SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

SRMINARSKI RAD

Viktor Mirić, Karlo Mišković i Tomislav Abičić

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Primjena neparametarskih testova: Kruskal - Wallis H-test, Friedman F - test, Spearman-ov koeficijent korelacije rangova u problemima iz područja proizvodnih (uslužnih) procesa

Mentori: Student:

Viktor Mirić

Karlo Mišković

Dr. sc. Hrvoje Cajner, dipl. ing. Tomislav Abičić

Zagreb, 2024.

A computer screen shot of a computer

Description automatically generated

Zadatak za seminarski rad iz kolegija   
**„Multivarijatne statističke metode“**

(2023./24.)

Studenti: Viktor Mirić, Karlo Mišković, Tomislav Abičić

Tema seminarskog rada: **Primjena neparametarskih testova: Kruskal - Wallis H-test, Friedman F - test, Spearman-ov koeficijent korelacije rangova u problemima iz područja proizvodnih (uslužnih) procesa**

Seminarski rad mora sadržavati uvod, teorijsku razradu, primjer iz područja industrijskog inženjerstva koji može biti realan ili za potrebe seminara simuliran. Primjer mora biti proveden u jednom od dostupnih programskih paketa po vlastitom odabiru (Matlab, Octave, Excel i sl.).

Za potrebe rješavanja seminarskog zadatka koristiti relevantnu literaturu koja je ujedno i predložena na uvodnom nastavnom satu. Opisi grafičkih prikaza moraju biti prevedeni na hrvatski jezik. Po završetku izrade seminarski rad dostaviti u elektronskom obliku.

U Zagrebu, 29.12.2023.

Zadao:

Izv. prof. dr. sc. Hrvoje Cajner

**SADRŽAJ**

[POPIS SLIKA II](#_Toc154184755)

[POPIS TABLICA III](#_Toc154184756)

[POPIS KRATICA IV](#_Toc154184757)

[1. UVOD 1](#_Toc154184758)

[2. OSNOVE GENETSKIH ALGORITAMA 2](#_Toc154184759)

[2.1. Genetski operatori 4](#_Toc154184760)

[2.1.1. Kodirane sheme 4](#_Toc154184761)

[2.1.2. Selekcija 5](#_Toc154184762)

[2.1.3. Križanje 6](#_Toc154184763)

[2.1.4. Mutacija 7](#_Toc154184764)

[3. SHEMATSKA METODOLOGIJA 9](#_Toc154184765)

[3.1. Kod genetskog algoritma 9](#_Toc154184766)

[3.2. Pseudokod 12](#_Toc154184767)

[3.3. Matematički Model 14](#_Toc154184768)

[3.4. Blok dijagram 16](#_Toc154184769)

[4. PREDNOSTI I NEDOSTACI 17](#_Toc154184770)

[5. PRIMJENE GENETSKIH ALGORITAMA U INDUSTRIJSKOM INŽENJERSTVU 20](#_Toc154184771)

[6. SOFTVERSKA RJEŠENJA 23](#_Toc154184772)

[6.1. MATLAB 23](#_Toc154184773)

[6.2. Python 23](#_Toc154184774)

[6.2.1. Pyvolution 23](#_Toc154184775)

[6.2.2. deap 24](#_Toc154184776)

[6.3. Java 24](#_Toc154184777)

[6.3.1. Jenetics 24](#_Toc154184778)

[6.3.2. EpochX 24](#_Toc154184779)

[6.4. C++ 25](#_Toc154184780)

[6.4.1. openGA 25](#_Toc154184781)

[6.4.2. BEAGLE 25](#_Toc154184782)

[6.5. Darwin 25](#_Toc154184783)

[7. ZAKLJUČAK 27](#_Toc154184784)

[LITERATURA 28](#_Toc154184785)

# POPIS SLIKA

[Slika 1. Proces genetskog algoritma [3] 2](#_Toc153828445)

[Slika 2. Genetski operatori [1] 9](#_Toc153828446)

[Slika 3. Generiranje genetskog algoritma "Ja volim operacijska istraživanja <3" 13](#_Toc153828447)

# POPIS TABLICA

[Tablica 1. Prednosti i nedostaci [13][12][11][1] 19](#_Toc153828448)

# POPIS KRATICA

|  |  |
| --- | --- |
| Oznaka | Opis |
| GA | Genetski algoritam |
| SUS | Stohastičko univerzalno uzorkovanje |
| PMX | Djelomično podudarajuće križanje |
| OX | Redoslijedno križanje |
| PPX | Očuvanje prioriteta križanja |
| RCX | Reducirano zamjensko križanje |
| DM | Mutacija premještanja |
| SIM | Operator jednostavne inverzije mutacije |
| SM | Mutacija nasumičnim poretkom |
| TS | Tabu pretraga |
| PSO | Optimizacija roja čestica |
| ACO | Optimizacija kolonije mrava |
|  |  |
|  |  |

# UVOD

Genetski algoritmi, složeni optimizacijski alati, predstavljaju inovativni pristup rješavanju problema inspiriran principima evolucije. Ovaj uvod pružit će sažeti uvid u svijet genetskih algoritama, definirajući ih kao specifičnu vrstu općeg heurističkog pristupa. Njihova važnost i relevantnost u kontekstu rješavanja problema postaju očigledni, stvarajući most između koncepta evolucijskih procesa i suvremenih tehnika optimizacije.

Genetski algoritmi, kako ih nazivamo, simuliraju prirodne mehanizme selekcije, križanja i mutacije kako bi pronašli najbolje rješenje u različitim problemima optimizacije. Ova metoda, inspirirana evolucijskim procesima u prirodi, pokazala se izuzetno moćnom u suočavanju s problemima koji uključuju veliki broj varijabli i složene prostore pretraživanja. Kroz iterativno poboljšavanje populacije rješenja, genetski algoritmi nude elegantan pristup rješavanju problema na način koji simulira procese evolucije.

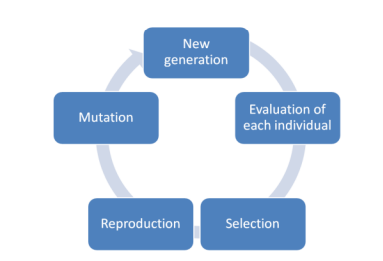
Važnost genetskih algoritama manifestira se u njihovoj sposobnosti prilagodbe raznolikim problemima, bilo da se radi o optimizaciji rasporeda, projektiranju mreža, ili pronalaženju najučinkovitijeg puta u prostoru pretraživanja. Njihova široka primjenjivost čini ih nezaobilaznim alatom u području operacijskog istraživanja, gdje su kompleksni problemi često sastavljeni od brojnih varijabli koje međusobno ovise.

Kroz ovaj uvod, otkrit ćemo ključne elemente genetskih algoritama koji ih čine tako učinkovitima. Daljnje istraživanje ove teme pružit će nam priliku dublje zaroniti u shematsku metodologiju, prednosti i nedostatke, stvarne primjene u domeni industrijskog inženjerstva, te prateća softverska rješenja koja podržavaju implementaciju ove inovativne metode rješavanja problema.

# OSNOVE GENETSKIH ALGORITAMA

Genetski algoritam (GA) predstavlja računalni model koji simulira prirodnu selekciju biološke evolucije prema Darwinovom konceptu, uz primjenu genetskog mehanizma. Ovaj pristup uključuje pretraživanje optimalnog rješenja putem simulacije prirodnog evolucijskog procesa. GA se temelji na pronalaženju potencijalnih rješenja unutar populacije. Nakon generiranja originalne populacije, svaka sljedeća generacija evoluira prema boljem približnom rješenju uzimajući u obzir razinu prilagodbe i veličinu domene problema za odabir jedinki u svakoj generaciji. Procesi križanja i mutacije potom se odvijaju s genetskim operatorima, rezultirajući novom populacijom koja predstavlja novi skup rješenja. Ovaj proces rezultira epigenetskom populacijom s boljom izvedbom u pogledu prilagodljivosti okolišu. Također, dekodiranjem optimalnog pojedinca iz prethodne generacije moguće je dobiti približno optimalno rješenje problema.

Različiti pristupi GA-u, kao što su hibridni genetski algoritmi, uključuju kombinaciju s drugim algoritmima za poboljšanje učinkovitosti i rješavanje određenih nedostataka. Sve više istraživanja uključuje uvođenje informacijske entropije i teorije igara u GA kako bi se dodatno optimizirali ovi algoritmi. Iako su primjene entropije, teorije igara i GA za optimizaciju nedostatne u Kini, istraživanje o njihovoj kombinaciji pokazuje obećavajuće smjerove za daljnje unapređenje ovih algoritama [3].



Slika 1. Proces genetskog algoritma [3]

Ključni koncepti genetskih algoritama [1]:

1. **Kromosomi i Geni**

Kromosomi u genetskom algoritmu predstavljaju nositelje genetske informacije, odnosno potencijalna rješenja problema. Svaki kromosom sastoji se od gena, koji su varijable ili parametri definirani unutar problema. Analogija s biološkim kromosomima i genima čini temeljnu jedinicu informacije unutar algoritma.

1. **Populacija**

Populacija je skup kromosoma koji zajedno čine trenutnu generaciju potencijalnih rješenja. Kroz iteracije genetskog algoritma, populacija se dinamički mijenja kako bi evoluirala prema boljim rješenjima tijekom vremena. Ovaj evolucijski pristup omogućuje algoritmu prilagodbu i optimizaciju.

1. **Selekcija**

Selekcija je ključna faza u kojoj se odabiru kromosomi iz trenutne populacije na temelju njihove prilagodbe. Kromosomi s višom prilagodbom imaju veću vjerojatnost sudjelovanja u reprodukcijskim procesima, simulirajući princip preživljavanja najsposobnijih u prirodi.

1. **Križanje (Rekombinacija)**

Proces križanja oponaša biološki mehanizam genetskog rekombiniranja. Dva roditeljska kromosoma kombiniraju svoje genetske informacije putem križanja, stvarajući potomka s kombinacijom karakteristika oba roditelja. Ovaj inovativni pristup omogućuje stvaranje novih rješenja koja nasljeđuju pozitivne aspekte prethodnih generacija.

1. **Mutacija**

Mutacija predstavlja mehanizam koji unosi slučajne promjene u genetsku informaciju jednog ili više gena u kromosomu. Ova varijabilnost doprinosi raznolikosti unutar populacije, stvarajući prostor za istraživanje novih, potencijalno boljih rješenja. Mutacija osigurava dinamiku koja održava adaptabilnost populacije kroz evolucijski proces.

1. **Funkcija Prilagodbe**

Funkcija prilagodbe ključna je za vrednovanje kvalitete svakog kromosoma unutar populacije. Ova evaluacija usmjerena je prema ciljevima optimizacije, a kromosomi s visokom prilagodbom preferirani su u procesima selekcije i reprodukcije. Stvaranje i očuvanje visoko prilagođenih rješenja ključno je za uspješnost genetskog algoritma.

## Genetski operatori

GA koristi različite operatore tijekom procesa pretraživanja. Ti operatori uključuju kodiranje, križanje, mutaciju i selekciju. Slika 2 prikazuje operatore koji se koriste u GA [1].

### Kodirane sheme

Kodirane sheme predstavljaju ključan aspekt rješavanja većine računalnih problema, igrajući važnu ulogu u pretvaranju informacija u određeni format. Informacije se obično kodiraju u niz bitova, a vrsta kodirane sheme razlikuje se prema domeni problema. Poznate kodirane sheme uključuju:

1. **Binarna kodna shema**: Najčešće korištena kodna shema gdje se svaki gen ili kromosom predstavlja nizom 0 ili 1. Brza je za implementaciju operatora križanja i mutacije, ali zahtijeva dodatni trud pri konverziji u binarni oblik, a točnost algoritma ovisi o binarnoj konverziji. Nije uvijek prikladna za inženjerske probleme dizajna zbog epistaze i prirodne reprezentacije.
2. **Oktalna i heksadecimalna kodna shema**: Gen ili kromosom predstavljaju se oktalnim (0–7) ili heksadecimalnim brojevima (0–9, A-F).
3. **Permutacijska kodna shema**: Često se koristi za probleme sortiranja, pri čemu se gen ili kromosom predstavljaju nizom brojeva koji predstavljaju poziciju u sekvenci.
4. **Vrijednosna kodna shema**: Gen ili kromosom predstavljaju se nizom vrijednosti koje mogu biti realni brojevi, cijeli brojevi ili znakovi. Korisna je za rješavanje problema s kompliciranim vrijednostima, gdje binarna kodna shema može zakazati. Često se primjenjuje u neuronskim mrežama za pronalaženje optimalnih težina.
5. **Kodna shema stabla**: Gen ili kromosom predstavljaju se stablom funkcija ili naredbi, slično reprezentaciji izraza u formatu stabla. Često se koristi u evoluciji programa ili izraza [1].

### Selekcija

Selekcija predstavlja važan korak u genetskim algoritmima koji određuje hoće li određeni niz sudjelovati u procesu reprodukcije ili ne. Ponekad se korak selekcije naziva i operatorom reprodukcije. Brzina konvergencije genetskog algoritma ovisi o selekcijskom pritisku. Poznate tehnike selekcije uključuju rulet, rang, turnir, Boltzmann i stohastičko univerzalno uzorkovanje.

1. **Rulet selekcija**: Svi mogući nizovi mapiraju se na kotač s dijelom kotača dodijeljenim prema njihovoj vrijednosti prilagodbe. Kotač se zatim nasumično okreće kako bi odabrao određena rješenja koja će sudjelovati u formiranju sljedeće generacije. Međutim, pati od problema uvođenja pogrešaka zbog svoje stohastičke prirode. De Jong i Brindle modificirali su metodu odabira ruleta kako bi uklonili pogreške, uvodeći koncept determinizma u postupak odabira.
2. **Rang selekcija**: Modificirana verzija rulet selekcije koja koristi rangove umjesto vrijednosti prilagodbe. Rangovi se dodjeljuju prema vrijednosti prilagodbe kako bi svaki pojedinac imao šansu biti odabran prema svojem rangu. Ova metoda smanjuje šanse za prijevremeno konvergiranje rješenja lokalnom minimumu.
3. **Turnir selekcija**: Tehnika turnirskog odabira prvi je puta predložena od strane Brindlea 1983. godine. Osobe se biraju prema svojoj vrijednosti prilagodbe u parovima. Nakon odabira, osobe s većom vrijednošću prilagodbe dodaju se skupu sljedeće generacije.
4. **Stohastičko univerzalno uzorkovanje (SUS)**: Proširenje postojeće metode odabira ruleta koje koristi slučajno početno mjesto na popisu pojedinaca iz generacije. Novi pojedinac odabire se na ravnom razmaku . Pruža jednaku šansu svim pojedincima da sudjeluju u križanju sljedeće generacije.
5. **Boltzmann selekcija**: Temelji se na entropiji i metodama uzorkovanja korištenima u Monte Carlo simulaciji. Pomaže u rješavanju problema prijevremenog konvergiranja ], s visokom vjerojatnošću odabira najboljeg niza, ali uz mogućnost gubitka informacija, što se može upravljati kroz elitizam .
6. **Elitizam selekcija**: Predložena za poboljšanje izvedbe rulet selekcije, osigurava da elitni pojedinac u generaciji uvijek prelazi u sljedeću generaciju. Ako pojedinac s najvišom vrijednošću prilagodbe nije prisutan u sljedećoj generaciji nakon normalnog postupka odabira, elitni pojedinac automatski se uključuje u sljedeću generaciju [1].

### Križanje

Križanje (crossover) operatori koriste se za generiranje potomstva kombiniranjem genetskih informacija dva ili više roditelja. Poznati križanje operatori uključuju križanje jedne točke, križanje dvije točke, k-point, uniformno, djelomično podudarajuće, redoslijedno, očuvanje prioriteta križanja, shuffle, reducirano zamjensko i cikličko križanje.

1. **Križanje jedne točke**: Odabire se nasumična točka križanja, nakon čega se genetske informacije roditelja koje su iza te točke zamjenjuju. Novo potomstvo dobiva se zamjenom informacija o repu oba roditelja.
2. **Križanje dvije točke i K-point**: Odabiru se dvije ili više nasumičnih točaka križanja, te se genetske informacije roditelja zamjenjuju prema segmentima koji su stvoreni. Srednji segment roditelja zamijenjen je radi generiranja novog potomstva.
3. **Uniformno križanje**: Roditelj se ne može razložiti na segmente; svaki gen roditelja tretira se zasebno. Nasumično se odlučuje treba li se gen zamijeniti s istom lokacijom na drugom kromosomu.
4. **Djelomično podudarajuće križanje (PMX)**: Često korišteni operator koji izvrsno funkcionira. Dva roditelja odabrana su za parenje, pri čemu jedan roditelj donira dio genetskog materijala, a odgovarajući dio drugog roditelja sudjeluje u stvaranju djeteta. Preostali dijelovi kopiraju se iz drugog roditelja.
5. **Redoslijedno križanje (OX)**: Kopira jedan (ili više) dijelova roditelja na potomka iz odabranih točaka rezanja i popunjava preostali prostor vrijednostima koje nisu uključene u kopirani dio. Koristan je za probleme poretka, ali manje učinkovit za Probleme trgovačkog putnika.
6. **Očuvanje prioriteta križanja (PPX)**: Očuva poredak individualnih rješenja prisutnih u roditelju potomka prije primjene križanja. Inicijalizira se potomstvo nizom slučajnih 1 i 0 koji odlučuje hoće li se pojedinci iz oba roditelja odabrati ili ne. Modificirana verzija PPX-a predložena je za višekriterijske problemske rasporede.
7. **Shuffle križanje**: Predloženo je kako bi se smanjila pristranost uvedena drugim tehnikama križanja. Miješa vrijednosti jednog pojedinačnog rješenja prije križanja i vraća ih u prethodno stanje nakon provedenog križanja, čime se osigurava da točka križanja ne uvodi pristranost u križanje.
8. **Reducirano zamjensko križanje (RCX)**: Smanjuje nepotrebna križanja ako roditelji imaju isti genski slijed za reprezentacije rješenja. Temelji se na pretpostavci da GA proizvodi bolje pojedince ako su roditelji dovoljno raznoliki u svojem genetskom sastavu.
9. **Cikličko križanje**: Pokušava generirati potomstvo koristeći roditelje gdje svaki element zauzima poziciju prema poziciji njihovih roditelja. U prvoj iteraciji uzima neke elemente od prvog roditelja, a u drugoj iteraciji preostale elemente od drugog roditelja.

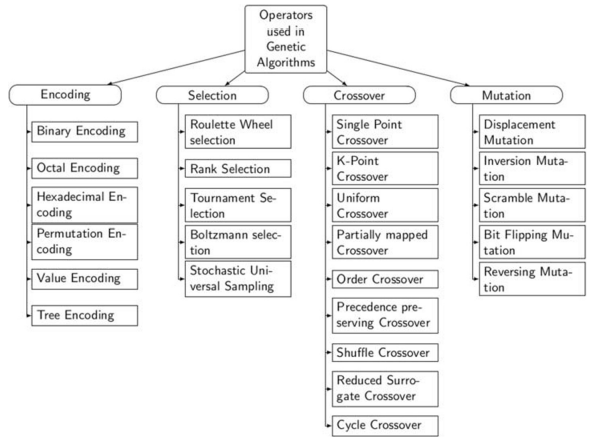
Križanje jedne točke i k-point križanje su jednostavni za implementaciju. Uniformno križanje pogodno je za velike podskupove. Redoslijedno i cikličko križanje pružaju bolje istraživanje od drugih tehnika križanja, dok djelomično podudarajuće križanje nudi bolje istraživanje od ostalih tehnika križanja. Međutim, reducirano zamjensko i cikličko križanje pate od prijevremenog konvergiranja [1].

### Mutacija

Mutacija je operator koji održava genetsku raznolikost od jedne populacije do sljedeće. Poznati operatori mutacije uključuju premještanje, jednostavnu inverziju i mutaciju nasumičnim poretkom.

1. **Mutacija premještanja (DM)**: Operator premještanja mutira podniz pojedinog rješenja unutar samog rješenja. Mjesto premještanja odabire se nasumično unutar zadanog podniza kako bi rezultirajuće rješenje bilo valjano, kao i nasumična promjena mjesta. Postoje varijante DM-a kao što su zamjena i umetanje mutacije, gdje se dio pojedinačnog rješenja zamjenjuje s drugim dijelom ili umetne na drugo mjesto, redom.
2. **Operator jednostavne inverzije mutacije (SIM)**: Ovaj operator invertira podniz između bilo koja dva odabrana mjesta u pojedinačnom rješenju. SIM je inverzni operator koji nasumično odabire niz, invertira ga i smješta na nasumično odabrano mjesto.
3. **Mutacija nasumičnim poretkom (SM)**: Operator SM postavlja elemente u određenom rasponu pojedinačnog rješenja u nasumičnom redoslijedu i provjerava poboljšava li se vrijednost dobivenog rješenja.

Uočeno je da se križanje jedne točke i uniformno križanje može koristiti s većinom operatora kodiranja i mutacije. Djelomično podudarajuće križanje koristi se s inverzijom mutacije, dok permuacija kodiranje pruža optimalno rješenje [1].



Slika 2. Genetski operatori [1]

# SHEMATSKA METODOLOGIJA

U ovom dijelu ovog seminarskog rada prikazati će se korišteni genetski algoritam.

## Kod genetskog algoritma

Genetski algoritam (GA) predstavljen u Python kodu ima za cilj generiranje ciljnog stringa "Ja volim Operacijska istraživanja II <3". GA koristi populaciju jedinki, gdje svaka jedinka predstavlja potencijalno rješenje problema. Evo opisa koda koristeći GA definiciju:

1. Inicijalizacija Populacije:

* Populacija se inicijalizira s 1000 jedinki, svaka s nasumičnim genima koji pripadaju definiranom skupu znakova.
* Ciljna sekvenca "Ja volim Operacijska istraživanja II <3" postavljena je kao referentni cilj za usporedbu.

1. Funkcija Prilagodbe:

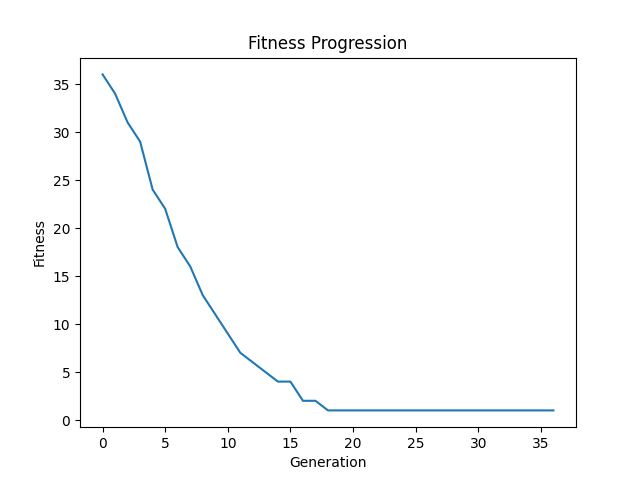
* Prilagodba svake jedinke računa se kao broj različitih znakova između gena jedinke i ciljnog stringa.
* Što je prilagodba manja, to je jedinka bliža ciljnoj sekvenci.

1. Operator Križanja i Mutacije:
   * Operator križanja između roditelja stvara potomka kombiniranjem njihovih gena.
   * Mutacija uvodi slučajne promjene u genetsku informaciju pojedinih gena.
   * Vjerojatnosti križanja i mutacije kontroliraju razinu varijabilnosti u populaciji.
2. Iterativni Proces:
   * Algoritam prolazi kroz iteracije, gdje se u svakoj iteraciji generira nova generacija jedinki.
   * Elitizam zadržava 10% najboljih jedinki iz prethodne generacije.
   * Ostatak nove generacije stvara se križanjem roditelja iz prethodne generacije.
   * Nakon svake generacije, ispisuje se trenutna informacija o najboljoj jedinki.
3. Uvjet Zaustavljanja:
   * Algoritam se zaustavlja kada se postigne ciljna prilagodba, odnosno kada je prilagodba najbolje jedinke nula.
4. Vizualizacija Progresije:

* Kroz iteracije, prilagodba najbolje jedinke bilježi se i vizualizira kako bi se pratila konvergencija algoritma.

U konačnici, ovaj genetski algoritam koristi princip prirodne evolucije za iterativno generiranje populacije prema ciljnom stringu. Kombinacija križanja, mutacije i selekcije omogućava algoritmu pretraživanje prostora rješenja kako bi pronašao optimalnu jedinku koja se podudara s ciljnim stringom.

import random  
import numpy as np  
import matplotlib.pyplot as plt  
import time  
  
POPULATION\_SIZE = 1000  
GENES = '''abcdefghijklmnopqrstuvwxyzčćđšžČĆĐŠŽABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890, .-;:\_!"#%&/()=?@${[]}<>'''  
TARGET = "Ja volim Operacijska istraživanja II <3"  
  
class Individual:  
 def \_\_init\_\_(self, chromosome):  
 self.chromosome = chromosome  
 self.fitness = self.cal\_fitness()  
  
 @classmethod  
 def mutated\_genes(cls):  
 return random.choice(GENES)  
  
 @classmethod  
 def generate\_mutated\_pool(cls, pool\_size=1000):  
 mutated\_pool = [cls.mutated\_genes() for \_ in range(pool\_size)]  
 return np.array(mutated\_pool, dtype=np.dtype('U1'))  
  
 @classmethod  
 def create\_gnome(cls):  
 return np.array([cls.mutated\_genes() for \_ in range(len(TARGET))], dtype=np.dtype('U1'))  
  
 def mate(self, partner, mutated\_pool):  
 child\_chromosome = []  
 for gp1, gp2 in zip(self.chromosome, partner.chromosome):  
 prob = random.random()  
 if prob < 0.45:  
 child\_chromosome.append(gp1)  
 elif prob < 0.90:  
 child\_chromosome.append(gp2)  
 else:  
 child\_chromosome.append(random.choice(mutated\_pool)) # Using pre-generated mutated\_pool  
 return Individual(np.array(child\_chromosome, dtype=np.dtype('U1')))  
  
 def cal\_fitness(self):  
 return np.sum(self.chromosome != np.array(list(TARGET), dtype=np.dtype('U1')))  
  
def main():  
 generation = 1  
 found = False  
 population = [Individual(Individual.create\_gnome()) for \_ in range(POPULATION\_SIZE)]  
 fitness\_progress = []  
  
 # Generate a pool of mutated genes  
 mutated\_pool = Individual.generate\_mutated\_pool()  
  
 while not found:  
 population = sorted(population, key=lambda x: x.fitness)  
  
 if population[0].fitness <= 0:  
 found = True  
 break  
  
 new\_generation = []  
 s = int((10 \* POPULATION\_SIZE) / 100)  
 new\_generation.extend(population[:s])  
  
 s = int((90 \* POPULATION\_SIZE) / 100)  
 for \_ in range(s):  
 parent1 = random.choice(population[:50])  
 parent2 = random.choice(population[:50])  
 child = parent1.mate(parent2, mutated\_pool) # Pass the pre-generated mutated\_pool  
 new\_generation.append(child)  
  
 population = new\_generation  
 fitness\_progress.append(population[0].fitness)  
  
 print(f"Generation: {generation}\tString: {''.join(population[0].chromosome)}\tFitness: {population[0].fitness}")  
 generation += 1  
  
 print(f"Generation: {generation}\tString: {''.join(population[0].chromosome)}\tFitness: {population[0].fitness}")  
  
 # Visualization using Matplotlib  
 plt.plot(fitness\_progress)  
 plt.xlabel('Generation')  
 plt.ylabel('Fitness')  
 plt.title('Fitness Progression')  
 plt.show()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()



Slika 3. Generiranje genetskog algoritma "Ja volim operacijska istraživanja <3"

## Pseudokod

* Inicijalizacija:

Pseudokod započinje definicijom varijabli koje određuju veličinu populacije, genski skup, ciljni string te klasu Individual koja predstavlja jedinku u genetskom algoritmu. Ova klasa sadrži metode za stvaranje mutiranih gena, generiranje mutiranog skupa, i stvaranje početne jedinke.

Metoda `mate` predstavlja operator križanja između roditelja, simulirajući genetsko rekombiniranje. Izbor gena za dijete ovisi o slučajnim vjerojatnostima, uz korištenje prethodno generiranog mutiranog skupa.

Metoda `cal\_fitness` računa prilagodbu jedinke na temelju usporedbe s ciljnim stringom.

1. POPULATION\_SIZE = 1000  
2. GENES = 'abcdefghijklmnopqrstuvwxyzčćđšžČĆĐŠŽABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890, .-;:\_!"#%&/()=?@${[]}<>'  
3. TARGET = "Ja volim Operacijska istraživanja II <3"  
  
4. class Individual:  
5. def \_\_init\_\_(self, chromosome):  
6. self.chromosome = chromosome  
7. self.fitness = self.cal\_fitness()  
  
8. @classmethod  
9. def mutated\_genes(cls):  
10. return random.choice(GENES)  
  
11. @classmethod  
12. def generate\_mutated\_pool(cls, pool\_size=1000):  
13. mutated\_pool = [cls.mutated\_genes() for \_ in range(pool\_size)]  
14. return np.array(mutated\_pool, dtype=np.dtype('U1'))  
  
15. @classmethod  
16. def create\_gnome(cls):  
17. return np.array([cls.mutated\_genes() for \_ in range(len(TARGET))], dtype=np.dtype('U1'))  
  
18. def mate(self, partner, mutated\_pool):  
19. child\_chromosome = []  
20. for gp1, gp2 in zip(self.chromosome, partner.chromosome):  
21. prob = random.random()  
22. if prob < 0.45:  
23. child\_chromosome.append(gp1)  
24. elif prob < 0.90:  
25. child\_chromosome.append(gp2)  
26. else:  
27. child\_chromosome.append(random.choice(mutated\_pool)) # Using pre-generated mutated\_pool  
28. return Individual(np.array(child\_chromosome, dtype=np.dtype('U1')))  
  
29. def cal\_fitness(self):  
30. return np.sum(self.chromosome != np.array(list(TARGET), dtype=np.dtype('U1')))

* Glavni Algoritam:

Glavni algoritam započinje generiranjem početne populacije. U svakoj iteraciji, populacija se sortira prema prilagodbi, a ako je prilagodba najbolje jedinke nula, algoritam završava.

Zatim se generira nova generacija, pri čemu se elitnih 10% najboljih jedinki direktno prenosi u novu generaciju. Ostatak nove generacije stvara se kombiniranjem roditelja iz trenutne populacije korištenjem operatora križanja.

Na kraju svake iteracije ispisuje se generacija, trenutni string najbolje jedinke i prilagodba.

31. def main():  
32. generation = 1  
33. found = False  
34. population = [Individual(Individual.create\_gnome()) for \_ in range(POPULATION\_SIZE)]  
35. fitness\_progress = []  
  
36. # Generate a pool of mutated genes  
37. mutated\_pool = Individual.generate\_mutated\_pool()  
  
38. while not found:  
39. population = sorted(population, key=lambda x: x.fitness)  
  
40. if population[0].fitness <= 0:  
41. found = True  
42. break  
  
43. new\_generation = []  
44. s = int((10 \* POPULATION\_SIZE) / 100)  
45. new\_generation.extend(population[:s])  
  
46. s = int((90 \* POPULATION\_SIZE) / 100)  
47. for \_ in range(s):  
48. parent1 = random.choice(population[:50])  
49. parent2 = random.choice(population[:50])  
50. child = parent1.mate(parent2, mutated\_pool) # Pass the pre-generated mutated\_pool  
51. new\_generation.append(child)  
  
52. population = new\_generation  
53. fitness\_progress.append(population[0].fitness)  
  
54. print(f"Generation: {generation}\tString: {''.join(population[0].chromosome)}\tFitness: {population[0].fitness}")  
55. generation += 1  
  
56. print(f"Generation: {generation}\tString: {''.join(population[0].chromosome)}\tFitness: {population[0].fitness}")  
  
57. # Visualization using Matplotlib  
58. plt.plot(fitness\_progress)  
59. plt.xlabel('Generation')  
60. plt.ylabel('Fitness')  
61. plt.title('Fitness Progression')  
62. plt.show()  
  
63. if \_\_name\_\_ == '\_\_main\_\_':  
64. main()

## Matematički Model

Neka su:

- veličina populacije,

- duljina ciljnog stringa,

- vjerojatnost mutacije,

- vjerojatnost križanja.

* Računanje prilagodbe:

gdje je -ti karakter jedinke, a -ti karakter ciljnog stringa.

* Operator križanja (mate operacija):

Ovaj model predstavlja pojednostavljen prikaz genetskog algoritma korištenog u programskom jeziku Python. Prilagodbe se mogu napraviti prema specifičnim zahtjevima ili varijacijama u algoritmu.

## Blok dijagram

NE

Generiraj novu populaciju

DA

Najbolji pojedinci

Evaluacija s funkcijom prilagođenosti

Zadovoljavajuće

Križanje

Selekcija

Mutacija

Početna populacija

# PREDNOSTI I NEDOSTACI

U optimizaciji, metaheuristički algoritmi pružaju svestran pristup rješavanju različitih problema. Ova usporedba fokusira se na nekoliko takvih algoritama, razmatrajući njihove karakteristike, prednosti i nedostatke.

1. **Genetski algoritam (GA)**

Genetski algoritam temelji se na principima prirodne evolucije. Koristi populaciju kandidata za rješenja, čime održava raznolikost i sprječava zaglavljivanje u lokalnim optimumima. GA je sposoban rješavati složene probleme, ali može zahtijevati više računalnih resursa zbog pristupa temeljenog na populaciji.

1. **Simulirano kaljenje**

Simulirano kaljenje je metaheuristički algoritam temeljen na jednom rješenju. Koristi lokalno pretraživanje kako bi poboljšao trenutno rješenje. Iako je učinkovit za određene optimizacijske probleme, može se zaglaviti u lokalnim optimumima.

1. **Tabu pretraga (TS)**

Tabu pretraga je još jedan metaheuristički algoritam temeljen na jednom rješenju. Ovaj algoritam može pružiti visokokvalitetna rješenja, ali postoji rizik od zaglavljivanja u lokalnim optimumima.

1. **Optimizacija rojem čestica (PSO)**

Optimizacija rojem čestica modelirana je prema ponašanju jata ptica ili škole riba. Održava raznolikost u rješenjima i sposobna je rješavati složene probleme. Kao i GA, može zahtijevati više računalnih resursa zbog pristupa temeljenog na populaciji.

1. **Optimizacija kolonijom mrava (ACO)**

Optimizacija kolonijom mrava inspirirana je ponašanjem mrava u pronalaženju putova od kolonije do hrane. Također održava raznolikost u rješenjima i sposobna je rješavati složene probleme, ali može zahtijevati više računalnih resursa zbog pristupa temeljenog na populaciji.

Tablica 1. Prednosti i nedostaci [13][12][11][1]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algoritam** | **Opis** | **Primjena** | **Prednosti** | **Nedostaci** |
| **Genetski algoritam (GA)** | GA je metaheuristički algoritam temeljen na populaciji inspiriran prirodnim procesom evolucije. Koristi više kandidata za rješenja tijekom procesa pretraživanja, održava raznolikost u populaciji i izbjegava da se rješenja zaglave u lokalnim optimama. | Rješavanje složenih problema, evolucijski dizajn, raspodjela resursa | Može riješiti složene probleme, održava raznolikost u rješenjima. | Može zahtijevati više računalnih resursa zbog pristupa temeljenog na populaciji. |
| **Simulirano kaljenje** | Metaheuristički algoritam temeljen na jednom rješenju koji koristi jedno kandidatsko rješenje i poboljšava ovo rješenje koristeći lokalno pretraživanje. | Planiranje rasporeda, optimizacija transporta, sustavi upravljanja zalihama | Učinkovit za određene optimizacijske probleme. | Može se zaglaviti u lokalnim optimama. |
| **Tabu pretraga (TS)** | Još jedan metaheuristički algoritam temeljen na jednom rješenju. | Raspoređivanje vozila, raspoređivanje poslova, optimizacija procesa proizvodnje | Može pružiti rješenja visoke kvalitete. | Može se zaglaviti u lokalnim optimama. |
| **Optimizacija rojem čestica (PSO)** | Metaheuristički algoritam temeljen na populaciji inspiriran ponašanjem jata ptica ili škole riba. | Obrada slika, optimizacija sustava upravljanja, prilagodba neuronskih mreža | Može riješiti složene probleme, održava raznolikost u rješenjima. | Može zahtijevati više računalnih resursa zbog pristupa temeljenog na populaciji. |
| **Optimizacija kolonijom mrava (ACO)** | Metaheuristički algoritam temeljen na populaciji inspiriran ponašanjem mrava u pronalaženju putova od kolonije do hrane. | Rute vozila, mrežno usmjeravanje, problem trgovačkog putnika | Može riješiti složene probleme, održava raznolikost u rješenjima. | Može zahtijevati više računalnih resursa zbog pristupa temeljenog na populaciji. |

U zaključku, odabir metaheurističkog algoritma ovisi o specifičnostima problema i resursima na raspolaganju. Svaki od navedenih algoritama ima svoje prednosti i nedostatke, te se primjenjuje prema zahtjevima konkretne optimizacijske situacije.

# PRIMJENE GENETSKIH ALGORITAMA U INDUSTRIJSKOM INŽENJERSTVU

Integracija genetskih algoritama u industrijskom inženjerstvu revolucionirala je tradicionalne pristupe rješavanju problema i optimizaciji. Ovi algoritmi, nadahnuti prirodnom selekcijom i evolucijom, nude inovativna rješenja za zamršene izazove u različitim industrijskim domenama. Od optimizacije dizajna do logističke učinkovitosti i poboljšanja procesa, genetski algoritmi postali su nezamjenjivi alati u modernom industrijskom inženjerstvu. U nastavku će se prikazati neke od mnogih spomenutih primjena genetskih algoritama.

* 1. Genetski algoritmi u optimizaciji industrijskog dizajna

Genetski algoritmi igraju ključnu ulogu u optimizaciji industrijskog dizajna revolucioniranjem iterativnog procesa stvaranja i usavršavanja dizajna proizvoda. Industrije poput automobilske ili zrakoplovne koriste genetske algoritme za istraživanje golemog dizajnerskog prostora, čime mogu lako generirati i procjenjivati brojne alternative dizajna. Konkretno, ovi algoritmi simuliraju prirodnu selekciju, razvijajući i poboljšavajući dizajne tijekom uzastopnih generacija.

Na primjer, u automobilskom dizajnu, genetski algoritmi pomažu u optimizaciji aerodinamike vozila, strukturalnog integriteta i ukupne izvedbe. Inženjeri unose parametre i ograničenja koja se odnose na aerodinamičke koeficijente, čvrstoću materijala i raspodjelu težine. Algoritam zatim istražuje te varijable, generirajući različite mogućnosti dizajna. Kroz iteracije i evaluacije na temelju unaprijed definiranih kriterija (npr. učinkovitost goriva, sigurnosni standardi), konvergira prema optimalnim projektnim rješenjima [17].

* 1. Genetski algoritmi u planiranju proizvodnog procesa

Planiranje proizvodnog procesa uključuje redoslijed operacija i optimalnu raspodjelu resursa kako bi se osigurala učinkovita proizvodnja. Genetski algoritmi to olakšavaju rješavanjem složenih odluka uključenih u slijed proizvodnje. Oni optimiziraju raspored proizvodnje uzimajući u obzir faktore kao što su mogućnosti stroja, ovisnosti o tijeku rada te proizvodne rokove.

U praktičnim primjenama, genetski algoritmi pomažu u optimiziranju proizvodnih linija za industrije poput elektronike ili farmaceutskih proizvoda, smanjujući vrijeme postavljanja i uska grla u proizvodnji. Procjenom različitih sljedova operacija, ovi algoritmi identificiraju najučinkovitiji raspored, u konačnici rezultiraju smanjenjem vremena isporuke i maksimiziranjem iskorištenja opreme [18].

* 1. Genetski algoritmi u logistici i distribuciji

Logistika i distribucija podrazumijevaju upravljanje kretanjem i skladištenjem robe. Genetski algoritmi su ključni u optimizaciji rute, posebno u scenarijima *isporuke zadnje milje* (last-mile delivery). Oni rješavaju problem trgovačkog putnika (TSP) određivanjem najučinkovitijih ruta isporuke za više odredišta, uzimajući u obzir čimbenike kao što su promet, udaljenost i rokovi isporuke.

Za tvrtke koje se bave e-trgovinom ili isporukom paketa, genetski algoritmi optimiziraju rute dostave za više vozila, osiguravajući pravovremene isporuke uz smanjenje potrošnje goriva i mehaničkog trošenja vozila. Takva optimizacija povećava operativnu učinkovitost i zadovoljstvo kupaca [19].

* 1. Genetski algoritmi u problemima industrijskog rasporeda

Industrije sa složenim zahtjevima za planiranje, kao što su proizvodni pogoni ili montažne trake, imaju koristi od genetskih algoritama u optimizaciji zadataka raspoređivanja. Ovi algoritmi rješavaju NP- teške probleme raspoređivanja generiranjem rasporeda koji minimiziraju *makespan* (ukupno vrijeme završetka) ili maksimiziraju korištenje resursa.

U proizvodnji, genetski algoritmi optimiziraju redoslijed poslova, smanjujući vrijeme prebacivanja između različitih proizvoda i osiguravajući učinkovito korištenje stroja. Uzimajući u obzir različita ograničenja i ciljeve, ovi algoritmi stvaraju rasporede koji poboljšavaju učinkovitost proizvodnje i smanjuju vrijeme mirovanja [20].

* 1. Genetski algoritmi u industrijskim procesima proizvodnje čelika

U proizvodnji čelika optimizacija procesa poput toplinske obrade ili sastava materijala presudna je za postizanje željenih svojstava materijala. Genetski algoritmi, zajedno s metodama konačnih elemenata, optimiziraju te procese istražujući ogroman prostor parametara kako bi pronašli najbolju kombinaciju koja zadovoljava specifične kriterije izvedbe.

Na primjer, u toplinskoj obradi, genetski algoritmi pomažu u određivanju idealnih temperaturnih profila i trajanja, osiguravajući željena svojstva materijala poput tvrdoće ili duktilnosti. Ova optimizacija smanjuje eksperimentiranje pokušaja i pogrešaka, što dovodi do uštede troškova i poboljšane kvalitete proizvoda [21].

Genetski algoritmi stoje kao preteče inovacija u industrijskom inženjerstvu, nudeći svestrana rješenja u mnoštvu aplikacija. Od pojednostavljenja proizvodnih procesa do optimizacije logističkih operacija i poboljšanja učinkovitosti dizajna, njihove mogućnosti svakim danom nude sve veću primjenu. Kako se industrije razvijaju, stalna integracija genetskih algoritama obećava daljnji napredak u učinkovitosti, produktivnosti i kvaliteti u svim industrijskim sektorima.

# SOFTVERSKA RJEŠENJA

Genetsko programiranje i algoritmi postaju jedno od najtraženijih područja u umjetnoj inteligenciji i strojnom učenju. Ovi algoritmi koriste se za proučavanje i analizu modifikacija gena i evolucije, procjenjujući genetsku konstituciju. S rastućim interesom za ovu domenu, također se razvijaju mnogi alati i tehnologije kako bi omogućili brže i učinkovitije istraživanje i progres. Od osnovnih do naprednih razina, danas postoji velik obujam alata koji omogućuju napredno istraživanje u području genetskog programiranja. Ovdje se navodi pet najčešće korištenih jezika za genetsko programiranje.

## MATLAB

Ovaj licencirani alat najčešće koriste istraživači za pisanje genetskih algoritama jer pruža fleksibilnost za uvoz podataka iz .xls datoteka, CSV datoteka itd. Posjeduje moćne, već ugrađene alate za crtanje koji omogućuju jednostavnu vizualizaciju podataka. Smatra se jednim od najboljih alata za genetske algoritme. Kada su u pitanju alati u MATLAB-u, jedan od najpopularnijih je GEATbx, alatka za genetske i evolucijske algoritme. Pruža mogućnosti globalne optimizacije u MATLAB-u za rješavanje problema koji nisu prikladni za tradicionalne pristupe optimizaciji. Također omogućuje rješavanje velikih i složenih problema uz laku vizualizaciju, višestruku optimizaciju, rukovanje ograničenjima i slično [22].

## Python

Python se ističe kao jedan od najpreferiranijih alata za genetsko programiranje zbog svoje jednostavnosti, fleksibilnosti i bogatstva biblioteka koje podržavaju različite aspekte genetskih algoritama. Osim toga, neke od najpoznatijih biblioteka za genetsko programiranje u Pythonu pružaju iscrpne mogućnosti koje unapređuju istraživanje i primjenu u različitim područjima:

### Pyvolution

Pyvolution se ističe kao biblioteka koja omogućuje izradu interaktivnih demonstracijskih aplikacija specijaliziranih za grafički prikaz. Ova biblioteka olakšava vizualizaciju genetskih algoritama, omogućujući korisnicima da prate evolucijske procese kroz dinamične i intuitivne grafičke prikaze [23].

### deap

Deap je jedna od najčešće korištenih biblioteka za genetsko programiranje u Pythonu. Pruža širok spektar alata za evolucijsko računanje, uključujući operatore za selekciju, križanje, mutaciju i evaluaciju jedinki. Ova biblioteka je sveobuhvatna i koristi se za rješavanje različitih optimizacijskih problema te podržava višestruke strategije genetskih algoritama [24].

Ove biblioteke u Pythonu omogućuju širok spektar primjena, od vizualizacije i optimizacije do specijaliziranih aplikacija u robotici, čime pridonose raznolikosti i bogatstvu genetskog programiranja u ovom programskom jeziku.

## Java

Java se često preferira zbog svoje objektno-orijentirane prirode i jednostavnosti programiranja genetskih algoritama. Korištenje Jave pruža potpunu prilagodljivost i uklanja neizvjesnost prilikom razvoja genetskih algoritama. Nakon što se definira skup klasa ili alata, modificiranje istih za različite zadatke postaje prilično jednostavno. Međutim, zahtijeva od korisnika da posjeduje određeno znanje programiranja, a odgovornost za greške leži na samom korisniku.

### Jenetics

Jenetics se ističe kao jedna od popularnih biblioteka za genetsko programiranje u Javi. Pruža bogat set alata za implementaciju genetskih algoritama s naglaskom na evolucijske strategije. Ova biblioteka olakšava stvaranje različitih varijacija genetskih algoritama, omogućujući korisnicima da prilagode strategije prema svojim potrebama [25].

### EpochX

EpochX je još jedna istaknuta biblioteka u Javi koja podržava razvoj genetskih algoritama. Ova biblioteka pruža fleksibilnost u implementaciji i evaluaciji genetskih algoritama, s naglaskom na jednostavnosti korištenja i prilagodljivosti za različite vrste optimizacijskih problema [26].

Java nudi alate koji omogućuju raznolikost i prilagodljivost u implementaciji genetskih algoritama, pridonoseći razvoju u području evolucijskog računanja i optimizacije. Ove biblioteke olakšavaju istraživačima razvoj i prilagodbu genetskih algoritama za širok spektar primjena.

## C++

C++ se ističe kao jedan od izvrsnih izbora za genetsko programiranje zbog svoje računalne efikasnosti. Pruža visok nivo softverskog okruženja za obavljanje kompleksnih zadataka u genetskom programiranju, uključujući alate za različite vrste genetskih algoritama:

### openGA

OpenGA je robusna i besplatna C++ biblioteka dizajnirana posebno za optimizaciju genetskog algoritma (GA). Ova svestrana biblioteka nudi mogućnosti za izvršavanje GA u jednostrukim, višestrukim i interaktivnim načinima rada. Za razliku od određenih ograničenja s kojima se MATLAB suočava, openGA pruža široku fleksibilnost, osnažujući korisnike s prilagodljivim značajkama. [27].

### BEAGLE

Open BEAGLE, okvir za evolucijsko računanje (EC) razvijen u C++-u, obuhvaća principe objektno orijentiranog (OO) programiranja. Arhitektura Open BEAGLE-a usklađena je s OO programiranjem, naglašavajući labavo spajanje objekata i mogućnost ponovne upotrebe koda, osiguravajući jednostavan, prenosiv i robustan EC okvir. Osim toga, elegancija Open BEAGLE-a ogleda se u njegovom koherentnom dizajnu koji slijedi OO i generička programska načela, promičući čitljivost, razumijevanje i modifikaciju njegovog C++ koda [28].

Korištenje C++-a za genetsko programiranje omogućuje programerima i istraživačima da se bave računalno intenzivnim zadacima uz napredne mogućnosti optimizacije i prilagodljivost. Ove biblioteke u C++-u olakšavaju razvoj i implementaciju genetskih algoritama za širok spektar primjena, pridonoseći raznolikosti i snazi u području evolucijskog računanja.

## Darwin

Darwin se ističe kao jezik genetskog algoritma koji pojednostavljuje eksperimentiranje s reprezentacijama GA rješenja, operatorima i parametrima, omogućavajući korisnicima da definiraju GA strukture s minimalnim definicijama i automatskim generiranjem većine programske logike. Ovaj jezik nudi jednostavnu sintaksu koja olakšava implementaciju i omogućuje korisnicima da pregledno prate implementaciju kroz proces prevođenja.

Darwin je posebno koristan za korisnike koji već posjeduju osnovno znanje o genetskim algoritmima, programskim jezicima i kompilatorima. Pruža intuitivan pristup razvoju genetskih algoritama uz minimalno trošenje vremena na kompleksne definicije i generiranje programske logike, čime olakšava eksperimentiranje i iteriranje kroz različite GA pristupe. Ovaj jezik pruža korisnicima mogućnost da se brzo fokusiraju na suštinu algoritma, što ga čini efikasnim alatom za napredne korisnike u domenu genetskog programiranja [29].

Softverska rješenja koja podržavaju genetsko programiranje i algoritme pružaju raznolike mogućnosti istraživanja i primjene u domenama umjetne inteligencije, strojnog učenja i operacijskih istraživanja. Kroz korištenje alata poput MATLAB-a, Pythona, Jave, C++-a i Darwinovog jezika, istraživači i programeri dobivaju moćne resurse za razvoj, optimizaciju i analizu genetskih algoritama.

Svaki od ovih jezika i biblioteka nudi specifične prednosti, od fleksibilnosti i jednostavnosti korištenja do visokih performansi i prilagodljivosti. Primjena ovih alata omogućuje rješavanje kompleksnih problema optimizacije, evolucije i prilagodbe s različitim pristupima i perspektivama.

Stalno napredovanje softverskih rješenja u podršci genetskih algoritama širi granice istraživanja, potiče inovacije te doprinosi razvoju novih tehnologija i primjena. Kroz ove softverske alate, istraživači i inženjeri dobivaju snažne resurse za istraživanje i primjenu genetskih algoritama u širokom spektru područja, od optimizacije do evolucijskog dizajna i analize podataka.

# ZAKLJUČAK

Kroz ovaj seminar se napravilo istraživanje genetskih algoritama (GA) gdje su se obradili temeljni elementi, shematske metodologije, prednosti i nedostatci te različite primjene u industrijskom inženjerstvu. Ovaj pregled je razotkrio fundamentalne genetske operatore - sheme kodiranja, selekciju, križanje i mutacije - ističući njihove ključne uloge u GA operacijama.

Shematske metodologije detaljno su opisale strukturne komponente GA-ova, razjašnjavajući njihovo kodiranje, pseudokod, matematičke modele i blok dijagrame. Ovi vizualni prikazi ponudili su sveobuhvatan uvid u unutarnji rad GA-a, pomažući i razumijevanju i implementaciji.

Udubljivanje u prednosti i nedostatke istaknulo je nijansirane aspekte GA-a. Dok GA-ovi predstavljaju prilagodljivost, robusnost i raznolikost rješenja, oni također nailaze na ograničenja koja se tiču računske složenosti i podešavanja parametara.

Primjene GA-ova u industrijskom inženjeringu isplivale su na površinu kao ključna žarišna točka, pokazujući njihovu ključnu ulogu u procesima optimizacije, planiranja, logistike i projektiranja. GA-ovi pokazuju snagu u rješavanju složenih industrijskih problema, optimiziranju procesa i povećanju učinkovitosti u raznim domenama.

Prelazeći na softverska rješenja, rasprava je istaknula niz programskih jezika i biblioteka za implementaciju GA. Od MATLAB-a, Pythona, Jave i C++ do specijaliziranih alata kao što su openGA, BEAGLE i Darwin, svako softversko rješenje nudi različite prednosti i sučelja prilagođena za razvoj GA.

Seminar je otkrio sveobuhvatnost, djelokrug i potencijal GA-ova u rješavanju zamršenih industrijskih problema, dok se također istaknula bogata lepezu softverskih alata dostupnih za implementaciju samih GA-a. Zamršena ravnoteža između teorijskog razumijevanja i praktične primjene pojavila se kao kamen temeljac, obećavajući daljnji napredak i inovacije unutar područja genetskih algoritama.

U zaključku, genetski algoritmi stoje kao robusna računalna paradigma koja nudi niz mogućnosti u industrijskom inženjerstvu. Dubina njihovih aplikacija, zajedno s različitim softverskim rješenjima, utire put kontinuiranom istraživanju i usavršavanju, potičući napredak u rješavanju problema i optimizaciji unutar različitih industrijskih domena.

# LITERATURA

1. Sourabh Katoch & Sumit Singh Chauhan & Vijay Kumar: A review on genetic algorithm: past, present, and future, 2020, pristupljeno 15.12.2023.
2. Yara Rizka , Mariette Awad: A quantum genetic algorithm for pickup and delivery problems with coalition formation, 2019., pristupljeno 15.12.2023.
3. Dan Zhang, Cheng Zhang, Feng Xu, Dunwen Zuo: A hybrid genetic algorithm used in vehicle dispatching for JIT distribution in NC workshop, 2015., pristupljeno 15.12.2023.
4. Marco Rinaldi, Stefano Primatesta, Martin Bugaj, Ján Rostáš and Giorgio Guglier: Development of Heuristic Approaches for Last-Mile Delivery TSP with a Truck and Multiple Drones. 2023., pristupljeno 15.12.2023.
5. Yohei HAZAMA, Hitoshi IIMA, Yoshiyuki KARUNO and Kosuke MISHIMA: Genetic algorithm for scheduling of parcel delivery by drones, 2021., prostupljeno 15.12.2023.
6. Liu Xin, Peng Xu, and Gu Manyi: Logistics Distribution Route Optimization Based on Genetic Algorithm, 2022., pristupljeno 15.12.2023.
7. Zhou Lanfeng, Yang Lina and Fang Hua: Lunar Rover Path Planning Based on Comprehensive Genetic Algorithm Based on Slip Prediction, 2019., pristupljeno 15.12.2023.
8. Lindawati and Aldy Gunawan: Minimizing waiting time in last mile delivery for smart city, pristupljeno 15.12.2023.
9. A.I. Diveeva, O.V. Bobra: Variational genetic algorithm for NP-hard scheduling problem solution, 2016., pristupljeno 15.12.2023.
10. Emmanuel Karlo Nyarko, Robert Cupec, Damir Filko: A Comparison of Several Heuristic Algorithms for Solving High Dimensional Optimization Problems, 2014., pristupljeno 15.12.2023.
11. Jaoqin Perez O., Rdolofo A. Pazos R., Juan Frausto S., Guillermo Rodriguez O., Laura Cruz R. and Hector Fraire H.: Comparison and Selection of Exact and Heuristic Algorithm, 2004., pristupljeno 15.12.2023.

1. [heuristics - What are the real differences between genetic algorithms and evolutionary algorithms? - Stack Overflow](https://stackoverflow.com/questions/9176195/what-are-the-real-differences-between-genetic-algorithms-and-evolutionary-algori) , pristupljeno 15.12.2023.
2. Seyedali Mirjalili, Jin Song Dong, Ali Safa Sahiq and Hossam Faris: Genetic Algorithm :Theory, Literature Review and Application in Image Reconstruction, 2019., pristupljeno 15.12.2023.
3. Mohammad Zahid Rayaz Khan and Dr. A K Bajpai: Genetic Algorithm And Its Application In Mechanical Engineering, 2013., pristupljeno 15.12.2023.
4. Wireless Communications and Mobile Computing: Retracted: Research on Automobile Assembly Line Optimization Based on Industrial Engineering Technology and Machine Learning Algorithm, 2023., pristupljeno 15.12.2023.
5. A. Sanz-García, A.V. Pernía-Espinoza, R. Fernández-Martínez, F.J. Martínez-de-Pisón-Ascacíbar: Combining genetic algorithms and the finite element method to improve steel industrial processes, 2012., pristupljeno 15.12.2023.
6. Sáez-Gutiérrez, F. L., F. J. F. Cañavate, and A. Guerrero-González. “Review of Industrial Design Optimization by Genetic Algorithms.” *Temporal Modelling of Customer Behaviour*, edited by Jorge Casillas et al., Springer, 2019, pp. 336-346
7. Ma, Guohua, and Fu Zhang. "Genetic algorithms for manufacturing process planning." *Variants of Evolutionary Algorithms for Real-World Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. 205-244.
8. Xin, Liu, Peng Xu, and Gu Manyi. "Logistics distribution route optimization based on genetic algorithm." *Computational Intelligence and Neuroscience* 2022 (2022).
9. Diveev, A. I., and O. V. Bobr. "Variational genetic algorithm for np-hard scheduling problem solution." *Procedia Computer Science* 103 (2017): 52-58.
10. Sanz-García, Andrés, et al. "Combining genetic algorithms and the finite element method to improve steel industrial processes." *Journal of Applied Logic* 10.4 (2012): 298-308.
11. Pohlheim, Hartmut. "Genetic and Evolutionary Algorithm Toolbox for use with MATLAB." *Dept. Comput. Sci., Univ. Ilmenau, Ilmenau, Germany* (1998).
12. Panchapakesan, Ashwin. "Pyvolution 1.1." Genetic 1.1 documentation. Python.org, 12 2012. Web. 5 listopad 2023.
13. Fortin, Félix-Antoine, et al. "DEAP: Evolutionary algorithms made easy." *The Journal of Machine Learning Research* 13.1 (2012): 2171-2175.
14. Wilhelmstötter, Franz. "Jenetics." *URL: http://jenetics. io* (2021).
15. Otero, Fernando, Tom Castle, and Colin Johnson. "Epochx: Genetic programming in java with statistics and event monitoring." *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*. 2012.
16. Mohammadi, Arash, et al. "OpenGA, a C++ genetic algorithm library." *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2017.
17. Gagné, Christian, and Marc Parizeau. "Open BEAGLE: A New Versatile C++ Framework for Evolutionary Computation." *GECCO Late Breaking Papers*. 2002.
18. Arslan, Arslan and Göktürk Üçoluk. “DARWIN: A Genetic Algorithm Language.” *International Symposium on Computer and Information Sciences* (2013).