**TEHNIČKO VELEUČILIŠTE U ZAGREBU**

**STRUČNI STUDIJ RAČUNARSTVA**

Leon Štok

# OPTIMIZACIJA TRGOVAČKIH STRATEGIJA STROJNIM UČENJEM

ZAVRŠNI RAD br. 1524

Zagreb, rujan, 2022.

**TEHNIČKO VELEUČILIŠTE U ZAGREBU**

**STRUČNI STUDIJ RAČUNARSTVA**

Leon Štok

JMBAG: 0246088535

# OPTIMIZACIJA TRGOVAČKIH STRATEGIJA STROJNIM UČENJEM

ZAVRŠNI RAD br. 1524

Zagreb, rujan, 2022.

## Sažetak

U ovom radu će se prvo razjasniti kako se cijena formira na tržištu i što su to trgovačke strategije, te koje su njihove prednosti. Opisat će se optimizacijski algoritmi crne kutije, npr. mrežasto pretraživanje, nasumično, genetski algoritmi, Bayesova optimizacija kao i neki drugi hibridni algoritmi. Ti algoritmi će se usporediti nad nekim popularnim matematičkim funkcijama. Ujedno će se usporediti i s trgovačkom strategijom. Monte Carlo simulacijom će se ustanoviti optimalna duljina povijesnog perioda za optimizaciju. Objasnit će se problem memorijske složenosti kod optimizacije trgovačkih strategija i načini kako je ublažiti.

Ključne riječi: trgovačke strategije, optimizacija višedimenzionalnih funkcija, genetski algoritmi, Bayesova optimizacija

# Sadržaj

## Popis oznaka i kratica

API – Dio softvera koji služi za komunikaciju s drugim softverom (engl. application programming interface)

BO – Bayesova optimizacija

IEEE 754 – standard aritmetike realnog broja s putujućom točkom

GA – genetski algoritam

GiB – gibibajt, 1 GiB = $2^{30}$ bajtova

RAM – memorija s nasumičnim pristupom (engl. random access memory)

REST – stil razvoja softvera, najčešće web poslužitelja (engl. representational state transfer)

RSI – indeks relativne jačine (engl. relative strength index), tehnički indikator

SATA - Serial AT Attachment, tip priključka uređaja za pohranu

SSD – solid state drive, uređaj za pohranu visokih performansi

VO – vremenski okvir, način grupiranja cijena s obzirom na vrijeme

## Popis slika

## Popis programskih isječaka

## Popis tablica

# 1. Uvod

Danas se trgovina vrijednosnih papira sve više izvršava elektronički, čime je omogućeno automatiziranje i besprekidni rad trgovanja. Automatiziranjem se smanjuju ljudske emocije trgovca koje znatno utječu na njegovu zaradu [1, str. 108-111]. Velikom brzinom današnjih računala se mogu testirati i simulirati milijuni strategija bez troška. Prije razvoja automatizacije trgovac je sâm morao izvršavati transakcije i voditi bilješke o izvođenju strategije. To je težak, dugotrajan i mukotrpan proces.

Velika većina trgovca se koristi diskretnim trgovanjem gdje oni u realnom vremenu odlučuju razmjenu vrijednosnih papira. Te odluke je teško predvidjeti čak i za samog trgovca koji ih izvršava, jer na odluke utječe ljudska psiha (npr. raspoloženje, osjećaji prema lošim događajima, iscrpljenost, eksperimentiranje novih metoda…). Utjecaj ljudske psihe se ublažava prelaskom na sistematsko trgovanje gdje su odluke točno definirane. Emocije će i dalje imati utjecaj iako se trgovac služi sistemskim trgovanjem. Zbog toga automatizirani sustavi imaju prednost.

Automatizirani sustav omogućava brzo simuliranje strategija nad povijesnim podacima. S njime je lakše kvantificiranje i usporedba s drugim strategijama. Otvara mogućnost izvršavanja velikog broja razmjena vrijednosnih papira u minuti. Smanjuje utjecaj ljudske greške i manje zahtjeva ljudsku pažnju, a i vremena. S druge strane teško ih je dizajnirati da pobijede profesionalne trgovce s puno iskustva.

Izrada trgovačkih strategija se sastoji od puno koraka. Ovaj rad će se fokusirati na optimizaciju njih. Pod optimizacijom se misli na to kako odrediti parametre trgovačke strategije tako da ona realizira najveći mogući profit. Objasnit će se struktura svijeća i trgovačkih strategija. Monte Carlo simulacijom će se utvrditi koliko podataka je potrebno za točnu optimizaciju. Brojni pokusi će utvrditi koji optimizacijski algoritmi su najbolji, te u kojim slučajevima.

# 2. Osnovni pojmovi i srodni radovi

## 2.1. Struktura cijene

U smislu vrijednosnih papira cijena je zadnja vrijednost neke razmjene. Razmjena se dogodi kada prodajna strana želi prodati, a kupovna kupiti za istu cijenu.

### 2.1.1. Knjiga ponuda

Postoje 3 najpopularnija modela cijena koji se baziraju korištenjem:

1) knjige ponuda (engl. orderbook)
2) bazena likvidnosti [1]
3) segmentiranog bazena likvidnosti [2]

Za razumijevanje je dovoljna knjiga ponuda koja se koristi u 99% slučajeva.

| Red | Tip | Količina | Cijena |
|---|---|---|---|
| 1 | Prodaja | 10.20 | 10.60 |
| 2 | Prodaja | 1.95 | 10.53 |
| 3 | Prodaja | 1.23 | 10.52 |
| 4 | | | |
| 5 | Kupnja | 0.57 | 10.50 |
| 6 | Kupnja | 2.70 | 10.49 |
| 7 | Kupnja | 8.91 | 10.46 |

*Tablica 1: Primjer knjige ponuda.*

Knjiga ponuda je sortirana po cijeni i sastoji se od dva segmenta, prodajne ponude i kupovne ponude. Ako osoba želi kupiti dvije dionice onda ima dvije mogućnosti. Može odmah kupiti na prodajnom segmentu po najboljoj cijeni uzimanjem ponuda (2. i 3. red). Kupit će 1.23 po cijeni 10.52 i kupit će 0.77 po cijeni 10.53. To je ukupno dvije dionice s prosječnom cijenom od (1.23*10.52+0.77*10.53) / 2 = 10.52385. Nakon kupnje 3. red se briše, a na drugom redu se količina smanji na 1.18. Drugi način je da stavi ponudu u knjigu s definiranom cijenom (npr. 10.49). Razmjena će se dogoditi samo ako netko želi prodati odmah tako da uzima iz knjige i ako nema ponuda boljih od njegove za prodavača. Prvo će se izvršiti 5. red, pa 6. red (onih 2.7 koji su stajali prije njegove ponude) i tek onda dolazi njegov red.

Ponuda se može otkazati pa će se izbrisati iz knjige. Na elektroničkim burzama knjiga ponuda se strahovito brzo mijenja. U trenutku velike navale jedna burza je dosegnula brzinu prometa od 2.5 GiB/s [3]. Da stavimo to u kontekst čitanje sa SATA SSD je oko 0.5 GiB/s [4]. To su ogromne količine podataka kojih je teško uskladištiti i raditi s njima. Zbog toga burze ne spremaju te podatke.

### 2.1.2. Struktura razmjene

Umjesto da se spremi svaki događaj koji se dogodio u knjizi ponuda, može se napraviti kompromis tako da se spremanju samo razmjene. Tim načinom se gubi znanje kolika je trenutna ponuda ili potražnja.

Razmjena mora minimalno sadržavati: vrijeme događaja, cijenu, količinu i stranu (prodaja ili kupnja). Za to je potrebno 8, 4, 4, 0 bajta respektivno. Stranu se sprema u predznak količine, a realne brojeve kao IEEE 754 format [5] koji ima dovoljnu preciznost za simulaciju trgovačke strategije. Uvidom u REST API Binance [6] burze se utvrđuje koliko se razmjena dogodilo do sada (22.74 milijarda). S takvim formatom za spremanje svih razmjena na svim tržištima s Binance burze je potrebno oko 338.88 GiB. To je još uvijek dosta puno.

### 2.1.3. Svijeće

Točnost u milisekundi nije potrebna za trgovačke strategije koje izvršavaju malu količinu razmjena u danu. Možda je točnost od jedne minute dovoljna ili čak jedan sat. Taj period se zove vremenski okvir (engl. timeframe). Da bi se to postiglo, razmjene se grupiraju u svijeće. Svijeća je format grupa razmjena u nekom vremenskom razdoblju i sastoji se od sljedećih vrijednosti [7].

- Timestamp – trenutno vrijeme početka grupacije
- Open – početna cijena prve razmjene
- High – maksimalna cijena razmjene u grupi
- Low – minimalna cijena razmjene u grupi
- Close – cijena zadnje razmjene
- Volume – suma količina svih razmjena u grupi

Trenutno, Binance ima 1904 tržišta i u prosjeku su aktivna 661.7 dana. Ukupni broj svijeća od jedne minute je 1.81 milijarda. Za pohranu svijeća je potrebno od prilike 47.31 GiB. Za simulaciju strategija je to dovoljno jer će se provoditi na manjem broju tržišta.

## 2.2. Indikatori

Indikatori su najčešće matematičke funkcije koje naglašavaju neki događaj. Taj događaj može biti da će cijena ići u nekom smjeru. Može biti i kompleksniji npr. prestanak ili početak razvoja cijene po nekim pravilima. Neki rade bolje od drugih, na drugim skupinama imovine ili u specifičnim uvjetima. Cijenu je teško predvidjeti, zbog toga nikad neće biti 100% točni.


*Slika 1: Primjer indikatora na cijeni dionice Apple Inc.. Izvor: [8]*

Na Slika 1 je vidljivo puno različitih indikatora:

- Plave linije – linije kretanja (engl. trendline) pokazuju da će se cijena odbiti od njih.

- Zelena linija – podrška i otpor (engl. support and resistance). To je horizontalna linija koja nastane kada se cijena okrene u vrhu ili dolini. Isto pokazuje da će se cijena odbiti.
- Crvena linija – putujuća srednja vrijednost (engl. moving average). Ako svijeće pređu preko nje odozdo prema gore onda se cijena nastavlja kretati prema gore. Ako se cijena već kreće prema gore onda se može odbiti od nje i promijeniti smjer.
- Ljubičasta linija – relativni indeks jačine (engl. relative strengh indeks (RSI)). Ako prijeđe iznad gornje linije onda postoji šansa da će se cijena okrenuti. Ako cijena kroz neki dulji period od 50 svijeća cijena pada, a RSI raste onda bi se cijena trebala okrenuti.
- Crveno-zeleni stupci – volumen (engl. volume). Broj dionica/ugovora kojih je kupljeno ili prodano u jednoj svijeći. Velik broj govori da postoji panika, euforija ili neslaganje u cijeni.
- E znak – Trenutak kada će tvrtka objaviti zaradu (engl. earnings) za taj kvartal. Očekuje se velika promjena u cijeni nad kratkom vremenskom periodu. On spada u fundamentalnu analizu, to je način određivanja buduće vrijednosti dionice sa svim drugim faktorima osim povijesne cijene dionice. [9]

Ovaj rad će se fokusirati na matematičke indikatore jer ih je lakše isprogramirati. Oni mogu imati jedan ili više brojeva za svaku svijeću, a ovise o svijećama i parametrima funkcije. Može ih se izračunati i spremiti za kasniju obradu.

## 2.3. Trgovačke strategije

Trgovačka strategija je u biti algoritam s točno definiranim uvjetima za izvršavanje razmjene. Strategija mora biti deterministična što znači da za točno određeni ulaz daje točno određeni izlaz. Preko toga svojstva se može simulirati trenutno stanje iz povijesnih podataka i tako provjeriti je li strategija profitabilna (engl. backtesting) [10]. Strategije koriste više indikatora koji pomažu pri odluci razmjene. Svaka strategija ima nekoliko parametara koji utječu na rad strategije.

Strategije ne traju vječno, dinamika tržišta se uvijek mijenja. U povijesti neki indikatori su dobro radili, no kako ih je sve više ljudi počelo koristiti, efikasnost se smanjila.

Programeri strategija su izrazito tajnoviti kako ne bi procurili informaciju i tako smanjili svoje profite.

Cilj ovog rada je objasniti kako naći parametre gdje strategija postiže najveći profit uz što manji broj potrošenih resursa.

## 2.4. Optimizacijski algoritmi

Algoritmi koji pretražuju funkciju kako bi našli globalni maksimum ili minimum se zovu optimizacijski algoritmi. Funkcija koji ti algoritmi pretražuju se zove objektivna funkcija. Uzorak je točka za koju se izračunala (evaluirala) funkcija. Raspon ulaznih parametara objektivne funkcije se zove prostor pretraživanja. Postoje dvije grupe algoritama s obzirom na objektivnu funkciju:

- Optimizacija bijele kutije (engl. white-box optimization) – derivacija funkcije u bilo kojoj točki je poznata prije evaluacije funkcije
- Optimizacija crne kutije (engl. black-box optimization) – derivacija nije poznata prije izračunavanja funkcije

Za trgovačke strategije se koriste optimizacija crne kutije jer su strategije računalni algoritmi koji mogu biti kompleksni te im derivacija nije poznata ili je teška za izračunati.

### 2.4.1. Rešetkasto pretraživanje

Najjednostavnija metoda koja pretražuje sva moguća rješenja. Prostor se podijeli na što je više moguće razmaka i testira se svaka moguća kombinacija. Na primjer: ako je zadano da je moguća domena jednog parametra između 0 i 100 onda se taj prostor podijeli na 11 dijelova (0, 10, 20, … 90, 100). Što više dijelova postoji to je veća preciznost, ali je i duže vrijeme izvršavanja. Ako se dodaju još 2 parametra s istim brojem dijelova onda je broj ukupnih dijelova: 11 * 11 * 11 = 1331. Iz čega se zaključuje da prostor pretraživanja eksponencijalno raste brojem parametra.

Prednosti:

- Jednostavno za implementirati
- Lagano je paralelizirati

- Mogućnost pronalaženje pravog maksimuma s velikom preciznošću

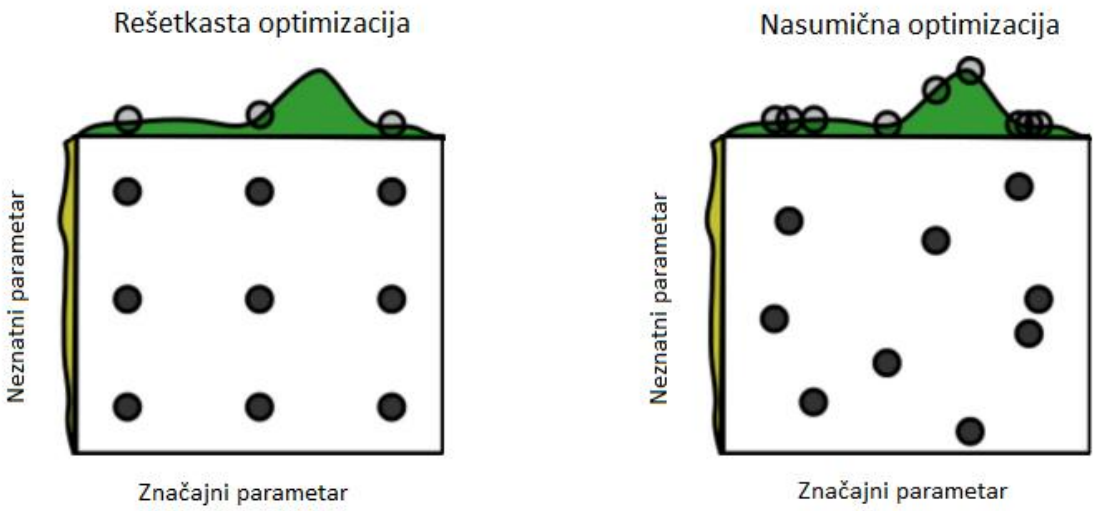
Nedostaci:

- Sporo pretraživanje u velikom prostoru
- Nije efektivno za funkcije koje ne ovise linearno o parametrima

Ono je pogodno za funkcije koje imaju mali prostor pretraživanja, koje se brzo se izvršavaju i ako je potreban globalni maksimum.

### 2.4.2. Nasumično pretraživanje

Za svaku konfiguraciju parametri se nasumično odaberu. Postoji mogućnost da se ista konfiguracija odabere više puta. Veće su šanse da će se pronaći maksimum nego rešetkasto pretraživanje manje gustoće nad velikim prostorima. Zbog sporog izvršavanja potrebno je smanjiti gustoću rešetka. Manjom gustoćom se premašuje optimum kao što je vidljivo na slici (Slika 2). Na slici su prikazane ovisnosti o tome koliko parametar pridonosi optimalnom rješenju.




*Slika 2: Usporedba rešetkastog i nasumičnog pretraživanja. Izvor: [11, p. 284]*

Mehanizam zaustavljanja mora biti definiran. On može biti da se zaustavi nakon nekoliko iteracija ili da vrijednost objektivne funkcije zadovoljava naše potrebe.

Prednosti:

- Efektivnije korištenje resursa naprema rešetkastog pretraživanja
- Na jako malom broju evaluacija može biti najefektivniji

Nedostaci:

- Globalni optimum često nije pogođen
- Pronalazi dosta nizak optimum

### 2.4.3. Genetski algoritmi (GA)

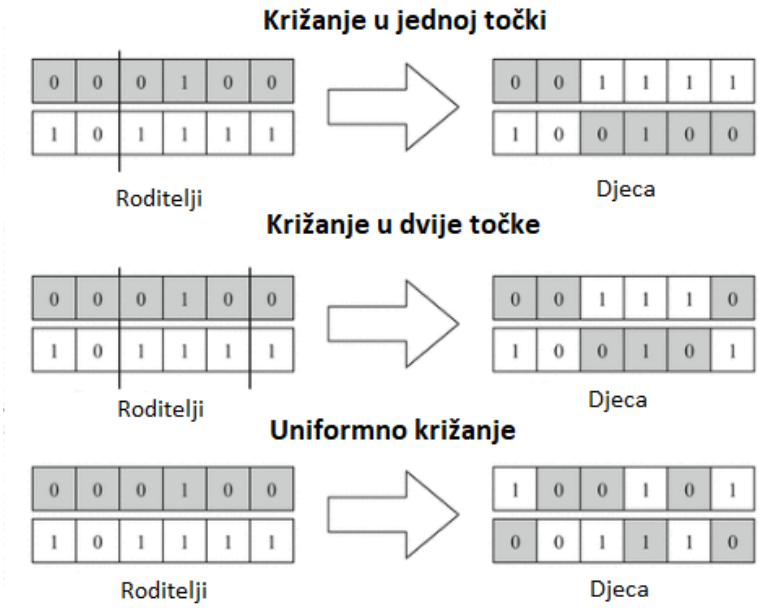
Genetski algoritmi su osmišljeni tako da oponašaju prirodu. Tako u svijetu postoje životinje koje se reproduciraju i prenose gene na djecu. Životinje koje imaju bolje gene su snažnije, dulje žive i reproduciraju se više. Prilikom reprodukcije postoji mala šansa mutacije. Ta mutacija može ojačati životinju ili je oslabiti. Kroz veliki broj generacija preživjet će jače životinje s boljim mutacijama.

Da se objasne genetski algoritmi potrebno je prvo znati terminologiju:

- Gen – najmanji dio neke karakteristike koja opisuje jedinku (npr. binarna, numerička, tekstualna vrijednost)
- Jedinka, kromosom, životinja – skupina gena koji rade zajedno
- Populacija – skupina jedinki
- Generacija – trenutna populacija
- Funkcija dobrote (engl. fitness function) – funkcija koja evaluira sposobnost (jačinu) jedinke
- Križanje, rekombinacija – način prijenosa gena s majke i oca na dijete
- Selekcija – način biranja jedinka za reprodukciju

U optimizacijskom kontekstu gen je jedna vrijednost parametra objektivne funkcije. Funkcija sposobnosti je izlazna vrijednost objektivne funkcije. Na početku populacija se inicijalizira nasumičnim vrijednostima gena.

U širem smislu algoritam se svodi na rangiranje jedinke funkcijom dobrote. Metodom selekcije se odaberu jedinke za reprodukciju. Stvaraju se nove jedinke koristeći gene oca i majke. Promijene se vrijednosti gena malih broja jedinka (mutacija). Ponovno se evaluira funkcija sposobnosti  i rangira populacija.




*Slika 3: Usporedba različitih operatora križanja. Izvor: [12]*

Postoji mnogo različitih operatora križanja (Slika 3) od kojih su 3 najpopularnija [12]:

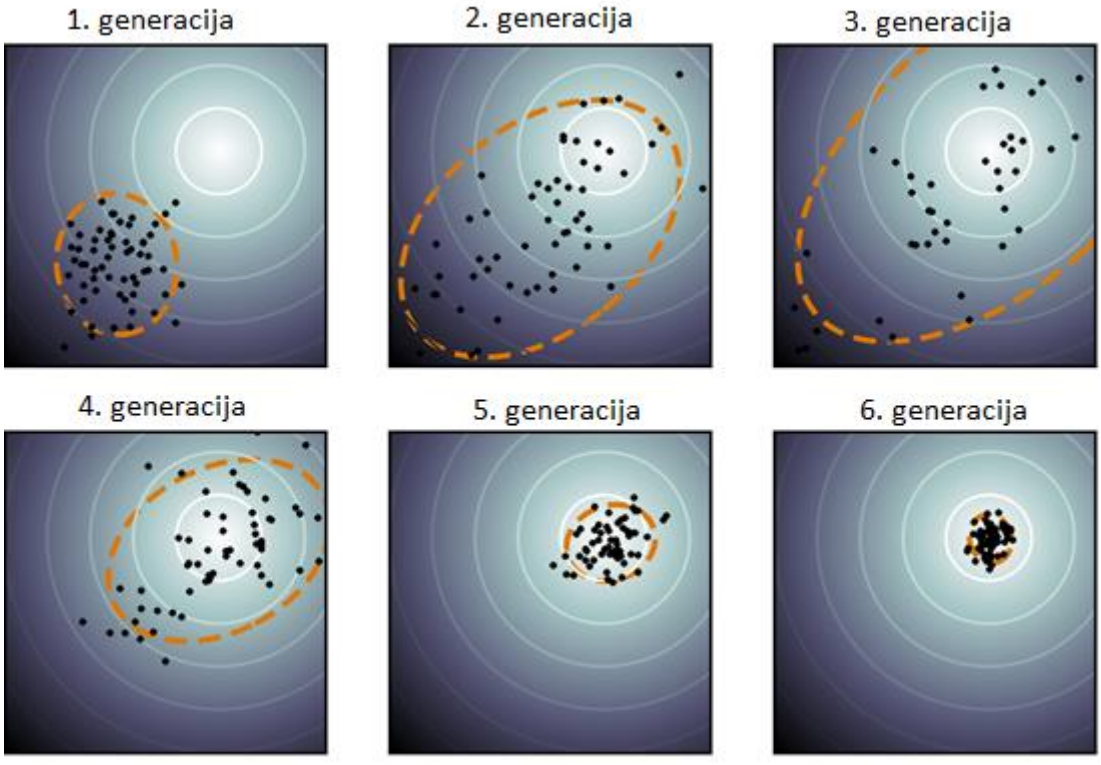
- Križanje u jednoj točki: geni se zamijene nakon nasumične točke
- Križanje u dvije točke: geni se zamijene nakon nasumične točke i prije još jedne nasumične točke
- Uniformno križanje: svaki gen ima 50% šanse da se zamijeni

Daljnje, postoje različite metode odabira jedinki:

- Odabir kotačem za rulet: Vjerojatnost biranja jedinke je proporcionalna njenoj sposobnosti. Postoji šansa da se ista jedinka odabere više puta.
- Odabir rangiranjem: Svaka jedinka ima istu vjerojatnost. Koristi se u slučaju kada se genetski algoritam izvršavao neko vrijeme i sve jedinke su slične. U tom slučaju veće su šanse da dijete ima roditelje koji su lošiji i djeca će biti raznolikija.
- Odabir turnirom: Nasumično se odaberu više jedinki. Jedinka s najboljom sposobnosti pobjeđuje. Pobjednici 2 turnira se reproduciraju.

Najpopularniji optimizacijski algoritam u evolucijskoj kategoriji je Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [13]. On se bazira na 2 principa:

1. Princip maksimalne vjerojatnosti, koji se temelji na ideji povećanja vjerojatnosti uspješnih kandidata i koraka pretraživanja. Srednja vrijednost distribucije ažurira se tako da je vjerojatnost prethodno uspješnih rješenja kandidata maksimizirana. Matrica kovarijacije distribucije ažurira se (postupno) tako da se povećava vjerojatnost prethodno uspješnih koraka pretraživanja [14].
2. Bilježe se dvije putanje vremenske evolucije distribucijske sredine strategije koja se nazivaju putevi pretraživanja ili evolucije. Ti putevi sadrže značajne informacije o korelaciji između uzastopnih koraka. Konkretno, ako se poduzmu uzastopni koraci u sličnom smjeru, putevi evolucije postaju dugi. Putevi evolucije se iskorištavaju na dva načina. Jedan put se koristi za postupak prilagodbe matrice kovarijance umjesto pojedinačnih uspješnih koraka pretraživanja i olakšava moguće puno brže povećanje varijance u povoljnim smjerovima. Drugi put se koristi za provođenje dodatne kontrole veličine koraka. Kontrola veličine koraka učinkovito sprječava preranu konvergenciju, a istovremeno omogućuje brzu konvergenciju do optimalne vrijednosti [14].




*Slika 4: Tijek traženja optimalne vrijednosti genetskim algoritmom. Izvor: [14]*

Na Slika 4 crne točke su jedinke gdje svaka jedinka ima 2 gena X i Y koji su parametri objektivne funkcije. Svjetlina plave boje opisuje optimalno rješenje. Narančasta linija je distribucija populacije.

U početku populacija je inicijalizirana nasumično, ali tako da jedinke slijede Gaussovu distribuciju. Sljedeće, odabere se najboljih 20% jedinka. Izračuna se srednja vrijednost i modificira se standardna distribucija nad najboljim jedinkama. Stvori se novi set jedinki iz te distribucije i odredi im se sposobnost.

Ovaj algoritam radi dobro u puno različitih situacija. U loše uvjetovanim, neodvojivim, ne konveksnim, više modalnim (više optimalnih rješenja) i šumnim funkcijama [15, p. 7].

Prednosti:

- Sposobnost konteksta trenutne okoline što znatno ubrzava pronalaženje optimuma
- Radi dobro nad različitim funkcijama velikih prostora pretraživanja
- Izvršavanje se može paralelizirati unutar jedne generacije

Nedostaci:

- Sporiji nad funkcijama manjih dimenzija (< 5)
- Sporiji nad jednostavnim funkcijama

### 2.4.4. Bayesova optimizacija (BO)

Bayesovsko gledište o vjerojatnosti povezano je sa stupnjem vjerovanja. To je mjera uvjerljivosti događaja ako potpuno znanje nije poznato. Bazirano je na tome da se može više saznati iz nekog sustava nego što je sadržano u podacima iz jednog eksperimenta. Bayesove metode se mogu koristiti da se spoje rezultati različitih eksperimenata.

Bazirane su na Bayesovom teoremu koji opisuje vjerojatnost događaja na temelju prethodnog znanja o uvjetima koji bi mogli biti povezani s događajem [16]. I on glasi:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Gdje su A i B događaji, $P(B) \neq 0$

$P(A \mid B)$ – vjerojatnost događaja A ako se dogodio događaj B

$P(B \mid A)$ – vjerojatnost događaja B ako se dogodio događaj A

$P(A), P(B)$ – vjerojatnost događaja A i B bez ikakvih uvjeta

Bayesovski pristupi prate prethodne rezultate evaluacije koji se koriste za formiranje probabilističkog modela transformiranjem parametara u vjerojatnost rezultata objektivne funkcije. Algoritam se sastoji od zamjenske i akvizicijske funkcije.

### Zamjenska funkcija

Funkcija koja aproksimira objektivnu funkciju. U literaturi se ovaj model naziva "surogat" i predstavlja se kao $\mathrm{P}(\mathrm{y} \mid \mathrm{x})$ (vjerojatnost izlaza objektivne funkcije y ako je poznat njen ulaz x). Ona je dizajnirana tako da je matematički jeftinija za izračunati, pa ju je stoga puno lakše optimizirati nego objektivnu. Bayesove metode rade tako da pronađu sljedeći skup parametara za procjenu stvarne ciljne funkcije odabirom parametara koji najbolje djeluju na zamjenskoj funkciji. Koristeći zamjensku funkciju, nije potreban nagib (derivacija) objektivne, a ipak se može procijeniti gdje se ona kreće.

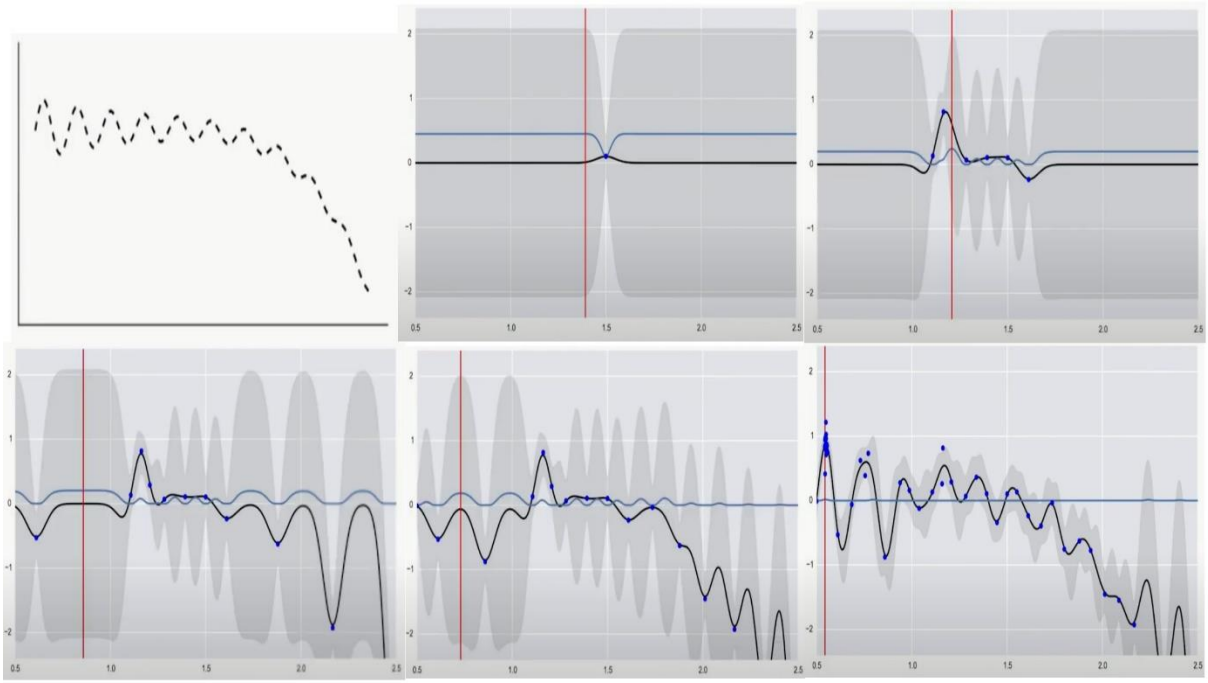
Postoje različite zamjenske funkcije od kojih su najznačajnije: Gaussovi processi, regresija slučajnom šumom (engl. Random forest regression) i TPE (engl. Tree-structured Parzen estimator) [17].

### Akvizicijska funkcija

Ažurira zamjensku funkciju i odgovorna je za predlaganje novih točaka za testiranje. Pokušava balansirati istraživanje novih i nepoznatih područja s područjima koje se već zna. Ako se više istražuje nepoznata područja, algoritam će biti sigurniji da je našao globalni optimum, ali za to je potrebno više testiranja objektivne funkcije. Na drugom kraju, ako je kut rasta vrijednosti funkcije na trenutnom području velik, algoritam će ići u smjeru rasta vrijednosti i tako brže doći do optimuma. U tom slučaju postoji velika vjerojatnost da je pronađen lokalni optimum, a ne globalni.

Uobičajene akvizicijske funkcije uključuju očekivano poboljšanje i maksimalnu vjerojatnost poboljšanja, sve koje mjere vjerojatnost da će se određeni uzorak isplatiti u budućnosti [17].

**Proces optimizacije**



*Slika 5: Proces pronalaska optimuma koristeći Bayesovu optimizaciju. Izvor: [18]*

Na prvom prikazu je izgled nepoznate (objektivne) funkcije. Na ostalim prikazima su vrijednosti funkcija nakon nekoliko iteracija. Crna linija prikazuje vrijednost zamjenske funkcije, plava linija akvizicijsku funkciju, a crvena linija sljedeću vrijednost koja će se evaluirati u objektivnoj funkciji. Zatamnjeni prostor je predviđena vrijednost nepoznate funkcije. Plave točke su evaluirane vrijednosti.

U početku se evaluira nekoliko nasumičnih vrijednosti da se istraži prostor. Koristeći te uzorke kreira se aproksimacija (surogat) objektivne funkcije (crna linija). Pretražuje se maksimum akvizicijske funkcije i koristi se njen ulaz za evaluaciju sljedeće točke (crvena linija). S pravom vrijednosti objektivne funkcije poboljšamo aproksimaciju. Optimizacija je gotova kada je izvršeno nekoliko iteracija ili je prošlo određeno vrijeme.

Prednosti:

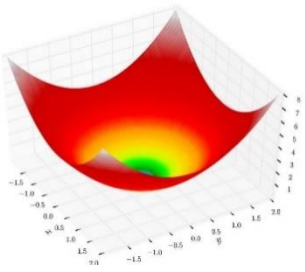
- Algoritam je točno dizajniran za pronalazak optimuma i stoga treba jako malo uzorka

Nedostaci:

- Za velike prostore se sporo izvršava (jer je potrebno pronaći maksimum akvizicijske funkcije). Da bi se to ublažilo preporučuje se koristiti nad funkcijama kojima su skupe za izračunati.
- Sekvencijalan slijed izvršavanja algoritma

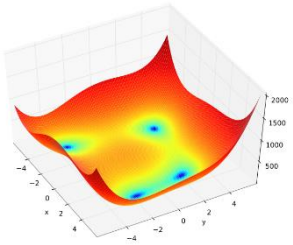
## 2.5. Usporedba različitih algoritama

Ovaj pokus nije osobno izvršen nego je oslonjen na rezultatima iz vanjskog izvora [19]. Optimizacijski algoritmi će biti uspoređeni nad 3 testne matematičke funkcije. Ako se koriste više različitih funkcija dobiti će se dovoljno dobra okvirna slika o tome koja optimizacija je najbolja.



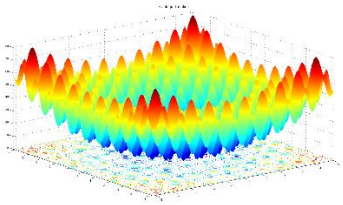
$$f(x) = \sum_{i=1}^{n} x_i^2$$

*Slika 6: Sphere funkcija i njezina jednadžba. Izvor* [20]



$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$$

*Slika 7: Himmelblau funkcija i njezina jednadžba. Izvor* [20]



$$f(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)]$$

*Slika 8: Rastrigin funkcija i njezina jednadžba. Izvor* [20]

Svaki algoritam će se izvršiti nekoliko tisuća puta dok se rezultati ne stabiliziraju. Algoritmi koji zahtijevaju znanje o rasporedu mreže (rešetkasto pretraživanje i Bayesova optimizacija) će nasumično dobiti mrežu koja ima različitu širinu područja i različite brojeve dijelova.

Algoritmi će se rangirati po sljedećim vrijednostima:

- Stopa uspješnosti (SU): Vjerojatnost pronalaska optimalnog rješenja [19].
- Važnost prethodnog uvjerenja (VPU): Utjecaj našeg prethodnog uvjerenja na stopu uspješnosti algoritma. Ova se vrijednost izračunava kao varijanca vjerojatnosti uspjeha u svim prethodnim testovima. Za svaki test, algoritam će imati nasumičnu konfiguraciju, stoga viša vrijednost podrazumijeva veću ovisnost algoritma o korisniku [19].
- Prosječan broj uzorka objektivne funkcije (PBUOF): Prosječan broj evaluacija da bi se našlo uspješno rješenje. Opisuje brzinu pronalaska optimalnog rješenja [19].

| Funkcija | Algoritam | SU | VPU | PBUOF |
|---|---|---|---|---|
| Sphere | Rešetkasto pretraživanje | 0.5% | 0.4 | 1720.9 |
| Sphere | Nasumično pretraživanje | 4.6% | 1.6 | 1944.9 |
| Sphere | Genetski algoritam | 99.8% | 26.6 | 213.2 |
| Sphere | Bayesova optimizacija | 99.9% | 0.1 | 29.3 |
| Himmelblau | Rešetkasto pretraživanje | 0.0% | 0.0 | 1765.1 |
| Himmelblau | Nasumično pretraživanje | 0.1% | 0.1 | 1917.0 |
| Himmelblau | Genetski algoritam | 99.7% | 60.4 | 554.1 |
| Himmelblau | Bayesova optimizacija | 99.9% | 0.1 | 53.5 |
| Rastrigin | Rešetkasto pretraživanje | 25.5% | 14.0 | 1766.0 |
| Rastrigin | Nasumično pretraživanje | 42.1% | 57.8 | 1178.1 |
| Rastrigin | Genetski algoritam | 96.2% | 40.7 | 1045.9 |
| Rastrigin | Bayesova optimizacija | 87.5% | 42.4 | 214.4 |

*Tablica 2: Rezultati testiranja optimizacijskih algoritama. Izvor [19]*

Iz gore navedene tablice (Tablica 2) se zaključuje:

- Nasumično pretraživanje pronalazi bolji optimum naprema rešetkastog algoritma
- GA ima najveću točnost i najviše ovisi o početnoj konfiguraciji
- Na svim funkcijama Bayesova optimizacija pronalazi optimalno rješenje s najmanjim brojem uzorka i gubi točnost na težim (rastrigin) funkcijama

# 3. Primjena optimizacije na trgovačke strategije

Najbolje je napraviti test gdje se koristi prava strategija kako bi se ustanovilo koji optimizacijski algoritam je najbolji za trgovačke strategije. Iz prijašnjih rezultata predosjećaj je na genetske algoritme.

## 3.1. Svojstva trgovačkih strategija

Prije biranja najboljeg optimizacijskog algoritma za trgovačke strategije važno je imati na umu kako izgleda objektivna funkcija i koja su njena svojstva. Prikazom se utvrđuje da li postoji neko zajedničko svojstvo (ili više njih) za većinu trgovačkih strategija. Za prikaz je korišteno detaljno rešetkasto pretraživanje i standardnu trgovačku strategiju koja koristi RSI indikator [21]. Pseudokôd te strategije glasi od prilike ovako:

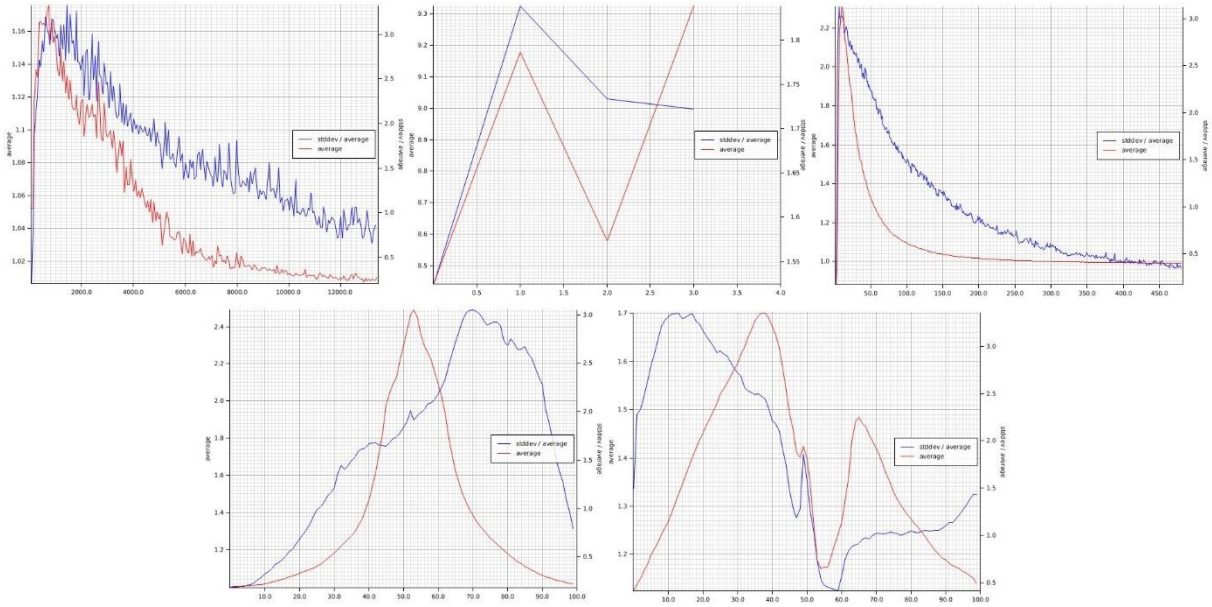
```
prijasnja_vrijednost_rsi = rsi(ulazne_tocke[0], param3);
for (i = 1; i < broj_ulaznih_tocaka; i++) {
      vrijednost_rsi = rsi(ulazne_tocke[i], param3);
      if (vrijednost_rsi prelazi preko param4 s nize na vise)
            kupi();
      else if(vrijednost_rsi prelazi preko param5 s vise na nize)
            prodaj();
      else if (trenutna cijena manja za 1% od kupljene)
            prodaj();
      else if (trenutna cijena veća za 1% od kupljene)
            kupi();
      prijasnja_vrijednost_rsi = vrijednost_rsi;
}
```

*Programski isječak 1: Pseudokôd RSI strategije*

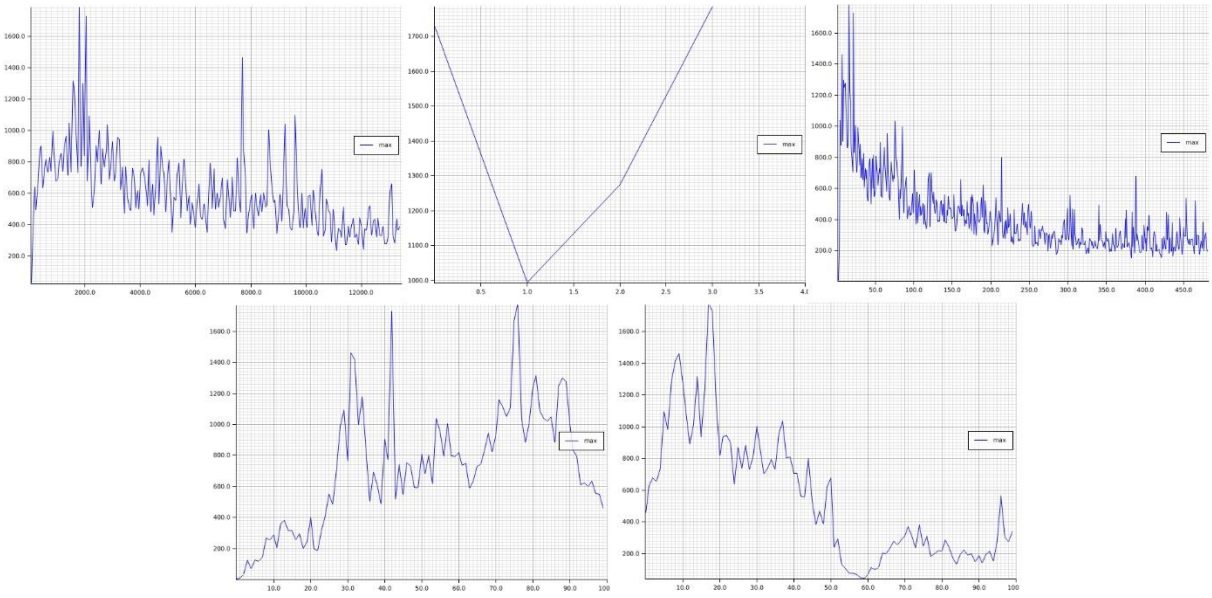
Strategija se sastoji od 5 parametra (dimenzija):

1) Vremenski okvir (engl. timeframe)
2) Tip ulaznih točaka – koji dio svijeća se uzima za izračun indikatora
3) Duljina RSI indikatora

4) Niži nivo RSI vrijednosti
5) Viši nivo RSI vrijednosti




*Slika 9: Prikaz krajnje bilance za svaku dimenziju nakon testiranja trgovačke strategije, crvena linija - srednja vrijednost, plava linija - standardna devijacija / srednja vrijednost, dimenzija raste s lijeva na desno po redu*




*Slika 10: Prikaz maksimalne krajnje bilance za svaku dimenziju nakon testiranja trgovačke strategije, dimenzija raste s lijeva na desno po redu*

Iz rezultata (Slika 10 i Slika 12) je vidljivo:

- Maksimum nije uvijek u sredini prosječne vrijednosti. Funkciju je teže predvidjeti jer optimalna rješenja nisu grupirana.

- Kod većina dimenzija srednja vrijednost ima jednostavnu distribuciju. Zbog toga je lakše predvidjeti smjer globalnog optimuma u odnosu na bilo koju točku.
- Šum je veći pri manjim dimenzijama.
- Maksimalne vrijednosti imaju veliki šum.

S toga, okvirni raspon vrijednosti za svaki parametar može biti:

1) Vremenski okvir od 1min do 60min.
2) Tip ulaza za RSI indikator nema baš veliki raspon te se može ignorirati.
3) Duljina RSI indikatora od 2 do 100, što je broj veći to je bilanca na kraju manja.
4) Od 0 do 100
5) Od 0 do 100

Zadnja dva parametra su specifični za RSI i ne mogu se primijeniti na druge strategije.

### 3.2. Memorijska svojstva

Svaka simulacija treba imati barem visoku, nisku i zadnju cijenu svijeća. Visoku i nisku za računanje rizika preko najmanje trenutne bilance, a zadnju za računanje razmjene kojoj je potrebna trenutna cijena. Simulacija se sastoji i od nekoliko indikatora. U nastavku slijedi izračun potrebne memorije za RSI strategiju koja se simulira vremenskim okvirom od jedne minute kroz pet godina.

Pet godina ima $60 \times 24 \times 365 \times 5 = 2\,628\,000$ minuta. Za svaki tip svijeće se upotrebljava broj s putujućom točkom od 4 bajta. Potrebno 4 broja po svijeći (visoka, niska, zadnja i RSI vrijednost). S toga je nužno $2\,628\,000 \times 4 \times 4 = 42\,048\,000\,B = 0.039\,GiB$ memorije. Za jednu simulaciju je dovoljno. No što ako se simulacije paralelno izvršavaju? Rešetkastim pretraživanjem je moguće paralelizirati zadnja 2 parametra bez povećane memorijske složenosti. Za genetske algoritme je prostor premali da bi bili efektivni. Iz prijašnjih rezultata genetski algoritmi su najbolja opcija i prema njima će biti dizajniran algoritam.

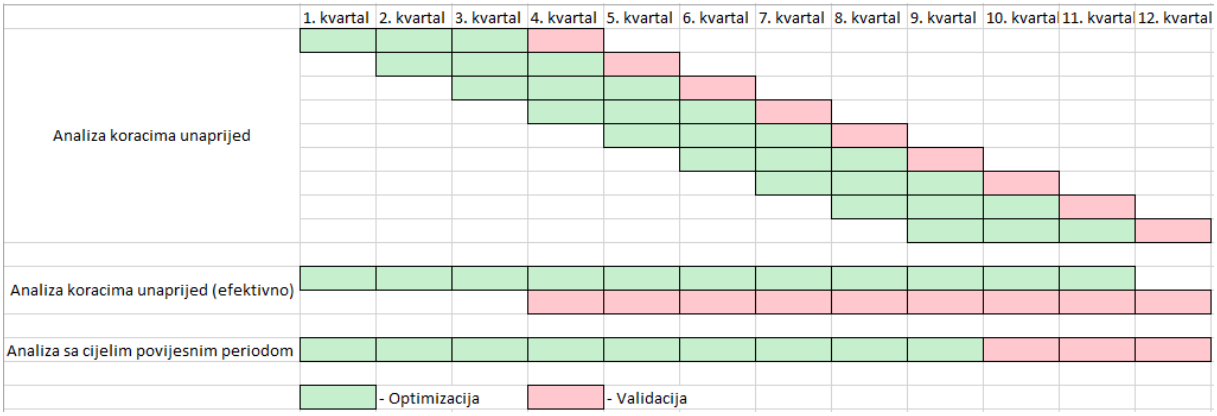
Svi parametri trebaju biti učitani unaprijed jer je nepoznato koji će parametar biti izabran genetskim algoritmom. Algoritam zahtjeva $2\,628\,000 \times 4 \times (98 + 3) =$

$0.989\ GiB$ za sve vrijednosti indikatora na vremenskom okviru od jedne minute. Ako se poveća vremenski okvir, broj svijeća se smanji. Faktor povećanja zbog vremenskog okvira koji se proteže od jedne minute do 60 minuta je $\sum_{i=1}^{60} \frac{1}{i} = 4.66$. Da bi se učitali svi indikatori na svim vremenskim okvirima potrebno je $0.989\ GiB \times 4.66 = 4.608\ GiB$.

Simulacije nad zadnjem vremenskom okviru će biti najkraće i imat će najmanje razmjena. Kako je manje uzorka, sigurnost da će se ta strategija izvršiti u stvarnom svijetu onako kako je simuliralo je manja. Preporučuje se nadopuna svijeća iz nekog drugog sličnog tržišta da bi točnost bila bolja. U tom slučaju je potrebno $0.989\ GiB \times 60 = 59.32\ GiB$. Toliko radne memorije imaju rijetka konvencionalna računala.

### 3.3. Analiza koracima unaprijed (engl. Walk forward analysis)

Analiza koracima unaprijed pomaže u smanjenju okupljenosti RAM. Umjesto da se optimizacija vrši nad cijelom povijesnom periodu, on se podijeli na više manjih dijelova, npr. jedna godina. Nakon što se izvrši optimizacija, slijedi testiranje strategije s najboljim parametrima na još kraćem vremenskom razdoblju, npr. jedan kvartal. Ako se strategija ponaša u zadovoljenim uvjetima, npr. profit nije manji od 50% u odnosu na optimizirani period, onda se smatra da je strategija zadovoljavajuća. Taj ciklus se ponavlja sve dok se ne iskoriste sve povijesne podatke.




*Slika 11: Ilustracija analize koracima unaprijed.*

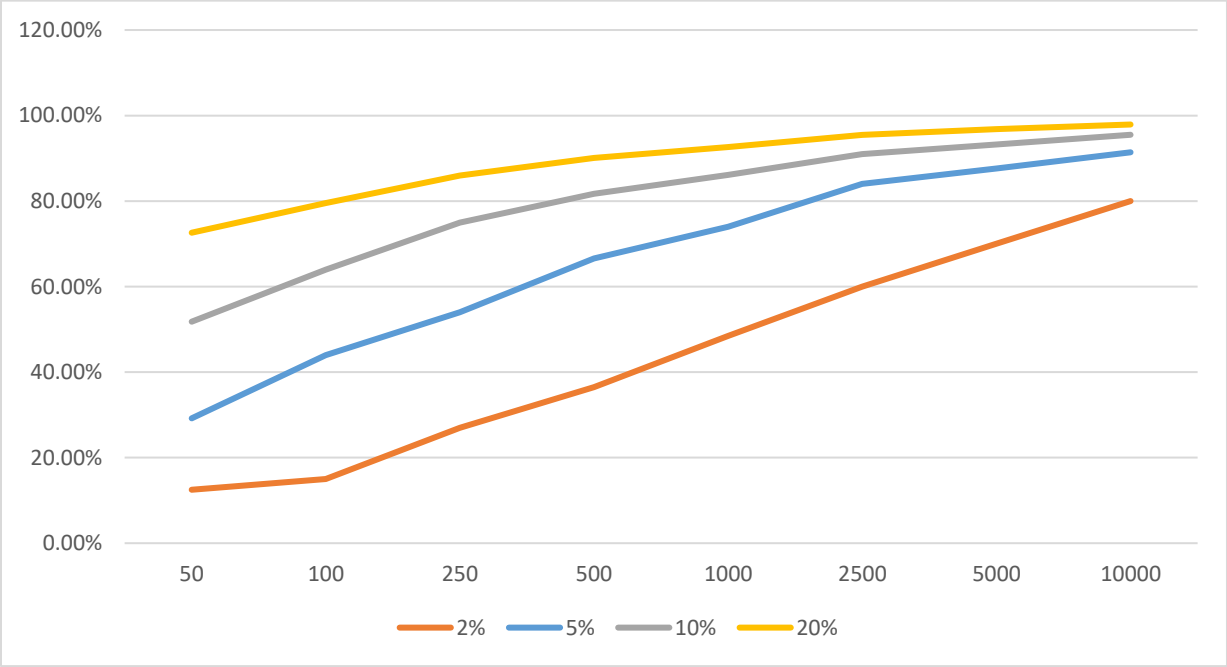
Prednost ove analize je da se model može prilagoditi trenutnoj situaciji na tržištu. Za optimizaciju se koristi jedna godina umjesto pet godina. Još jedna prednost je to što se model testira na više nepoznatim podacima nego bez ove metode. Na (Slika 11)

se vidi da ako se ne koristi metoda, za optimizaciju treba 9 ćelija, a za validaciju 3 ćelije. No, korištenjem ove metode efektivno se ispostavlja da se za optimizaciju koristi 11 ćelija, a za validaciju 9 od ukupno 12 ćelija.

### 3.4. Određivanje potrebnog broja uzoraka

Treba imati na umu da se optimizacijom aproksimira stvarni svijet. Optimizacija ne treba točno pratiti povijesno razvijanje cijene jer ne postoji dovoljno mjesta ili je premalo podataka. Strategija će tako biti generalizirana, radit će prosječno u većini slučajeva, nego odlično u samo jednom sve dok na kraju ne završi užasno.

Napravit će se još jedan eksperiment. Monte Carlo simulacija koja će imati 100 jednostavnih strategija. Svaka strategija će imati statističku prednost od -10% do +10% s razmakom od 0.2% ne uključujući nulu, ovisno o njenom indeksu. Svaka strategija će imati određenu vjerojatnost da ostvari profit. U obzir će se uzeti od 50 pa do 10 000 uzorka. Nakon simulacije strategije su rangirane po profitu gdje će prva imati najveći profit, a zadnja najlošiji. Zapisuje se postotak profita strategije s najboljom statističkom prednošću (10%) naprema trenutno najbolje strategije. Ta simulacija se ponovi 50 puta i izračuna srednja vrijednost.




*Slika 12: Prikaz postotka profita strategije s najboljom statističkom prednošću od simulirane najbolje strategije. Različite linije prikazuju za različite vrijednosti statističke prednosti. X os je broj uzorka. Izvor [22]*

Iz slike (Slika 12) je očito da što je manja prednost to je potrebno više uzorka. U statistici se najčešće uzima povjerenje od 90%.

Algoritam optimizacije bi mogao raditi u ovom smislu. Ako se pretražuje široko područje, povjerenje je smanjeno i zapamte se različiti rezultati koji su blizu jedni drugog. Nakon toga povjerenje se poveća i ponovno se pretražuje područje oko tih grupa rezultata.

### 3.5. Memorijska optimizacija

Vrijednost svijeća i indikatora kroz povijest se ne mijenjaju. Stoga se preporučuje skladištenje tih podataka.

Primjenom analize koracima unaprijed se učitava manji povijesni period. Tu postoji pitanje koliki povijesni period je dovoljan. U statistici ljudi preporučaju 30 ili 100 uzorka ovisno o tipu istraživanja [23]. Nagađanje nije baš najbolja opcija. Stoga se koristi prethodno objašnjena metoda.

Još jedan način je veća iskoristivost radne memorije. U simulaciju se uključe podaci za različite vremenske okvire koji su već učitani u RAM. Pritom, prvo se uzmu u obzir podaci najbližih vremenskih okvira originalnom. Vremenski okvir od 59 minuta je vrlo sličan onom od 60 min.

Na primjer. Pet godina će biti skraćeno na jednu godinu ($59.32\ GiB$ / 5 = 11.87 GiB). Jedna optimizacija će se podijeliti na 3 različita vremenska okvira (11.87 GiB / 3 = 3.95 GiB). Ako se u prosjeku razmjena dogodi svakih 1000 svijeća. Efektivno, povjerenje iznosi $\frac{3}{5} \times \frac{2\,628\,000}{1000} \approx 1577\ uzorka \approx 50\%$ za statističku prednost od 2%.

## 4. Konfiguracija i rezultati pokusa

Algoritmi koji će se analizirati su: rešetkasto pretraživanje, nasumično, genetski algoritam (CMA-ES), Bayesova optimizacija (Simple(x) algoritam) i dva hibridna algoritma.

Jedan od hibridnih koristi Bayesovu optimizaciju za vremenski okvir, a za ostale parametre genetski algoritam. Nad svakim parametrom vremenskog okvira će se izvršiti cijela optimizacija genetskim algoritmom. Tim algoritmom se drastično smanjuje memorijska složenost (60 puta manje u ovom slučaju). Neka se zove GA + BO.

Drugi algoritam će razlomiti područje pretraživanja na parametru za vremenski okvir. Genetska optimizacija će se izvršiti 10 puta gdje svaka optimizacija koristi raspon 6 vrijednosti parametra vremenskog okvira. Ovim načinom se koristi samo GA. Ima veću memorijsku složenost nego GA + BO, no možda je algoritam efektivniji jer ima manje invokacija genetske optimizacije uz veći raspon vremenskih okvira od prijašnjeg algoritma. Neka se taj algoritam zove podijeljeni GA.

Simulacija će predmemorirati vrijednosti svijeća za sve vremenske okvire. Neki algoritmi će i predmemorirati vrijednosti indikatora ako je to moguće. (rešetkasto pretraživanje).

Efikasnost će se izračunati tako da će se uzeti maksimalna pronađena bilanca strategije na kraju simulacije, ona se podijeli sa stvarnim maksimumom područja i s proteklim vremenom. Sve simulacije su izvršene na jednoj jezgri i5-3320m Intel procesora.

Usporedit će se i brzina algoritma ako se paralelno izvrši na 1000 jezgri procesora. Efikasnost se u tom slučaju podijeli brojem maksimalnih mogućih paralelnih izvršavanja odjednom imajući na umu dizajn algoritma. I u slučaju ako je u potpunosti moguće predmemorirati memoriju za indikatore i svijeće na svim vremenskim okvirima. To nije moguće na osobnim računalima zbog memorijske složenosti pa će se oduzeti vrijeme potrebno za izračunavanje tih podataka.

Algoritmi bazirani na BO će se testirati na 10, 15 i 30 uzorka vremenskog okvira od ukupno 60. Algoritmi bazirani na GA će se testirati s populacijom od 10, 100, 1000 i

brojem generacija od 10, 100, 1000. Ako CMA-ES ustanovi da ne može pronaći maksimum onda će biti i manji broj generacija.

### 4.1. Implementacija

Pokus je implementiran u programskom jeziku Rust. Značajne biblioteke koje se koriste su:

- „cmaes v0.2.1“ – implementacija CMA-ES algoritma [24]
- „simplers_optimization v0.4.3“ – implementacija Simple(x) algoritma [25]
- „plotters v0.31“ – biblioteka za vizualizaciju podataka, s njome je generirana slika 9 i 10 [26]

Simulacija će se izvršiti nad svijećama Bitcoin tržišta BitMEX burze (XBTUSD) od 25.9.2015 do 10.10.2021. Podaci su skinuti direktno sa BitMEX burze kao razmjene koristeći ručno izrađen REST API klijent (vidljivo u „nebuchadnezzar“ biblioteci). Razmjene se pretvaraju u svijeće nad vremenskim periodom od jedne sekunde i spremaju se na hard disk (vidljivo u „load_hlcv“ modulu). Koristi se prethodna RSI strategija samo s jednim tipom ulaza (zadnja vrijednost svijeće).

```
// Definiranje područja pretraživanja.
const MIN_TIMEFRAME: u32 = 60;
const TIMEFRAME_RANGE: u32 = 60;
const TIMEFRAME_STEP: u32 = 60;
const MAX_TIMEFRAME: u32 = MIN_TIMEFRAME + TIMEFRAME_RANGE *
TIMEFRAME_STEP;
const MAX_LEN: u32 = 100;
const MAX_HLINE: u32 = 101;
const MAX_LLINE: u32 = 101;
```

*Programski isječak 2: Definicija područja pretraživanja.*

Vremenski okvir je definiran minimalnim korakom od 60 sekundi. Prostor je definiran prema vrijednostima iz 19. stranice.

```rust
pub struct Context {
    // Predmemorija svijeća.
    pub timeframe_cache: RwLock<HashMap<u32, Arc<Vec<Hlcv>>>>,
    // Daje mogućnost isključenja predmemorije.
    pub cache_timeframe: bool,
    // Vrijeme početka svijeća.
    pub hlcvs_start_ts: u32,
    // Koliko maksimalno cijena smije pasti nakon kupnje. Ako padne ispod
    // onda se prodaje.
    pub max_risk: f32,
    // Ukupni broj proteklih mikrosekundi prilikom izračunavanja razmjena.
    pub micros_compute: AtomicU64,
    // Ukupni broj proteklih mikrosekundi prilikom izračunavanja
    // indikatora.
    pub micros_cache: AtomicU64,
    // Ukupni broj proteklih mikrosekundi prilikom izračunavanja svijeća.
    pub micros_timeframe: AtomicU64,
    // Da li se ignorira vremenski okvir prilikom izvšavanja genetskog
    // algoritma.
    pub hybrid: bool,
    // Trenutni vremenski okvir GA + BO algoritma.
    pub timeframe: AtomicU32,
    // Broj dijelova podijeljenog GA.
    pub n_shards: usize,
    // Trenutna podjela, služi za korektno limitiranje područja
    // pretraživanja vremenskog okvira.
    pub shard_i: usize,
    // Top 10 najboljih parametara.
    pub top: Mutex<Vec<Pair>>,
}
```

*Programski isječak 3: Prikaz definicije područja pretraživanja i „Context“ strukture.*

Simulacija se izvršava metodom „backtest“ od „Context“ strukture u „context“ modulu. Svijeće se čitaju sekvencijski, trenutna cijena je zadnja cijena svijeće. Kupnja ili prodaja se uvijek izvršava sa svim mogućim novcem. Ako nije otvorena pozicija (nije ništa trenutno kupljeno) onda se provjerava uvjet RSI indikatora. Ako je zadovoljen, kupuje se sa svim mogućim novcem. Prodaje se ako je dozvoljen uvjet RSI indikatora ili ako je minimalna cijena ispod 99% od cijene kada je kupljeno. Strategija radi i u obrnutom slučaju gdje može prodati ako nema ništa kupljeno. U stvarnom svijetu se to zove kratka prodaja (engl. short selling) gdje se prvo posude dionice od nekoga na burzi pa se prodaju na tržištu, kasnije se kupe po nižoj cijeni i vrate natrag vlasniku.

```rust
pub fn update(
    &mut self,
    hlcvs: &[Hlcv],
    i: usize,
    prev_close: f32,
    prev_rsi: f32,
    rsi: f32,
    max_risk: f32,
    hline: f32,
    lline: f32,
) {
    let hlcv = unsafe { hlcvs.get_unchecked(i) };
    if hlcv.close > prev_close {
        let hline_condition = prev_rsi < hline && rsi >= hline;
        let stop_price = self.entry_price * -(1. + max_risk);
        let stop_short = self.short_stopped(stop_price, hlcv.high);
        let open_short = self.entry_price == 0. && hline_condition;
        let close_long = self.entry_price > 0. && hline_condition;
        let close_position = close_long | stop_short;
        self.market_open_short(open_short, hlcv.close);
        self.market_close_long(close_long, hlcv.close);
        self.closed_position(close_position, i);
    } else if hlcv.close < prev_close {
        let lline_condition = prev_rsi >= lline && rsi < lline;
        let stop_price = self.entry_price * (1. - max_risk);
        let stop_long = self.long_stopped(stop_price, hlcv.low);
        let open_long = self.entry_price == 0. && lline_condition;
        let close_short = self.entry_price < 0. && lline_condition;
        let close_position = close_short | stop_long;
        self.market_open_long(open_long, hlcv.close);
        self.market_close_short(close_short, hlcv.close);
        self.closed_position(close_position, i);
    } else {
        let stop_short_price = self.entry_price * -(1. + max_risk);
        let stop_long_price = self.entry_price * (1. - max_risk);
        let stop_short = self.short_stopped(stop_short_price, hlcv.high);
        let stop_long = self.long_stopped(stop_long_price, hlcv.low);
        let close_position = stop_long | stop_short;
        self.closed_position(close_position, i);
    }
}
```

*Programski isječak 4: Metoda „update“ „Account“ strukture. Služi za izračunavanje bilance tijekom kupnje ili prodaje.*

Funkcija „update“ je malo drugačije implementirana nego kako je gore rečeno. Razlog tomu je da se brže izvrši.

Rešetkasto i nasumično pretraživanje neće biti prikazano jer su trivijalni za napraviti.

```rust
pub fn bayes_search(&self) {
    // Objektivna funkcija.
    let f = |x: &[f64]| {
        let timeframe = MIN_TIMEFRAME + TIMEFRAME_STEP * x[0] as u32;
        let len = x[1] as usize;
        let hline = x[2] as u32 as f32;
        let lline = x[3] as u32 as f32;
        let account = self.backtest(timeframe, len, hline, lline);
        account.balance
    };
    // Definiranje područja pretraživanja.
    let input_interval = vec![
        (0., TIMEFRAME_RANGE as f64),
        (2., MAX_LEN as f64),
        (0., MAX_HLINE as f64),
        (0., MAX_LLINE as f64),
    ];
    // Broj iteracija optimizatora.
    let nb_iterations = 100;

    // Pokretanje optimizatora.
    Optimizer::new(&f, &input_interval, false)
        // "Optimizer" implementira "Iterator" "trait" te se preko njega
        // pokreće.
        // Evaluiranje "nb_iterations" uzoraka.
        .skip(nb_iterations)
        // Traži se zadnja vrijednost kako bi se iterator izvršio.
        .next()
        .unwrap();
}
```

*Programski isječak 5: Prikaz programskog isječka koji pokreće Bayesovu optimizaciju.*

Bayesov optimizator koristi klauzulu (engl. closure) umjesto osobine. Korištenjem klauzule programski isječak je pregledniji. Pokretanje optimizatora je zbunjujuće jer se koristi iterator. Sučelje iteratora nije baš namijenjeno za optimizatore. Najbolje bi bilo da su medote.

Funkcija „Optimizer::new“ prima klauzulu, točno definirani prostor pretraživanja i da li će pronalaziti minimum ili maksimum.

```rust
// Implementacija osobine koja je potrebna za izvršavanje genetskog
// algoritma.
impl ObjectiveFunction for &Context {
    fn evaluate(&mut self, x: &DVector<f64>) -> f64 {
        // Pozivanje parallelne implementacije.
        ParallelObjectiveFunction::evaluate_parallel(self, x)
    }
}
```

```rust
pub fn genetic_search(&self) {
    let mut param_count = 4;
    // Ako se koristi GA + BO algoritam.
    if self.hybrid {
        param_count = 3;
    }
    let first_guess = vec![0.5; param_count];
    // Konfiguracija broja populacije i generacije.
    let pop = 1000;
    let gen = 1000;
    println!("{pop}pop {gen}gen");
    // Konfiguriranje CMA-ES algoritma.
    let mut cmaes_state = CMAESOptions::new(first_guess, 0.5)
        .mode(Mode::Maximize)
        .population_size(pop)
        .max_generations(gen)
        // "self" (Context struktura) ima implementiran "trait" koji sadrži
        // funkciju koja se pokreće za svaku jedinku.
        .build(self)
        // Termininiranje programa ako se pojavi neka greška.
        .unwrap();
    // Pokretanje optimizacije.
    cmaes_state.run();
}
impl ParallelObjectiveFunction for &Context {
    fn evaluate_parallel(&self, orig: &DVector<f64>) -> f64 {
        let mut offset = 0;
        // U slučaju hibridnog načina rada se ignorira vremenski period.
        if self.hybrid {
            offset = 1;
        }
        for x in orig {
            if *x > 1. || *x < 0. {
                return 0.;
            }
        }
        // “orig” je vektor brojeva između 0.0 i 1.0 on se skalira da bi
        // stao u definirano područje pretraživanja.
        let x = convert(orig, self.n_shards, self.shard_i);

        // Učitava se potrebni vremenski okvir u sljučaju GA + BO
        // algoritma.
        let timeframe = if self.hybrid {
            self.timeframe.load(Ordering::SeqCst)
        } else {
            x[0].0 as u32
        };
        let period = x[1 - offset].0 as usize;
        let hline = x[2 - offset].0 as u32 as f32;
        let lline = x[3 - offset].0 as u32 as f32;
        let account = self.backtest(timeframe, period, hline, lline);
        account.balance as f64
    }
}
```

*Programski isječak 6: Način pokretanja i definicija osobina genetskog algoritma.*

Funkcija „CMAESOptions::new“ kao prvi parametar prima prvu pretpostavku najboljih parametara, a drugi parametar opisuje za koliko se smije proći dalje od pretpostavke. Postoji mala vjerojatnost da će broj biti veći od 1 zbog zaokruživanja brojeva. Zato se mora napraviti provjera u objektivnoj funkciji. Nakon provjere parametri se pretvore u gore navedeno područje.

Biblioteka „cmaes“ koristi dvije osobine (engl. trait), jednu za izvršavanje na jednoj jezgri procesora i drugu na više. Struktura implementira obije zbog lakšeg razvoja programa. Da bi se algoritam izvodio paralelno potrebno je pozvati "run_parallel" metodu. Osobine su slične kao sučelja (engl. interface) u drugim programskim jezicima.

Podijeljeni GA i GA + BO algoritmi su jednostavni za implementaciju. Na početku se zapiše način rada genetskog algoritma čime se neće uključiti vremenski okvir u prostor pretraživanja. U petlji se poziva genetsko pretraživanje te se zabilježe najbolji parametri kako se ne bi izbrisali prilikom sljedeće iteracije.

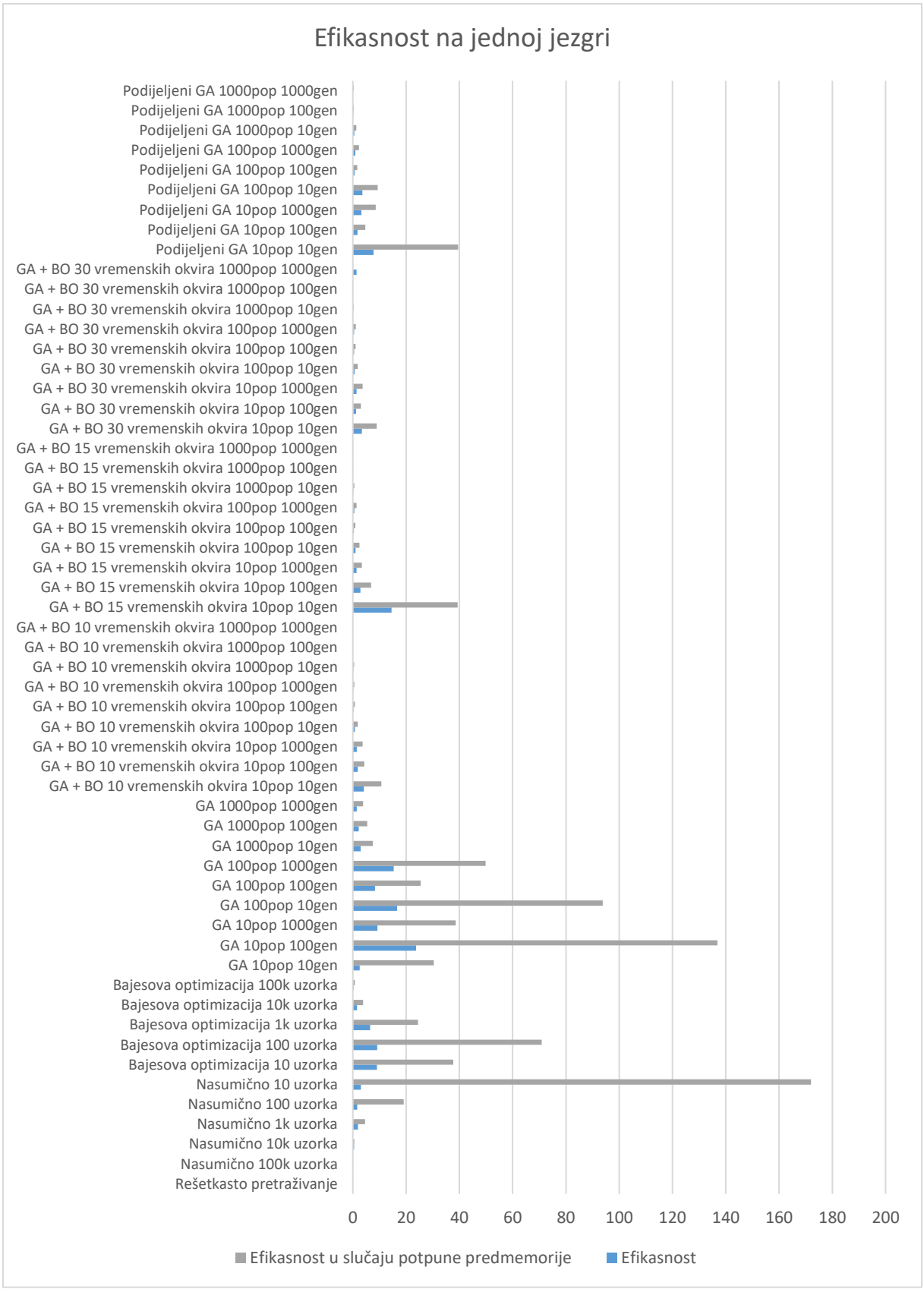
```
// . . .
let mut context = Context::new().await?;
// Omogućuje predmemoriranje svijeća.
context.cache_timeframe = true;
context.genetic_shearch();
// Ispiše rezultat u standardni izlaz.
context.summary();
context.bayes_shearch();
context.summary();
// . . .
```

*Programski isječak 7: Način pozivanja optimizacijskih algoritama.*

Za pokretanje pokusa je dovoljno ubaciti programski isječak 7 u „main“ metodu. Na standardnom izlazu će se prikazati rezultati optimizacije.

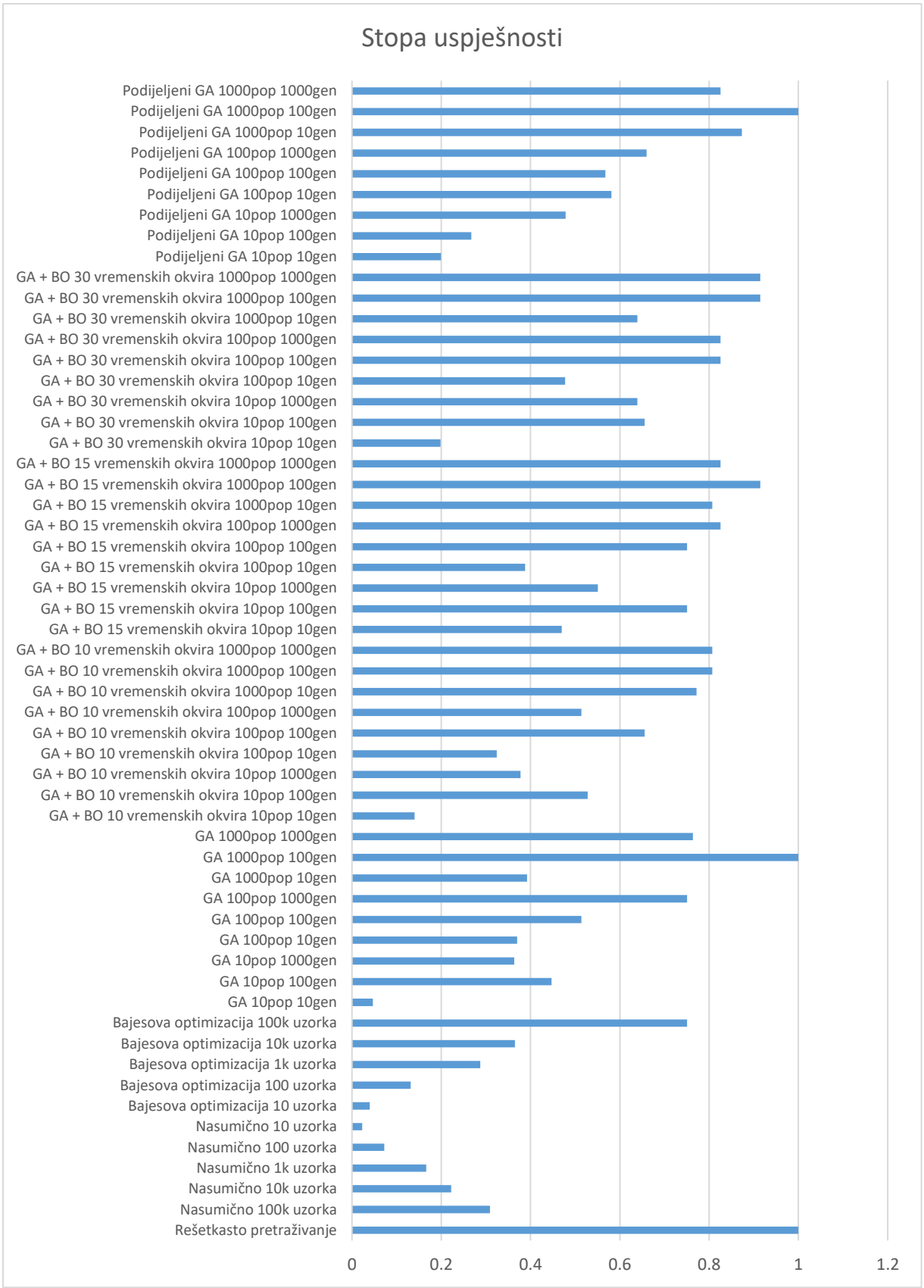
## 4.2. Rezultati pokusa

Izvršeno je ukupno 56 različitih optimizacija. Ukupno vrijeme izvršavanja svih optimizacija je 57 sati. Prikazi rezultata su na sljedećim stranicama. Detaljnije vrijednosti su vidljive u prilogu 1.

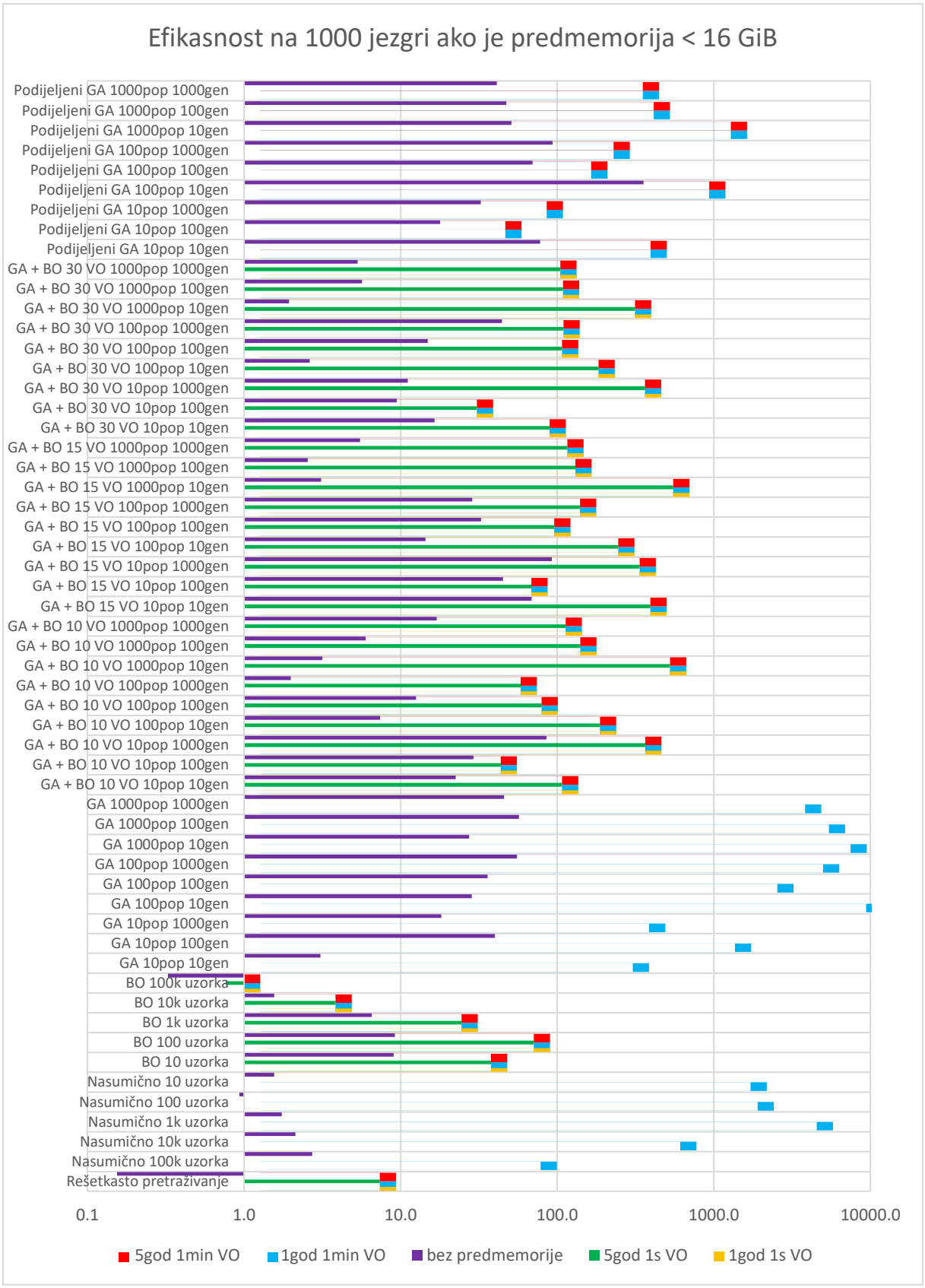



*Slika 13: Prikaz efikasnosti različitih optimizacijskih algoritama na jednoj jezgri.*


*Slika 14: Prikaz efikasnosti različitih optimizacijskih algoritama na 1000 jezgri. xgen – maksimalni broj generacija, xpop – veličina populacije, x vremenskih okvira – broj uzorka Bayesove optimizacije*




*Slika 15: Prikaz omjera maksimalne pronađene vrijednosti određenim algoritmom naprema maksimalne vrijednosti područja. xgen – maksimalni broj generacija, xpop – veličina populacije, x vremenskih okvira – broj uzorka Bayesove optimizacije*




*Slika 16: Prikaz efikasnosti algoritma na 1000 jezgri ako predmemorjia algoritma stane u 16 GiB. Na legendi prva riječ označava duljinu povijesnog perioda, a druga veličinu vremenskog okvira (VO). Za vrijednosti > 1 točna vrijednost se nalazi lijevo od pravokutnika, a za < 1 se nalazi na lijevoj strani pravokutnika odmaknuta za njegovu širinu. xgen – maksimalni broj generacija, xpop – veličina populacije, x VO – broj uzorka Bayesove optimizacije*

### 4.3. Rešetkasto pretraživanje

Primjenom guste rešetke pogodno je u one svrhe za koje je potreban globalni maksimum. No to dolazi s jako velikom cijenom jer je 3 reda veličine manje efektivniji od najboljeg algoritma.

Prilikom izrade strategije korisnik želi vrlo površno pogledati cijeli prostor pretraživanja kako bi ustanovio da li je uopće potrebno nastaviti izradu ili početi s novom idejom. Može služiti kako bi se otkrio manji prostor pretraživanja za ostale algoritme.

### 4.4. Nasumični algoritam

Svojom efikasnosti ne zaostaje puno za genetskim algoritmom. No ipak se ne bi trebao koristiti jer ima najnižu stopu uspješnosti. Jedina situacija gdje bi se koristio je prilikom razvoja nove strategije gdje je potrebno vidjeti rezultate u jako kratkom vremenskom razdoblju.

### 4.5. Bayesova optimizacija

Nije efektivniji čak ni na jednojezgrenom načinu rada. U slučaju gdje se simulacija dulje izvršava bi bila bolja jer postiže najmanji broj invokacija objektivne funkcije. Strategije imaju previše šuma da bi bile efektivne. Stopa uspješnosti je slabija od GA. Ne preporučuje se uopće koristiti čak ni kada nema dovoljno predmemorije za druge algoritme.

### 4.6. Genetski algoritam

Najefektivniji, ali zato koristi najviše memorije. Stopa uspješnosti je manja od hibridnih. Zbog velike efektivnosti preporučuje se uvijek koristiti ako predmemorija stane u RAM. U slučaju da nema dovoljno RAM može se koristiti i bez predmemorije zbog jednostavnije implementacije.

### 4.7. GA + BO

Ima drugu najbolju stopu uspješnosti. Na 3. mjestu što se tiče brzine. Podržava velike simulacije jer ima malu memorijsku složenost. Koristiti nakon podijeljenog GA ako nema dovoljno RAM.

### 4.8. Podijeljeni GA

Ima najveću stopu uspješnosti (ako se izostavi rešetkasto pretraživanje). Najefikasniji algoritam u situaciji gdje nema dovoljno memorije. Stoga se preporučuje koristiti na optimizacijama najveće memorijske složenosti. Ne preporučuje se koristiti s predmemorijom jer ima istu efikasnost kao GA + BO, a taj algoritam ima manju memorijsku složenost.

### 4.9. Moguća unaprjeđenja

Jedina mana GA je to što pronalazi maksimum koji je velik 0.75 puta od maksimuma. Veća preciznost se postiže povećanjem populacije i brojem generacija. Efikasnije je povećati broj populacije jer se može paralelizirati. Današnji procesori imaju 4 – 8 jezgri, dok grafičke kartice imaju 2000 – 4000 jezgri. Ako su podaci predmemorirani onda je veći problem propusnost memorije. Algoritam će samo provjeravati da li su zadovoljeni uvjeti za razmjenu, a razmjena se događa svakih 100 – 500 svijeća. Procesori imaju 50 GiB/s memorijske propusnosti, a grafičke 300 GiB/s. Zbog toga, optimizacija će se brže izvršiti na grafičkim karticama, bit će točnija i manje energije će se iskoristiti.

Grafičke kartice su sposobne izvršavati veći broj simulacija od broja jezgri. Svojim velikim brojem registarske memorije izvršava zamjenu konteksta skupine niti u samo jednom ciklusu. Time imaju posla dok čekaju na čitanje memorije. Preporučuje se puno veći broj populacije od broja jezgri. Neki algoritam koji koristi uzastopno prepolovljivanje. I rano završavanje jedne simulacije ako se utvrdi da je strategija jako loša s određenom stopom sigurnošću.

Profit nije uvijek najvažnija stvar. U situaciji gdje se koristi puno strategija zajedno (portfolio strategija), bitno je da su strategije što više različite od drugih. Tako će ukupna bilanca manje varirati. Različitost modela se postiže izračunavanjem

korelacije kretanja bilance. Optimizacijska vrijednost se može svesti na jednu primjenom neke matematičke funkcije, npr: $krajnja\ bilanca \times (-korelacija)$. Takva optimizacija daje jedno rješenje koja nije ujedno i optimalno, optimalnije bi možda bila drugačija funkcija koja kvadrira krajnju bilancu. Primjenom multi-objektivne optimizacije se dobiva set najboljih parametra koji zadovoljavaju neki kriterij. Npr. maksimiziraj krajnju bilancu, minimiziraj korelaciju. Nakon optimizacije korisnik vidi cijeli raspon najboljih parametra, od najvećeg profita i osrednje korelacije pa sve do osrednjeg profita i najmanje korelacije. Tako odlučuje koji parametar mu je više važan. Jedan od kandidata bi mogao biti NSGA-II algoritam [27].

## 5. Zaključak

Sistemskim trgovanjem omogućeno je testiranje velikog broja trgovačkih strategija nad povijesnim podacima. Diskretnih trgovca ima sve manje. Oni se počinju orijentirati prema statistici i programiranju.

Objašnjeno je da cijena proizlazi iz knjige ponuda i njihovih razmjena. Za spremanje trenutnog stanja knjige ponuda je potrebna ogromna količina prostora. Za velik broj trgovačkih strategija ne trebaju razmjene iz knjige ponuda. Zbog toga je predloženo vremensko grupiranje korištenjem svijeća. Većina trgovačkih strategija ne trguje unutar jedne minute pa točnost u milisekundama nije potrebna.

Kreiranjem testne trgovačke strategije i temeljitim rešetkastim pretraživanjem nje, dobiju se okvirni podatci koji se mogu generalizirati na većinu ostalih strategija. Iz tih podataka je zaključeno da globalni maksimum nije blizu maksimalne srednje vrijednosti tog parametra i da maksimalne vrijednosti imaju dosta šuma.

Pojašnjen je problem memorijske složenosti kod trgovačkih strategija. Razvijeno je nekoliko načina bolje efikasnosti memorije. Ustanovljeno je koliko je povijesnih podataka potrebno za određenu točnost simulacije.

Pokusom je utvrđeno da nijedna strategija nije najbolja. Generalno genetski algoritam je najbolji ako računalo ima dovoljno RAM. Zbog memorijske ograničenosti preporučuje se biranje algoritma ovim redoslijedom: GA, podijeljeni GA, GA + BO, GA bez predmemorije. Rešetkasti, nasumični i BO algoritmi se ne bi trebali koristiti jer postoje efikasnije varijante koje bolje mijenjaju procesorsku snagu za pronalaženje većeg profita.

Rešetkasti algoritam visoke rezolucije će jedini pronaći maksimum, drugim optimizacijskim algoritmima se efektivno gubi 25% profita zbog kraćeg vremena izvršavanja. Isto tako odabirom količine podataka za simulaciju se gubi do 20% profita.

_____

(potpis studenta)

# 6. Literatura


[1] Z. Yhang, X. Chen i D. Park, »Formal specification of constant product (xy=k) market maker model and implementation,« October 2018. [Mrežno]. Available: https://github.com/runtimeverification/verified-smart-contracts/blob/uniswap/uniswap/x-y-k.pdf. [Pokušaj pristupa 20 1 2022].

[2] H. Adams, N. Zinsmeister, M. Salem, R. Keefer i D. Robinson, »Uniswap v3 Core,« March 2021. [Mrežno]. Available: https://uniswap.org/whitepaper-v3.pdf.

[3] C. Zhao, »Performance Incident Report,« 17 2 2020. [Mrežno]. Available: https://www.binance.com/en/blog/all/performance-incident-report-421499824684900421. [Pokušaj pristupa 21 1 2022].

[4] »Samsung SSD 860 PRO 1TB Benchmark results,« PassMark®, 2 11 2021. [Mrežno]. Available: https://www.harddrivebenchmark.net/hdd.php?hdd=Samsung%20SSD%20860%20PRO%201TB&id=16079. [Pokušaj pristupa 21 1 2022].

[5] »IEEE Standard for Binary Floating-Point Arithmetic,« 1985, pp. 1-20. doi:10.1109/IEEESTD.1985.82928.

[6] Binance, »Binance API,« 21 1 2022. [Mrežno]. Available: https://binance-docs.github.io/apidocs/spot/en/#public-api-definitions. [Pokušaj pristupa 21 1 2022].

[7] A. Hayes, »Candlestick Definition,« 29 7 2020. [Mrežno]. Available: https://www.investopedia.com/terms/c/candlestick.asp. [Pokušaj pristupa 21 1 2022].

[8] »TradingView, Apple Inc.,« [Mrežno]. Available: https://tradingview.com/chart. [Pokušaj pristupa 19 8 2022].

[9] J. Rubano, Trader Construction Kit: Fundamental & Technical Analysis, Risk Management, Directional Trading, Spreads, Options, Quantitative Strategies, Execution, Position Management, Data Science & Programming, 2020.

[10] E. Chan, »Quantitative Trading: How to Build Your Own Algorithmic Trading Business,« 2009, pp. 31-66.

[11] J. Bergstra i Y. Bengio, »Random Search for Hyper-Parameter Optimization,« *Journal of Machine Learning Research,* svez. 13, br. 10, p. 281−305, 2012.

[12] N. Senaratna, »Genetic Algorithms: The Crossover-Mutation Debate,« 2005. [Mrežno]. Available: https://www.semanticscholar.org/paper/Genetic-Algorithms%3A-The-Crossover-Mutation-Debate-Senaratna/73a50124700c7b2e44e3a72a298f6279a8b54ac3. [Pokušaj pristupa 24 1 2022].

[13] N. Hansen, The CMA Evolution Strategy: A Tutorial, arXiv:1604.00772, 2016.

[14] Wikipedia, »CMA-ES,« 22 1 2022. [Mrežno]. Available: https://en.wikipedia.org/wiki/CMA-ES. [Pokušaj pristupa 24 1 2022].

[15] N. Hansen, »The CMA Evolution Strategy: A Tutorial,« arXiv:1604.00772, 2016, p. 7.

[16] M. P. Lee, Bayesian Statistics: An Introduction, 4th Edition, 2012.

[17] W. Koehrsen, »A Conceptual Explanation of Bayesian Hyperparameter Optimization for Machine Learning,« 24 6 2018. [Mrežno]. Available: https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f. [Pokušaj pristupa 24 1 2022].

[18] W. M. Hoffman, »Bayesian Optimization,« Youtube, 4 10 2018. [Mrežno]. Available: https://www.youtube.com/watch?v=C5nqEHpdyoE. [Pokušaj pristupa 24 1 2022].

[19] D. Moran, »HYPERPARAMETER OPTIMIZATION,« 3 6 2019. [Mrežno]. Available: https://www.revelock.com/en/blog/hyperparameter-optimization. [Pokušaj pristupa 24 1 2022].

[20] »Test functions for optimization,« Wikipedia, 27 6 2021. [Mrežno]. Available: https://en.wikipedia.org/wiki/Test_functions_for_optimization. [Pokušaj pristupa

24 1 2022].

[21] J. Fernando i C. Potters, »Relative Strength Index (RSI),« 30 10 2021. [Mrežno]. Available: https://www.investopedia.com/terms/r/rsi.asp. [Pokušaj pristupa 21 1 2022].

[22] T. Martyn, »2.3) Why Trading Optimizations need a Statistically Significant Sample Size (Number of Trades),« Darwinex, 23 4 2020. [Mrežno]. Available: https://www.youtube.com/watch?v=3fqyk5RUsaU. [Pokušaj pristupa 31 8 2022].

[23] e. a. Whitehead L Amy, »Estimating the sample size for a pilot randomised trial to minimise the overall trial sample size for the external pilot and main trial for a continuous outcome variable,« *SAGE Journals, doi: 10.1177/0962280215588241,* svez. 25, br. 3, 2016.

[24] S. Owen, »Crates.io: cmaes,« 18 5 2022. [Mrežno]. Available: https://crates.io/crates/cmaes. [Pokušaj pristupa 6 9 2022].

[25] D. Nestor, »Crates.io: simplers_optimization,« 26 10 2021. [Mrežno]. Available: https://crates.io/crates/simplers_optimization. [Pokušaj pristupa 6 9 2022].

[26] H. Hao, »Crates.io plotters,« 21 5 2021. [Mrežno]. Available: https://crates.io/crates/plotters. [Pokušaj pristupa 6 9 2022].

[27] K. Deb, »A fast and elitist multiobjective genetic algorithm: NSGA-II,« *IEEE Transactions on Evolutionary Computation, doi: 10.1109/4235.996017,* svez. 6, br. 2, pp. 182-197, 2002.

# 7. Prilozi

Prilog 1: Rezultati optimizacijskih algoritama za RSI trgovačku strategiju (kroz 4 stranice), bodovi = efikasnost

| | proteklo vrijeme [s] | | | | | | | | | | | | | | | | bodovi | | | | bodovi na 1000 jezg. ako predmemorija < 16 GiB | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | potrebno memorije [GiB] | | | 1 jezgra | | | | 1000 jezgri | | | | | | | | | računanje | | predmemorirano | | 1min VO | | | 1s VO | | | top 10 bilanca | | | | |
| ime algoritma | min | max | predmemorirano | simulacija | indikator | VO | ukupno | simulacija | indikator | VO | ukupno | efektivni max | 1 jezgra | 1000 jezgri | 1 jezgra | 1000 jezgri | 1god | 5god | 1god | 5god | 1god | 5god | min | prosjek | max | steddev |
| Rešetkasto pretraživanje | 0.0078 | 0.0078 | 0.0078 | 101000 | 4360 | 362.00 | 106000 | 101 | 4360 | 362 | 4820 | 100% | 0.01 | 0.15 | 0.01 | 7.37 | 7.37 | 7.37 | 7.37 | 7.37 | 7.37 | 7.37 | 596 | 634.00 | 743 | 46.10 |
| Nasumično 100k uzorka | 0.0078 | 11.9 | 11.9 | 2940 | 3350 | 78.00 | 3180 | 2.94 | 3.35 | 78.0 | 84.3 | 31% | 0.07 | 2.73 | 0.08 | 78.20 | 78.2 | 78.2 | 78.2 | 78.2 | 78.2 | 78.2 | 185 | 210.00 | 230 | 15.50 |
| Nasumično 10k uzorka | 0.0078 | 11.9 | 11.9 | 270.00 | 318 | 77.10 | 332.00 | 0.27 | 0.32 | 77.1 | 77.7 | 22% | 0.50 | 2.12 | 0.61 | 611.0 | 611 | 611 | 611 | 611 | 611 | 611 | 110 | 129.00 | 165 | 20.50 |
| Nasumično 1k uzorka | 0.0078 | 2.31 | 11.9 | 27.20 | 31.5 | 70.90 | 64.80 | 0.03 | 0.03 | 70.9 | 71.0 | 17% | 1.91 | 1.74 | 4.55 | 4550 | 4550 | 4550 | 4550 | 4550 | 4550 | 4550 | 58.6 | 83.80 | 124 | 22.20 |
| Nasumično 100 uzorka | 0.0078 | 0.55 | 11.9 | 2.83 | 3.27 | 57.70 | 31.90 | 0.03 | 0.03 | 57.7 | 57.8 | 7% | 1.69 | 0.93 | 19.10 | 1910 | 1910 | 1910 | 1910 | 1910 | 1910 | 1910 | 6.51 | 22.40 | 54.0 | 16.40 |
| Nasumično 10 uzorka | 0.0078 | 0.078 | 11.9 | 0.10 | 0.12 | 11.00 | 5.61 | 0.01 | 0.01 | 11.0 | 11.0 | 2% | 3.07 | 1.56 | 172.0 | 1720 | 1720 | 1720 | 1720 | 1720 | 1720 | 1720 | 0.00 | 3.12 | 17.2 | 5.92 |
| BO 10 uzorka | 0.0078 | 0.0078 | 0.0078 | 0.78 | 0.93 | 1.57 | 3.27 | 0.78 | 0.93 | 1.57 | 3.27 | 4% | 9.03 | 9.03 | 37.80 | 37.80 | 37.8 | 37.8 | 37.8 | 37.8 | 37.8 | 37.8 | 1.00 | 8.18 | 29.6 | 11.10 |
| BO 100 uzorka | 0.0078 | 0.0078 | 0.0078 | 1.38 | 1.82 | 7.44 | 10.60 | 1.38 | 1.82 | 7.44 | 10.6 | 13% | 9.18 | 9.18 | 70.90 | 70.90 | 70.9 | 70.9 | 70.9 | 70.9 | 70.9 | 70.9 | 61.4 | 78.60 | 97.7 | 14.30 |
| BO 1k uzorka | 0.0078 | 0.0078 | 0.0078 | 8.73 | 10.6 | 13.30 | 32.70 | 8.73 | 10.60 | 13.3 | 32.7 | 29% | 6.53 | 6.53 | 24.50 | 24.50 | 24.5 | 24.5 | 24.5 | 24.5 | 24.5 | 24.5 | 120 | 160.00 | 214 | 34.20 |
| BO 10k uzorka | 0.0078 | 0.0078 | 0.0078 | 70.60 | 93.1 | 10.40 | 174.00 | 70.60 | 93.10 | 10.4 | 174 | 37% | 1.56 | 1.56 | 3.84 | 3.84 | 3.84 | 3.84 | 3.84 | 3.84 | 3.84 | 3.84 | 195 | 223.00 | 271 | 21.80 |
| BO 100k uzorka | 0.0078 | 0.0078 | 0.0078 | 732.00 | 967 | 14.70 | 1710 | 732 | 967.0 | 14.7 | 1710 | 75% | 0.33 | 0.33 | 0.76 | 0.76 | 0.76 | 0.76 | 0.76 | 0.76 | 0.76 | 0.76 | 517 | 542.00 | 558 | 20.90 |
| GA 10pop 10gen | 0.025 | 0.078 | 11.9 | 1.15 | 1.33 | 11.10 | 13.60 | 0.12 | 0.13 | 11.1 | 11.4 | 5% | 2.57 | 3.08 | 30.40 | 304.0 | 304 | 304 | 304 | 304 | 304 | 304 | 15.9 | 24.80 | 34.9 | 7.04 |
| GA 10pop 100gen | 0.025 | 0.078 | 11.9 | 2.43 | 3.86 | 7.67 | 14.00 | 0.24 | 0.39 | 7.67 | 8.30 | 45% | 23.80 | 40.00 | 137.0 | 1370 | 1370 | 1370 | 1370 | 1370 | 1370 | 1370 | 313 | 315.00 | 332 | 6.12 |
| GA 10pop 1000gen | 0.025 | 0.078 | 11.9 | 7.00 | 8.88 | 13.30 | 29.10 | 0.70 | 0.89 | 13.3 | 14.8 | 36% | 9.27 | 18.20 | 38.60 | 386.0 | 386 | 386 | 386 | 386 | 386 | 386 | 270 | 270.00 | 270 | 0.00 |

| | | | | proteklo vrijeme [s] | | | | | | | | | bodovi | | | | bodovi na 1000 jezg. ako predmemorija < 16 GiB | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | potrebno memorije [GiB] | | | 1 jezgra | | | | 1000 jezgri | | | | | računanje | | predmemorirano | | 1min VO | | 1s VO | | top 10 bilanca | | | |
| ime algoritma | min | max | predmemorirano | simulacija | indikator | VO | ukupno | simulacija | indikator | VO | ukupno | efektivni max | 1 jezgra | 1000 jezgri | 1 jezgra | 1000 jezgri | 1god | 5god | 1god | 5god | min | prosjek | max | steddev |
| GA 100pop 10gen | 0.21 | 0.54 | 11.9 | 2.93 | 3.96 | 9.61 | 16.50 | 0.03 | 0.04 | 9.61 | 9.67 | 37% | 16.70 | 28.40 | 93.80 | 9380 | 9380 | 9380 | 9380 | 9380 | 138 | 161.00 | 275 | 40.90 |
| GA 100pop 100gen | 0.21 | 0.54 | 11.9 | 15.00 | 20.7 | 10.30 | 46.00 | 0.15 | 0.21 | 10.3 | 10.6 | 51% | 8.30 | 35.90 | 25.50 | 2550 | 2550 | 2550 | 2550 | 2550 | 313 | 361.00 | 382 | 28.10 |
| GA 100pop 1000gen | 0.21 | 0.54 | 11.9 | 11.20 | 15.4 | 9.80 | 36.40 | 0.11 | 0.15 | 9.80 | 10.1 | 75% | 15.30 | 55.40 | 49.90 | 4990 | 4990 | 4990 | 4990 | 4990 | 558 | 558.00 | 558 | 0.00 |
| GA 1000pop 10gen | 2.06 | 2.31 | 11.9 | 39.00 | 50.4 | 10.60 | 100.00 | 0.04 | 0.05 | 10.6 | 10.7 | 39% | 2.92 | 27.40 | 7.47 | 7470 | 7470 | 7470 | 7470 | 7470 | 202 | 227.00 | 292 | 27.40 |
| GA 1000pop 100gen | 2.06 | 2.31 | 11.9 | 136.00 | 180 | 12.70 | 329.00 | 0.14 | 0.18 | 12.7 | 13.1 | 100% | 2.26 | 56.90 | 5.45 | 5450 | 5450 | 5450 | 5450 | 5450 | 567 | 585.00 | 743 | 55.60 |
| GA 1000pop 1000gen | 2.06 | 2.31 | 11.9 | 148.00 | 222 | 12.00 | 382.00 | 0.15 | 0.22 | 12.0 | 12.4 | 76% | 1.49 | 45.90 | 3.83 | 3830 | 3830 | 3830 | 3830 | 3830 | 567 | 567.00 | 567 | 0.00 |
| GA + BO 10 VO 10pop 10gen | 0.025 | 0.078 | 0.0078 | 9.70 | 13.1 | 2.35 | 25.20 | 0.97 | 1.31 | 2.35 | 4.63 | 14% | 4.14 | 22.50 | 10.70 | 107.0 | 107 | 107 | 107 | 107 | 47.0 | 72.50 | 104 | 17.80 |
| GA + BO 10 VO 10pop 100gen | 0.025 | 0.078 | 0.0078 | 90.10 | 115 | 3.29 | 208.00 | 9.01 | 1.15 | 3.29 | 13.5 | 53% | 1.88 | 29.20 | 4.36 | 43.60 | 43.6 | 43.6 | 43.6 | 43.6 | 98.6 | 201.00 | 393 | 90.90 |
| GA + BO 10 VO 10pop 1000gen | 0.025 | 0.078 | 0.0078 | 77.00 | 106 | 2.41 | 186.00 | 0.77 | 0.11 | 2.41 | 3.29 | 38% | 1.51 | 85.40 | 3.64 | 364.0 | 364 | 364 | 364 | 364 | 58.4 | 171.00 | 281 | 87.30 |
| GA + BO 10 VO 100pop 10gen | 0.21 | 0.54 | 0.0078 | 129.00 | 159 | 15.40 | 303.00 | 1.29 | 15.90 | 15.4 | 32.6 | 32% | 0.79 | 7.39 | 1.87 | 187.0 | 187 | 187 | 187 | 187 | 136 | 198.00 | 241 | 35.70 |
| GA + BO 10 VO 100pop 100gen | 0.21 | 0.54 | 0.0078 | 612.00 | 795 | 24.80 | 1430.0 | 6.12 | 7.95 | 24.8 | 38.8 | 66% | 0.34 | 12.50 | 0.80 | 79.60 | 79.6 | 79.6 | 79.6 | 79.6 | 159 | 287.00 | 487 | 93.90 |
| GA + BO 10 VO 100pop 1000gen | 0.21 | 0.54 | 0.0078 | 653.00 | 824 | 185.00 | 1660.0 | 6.53 | 0.82 | 185 | 192 | 51% | 0.23 | 1.99 | 0.58 | 58.40 | 58.4 | 58.4 | 58.4 | 58.4 | 207 | 292.00 | 382 | 58.30 |
| GA + BO 10 VO 1000pop 10gen | 2.06 | 2.31 | 0.0078 | 1090 | 1420 | 38.50 | 2550.0 | 1.09 | 142.0 | 38.5 | 182 | 77% | 0.23 | 3.16 | 0.53 | 526.0 | 526 | 526 | 526 | 526 | 214 | 338.00 | 574 | 103.00 |
| GA + BO 10 VO 1000pop 100gen | 2.06 | 2.31 | 0.0078 | 4280 | 5780 | 38.20 | 10100 | 4.28 | 57.80 | 38.2 | 100 | 81% | 0.06 | 5.98 | 0.14 | 140.0 | 140 | 140 | 140 | 140 | 300 | 380.00 | 600 | 103.00 |

| | | | | proteklo vrijeme [s] | | | | | | | | | bodovi | | | | bodovi na 1000 jezg. ako predmemorija < 16 GiB | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | potrebno memorije [GiB] | | | 1 jezgra | | | | 1000 jezgri | | | | | računanje | | predmemorirano | | 1min VO | | 1s VO | | top 10 bilanca | | | |
| ime algoritma | min | max | predmemorirano | simulacija | indikator | VO | ukupno | simulacija | indikator | VO | ukupno | efektivni max | 1 jezgra | 1000 jezgri | 1 jezgra | 1000 jezgri | 1god | 5god | 1god | 5god | min | prosjek | max | steddev |
| GA + BO 10 VO 1000pop 1000gen | 2.06 | 2.31 | 0.0078 | 5290 | 7230 | 22.70 | 12500 | 5.29 | 7.23 | 22.7 | 35.2 | 81% | 0.05 | 17.00 | 0.11 | 113.0 | 113 | 113 | 113 | 113 | 300 | 392.00 | 600 | 119.00 |
| GA + BO 15 VO 10pop 10gen | 0.025 | 0.078 | 0.0078 | 8.87 | 12.2 | 2.98 | 24.00 | 0.89 | 1.22 | 2.98 | 5.09 | 47% | 14.50 | 68.70 | 39.40 | 394.0 | 394 | 394 | 394 | 394 | 58.3 | 111.00 | 349 | 85.60 |
| GA + BO 15 VO 10pop 100gen | 0.025 | 0.078 | 0.0078 | 81.70 | 113 | 3.09 | 198.00 | 8.17 | 1.13 | 3.09 | 12.4 | 75% | 2.82 | 45.00 | 6.83 | 68.30 | 68.3 | 68.3 | 68.3 | 68.3 | 233 | 341.00 | 558 | 100.00 |
| GA + BO 15 VO 10pop 1000gen | 0.025 | 0.078 | 0.0078 | 122.00 | 168 | 3.05 | 293.00 | 1.22 | 0.17 | 3.05 | 4.44 | 55% | 1.40 | 92.30 | 3.36 | 336.0 | 336 | 336 | 336 | 336 | 161 | 258.00 | 410 | 81.20 |
| GA + BO 15 VO 100pop 10gen | 0.21 | 0.54 | 0.0078 | 118.00 | 158 | 3.05 | 278.00 | 1.18 | 15.80 | 3.05 | 20.0 | 39% | 1.04 | 14.40 | 2.45 | 245.0 | 245 | 245 | 245 | 245 | 189 | 221.00 | 288 | 35.40 |
| GA + BO 15 VO 100pop 100gen | 0.21 | 0.54 | 0.0078 | 583.00 | 805 | 3.21 | 1390.0 | 5.83 | 8.05 | 3.21 | 17.1 | 75% | 0.40 | 32.70 | 0.96 | 95.70 | 95.7 | 95.7 | 95.7 | 95.7 | 264 | 346.00 | 558 | 82.30 |
| GA + BO 15 VO 100pop 1000gen | 0.21 | 0.54 | 0.0078 | 438.00 | 599 | 16.40 | 1050.0 | 4.38 | 0.60 | 16.4 | 21.4 | 83% | 0.58 | 28.70 | 1.40 | 140.0 | 140 | 140 | 140 | 140 | 298 | 427.00 | 614 | 120.00 |
| GA + BO 15 VO 1000pop 10gen | 2.06 | 2.31 | 0.0078 | 1090 | 1500 | 42.30 | 2630.0 | 1.09 | 150.0 | 42.3 | 193 | 81% | 0.23 | 3.11 | 0.55 | 551.0 | 551 | 551 | 551 | 551 | 267 | 374.00 | 600 | 112.00 |
| GA + BO 15 VO 1000pop 100gen | 2.06 | 2.31 | 0.0078 | 5190 | 6750 | 193.00 | 12100 | 5.19 | 67.50 | 193 | 266 | 91% | 0.06 | 2.55 | 0.13 | 131.0 | 131 | 131 | 131 | 131 | 313 | 461.00 | 680 | 145.00 |
| GA + BO 15 VO 1000pop 1000gen | 2.06 | 2.31 | 0.0078 | 5290 | 6980 | 99.20 | 12400 | 5.29 | 6.98 | 99.2 | 111 | 83% | 0.05 | 5.50 | 0.12 | 116.0 | 116 | 116 | 116 | 116 | 323 | 446.00 | 614 | 124.00 |
| GA + BO 30 VO 10pop 10gen | 0.025 | 0.078 | 0.0078 | 16.40 | 22.4 | 5.06 | 43.90 | 1.64 | 2.24 | 5.06 | 8.94 | 20% | 3.36 | 16.50 | 8.95 | 89.50 | 89.5 | 89.5 | 89.5 | 89.5 | 92.1 | 112.00 | 147 | 15.00 |
| GA + BO 30 VO 10pop 100gen | 0.025 | 0.078 | 0.0078 | 159.00 | 205 | 33.50 | 397.00 | 15.90 | 2.05 | 33.5 | 51.5 | 66% | 1.23 | 9.46 | 3.06 | 30.60 | 30.6 | 30.6 | 30.6 | 30.6 | 188 | 280.00 | 487 | 90.60 |
| GA + BO 30 VO 10pop 1000gen | 0.025 | 0.078 | 0.0078 | 131.00 | 174 | 41.40 | 346.00 | 1.31 | 0.17 | 41.4 | 42.8 | 64% | 1.37 | 11.10 | 3.64 | 364.0 | 364 | 364 | 364 | 364 | 237 | 339.00 | 475 | 88.90 |
| GA + BO 30 VO 100pop 10gen | 0.21 | 0.54 | 0.0078 | 193.00 | 251 | 108.00 | 552.00 | 1.93 | 25.10 | 108 | 135 | 48% | 0.64 | 2.62 | 1.84 | 184.0 | 184 | 184 | 184 | 184 | 226 | 273.00 | 355 | 43.60 |

| | | | | proteklo vrijeme [s] | | | | | | | | | bodovi | | | | bodovi na 1000 jezg. ako predmemorija < 16 GiB | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | potrebno memorije [GiB] | | | 1 jezgra | | | | 1000 jezgri | | | | | računanje | | predmemorirano | | 1min VO | | 1s VO | | top 10 bilanca | | | |
| ime algoritma | min | max | predmemorirano | simulacija | indikator | VO | ukupno | simulacija | indikator | VO | ukupno | efektivni max | 1 jezgra | 1000 jezgri | 1 jezgra | 1000 jezgri | 1god | 5god | 1god | 5god | min | prosjek | max | steddev |
| GA + BO 30 VO 100pop 100gen | 0.21 | 0.54 | 0.0078 | 573.00 | 775 | 27.70 | 1380.0 | 5.73 | 7.75 | 27.7 | 41.1 | 83% | 0.45 | 14.90 | 1.07 | 107.0 | 107 | 107 | 107 | 107 | 391 | 545.00 | 614 | 76.10 |
| GA + BO 30 VO 100pop 1000gen | 0.21 | 0.54 | 0.0078 | 560.00 | 760 | 7.47 | 1330.0 | 5.60 | 0.76 | 7.47 | 13.8 | 83% | 0.46 | 44.30 | 1.09 | 109.0 | 109 | 109 | 109 | 109 | 382 | 512.00 | 614 | 96.20 |
| GA + BO 30 VO 1000pop 10gen | 2.06 | 2.31 | 0.0078 | 1510 | 2060 | 37.90 | 3610.0 | 1.51 | 206.0 | 37.9 | 245 | 64% | 0.13 | 1.94 | 0.31 | 314.0 | 314 | 314 | 314 | 314 | 351 | 420.00 | 475 | 44.00 |
| GA + BO 30 VO 1000pop 100gen | 2.06 | 2.31 | 0.0078 | 6240 | 8450 | 29.00 | 14700 | 6.24 | 84.50 | 29.0 | 120 | 91% | 0.05 | 5.67 | 0.11 | 109.0 | 109 | 109 | 109 | 109 | 475 | 565.00 | 680 | 59.70 |
| GA + BO 30 VO 1000pop 1000gen | 2.06 | 2.31 | 0.0078 | 6500 | 8680 | 113.00 | 487.00 | 6.50 | 8.68 | 113 | 128 | 91% | 1.39 | 5.32 | 0.11 | 105.0 | 105 | 105 | 105 | 105 | 475 | 532.00 | 680 | 175.00 |
| Podijeljeni GA 10pop 10gen | 0.055 | 0.055 | 1.21 | 3.75 | 5.28 | 10.10 | 19.10 | 0.38 | 0.53 | 1.01 | 1.91 | 20% | 7.77 | 77.70 | 39.50 | 395.0 | 395 | 395 | 395 | 395 | 475 | 67.70 | 148 | 38.40 |
| Podijeljeni GA 10pop 100gen | 0.055 | 0.055 | 1.21 | 42.60 | 58.9 | 9.72 | 111.00 | 4.26 | 5.89 | 0.97 | 11.1 | 27% | 1.79 | 17.90 | 4.66 | 46.60 | 46.6 | 46.6 | 46.6 | 46.6 | 475 | 199.00 | 199 | 0.00 |
| Podijeljeni GA 10pop 1000gen | 0.055 | 0.055 | 1.21 | 41.50 | 58.4 | 9.72 | 110.00 | 4.15 | 5.84 | 0.97 | 11.0 | 48% | 3.24 | 32.40 | 8.58 | 85.80 | 85.8 | 85.8 | 85.8 | 85.8 | 475 | 266.00 | 355 | 31.50 |
| Podijeljeni GA 100pop 10gen | 0.231 | 0.231 | 1.21 | 46.20 | 64.9 | 10.10 | 121.00 | 0.46 | 0.65 | 0.10 | 1.21 | 58% | 3.56 | 356.0 | 9.35 | 935.0 | 935 | 935 | 935 | 935 | 475 | 161.00 | 432 | 107.00 |
| Podijeljeni GA 100pop 100gen | 0.231 | 0.231 | 1.21 | 256.00 | 336 | 13.80 | 606.00 | 2.56 | 3.36 | 0.14 | 6.06 | 57% | 0.70 | 69.60 | 1.65 | 165.0 | 165 | 165 | 165 | 165 | 475 | 422.00 | 422 | 0.00 |
| Podijeljeni GA 100pop 1000gen | 0.231 | 0.231 | 1.21 | 214.00 | 302 | 10.10 | 526.00 | 2.14 | 3.02 | 0.10 | 5.26 | 66% | 0.93 | 93.20 | 2.29 | 229.0 | 229 | 229 | 229 | 229 | 475 | 490.00 | 490 | 0.00 |
| Podijeljeni GA 1000pop 10gen | 1.21 | 1.21 | 1.21 | 505.00 | 695 | 11.50 | 1210.0 | 0.51 | 0.70 | 11.5 | 12.7 | 87% | 0.54 | 51.00 | 1.28 | 1280 | 1280 | 1280 | 1280 | 1280 | 475 | 335.00 | 649 | 145.00 |
| Podijeljeni GA 1000pop 100gen | 1.21 | 1.21 | 1.21 | 1790 | 2490 | 11.40 | 4290.0 | 1.79 | 2.49 | 11.4 | 15.7 | 100% | 0.17 | 47.40 | 0.41 | 414.0 | 414 | 414 | 414 | 414 | 475 | 627.00 | 743 | 41.00 |
| Podijeljeni GA 1000pop 1000gen | 1.21 | 1.21 | 1.21 | 1740 | 2410 | 10.80 | 4170.0 | 1.74 | 2.41 | 10.8 | 14.9 | 83% | 0.15 | 41.10 | 0.35 | 352.0 | 352 | 352 | 352 | 352 | 475 | 614.00 | 614 | 0.00 |