Optične in nanotehnologije

Vpliv parametrov na uspešnost simulatorja pri iskanju želene strukture

Miha Nagelj – 63050062
Ivo Križman – 63050060
Tom Vodopivec – 63050129
Davor Sluga – 63050108

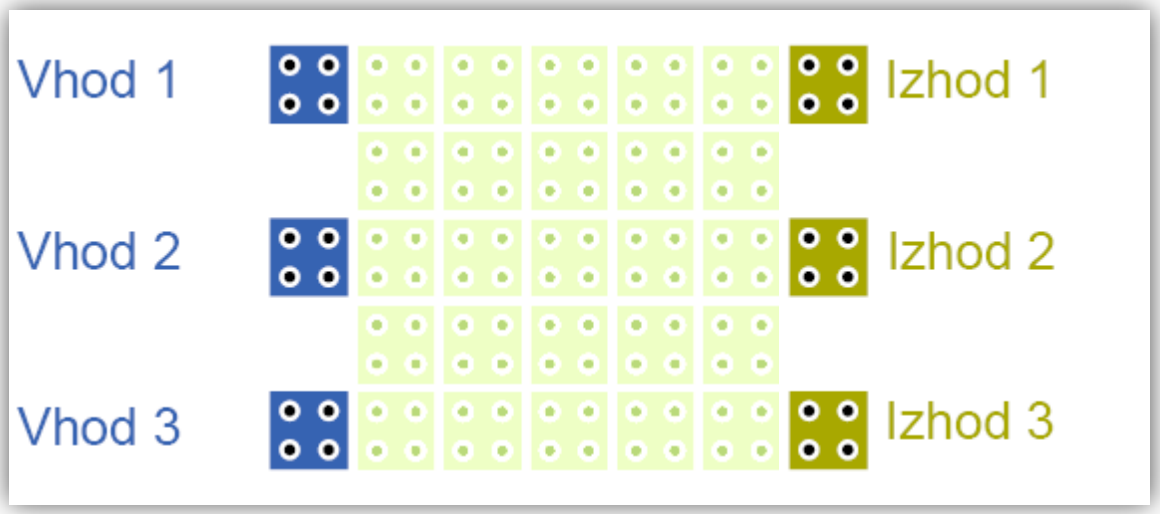
12. januar 2008

Kazalo

1. Uvod

1.1. Opis problema

Namen seminarske naloge je analiza vpliva vrednosti parametrov genetskega algoritma na učinkovitost iskanja želene logične strukture v kvantnih celularnih avtomatih. Genetski algoritem (GA) išče strukturo celic znotraj polja 5x5, ki bi čim točneje opravljala podano logično funkcijo. Struktura celic ima tri vhode in tri izhode, kar prikazuje tudi spodnja slika.



Na polje je možno postaviti 25 celic. Vse morebitne celice so vezane na isto uro in nobena nima fiksirane vrednosti. Naloga algoritma je, da ustrezno razvrsti celice po polju. Ker je vseh možnih razvrstitev 2^{25} = 33,5 milijona, je uporaba genetskih pristopov nujna.

V našem primeru moramo realizirati dvovhodno funkcijo z enim izhodom, zato enega vhoda in dveh izhodov ne bomo uporabljali.

1.2. Delovanje genetskega algoritma

GA uporablja standardne operacije za iskanje rešitve kot so na primer križanje, mutacija in zamenjava. Osebke ocenjuje s pomočjo simulatorja QCA struktur. Ta zna iz podanih vrednosti vhodov in postavitve celic izračunati v kakšnem stanju se nahaja izhod strukture. Kompleksnost izračuna je odvisna od postavitve in števila celic, zato je čas izračuna zelo različen.

GA oceno izračuna na podlagi ujemanja vrednosti izhoda z vrednostjo logične funkcije za vse možne vhodne kombinacije. Ker za vsako ujemanje oceni prišteje 1, je ocena v mejah med 0 in 24. V našem primeru ko nas zanimata samo dva vhoda in en izhod, vrednosti ostalih vhodov in izhodov vedno vzamemo za prave. Tako GA vedno izračuna oceno med 20 in 24. Zaradi bolj smiselnega prikazovanja rezultatov smo v nadaljevanju uporabili prikazovanje fitnesov z vrednostmi od 0 (ki predstavlja 20) do 4 (predstavlja 24).

1.3. Izvajanje meritev

Preverjanje učinkovitosti pri različnih vrednostih parametrov, zahteva veliko število zaganjanj algoritma. Ker bi bilo ročno zaganjanje zelo dolgotrajno, predvsem pa neproduktivno, smo se odločili napisati program, ki bo zaganjal program z ustrezno vrednostjo posameznega parametra, shranjeval rezultate in zgeneriral končni povzetek rezultatov.

Program izvajamo za 3 logične funkcije in sicer AND, NAND in OR.

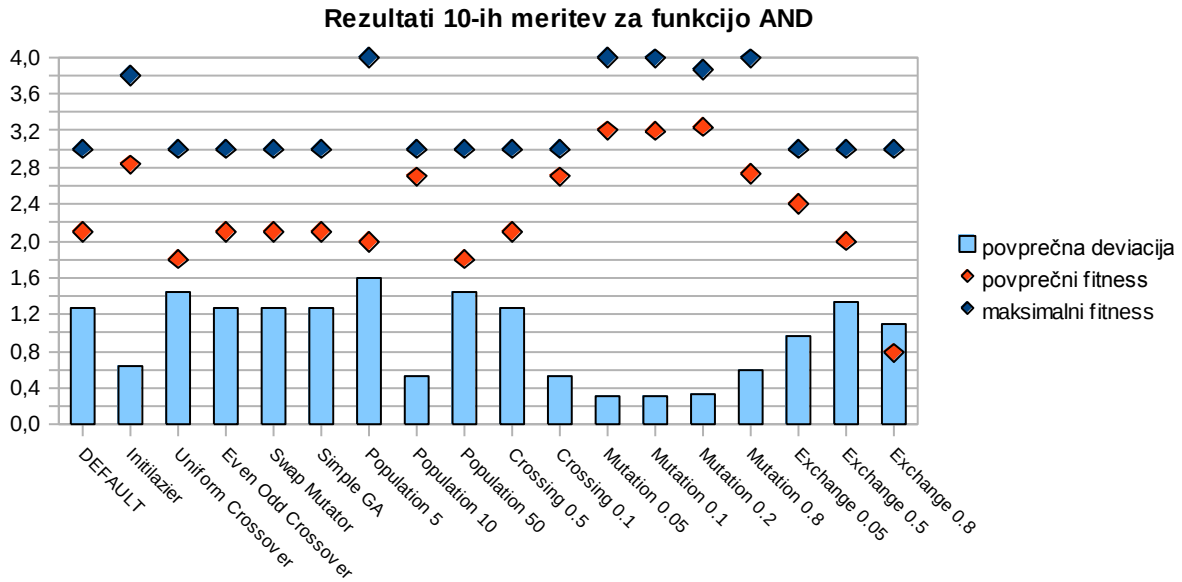
2. Parametri

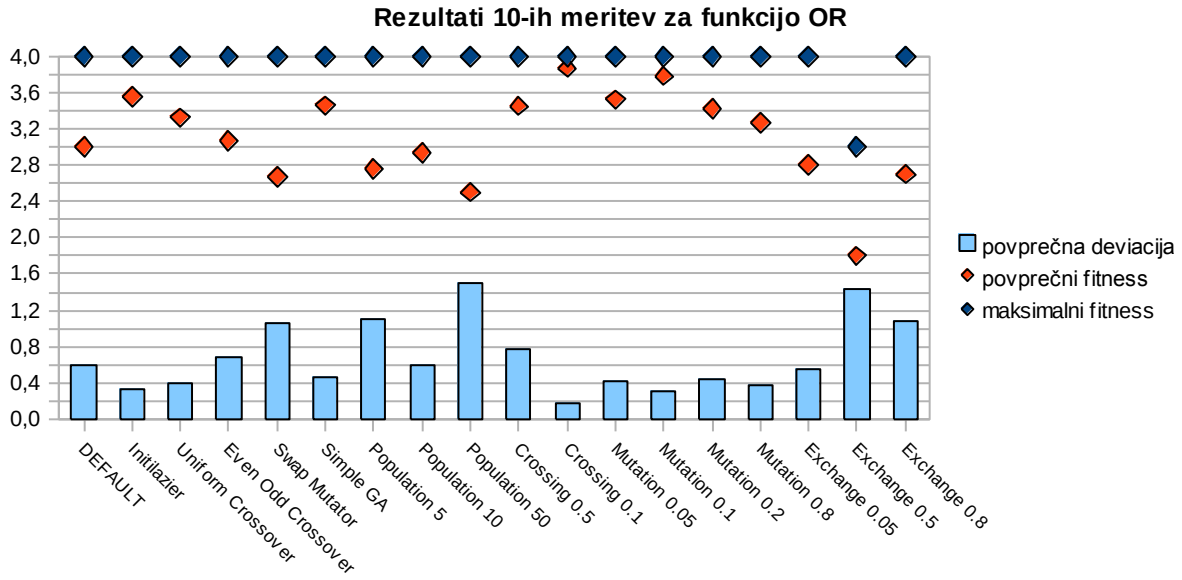
Parametri katerih vrednosti analiziramo, se nahajajo v spodnji tabeli.

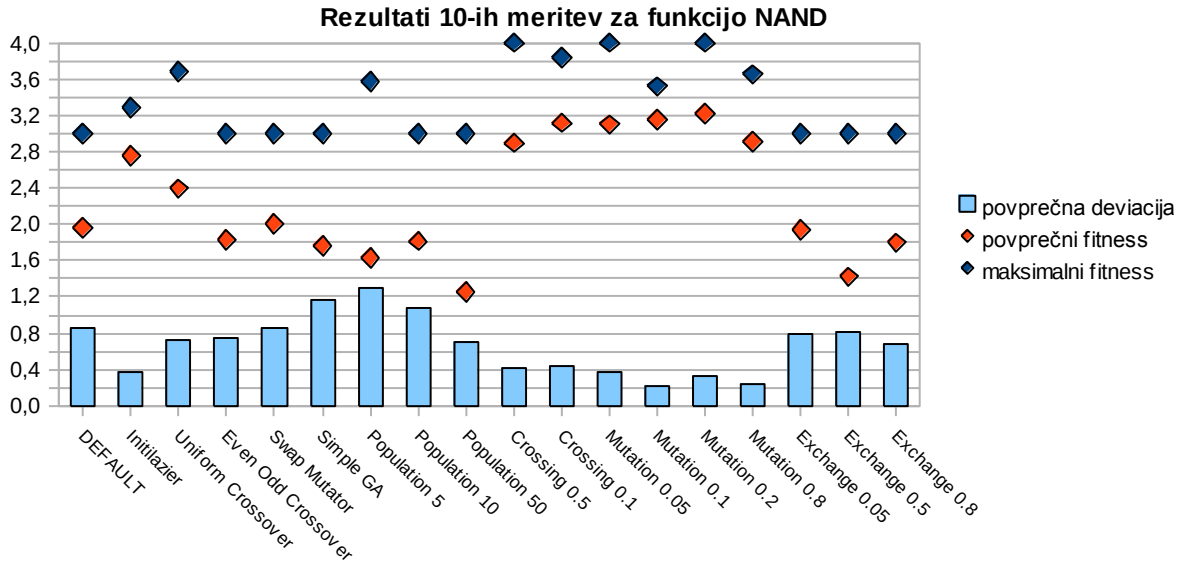
Parameter	**Privzeta vrednost**	**Ostale vrednosti parametrov**			
Initializer	NoInitializer	Initializer			
Križanje	OnePointCrossover	UniformCrossover	EvenOddCrossover		
Mutacija	FlipMutator	SwapMutator			
Tip GA	SteadyState	Simple			
Velikost populacije	20	5	10	50	
Verjetnost križanja	0,8	0,5	0,1		
Verjetnost mutacije	0,01	0,05	0,1	0,2	0,8
Verjetnost zamenjave	0,2	0,05	0,5	0,8	

Skupaj imamo 8 parametrov. Analizirali bomo spremembo vsakega parametra posebej, torej tako da spreminjamo vrednost samo enemu parametru, ostali pa imajo privzeto vrednost. Opazimo lahko, da imamo s tem 18 različnih kombinacij. Za bolj realne rezultate moramo vsako kombinacijo izvesti večkrat npr. 10-krat. Ker smo omejeni s časom in ker je smiselno postaviti časovno mejo izvajanja, postavimo tudi omejitev izvajanja na 5 min. Za izvajanje algoritma tako rabimo 18*10*5/60=15 ur časa za vsako logično funkcijo. Za funkcije AND, NAND in OR tako potrebujemo skoraj 2 dni časa.

3. Rezultati in ugotovitve







3.1. Opis postopka in pomen posameznih vrednosti

Zgornji trije grafi nakazujejo povprečne vrednosti za 10 meritev vseh 18 kombinacij parametrov za vsako logično funkcijo posebej. Ker imajo izbrane logične funkcije dveh spremenljivk vsaka po 4 možne kombinacije, se za cenitev učinkovitosti strukture izračuna število pravilnih kombinacij na izhodu - torej fitness ima razpon med 0 (vse napačne) do 4 (vse pravilne). Povprečni fitness je izračunan na podlagi najvišjega fitnessa, ki ga je dosegel vsak test posebej, maksimalni fitness je najvišji izmed vseh desetih. Namen povprečne deviacije je nakazati kako velika so odstopanja od povprečne vrednosti pri vsakem parametru posebej in ima zgolj relativni, ne pa absolutni, pomen - ta podatek je namenjen primerjavi zanesljivosti posameznih povprečenih rezultatov.

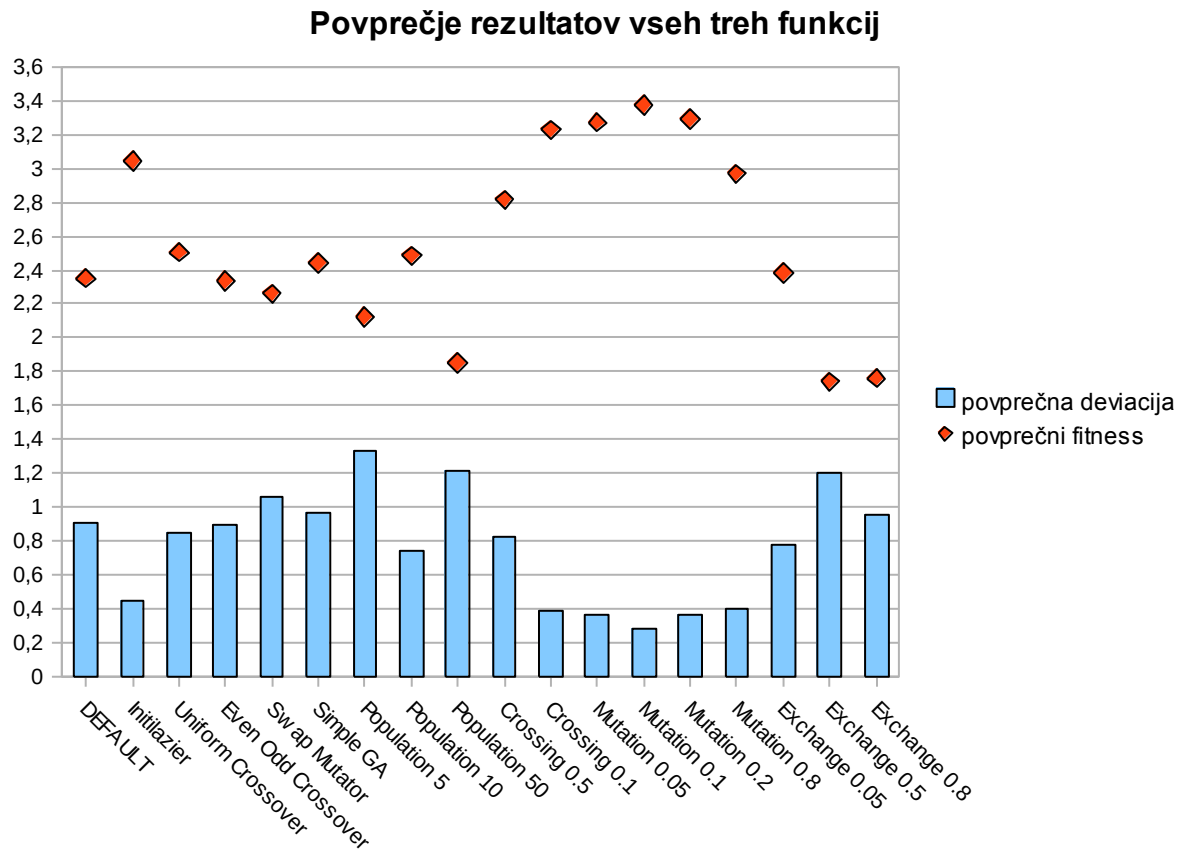
Meritve za funkcijo OR so bile poganjane na močnejšem računalniku, zato je algoritem v enakem času naredil večje število korakov, kar je razvidno iz boljših rezultatov - predvsem iz dejstva, da so praktično vse kombinacije parametrov (razen enega) vsaj enkrat dosegle vrednost 4. Da smo dobili verodostojne rezultate smo morali parameter “initializer” zaganjati 3x dlje kot ostale - težava je namreč v komplekstnosti struktur pri katerih simulator potrebuje veliko več časa za posamezni izračun izhoda.

3.2. Analiza rezultatov

Najbolj negativen vpliv na učinkovitost algoritma imata parameter “population”, ko je le ta premajhen ali prevelik - 5 ali 50 ter parameter “exchange”, ko je le ta previsok - 0.5 ali 0.8. V teh primerih so tudi odstopanja največja.

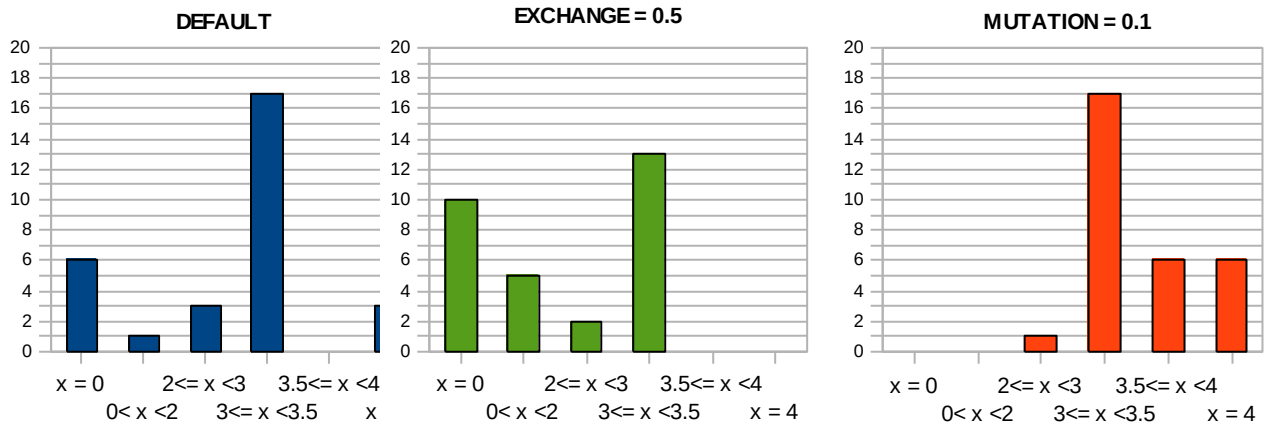
Več od testiranih kombinacij ne bistveno vpliva na končni rezultat. “Uniform Crossover”, “Simple GA”, “Population = 10” ter “Exchange = 0.05” minimalno izboljšajo rezultat ter imajo rahlo nižje odstopanje. Na drugi strani “Even Odd Crossover” in “Swap Mutator” minimalno poslabšajo rezultat.

Parametri, ki bistveno izboljšajo učinkovitost so verjetnosti križanja “Crossing”, verjetnosti mutacije “mutation” ter zastavica “Initializer”. Od naštetih dajejo najboljše rezultate “Crossing = 0.1” ter “Mutation” na vrednostih 0.05, 0.1 ter 0.2. Očitno je privzeta vrednosta “Crossing = 0.8” previsoka ter “Mutation = 0.01” prenizka. “Initializer” - to je začetna naključna postavitev celic v polje - daje dobre rezultate, toda zahteva veliko večjo porabo časa in je zaradi tega postavljena pod vprašaj njegova resnična učinkovitost. Pri vseh teh parametrih (razen pri “Crossing = 0.5”) je povprečno odstopanje zelo nizko, kar bistveno povečuje zanesljivost rezultatov.



Zgornji graf prikazuje povprečje rezultatov vseh treh funkcij. Iz njega je razvidno, da vse prejšnje ugotovitve držijo - tudi pri povprečenju meritev čez vse 3 funkcije (30 meritev), še vedno bistveno izstopajo iste kombinacije parametrov.

Grafi porazdelitev najboljših fitnesov čez 30 meritev za tri izbrane primere parametrov:



Izbrana je bila privzeta kombinacija prametrov “DEFAULT” s katero smo primerjali domnevno najslabšo (“Exchange = 0.5”) ter najboljšo (“Mutation = 0.1”) kombinacijo parametrov. Razberemo, da je učinkovitost posameznega parametra na končno delovanje algoritma bistveno različna, kar spet potrjuje naše prejšnje ugotovitve.

3.3. Najboljša kombinacija parametrov ter primerjava z ostalimi kombinacijami

Na podlagi vseh prejšnjih ugotovitev lahko sestavimo načeloma optimalno kombinacijo parametrov:

PARAMETER	VREDNOST
Initializer	Initializer
Križanje	UniformCrossover
Mutacija	FlipMutator
Tip GA	Simple
Velikost populacije	10
Verjetnost križanja	0,1
Verjetnost mutacije	0,1
Verjetnost zamenjave	0,05

Pri takem kombiniranju pridemo do težave, da ne poznamo medsebojnih odvisnosti med parametri in lahko tako pridemo do slabših rezultatov kot jih pričakujemo. Ker prvi štirje parametri določajo način, kako algoritem deluje, je mogoče, da so optimalne vrednosti parametrov "velikost populacije" ter "verjetnosti križanja, mutacije in zamenjave" različne od načina do načina. Poleg tega parameter "initializer" zelo upočasni izvajanje in nam tako onemogoča pridobitev verodostojnih rezultatov v času, ki ga imamo na razpolago.

Iz teh dveh razlogov smo se odločili uporabiti tako kombinacijo parametrov, ki naj bi dala optimalne rezultate pri privzetih vrednostih parametrov "initializer", "križanje", "mutacija" ter "tip GA".

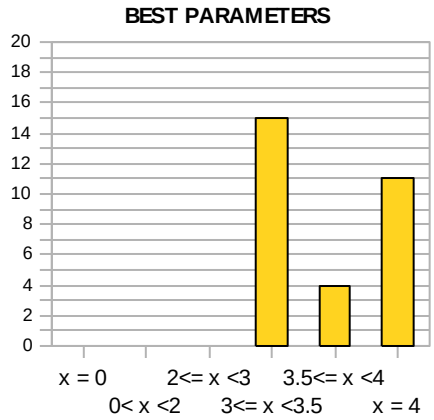
Naša končna izbrana kombinacija najboljših vrednosti parametrov:

PARAMETER	VREDNOST
Initializer	NoInitializer
Križanje	OnePointCrossover
Mutacija	FlipMutator
Tip GA	SteadyState
Velikost populacije	10
Verjetnost križanja	0,1
Verjetnost mutacije	0,1
Verjetnost zamenjave	0,05

3.4. Primerjava najboljše kombinacije parametrov z ostalimi

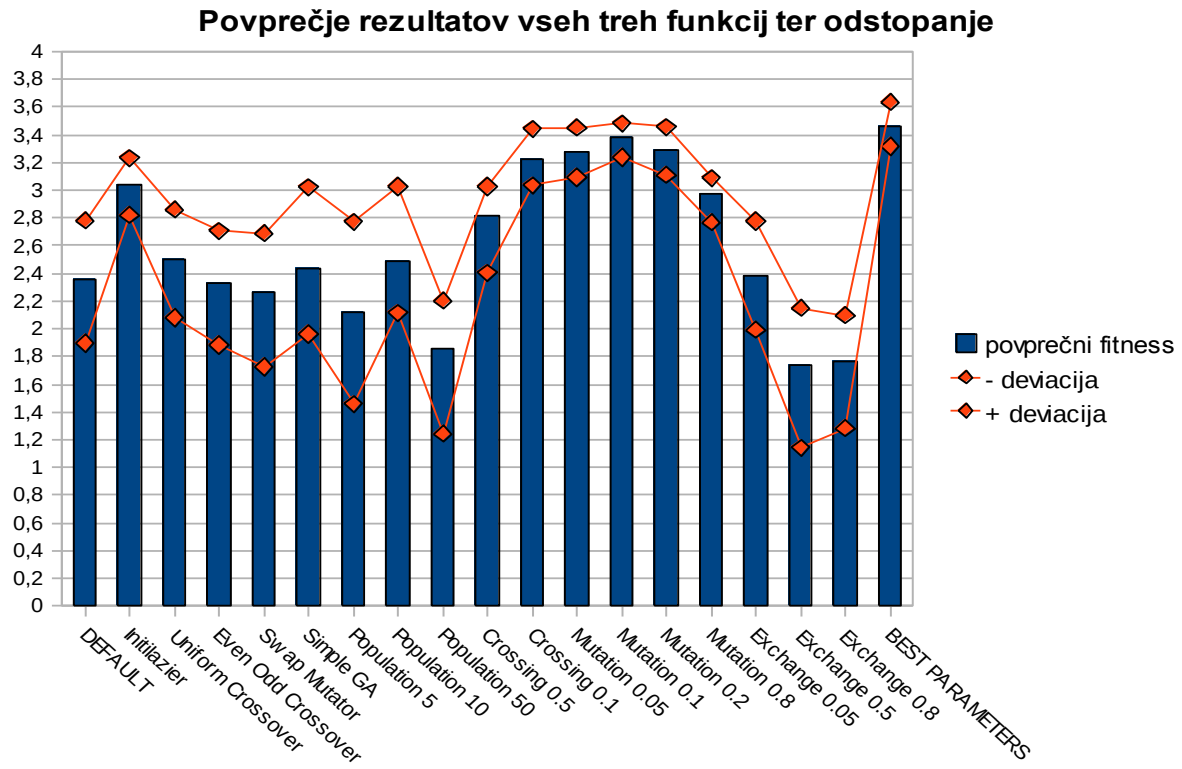
Z dodatnimi meritvami smo preverili ali je izbrana kombinacija boljša ali mogoče slabša od ostalih.

Graf porazdelitev najboljših fitnessov čez 30 meritev za najboljšo kombinacijo parametrov:

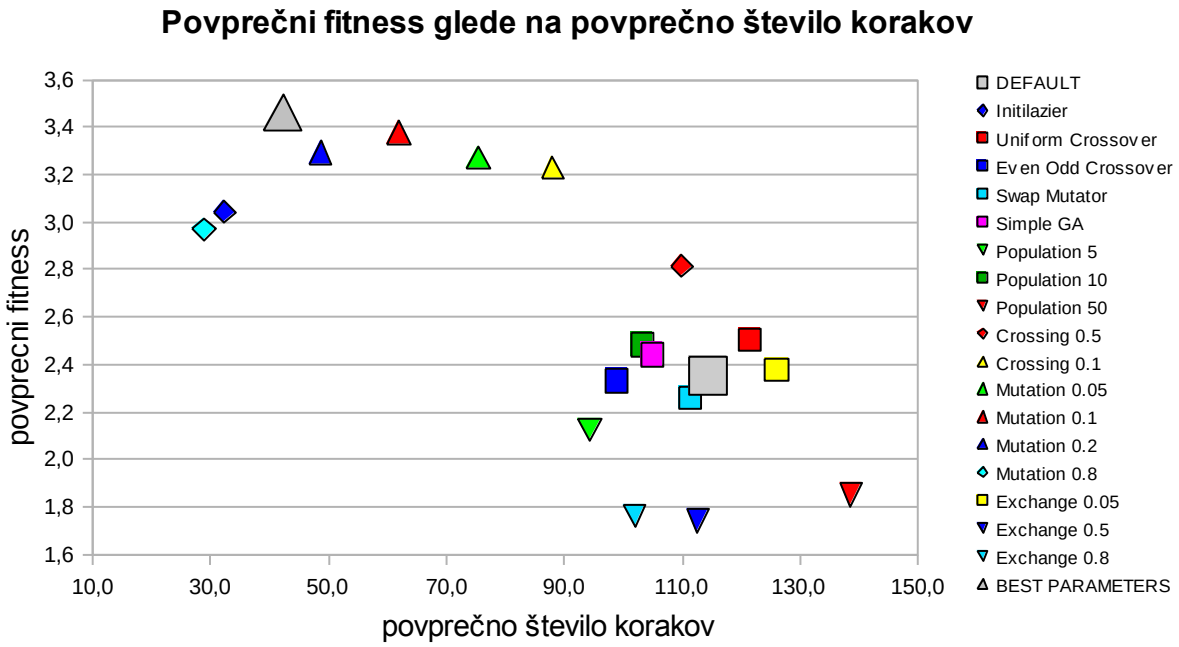


Vidimo, da v primerjavi z ostalimi grafi porazdelitev, je nova kombinacija boljša.

Naslednji graf je modifikacija enega izmed prvih grafov, ki prikazuje učinkovitost posamezne kombinacije v povprečju za vseh 30 meritev. Razlikuje se v tem, da je tu dodana še optimalna kombinacija parametrov (skrajno desno), da je povprečni fitnes prikazan stolpcično ter da je odstopanje prikazano absolutno (rdeče točke):

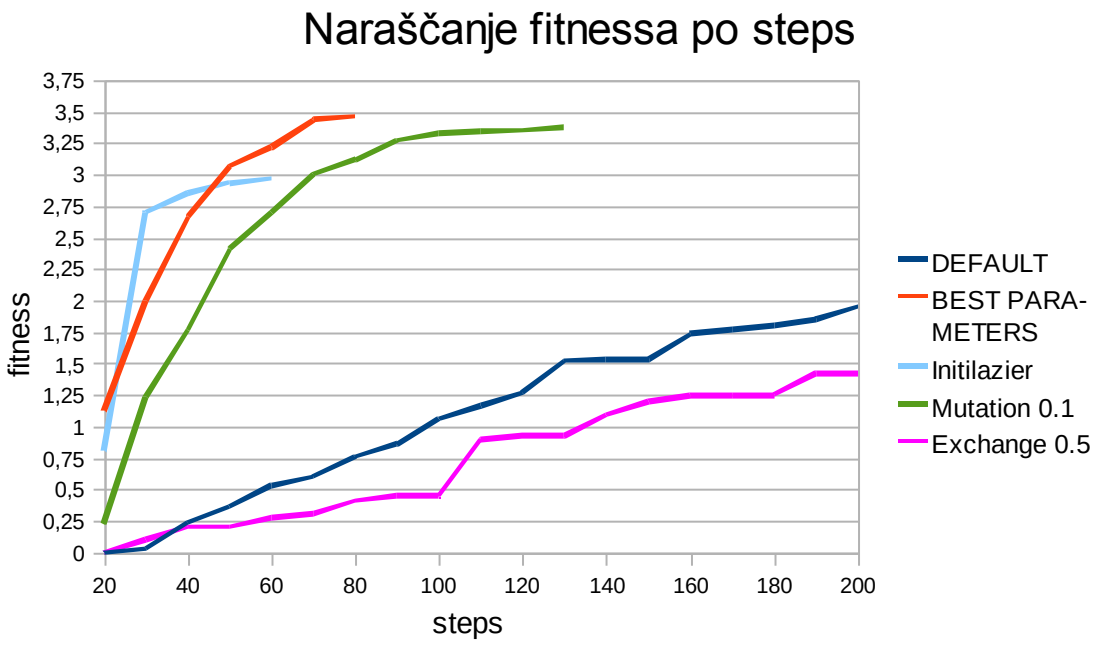


Vidimo, da v je povprečju izbrana kombinacija - čeprav za malo - najboljša ter da je njeno odstopanje zelo majhno. V primerjavi z “DEFAULT” je izboljšanje zelo očitno.



Zgornji graf je prvi, ki prikazuje hitrost (število korakov) konvergence, ki je ravno tako ključnega pomena pri učinkovitosti algoritma. Z trikotnikom obrnjenim navzgor so označene kombinacije parametrov, ki bistveno izboljšajo učinkovitost algoritma. Rombi tudi ponazarjajo izboljšanje, toda v manjši meri. Kvadrati ponazarjajo tiste parametre, ki imajo majhen vpliv. Trikotniki obrnjeni navzdol prikazujejo parametre z negativnim vplivom. Velik siv kvadrat je privzeta vrednost parametrov, velik siv trikotnik obrnjen navzgor pa izbrana optimalna vrednost parametrov.

Vidimo, da parametri, ki izboljšajo učinkovitost, izboljšajo tudi hitrost konvergence (z izjemo “crossing = 0.5”) - zmanjšajo potrebno število korakov tudi za 2 do 3-krat. Izbrana kombinacija parametrov ima zelo dobro hitrost konvergence.

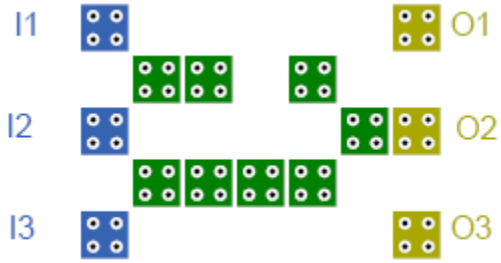


Tukaj vidimo najboljši prikaz hitrosti naraščanja fitnessa po korakih. Zaradi časovne zahtevnosti meritev in pomanjkanja časa je graf nepopoln - velik potencial ima parameter “Initializer”, ki pa nujno potrebuje več meritev. Za optimalno kombinacijo se izkaže, da bolje konvergira kot kombinacija z “mutation = 0.1”, ki je pri meritvah dela ene izmed najboljših rezultatov. Vidimo, da so vse te kombinacije bistveno boljše od “DEFAULT”, parameter “exchange = 0.5” pa je slabši.

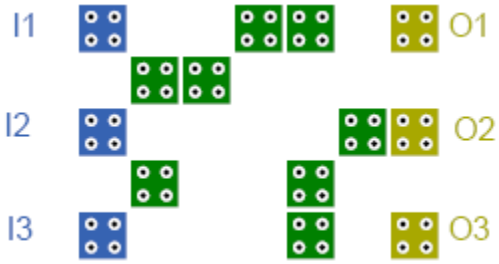
3.5. Strukture dobljene z najboljšo konfiguracijo parametrov

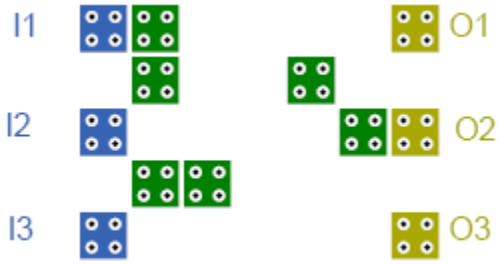
Na spodnjih slikah so prikazane dobljene strukture s fitnessom 24.

AND

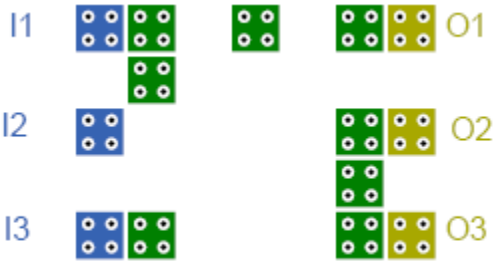


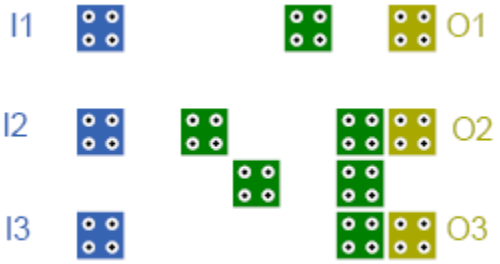
NAND





OR





Poudariti je potrebno, da prikazane strukture preverjeno delujejo samo ko so povsem izolirane, torej ko nobena druga zunanja celica ne vpliva na strukturo.

4. Zaključek

Pri analizi vpliva posameznih parametrov na učinkovitost genetskega algoritma smo naleteli na kar nekaj težav. Preveriti je bilo treba veliko različnih konfiguracij, kar traja veliko časa, ker pa smo bili časovno omejeni, smo vsako konfiguracijo pognali le 10x z časom izvajanja omejenim na 5 minut.

Naši sklepi tako nimajo zelo močne statistične podlage, vendar smo iz njih vseeno izluščili določene zaključke. Za dosego rezultatov, ki bi zagotovili jasnejšo sliko vpliva parametrov na algoritem, bi bilo potrebno izvesti dolgotrajnejše testiranje.

Neraziskan je ostal predvsem parameter Initalizer, saj se je pri postavitvi te zastavice simulator močno upočasnil in zaradi tega opravil precej manj iteracij kot sicer. Glede na to, da je njegov vpliv na genetski algoritem precej velik, bi bila tu potrebna dodatna testiranja, ali lahko z njim še izboljšamo rezultat našega optimalnega nabora parametrov in ali bi se ostali parametri morali v tem primeru prilagoditi (tu je vprašljiva predvsem verjetnost mutacije). Kljub vsemu so naša testiranja privedla do uporabnih rezultatov, saj naš optimalni nabor parametrov precej izboljša hitrost konvergence napram izhodiščnemu.