења. У многим проблемима који су решавани путем генетског алгоритма је тражен фиксни број величине популације који даје најбоље резултате али као и код свих параметара нису пронађени одговарајући резултати. Ипак у примеру решавања проблема *N* краљица уз помоћ генетског алгоритма су добијени занимљиви резултати. Посматран је просечан утицај величине популације на број генерација потребним генетском алгоритму да дође до решења. Резултати су приказани на слици 31.

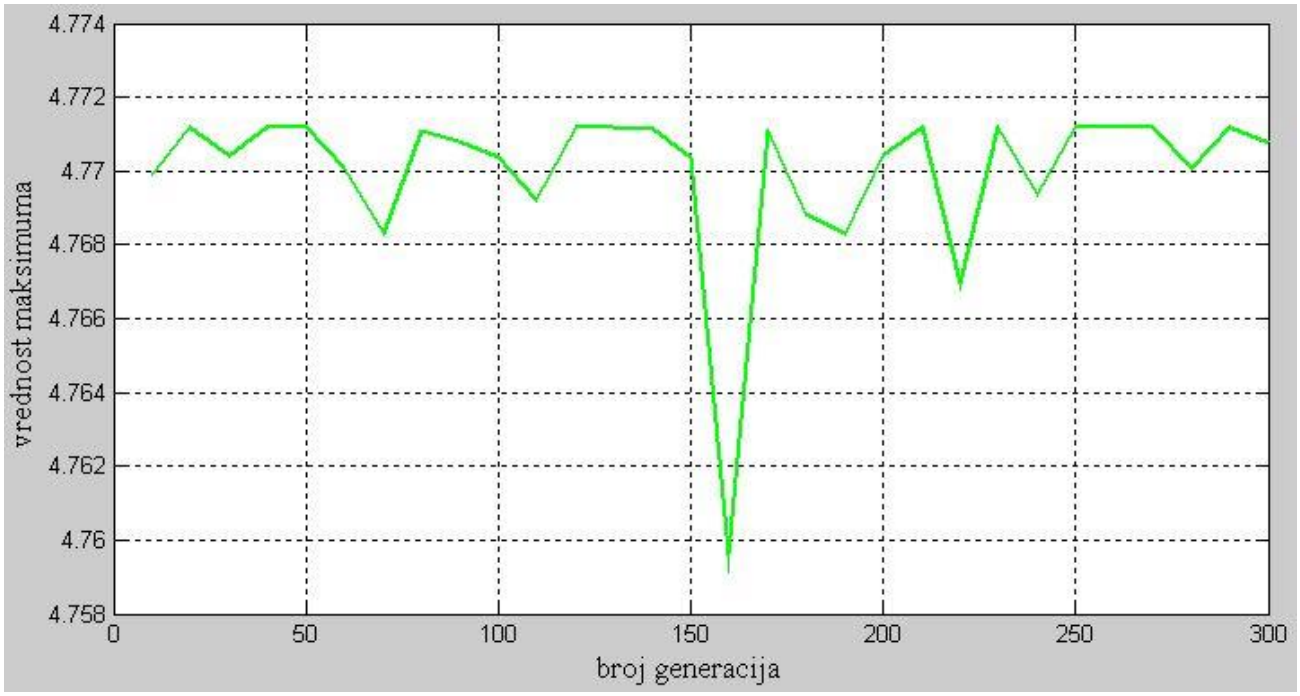


Слика 31. Утицај величине популације на проналазак решења

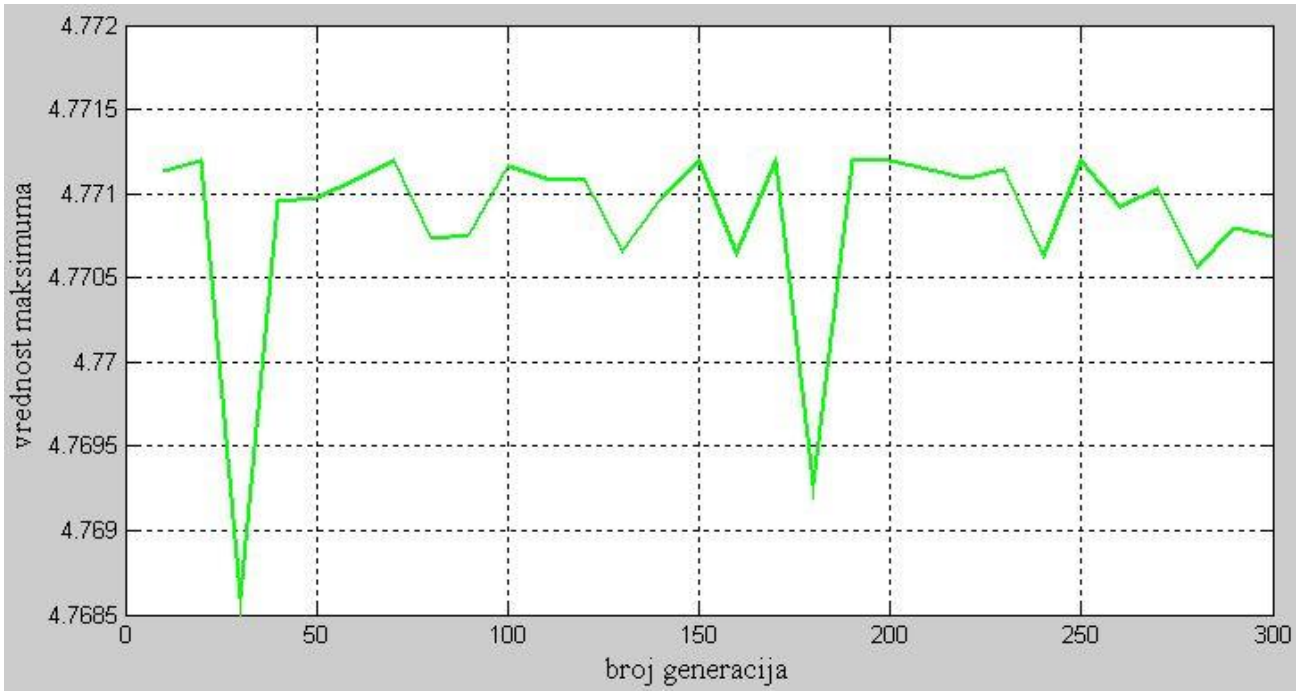
Са слике 31 се види неконзистентност генетског алгоритма код броја генерација потребних за проналазак решења уз различите бројке величине популација. Ефикасност алгоритма опада са повећањем броја јединки. Код решавања овог проблема алгоритму више одговара мања популација, односно решење проблема је пре достижно са мањим бројем јединки у популацији[36].

4.2.4 Утицај вероватноће мутације

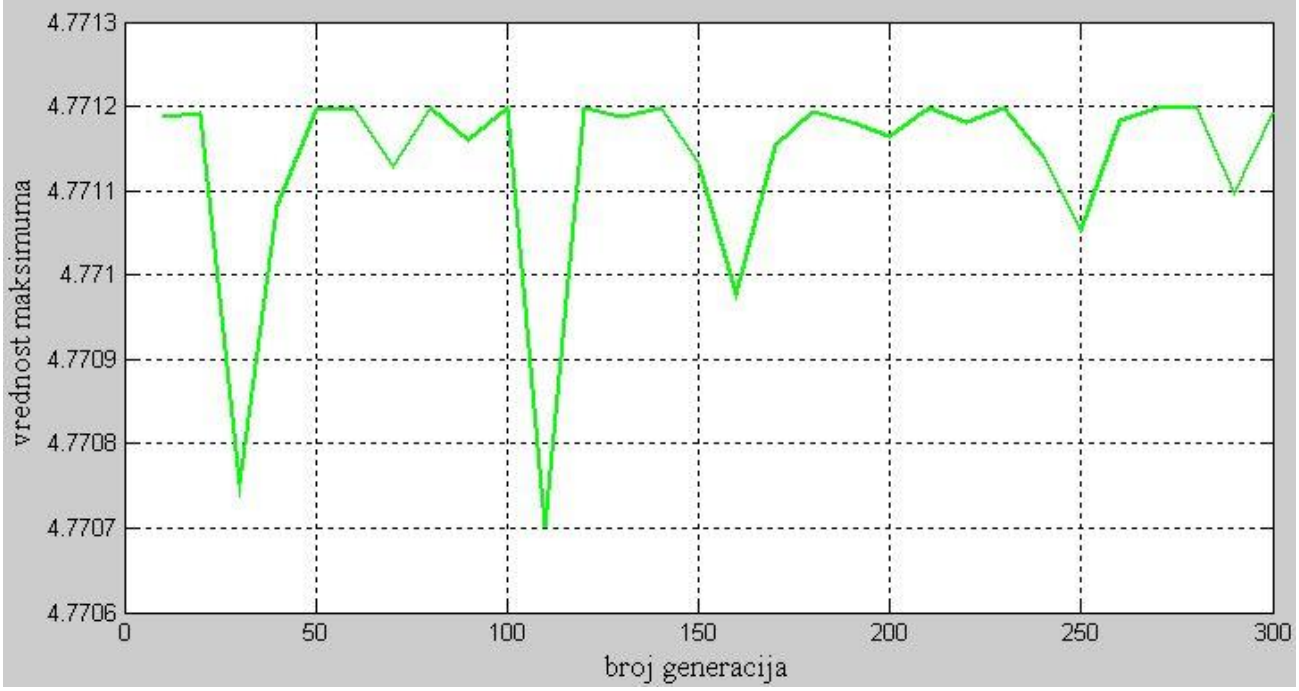
Вероватноћа мутације је један од најважнијих параметара генетског алгоритма. Поставке мутације одређују како генетски алгоритам ради мале случајне промене јединки популације. Мутацијом као параметром се обезбеђује генетска разноликост и генетском алгоритму шири простор претраге решења, а управо је мутација механизам за избегавање локалног оптимума. Утицај вероватноће мутације се може огледати да ако читава популација заврши у неком од локалних минимума односно максимума, једино случајним претраживањем решења у још неистраженим деловима простора, проналази се боље решење. Сасвим је довољно да једна јединка настала мутацијом буде боља од осталих, па да се у следећих неколико генерација све јединке преселе у простор са бољим решењем.



Слика 32. График добијеног решења када је вероватноћа мутације 0.01

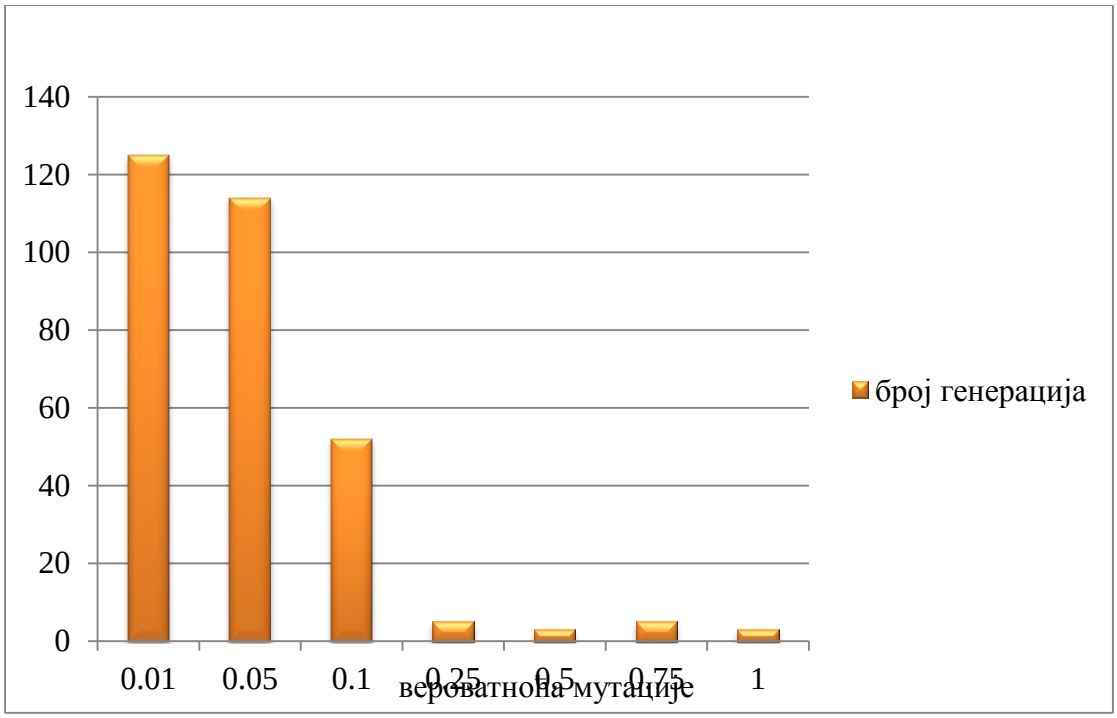


Слика 33. График добијеног решења када је вероватноћа мутације 0.1



Слика 34. График добијеног решења када је вероватноћа мутације 0.5

У примеру са три функције (4.2.1) одрађен је генетски алгоритам у овом случају за прву функцију за коју је тражен глобални максимум. У првом случају (слика 32) је коришћена униформна мутација са вероватноћом од 0.01, уз величину популације 300 и за број генерација 300. У другом случају (слика 33) је коришћен такође исти тип мутације али са вероватноћом од 0.1 и свим истим осталим параметрима као и у првом случају. Такође у трећем случају (слика 34) је у MATLAB коду постављена униформна мутација са степеном од 0.5 и величином популације 300, а и број генерација је исти као у претходним примерима 300. Са слика се може потврдити да претходна констатација важи, јер се у првом случају са малом вероватноћом мутације јављају локални максимуми и већа одступања од глобалног оптимума. Док се и у другом па још више приметно и у трећем случају са повећањем вероватноће мутације добијају много боља решења, из генерације у генерацију је готово добијен глобални максимум. Док графици изгледају у сва три примера хаотично и линија која показује кретање резултата кроз генерације изгледа приближно исто, треба обратити пажњу на бројеве на вертикалној скали где се види да се сужава простор решења глобалном максимуму а то је 4.7712.



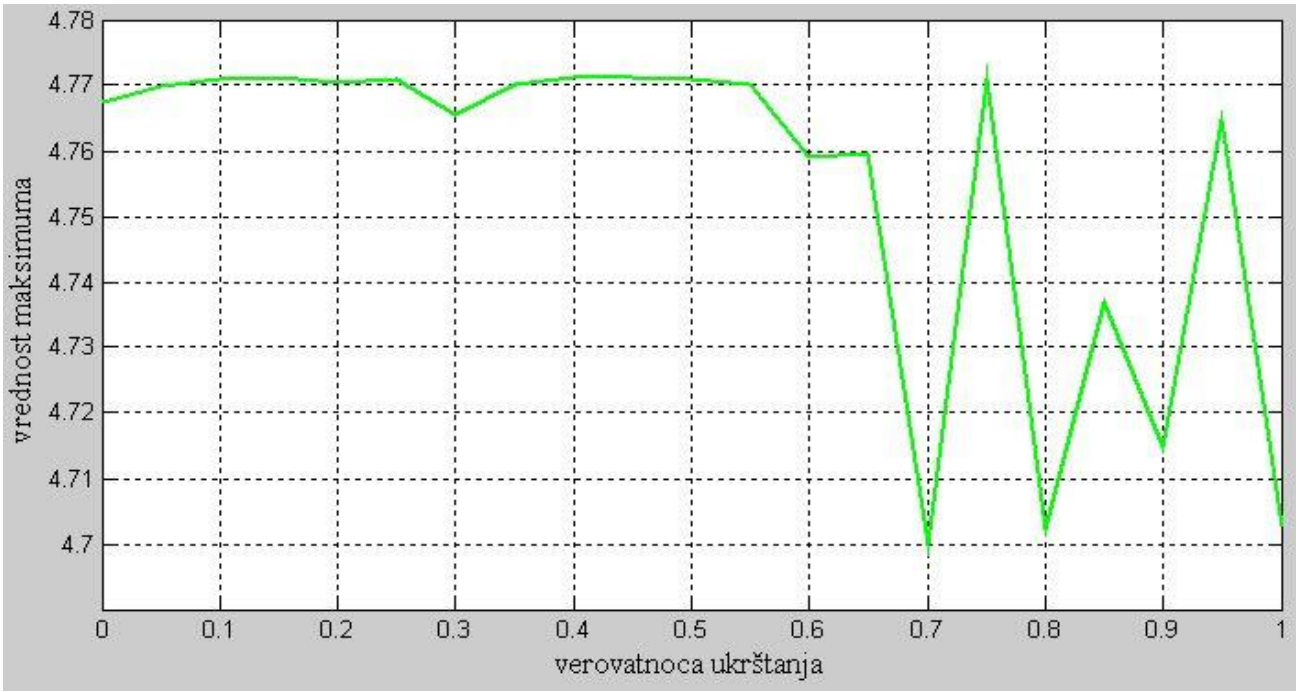
Слика 35. Утицај вероватноће мутације на проналазак решења

На слици 35 је дат просечан утицај вероватноће мутације на проналазак решења одрађен на неколико примера. Примећује се да мала вероватноћа мутације (мања од 0.2) утиче на повећање броја генерација како би се пронашло ваљано решење, понекад смањује и сам учинак проналаска решења. Најбоље резултате даје вероватноћа популације вредности 0.5, мада су готово идентични резултати и код вероватноће 0.25 до 1.

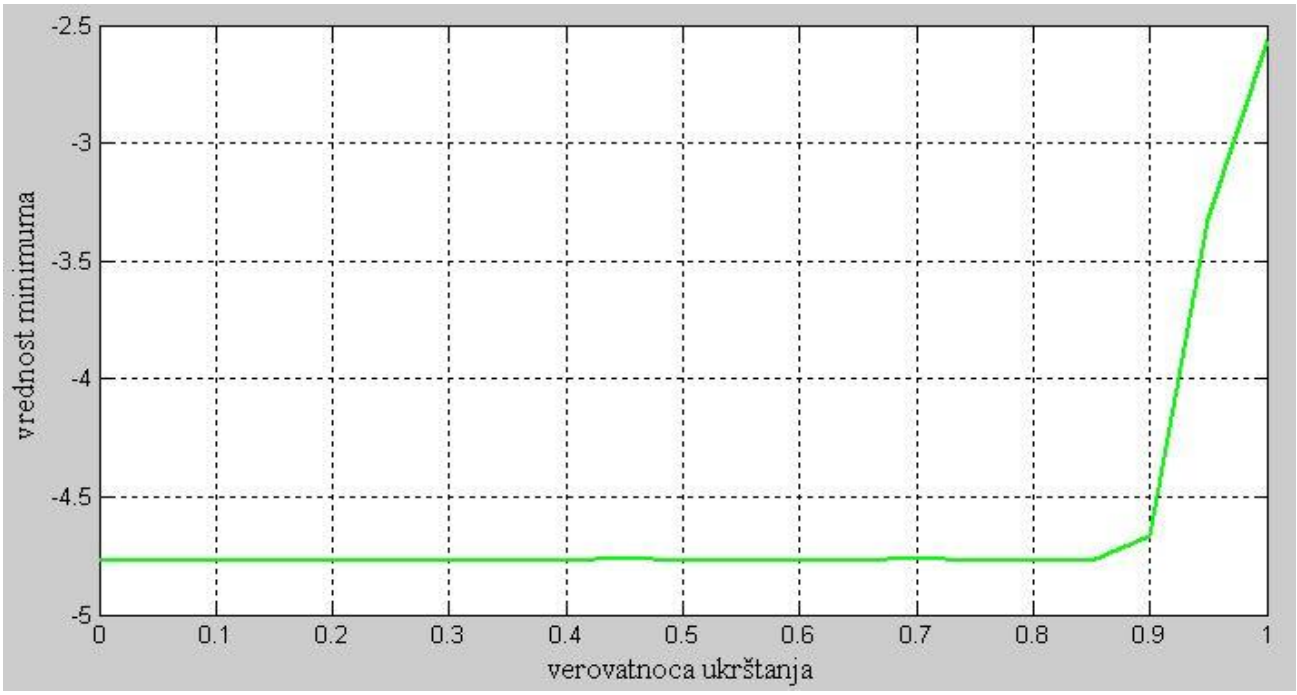
Због улоге коју има овај параметар, потребно је бити пажљив у раду са њим и одабрати правилне вредности. Генетски алгоритам јако реагује и на најмање промене овог параметра.

4.2.5 Утицај вероватноће укрштања

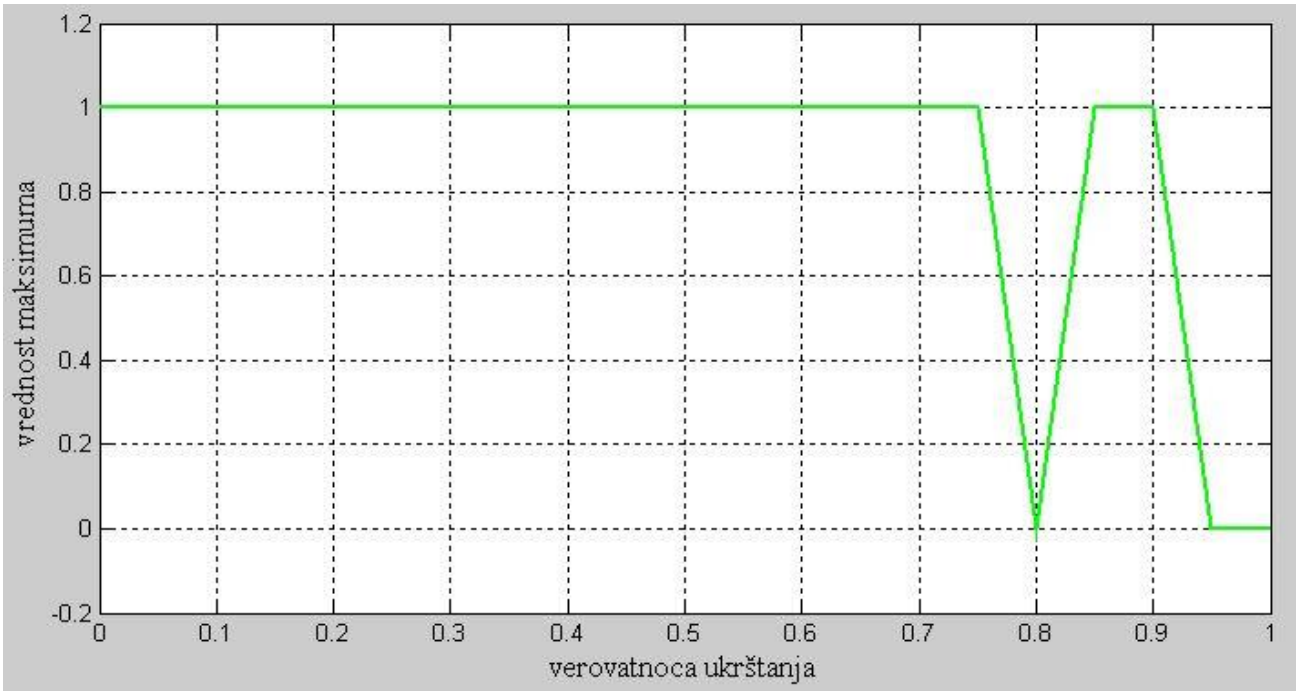
Вероватноћа укрштања је параметар који се необавезно појављује у генетском алгоритму. Треба напоменути битне ставке везане за вероватноћу укрштања која је у корелацији са вероватноћом мутације а током рада са генетским алгоритмом. Ако су решења распршена потребно је повећати вероватноћу укрштања а смањити мутацију, односно ако су решења униформна потребно је реаговати супротно: повећати вероватноћу мутације а смањити вероватноћу укрштања, јер укрштањем два иста решења неће се постићи ништа ново већ ће се добити трећи исти хромозом.



Слика 36. График добијеног решења за различите проценте укрштања за добијање максимума прве функције из примера 4.2.1



Слика 37. График добијеног решења за различите проценте укрштања, за добијање минимума прве функције из примера 4.2.1



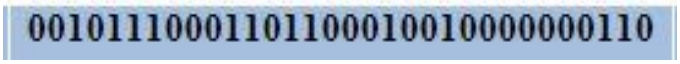
Слика 38. График добијеног решења за различите проценте укрштања, за добијање максимума треће функције из примера 4.2.1

Са приказаних слика (36, 37, 38) може се приметити да као и код осталих параметара и утицај вероватноће укрштања варира у зависности од примера до примера као и од циља задатка. А под циљем се подразумева да ли се у задатку тражи минимум или максимум функције (као што је то случај у датом примеру 4.2.1). На слици 36 и 38 је добијено решење за максимум прве односно треће функције, види се да је глобални максимум добијен од старта и у оба случаја је константан до, у првом случају око 65% вероватноће укрштања а у другом 75% вероватноће укрштања. До тог тренутка добијена решења су одлична, а онда се примећује варирање и већа одскакања од глобалног оптимума. Док на слици 32 на којој је приказан график добијеног решења за минимум прве функције, глобални минимум је добијен од самог почетка и решења су константна све до 90% вероватноће укрштања, одакле се види нагли губитак глобалног оптимума и све до краја се бележе све лошија решења. Дакле тешко је прецизирати општу констатацију за утицај вероватноће укрштања али се може рећи да је дејство вероватноће до приближно 70% добро на рад генетског алгоритма и да је у том случају добијен глобални оптимум.

4.2.6 Утицај дужине бинарног кода

Дужина бинарног кода је параметар који је скоро увек задат унапред и што је важно рећи не мења се током рада генетског алгоритма. У бинарном приказу овај параметар се приказује као димензија хромозома *x* броја битова. То је у одељку о примеру

рада генетског алгоритма била величина од 32, дакле димензија хромозома је била 32 бита, како је и на слици 39 лако пребројати.

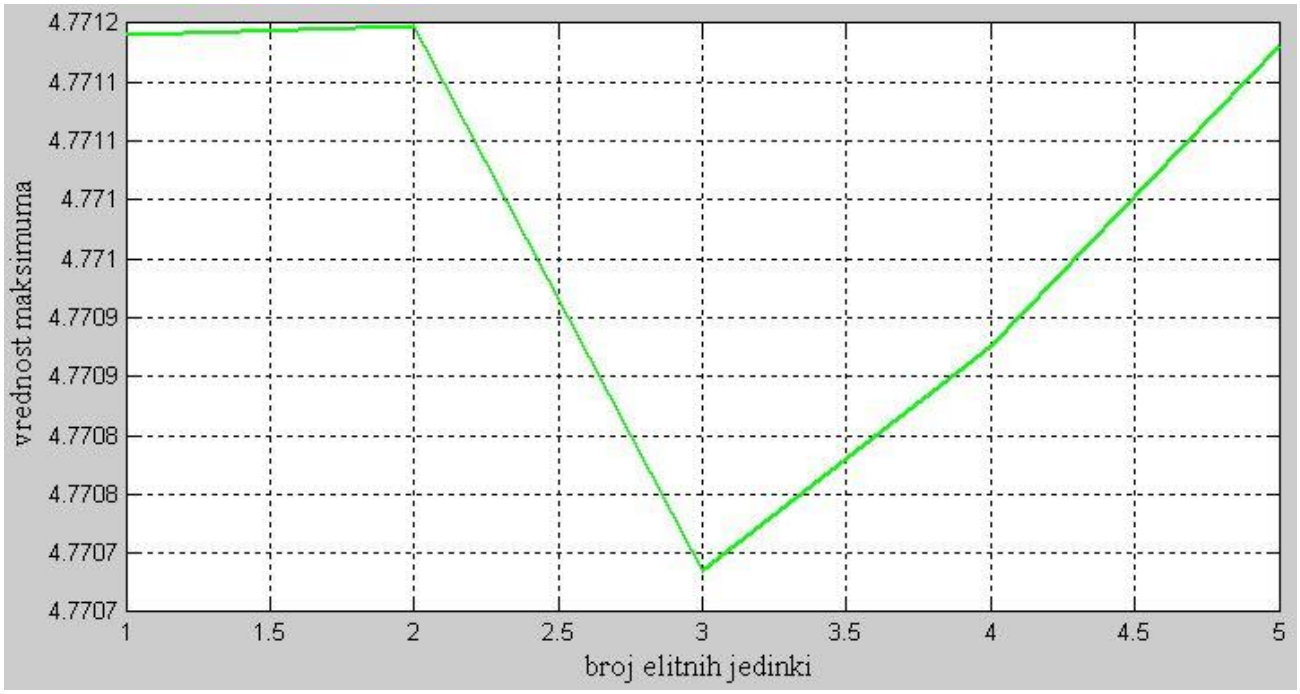


Слика 39. Приказ димензије хромозома од 32 бита

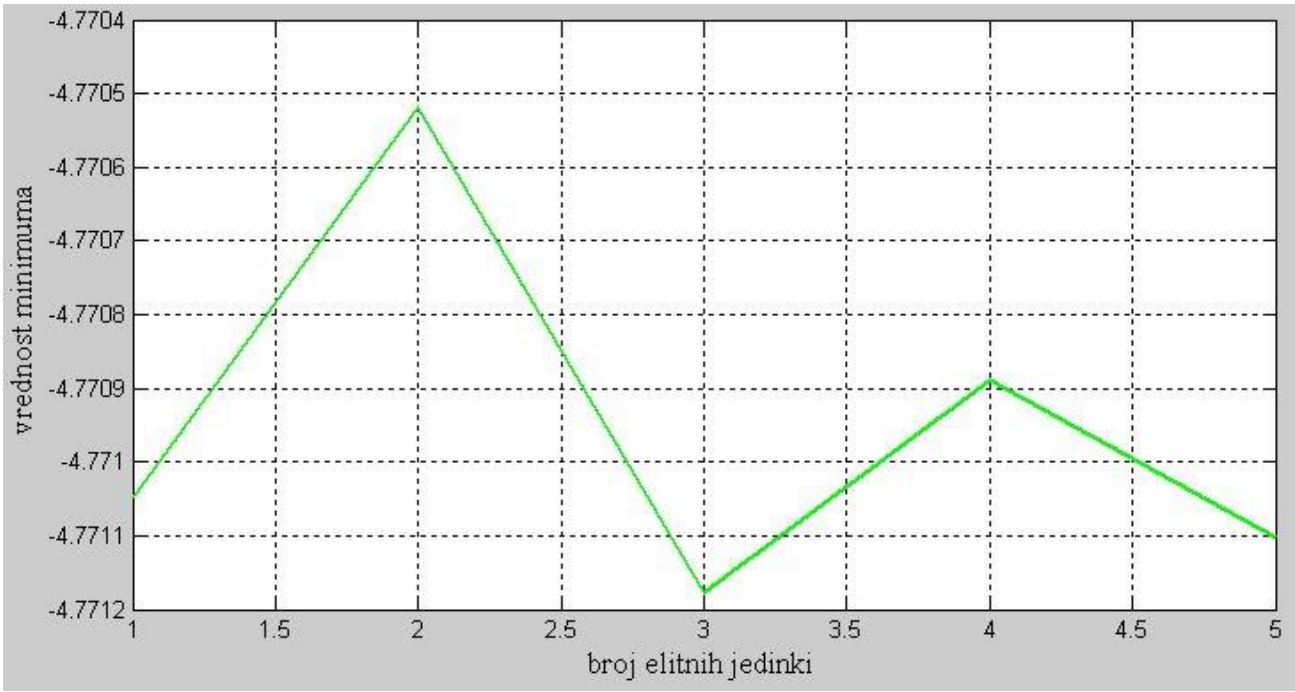
Како је овај параметар претходно дефинисан, тако је потребно успоставити баланс у подешавању код осталих параметара генетског алгоритма, да би се задовољавајућем времену постизала задовољавајућа решења.

4.2.7 Утицај елитних јединки

Већ је речено да је елитизам, механизам чувања најбољих јединки од њиховог губитка или измене од стране генетских оператора. Само увођење овог параметра у генетски алгоритам је јако важно и има велики утицај на његов рад, јер директно утиче на квалитет решења. Дефинитивно се овим параметром долази до прецизнијег решења јер је најбоља јединка у свакој генерацији заштићена.



Слика 40. График добијеног решења различитог броја елитних јединки, за добијање максимума прве функције из примера 4.2.1



Слика 41. График добијеног решења различитог броја елитних јединки, за добијање минимума прве функције из примера 4.2.1

Имплементација елитних јединки кроз генетски алгоритам је потребна, како би се кроз потрагу за што бољим решењем и дошло до њега. Међутим са слика 40 и 41 где је тражен максимум и минимум прве функције примера 4.2.1 се види да се за различит број елитних јединки добијају различита решења упоредивши оба случаја. Коришћена је величина популације 100 и број генерација 30. Може се рећи да је за највећи број елитних јединки у свакој генерацији, у овом примеру 5, добијено добро решење и у једном и у другом случају, а да за остале бројеве се показује недефинисаност. На пример за број елитних јединки 3, у првом случају је добијено лоше а у другом одлично решење. Ово само показује да ипак највећи утицај има примена елитизма а мање квантитет елитних јединки. Ипак када се постави ова заштита као параметар, утиче се на време извршавања генетског алгоритма јер је потребно доста више времена за његову финализацију.

4.3 Неке од препоручених поставки параметара

Већ је речено да је тешко истаћи фиксни избор параметара за било који случај проблема. Ефикасност рада генетског алгоритма зависи од поставки параметара и да је препоручљиво да се поставке изврше према сопственом прорачуну за свој конкретан проблем. Ипак неки научници су покушали да одгонетну избор параметара за већину случајева, препоручили су неке од поставки и да тако одабрани параметри дају добре резултате генетског алгоритма.

Дејонгова подешавања

Поставке које је предложио Дејонг (енг. *DeJong*)[37] су у пракси стандард за већину генетских алгоритама. Он је показао да је комбинација параметара коју је поставио, ради много боље од многих других комбинација параметара за функцију оптимизације.

- Величина популације: 50
- Број генерација: 1000
- Начин укрштања: класично двопозиционо укрштање
- Вероватноћа укрштања: 0.6
- Начин мутације: обртање битова мутација
- Вероватноћа мутације: 0.02 и 0.04

Код начина укрштања се поред двопозиционог може изабрати и једнопозиционо укрштање. Ако се и употреби код проблематичних метода на пример униформно укрштање, потребно је користити нижу стопу вероватноће (на пример 0.5). Вероватноћа мутације је дата по биту, ипак у јавним објавама у кодовима вероватноћа мутације је постављена по хромозому. И треба повести рачуна да степен мутације није превише низак.

Грефенстетова подешавања

Ово је генерално други напопуларнији сет подешавања параметара. Обично се користи када рачунарски трошак за проналажење објективне функције, вас тера да имате што мању популацију[38].

- Величина популације: 30
- Број генерација: није прецизирано
- Начин укрштања: класично двопозиционо укрштање
- Вероватноћа укрштања: 0.9
- Начин мутације: обртање битова мутација
- Вероватноћа мутације: 0.01

И у овом случају подешавања параметара је могуће користити уместо двопозиционог једнопозиционо укрштање.

Микро подешавања за генетске алгоритме

Ове поставке тренутно немају широку примену, али они који су их користили за оптинизацију функције известили су да ове поставке смањују и до четири пута број потребних евалуација како би се достигао тражени ниво перформанси.

- Величина популације: 5
- Број генерација: 100
- Начин укрштања: униформно укрштање
- Вероватноћа укрштања: 0.5
- Начин мутације: гмизајућа мутација
- Вероватноћа популације: 0.02 и 0.04

Давидов код се непрекидно рестартује са случајним хромозомима када се открије конвергенција, тако да постоји велика могућност да ове препоруке параметара неће добро радити за друге кодове.

5. ЗАКЉУЧАК

У овом раду детаљно је описан генетски алгоритам, његови оператори и сваки параметар понаособ. Посебно је посвећена пажња стандардним параметрима и њиховом утицају на перформансе генетског алгоритма. Показана је велика зависност рада генетског алгоритма од избора параметара, односно њихових вредности.

Генетски алгоритми су веома добар и квалитетан алат за решавање многих проблема, ипак потребно је добро овладати њима како би се дошло до прихватљивих и задовољавајућих решења. Генетски алгоритми имају и својих предности:

- примењиви су на велики број проблема
- решавају све проблеме који се могу представити оптимизацијски
- једноставним понављањем поступка се може повећати поузданост резултата
- као резултат даје скуп решења а не једно решење и многе друге

али имају и својих мана:

- не може се постићи 100 % поузданост решења
- тешко се дефинише добра функција доброте
- често је потребно прилагођавање проблема алгоритму итд.

Једна од ставки која има велики утицај на ефикасност генетских алгоритама су параметри, које је тешко поставити и то представља веома сложен задатак. Ипак то је могуће постићи и у овом раду је дат поступак рада са параметрима, и како се њиховим избором може утицати на перформансе алгоритма. Као што су и примери показали, несумњиво је да постоји велика зависност између самих параметара. Промена једног параметра у већини случајева повлачи промену осталих како би се одржала иста квалитетна решења. Такође као битан сегмент који може скратити време проналаска решења је и да се одреди тачност решења. Унапред одређена тачност решења знатно скраћује време тражења скупа параметара који дају тражено решење.

Одрађен је пример у MATLAB програму са одабране три функције и могућношћу одабира параметара који се посматрају у току рада програма и на крају показан графички приказ добијених решења радом генетског алгоритма. Јасно су дефинисани параметри и њихове вредности у MATLAB-у како би се на што бољи начин омогућио приступ раду са њима у алгоритму. Добијена решења су показала како је речено да постоји велика зависност између свих параметара, али ипак може се закључити да је однос између неких

параметара нешто јачи. Величина популације показује велику зависност од броја генерација и обратно. Смањен број величине популације обично захтева повећавање броја генерација како би се дошло до што бољег и прецизнијег решења. Такође је приметна јача повезаност вероватноће укрштања и мутације, где се ово огледа у распшености решења и заглављивању генетског алгоритма у неком од локалних екстремума. Сви параметри јако утичу на време извођења алгоритма, број елитних јединки ако се примени као параметар знатно повећава рад генетског алгоритма.

Потребно је на крају нагласити да је фиксни избор параметара могућ за различите задатке и проблеме, али да у таквој ситуацији у већини случајева нису добијена ваљана решења. Сваки проблем тражи поставку и избор параметара који је погодан за њега. Потребно је сагледати све аспекте (време проналаска решења, ваљаност решења) и према њима се направио најбољи избор за одабир параметара. На крају су дате препоручене поставке параметара од стране неких научника које су показале значајне резултате, ипак ни оне не гарантују да се њиховом имплементацијом у сваком проблему може доћи до оптималног решења.

6. ЛИТЕРАТУРА

[1] Ingo Rechenberg, *"Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution (PhD thesis) "*, (1971).

[2] Holland J.H., *Adaptation in Natural and Artifcial Systems*, University of Michigan Press, Ann Arbor, (1975).

[3] Drezner Z., Hamacher H., *"Facility Theory: Applications and Theory"*, Springer-Verlag, Berlin-Heidelberg, (2002).

[4] Klose A., Drexl A., *"Facility location models for distribution system design"*, European Journal of Operational Research, Vol. 162, pp. 4-29, (2005).

[5] Vygen J., *"Approximation Algorithms for Facility Location Problems"* (Lecture Notes), Research Institute for Discrete Mathematics, University of Bonn, Report No. 05950-OR, (2005).

[6] Billionnet A., *"Integer programming to schedule a hierarchical workforce with variable demands"*, European Journal of Operational Research 114, pp. 105–114, (1999).

[7] Emmons H., "*Workforce scheduling with cyclic requirements and constraints on days off, and workstretch"*, IIE Transactions 17 pp. 8–16, (1985).

[8] Hung R., *"Using compressed workweeks to reduce work commuting"*, Transportation Research 30, pp. 1–19, (1996).

[9] Narasinhan R., *"Algorithm for single shift scheduling of hierarchical workforce"*, European Journal of Operational Research 96 (1), pp. 113–121, (1997).

[10] Topaloglu S., Ozkarahan I., *"Implicit optimal tour scheduling with flexible break assignments"*, Computers & Industrial Engineering 44, pp. 75–89, (2002).

[11] Darvin Č., *"Postanak vrsta"*, Nolit, Beograd, (1985).

[12] Watson, J. D., & Crick, F. H. C. *"A structure for deoxyribose nucleic acid. "* Nature 171, (1953).

[13] Bäck T., Fogel D. B., Michalewicz Z., *"Basic Algorithms and Operators"*, in: Evolutionary Computation 1, Institute of Physics Publishing, Bristol-Philadelphia, (2000).

[14] Beasley D., Bull D.R., Martin R.R., *"An Overview of Genetic Algorithms, Part 1, Fundamentals"*, University Computing, Vol. 15, No. 2, pp.58-69, (1993) ftp://ralph.cs.cf.ac.uk/pub/papers/GAs/ga_overview1.ps

[15] Goldberg D.E., *"Genetic Algorithms in Search, Optimization and Machine Learning"*, Addison-Wesley Publ. Comp., Reading, Mass., (1989).

[16] Michalewicz Z., *"Genetic Algorithms + Data Structures = Evolution Programs"*, Third Edition, Springer Verlag, Berlin Heideleberg, (1996).

[17] Mitchell M., *"An Introduction to Genetic Algorithms"*, MIT Press, Cambridge, Massachusetts, (1999).

[18] Mühlenbein H., *"Genetic Algorithms"*, Local Search in Combinatorial Optimization, eds. Aarts E.H.L., Lenstra J.K., John Wiley & Sons Ltd., pp. 137-172, (1997).

[19] Čangalović M., *"Opšte heuristike za rešavanje problema kombinatorne optimizacije"*, u: Kombinatorna optimizacija: Matematicka teorija i algoritmi, str. 320-350, (1996).

[20] Filipović V., *"Predlog poboljšanja operatora turnirske selekcije kod genetskih algoritama"*, Magistarski rad, Univerzitet u Beogradu, Matematički fakultet, (1998).

[21] Kratica J., *"Paralelizacija genetskih algoritama za rešavanje nekih Npkompletnih problema"*, Doktorska disertacija, Matematički fakultet, Beograd, (2000).

[22] Kratica J., Tošić D., *"Introduction to Genetic Algorithms and Some Applications"*, Proceedings of a Workshop on Computational Intelligence - Theory and Applications, pp. 57-68, Niš, Yugoslavia, February 27, (2001).

[23] Tošić D., Mladenović N., Kratica J., Filpović V., *"Genetski algoritmi"*, Matematički institut SANU, Beograd, (2004).

[24] Marin Golub, *"Genetski algoritmi"* , Zagreb, Oktobar (1997) http://www.zemris.fer.hr/~golub/ga/ga_skripta1.pdf

[25] Srinivas M., Patnaik L.M., *"Genetic Algorithms: A Survey"*, IEEE Computer, pp. 17-26, (1994).

[26] http://www.mathworks.com/help/gads/performing-a-genetic-algorithm-optimization.html

[27] http://www.mathworks.com/help/gads/performing-a-genetic-algorithm-optimization.html

[28] Beck T., *"Optimal Mutation Rates in Genetic Search"*, in: Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, Calif., pp. 2-8, (1993).

[29] Syswerda G., *"Uniform Crossover in Genetic Algorithms"*, In: Proceedings of the Third International Conference on Genetic Algorithms - ICGA '89, Morgan Kaufmann, San Mateo, Calif., pp. 2-9, (1989).

[30] Ackley D.H., *"An empirical study of bit vector function optimization"*, In: Genetic algorithms and Simulated Annealing, Davis L. (ed), chapter 13, pp. 170-204, (1987).

[31] Fogarty T.C., *"Varying the probability of mutation in the genetic algorithms"*, In: Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, Calif., pp. 61-69, (1989).

[32] Davis L., *"Adapting operator probabilities in genetic algorithms"*, In: Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, Calif., pp. 61-69, (1989).

[33] Davis L., *"Handbook of Genetic Algorithms"*, Van Nostrand Reinhold, New York, (1991).

[34] Bëck T., *"Self-adaptation in Genetic Algorithms"*, In: Proceedings of the First European Conference on Artificial Life, MIT Press, (1992).

[35] Beasley D., Bull D.R., Martin R.R., "*An Overview of Genetic Algorithms, Part 2, Research Topics"*, University Computing, Vol. 15, No. 4, pp. 170-181, (1993).

[36] Rene Huić, *"Rešavanje problema N kraljica uz pomoć genetskog algoritma"*, Zagreb, jun (2008).

[37] DeJong, K.A. and Spears, W.M. *"An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms"*, Proc. First Workshop Parallel Problem Solving from Nature, Springer-Verlag, Berlin, pp. 38-47, (1990).

[38] Grefenstette, J.J. *"Optimization of Control Parameters for Genetic Algorithms"*, IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-16, No. 1, Jan./Feb., pp. 122-128., (1986).