

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2022

**Pronalaženje Booleovih funkcija  
maksimalne nelinearnosti  
evolucijskim računanjem**

Kristijan Vulinović

Zagreb, lipanj 2019.

**SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA  
ODBOR ZA DIPLOMSKI RAD PROFILA**

Zagreb, 8. ožujka 2019.

## **DIPLOMSKI ZADATAK br. 2022**

Pristupnik: **Kristijan Vulinović (0036483282)**

Studij: **Računarstvo**

Profil: **Računarska znanost**

Zadatak: **Pronalaženje Booleovih funkcija maksimalne nelinearnosti evolucijskim računanjem**

**Opis zadatka:**

Booleove funkcije sastavni su element kriptografskih algoritama. Kako bi se povećala otpornost na napade linearnom kriptoanalizom, od posebnog je značaja svojstvo nelinearnosti Booleove funkcije. Booleove funkcije zadanog broja varijabli i maksimalne nelinearnosti nazivaju se Bent-funkcije, dok su sa stajališta primjene u kriptografskim algoritmima od posebnog interesa Booleove funkcije koje dodatno imaju i svojstvo balansiranosti.

U okviru ovog diplomskog rada potrebno je proučiti heurističke pristupe pronalaženja Booleovih funkcija maksimalne nelinearnosti te balansiranih Booleovih funkcija maksimalne nelinearnosti.

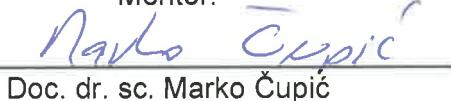
Potrebno je izraditi prototipne implementacije odabranih pristupa te prikazati i ocijeniti dobivene rezultate.

Radu priložiti izvorni kod razvijenih postupaka uz potrebna objašnjenja i dokumentaciju. Predložiti pravce budućeg razvoja. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 15. ožujka 2019.

Rok za predaju rada: 28. lipnja 2019.

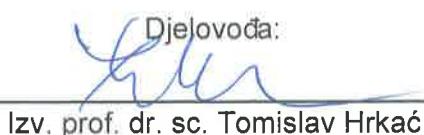
Mentor:

  
\_\_\_\_\_  
Doc. dr. sc. Marko Čupić

Predsjednik odbora za  
diplomski rad profila:

  
\_\_\_\_\_  
Doc. dr. sc. Marko Čupić

Djelovođa:

  
\_\_\_\_\_  
Izv. prof. dr. sc. Tomislav Hrkać

*Uz najiskrenije zahvale mentoru, doc. dr. sc. Marku Čupiću, na strpljenju, savjetima i pomoći prilikom izrade rada.*

*Također, veliko hvala obitelji na neizmjernoj podršci i motivaciji tijekom cijelog studija te hvala Luki i Dinu na pomoći tijekom izrade rada.*

# SADRŽAJ

<b>1. Uvod</b>	<b>1</b>
<b>2. Booleove funkcije i interesantna svojstva</b>	<b>3</b>
2.1. Načini prikaza . . . . .	3
2.2. Svojstva Booleovih funkcija . . . . .	4
2.3. Walshov spektar . . . . .	7
<b>3. Pregled područja</b>	<b>11</b>
<b>4. Pregled optimizacijskih algoritama</b>	<b>14</b>
4.1. Iterativni algoritam pretraživanja . . . . .	14
4.2. Metoda uspona na vrh . . . . .	15
4.3. Simulirano kaljenje . . . . .	15
4.4. Genetski algoritam . . . . .	16
4.5. Genetsko programiranje . . . . .	17
<b>5. Analiza problema</b>	<b>18</b>
5.1. Mjere vrednovanja rješenja . . . . .	19
5.2. Analiza susjedstva . . . . .	20
<b>6. Implementacija i rezultati</b>	<b>37</b>
6.1. Iterativni algoritam pretraživanja . . . . .	37
6.2. Metoda uspona na vrh . . . . .	38
6.3. Simulirano kaljenje . . . . .	40
6.4. Genetski algoritam . . . . .	42
6.5. Genetsko programiranje . . . . .	48
<b>7. Zaključak</b>	<b>52</b>
<b>Literatura</b>	<b>53</b>

# 1. Uvod

Kriptografija je grana računarske znanosti, bez čijih dostignuća bi današnji svakodnevni život izgledao nezamislivo drugačije. Većina se ljudi svakodnevno služi kriptografijom, premda toga nisu nužno ni svjesni. Bilo to korištenjem online bankarstva, online trgovine ili jednostavno bilo kojeg online servisa temeljenog na protokolu *HT-TPS*. Navedeni primjeri često se služe asimetričnim kriptografskim algoritmima kako bi klijent i poslužitelj razmijenili ključeve koje potom koriste u simetričnim kriptografskim algoritmima kojima se enkriptiraju njihove poruke.

Booleove funkcije predstavljaju jedan od sastavnih elemenata kriptografskih algoritama, poput S-kutija kod algoritma *DES* i *AES* [5]. Kako bi se postigla što veća sigurnost takvih sustava, potrebno je pomno odabratи korištene funkcije, ovisno o određenim svojstvima koje one posjeduju. Jedan od najpoznatijih napada je napad linearnom kriptoanalizom, prvi puta opisan u [7]. Kako bi se postigla veća otpornost na tu vrstu napada, od posebnog je značaja svojstvo nelinearnosti Booleove funkcije. Pored toga, za primjenu u kriptografskim algoritmima dodatno je važno i svojstvo balansiranosti. Osim navedenog, važna su i brojna druga svojstva poput svojstva korelacijske otpornosti (engl. *correlation immunity*), autokorelacije (engl. *autocorrelation*), algebarski stupanj (engl. *algebraic degree*), algebarske otpornosti (engl. *algebraic immunity*) i drugih opisanih u [4].

Primarni fokus ovog rada je pronađak Booleovih funkcija maksimalne nelinearnosti, kao i balansiranih Booleovih funkcija maksimalne nelinearnosti, pritom ne razmatrajući ostala svojstva. Posebna pozornost posvećena je balansiranim Booleovim funkcijama s 8 varijabli, koje se koriste u algoritmu *Achterbahn* [6], kao i u algoritmu *RAKAPOSNI* [1], za koje je pitanje maksimalne moguće nelinearnosti još uvijek otvoreno. Naime, najniža gornja granica nelinearnosti navedenih funkcija iznosi 118, dok je najviša do sad pronađena nelinearnost u iznosu 116.

Nastavak rada podijeljen je u 6 poglavlja. U drugom poglavlju objašnjen je pojам Booleove funkcije, kao i neka od njezinih svojstava korištena u dalnjem radu. Prikazani su i razni načini zapisa funkcije, kao i načini izračuna opisanih svojstava. Treće

poglavlje predstavlja pregled područja i opisuje dosadašnje pristupe pronalasku visoko nelinearnih Booleovih funkcija. U četvrtom poglavlju ukratko su objašnjene osnove optimizacijskih algoritama koji su korišteni, a to su iterativni algoritam pretrage, metoda uspona na vrh, simulirano kaljenje, genetski algoritam i genetsko programiranje. Detaljniji opis problema dan je u petom poglavlju. U istom poglavlju su navedene i objašnjene korištene mjere vrednovanja rješenja te je provedena analiza susjedstva. Implementacija optimizacijskih algoritama, kao i rezultati te analiza i usporedba rezultata prikazana je u šestom poglavlju.

## 2. Booleove funkcije i interesantna svojstva

Funkcija  $f$  u općenitom je smislu definirana kao preslikavanje člana jednog skupa, koji se naziva domena u točno određeni član drugog skupa, koji se naziva kodomena.

Ako su domena i kodomena skup realnih brojeva  $\mathbb{R}$ , funkcija  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  zove se **realna funkcija**.

**Booleova funkcija** je funkcija čija su domena i kodomena iz skupa  $\mathbb{B}$ , što je skup elemenata  $\mathbb{B} = \{0, 1\}$ .

### 2.1. Načini prikaza

Booleove funkcije moguće je jednoznačno definirati na brojne načine. Najjednostavniji i često korišteni način je prikaz **tablicom istinitosti**, što je prikaz vrijednosti funkcije za sve moguće kombinacije varijabli, poredanih prema leksikografskom poretku vrijednosti ulaznih varijabli. Tablica 2.1 predstavlja primjer prikaza Booleove funkcije 3 varijable tablicom istinitosti. Kako je redoslijed ulaznih varijabli u tablici

**Tablica 2.1:** Primjer prikaza Booleove funkcije tablicom istinitosti

$x_1$	$x_2$	$x_3$	$f(\vec{x})$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

istinitosti dogovoren, moguće je zapisivati samo zadnji stupac te istu funkciju prikazati kao 01101001, što je zapis korišten u ostatku rada.

Osim tablice istinitosti postoje i kanonski oblici zapisivanja Booleovih funkcija, što se ostvaruje zapisivanjem varijabli i operacija koje nad njima djeluju. Jedan od kanonskih oblika je suma minterma. Minterm je izraz za kojega funkcija poprima vrijednost 1 u samo jednom retku tablice istinitosti. Unutar minterma, varijable su povezane operacijom AND, dok su mintermi međusobno povezani operacijom OR. Zapis funkcije prikazane u tablici 2.1 kao suma minterma dan je u izrazu (2.1).

$$f(x_1, x_2, x_3) = \bar{x}_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 \bar{x}_3 + x_1 \bar{x}_2 \bar{x}_3 + x_1 x_2 x_3 \quad (2.1)$$

Iraz koji poprima vrijednost 0 u samo jednom retku tablice istinitosti zove se maksterm. Booleovu funkciju moguće je zapisati i kao umnožak maksterma, gdje su varijable unutar maksterma povezane operacijom OR, dok su makstermi međusobno povezani operacijom AND. Zapis funkcije iz prethodnih primjera umnoškom makssterma prikazan je u izrazu (2.2).

$$f(x_1, x_2, x_3) = (x_1 + x_2 + x_3) \cdot (x_1 + \bar{x}_2 + \bar{x}_3) \cdot (\bar{x}_1 + x_2 + \bar{x}_3) \cdot (\bar{x}_1 + \bar{x}_2 + x_3) \quad (2.2)$$

Prikazani kanonski oblici koriste ukupno 3 različite računske operacije; negaciju, AND i OR. Booleovu funkciju je moguće zapisati i uz korištenje samo 2 različite operacije, poput u zapisu u algebarskom normalnom obliku gdje se koriste samo operacije XOR i AND. Zapis funkcije iz prethodnih primjera u algebarskom normalnom obliku prikazan je u izrazu (2.3).

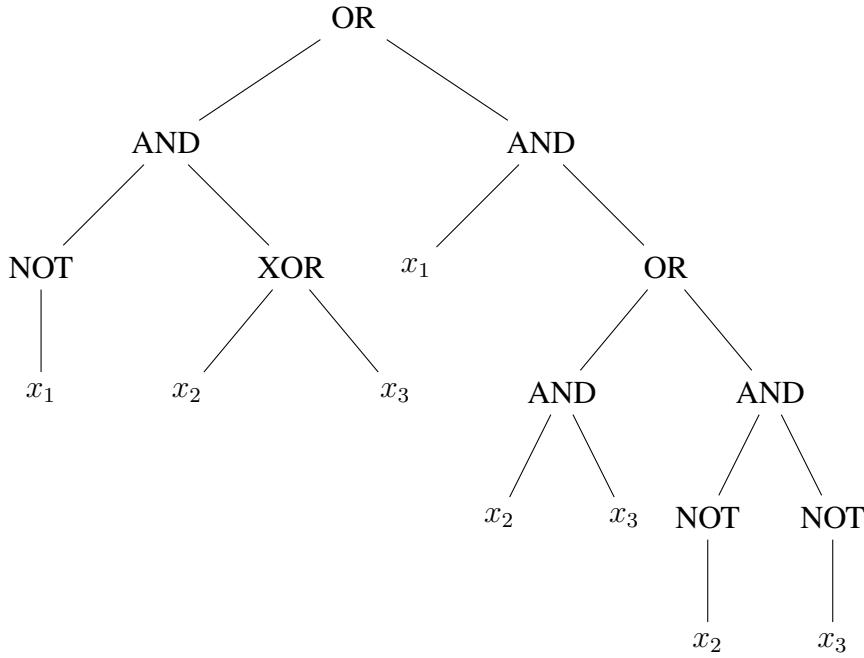
$$f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3 \quad (2.3)$$

Svi opisani načini prikaza Booleovih funkcija posjeduju zajedničko svojstvo da svakoj funkciji odgovara točno jedan mogući prikaz. To nije slučaj kod prikaza stablom operatora, gdje se funkcija grafički prikazuje kroz stablo, čiji s čvorovi operatori, a listovi varijable. Svaki operator iz stabla primjenjuje se na djecu tog čvora. Ovisno o korištenim operatorima, moguće je dobiti različite prikaze, a jedan od mogućih prikaza za prethodno korištenu funkciju dan je na slici 2.1. Stablo na slici izgrađeno je korištenjem operatora AND, OR, XOR i NOT.

## 2.2. Svojstva Booleovih funkcija

Realna funkcija  $f$  je **linearna funkcija** ako je oblika:

$$f(\vec{x}) = a_1 x_1 + a_2 x_2 + \cdots + a_n x_n, \quad (2.4)$$



**Slika 2.1:** Primjer prikaza Booleove funkcije stablom operatora

gdje su  $a_1, a_2, \dots, a_n \in \mathbb{R}$ ,  $x_1, x_2, \dots, x_n \in \mathbb{R}$  te  $f(\vec{x}) \in \mathbb{R}$ .

Realna funkcija je **afina funkcija** ako je oblika:

$$f(\vec{x}) = a_0 + a_1x_1 + a_2x_2 + \cdots + a_nx_n, \quad (2.5)$$

gdje su  $a_0, a_1, a_2, \dots, a_n \in \mathbb{R}$ ,  $x_1, x_2, \dots, x_n \in \mathbb{R}$  te  $f(\vec{x}) \in \mathbb{R}$ .

Slično kao i za realne funkcije, moguće je definirati linearnu i afinu Booleovu funkciju. Pritom se umjesto operacija zbrajanja koristi operacija logičkog isključivo ili (engl. *exclusive or*), odnosno XOR, dok se umjesto množenja koristi operacija konjunkcije, odnosno AND.

Booleova funkcija je linearна, ako je oblika:

$$f(\vec{x}) = a_1x_1 \oplus a_2x_2 \oplus \cdots \oplus a_nx_n, \quad (2.6)$$

gdje su  $a_1, a_2, \dots, a_n \in \mathbb{B}$ ,  $x_1, x_2, \dots, x_n \in \mathbb{B}$  te  $f(\vec{x}) \in \mathbb{B}$ . Primjer linearne Booleove funkcije dan je u prethodno korištenoj funkciji 01101001. To svojstvo je teško primijetiti korištenjem prikaza tablice istinitosti, ali ga je trivijalno za primijetiti korištenjem algebarskog normalnog oblika, kao što se vidi iz izraza (2.3).

Booleova funkcija je afina, ako je oblika:

$$f(\vec{x}) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus \cdots \oplus a_nx_n, \quad (2.7)$$

gdje su  $a_0, a_1, a_2, \dots, a_n \in \mathbb{B}$ ,  $x_1, x_2, \dots, x_n \in \mathbb{B}$  te  $f(\vec{x}) \in \mathbb{B}$ . Za svaku linearnu Booleovu funkciju vrijedi da je ona ujedno i afina Booleova funkcija, zbog čega je i

prethodno spomenuta funkcija afina. Obrat tvrdnje ne vrijedi, te je recimo funkcija 10010110 afina Booleova funkcija, ali nije linearna, što se vidi iz prikaza u algebarskom normalnom obliku danom u izrazu (2.8).

$$f(\vec{x}) = 1 \oplus x_1 \oplus x_2 \oplus x_3, \quad (2.8)$$

Booleova funkcija je **balansirana** ako u tablici istinitosti sadrži jednak broj vrijednosti 0 i vrijednosti 1. Funkcija prikazana tablicom 2.1 je balansirana jer u svojoj tablici istinitosti sadrži 4 nule i 4 jedinice. Funkcija 00100101 nije balansirana jer u tablici istinitosti ima 5 nula, a samo 3 jedinice.

Za dvije Booleove funkcije, definirana je **Hammingova udaljenost** kao broj bitova po kojima se njihove tablice istinitosti međusobno razlikuju. Tako je na primjer Hammingova udaljenost funkcija  $f_1 = 01001011$  i  $f_2 = 00100111$  iznosi 4, jer se funkcije razlikuju u drugom, trećem, petom i šestom bitu.

**Nelinearnost** Booleove funkcije definirana je kao minimalna Hammingova udaljenost od svih afinih Booleovih funkcija.

Booleova funkcija  $f$  je **Bent-funkcija** ako za dani broj varijabli posjeduje svojstvo maksimalne nelinearnosti te je jednako udaljena od svih afinih Booleovih funkcija. Bent funkcija postoji samo za paran broj varijabli, ali postoji i skupina semi-bent-funkcija definirana za neparan broj varijabli, što je opisano u knjizi [4].

Na primjeru funkcije s dvije varijable, postoji ukupno 8 afinih funkcija. To su redom:

$$\begin{array}{ll} f_1(\vec{x}) = 0 & f_5(\vec{x}) = 1 \\ f_2(\vec{x}) = x_1 & f_6(\vec{x}) = 1 \oplus x_1 \\ f_3(\vec{x}) = x_2 & f_7(\vec{x}) = 1 \oplus x_2 \\ f_4(\vec{x}) = x_1 \oplus x_2 & f_8(\vec{x}) = 1 \oplus x_1 \oplus x_2. \end{array}$$

Radi jednostavnosti izračuna Hammingove udaljenosti, u nastavku je dan prikaz afinih Booleovih funkcija dviju varijabli korištenjem tablice istinitosti.

$$\begin{array}{ll} f_1(\vec{x}) = 0000 & f_5(\vec{x}) = 1111 \\ f_2(\vec{x}) = 0011 & f_6(\vec{x}) = 1100 \\ f_3(\vec{x}) = 0101 & f_7(\vec{x}) = 1010 \\ f_4(\vec{x}) = 0110 & f_8(\vec{x}) = 1001 \end{array}$$

Na primjeru funkcije 1011, moguće je izračunati da Hammingova udaljenost te funkcije od navedenih afinih funkcija iznosi redom: 3, 1, 3, 3, 1, 3, 1 i 1. Najmanja udaljenost iznosi 1, zbog čega i nelinearnost funkcije 1011 iznosi također 1. Kako je

udaljenost od svake grupe afinih funkcija jednaka (pritom se istom grupom smatraju funkcije  $f_1$  i  $f_2$  za koje vrijedi da je  $f_1 = 1 \oplus f_2$ ), ova funkcija je ujedno i Bent-funkcija.

## 2.3. Walshov spektar

Walshova transformacija Booleove funkcije  $f$  definirana je kao:

$$W_f(\vec{w}) = \sum_{\vec{x} \in \mathbb{B}^n} f(\vec{x})(-1)^{\vec{w}\vec{x}}, \quad (2.9)$$

čime je definiran Walshov koeficijent  $W_f(\vec{w})$ . Skup Walshovih koeficijenata za sve  $\vec{w} \in \mathbb{B}^n$ , zove se **Walshov spektar** Booleove funkcije [4].

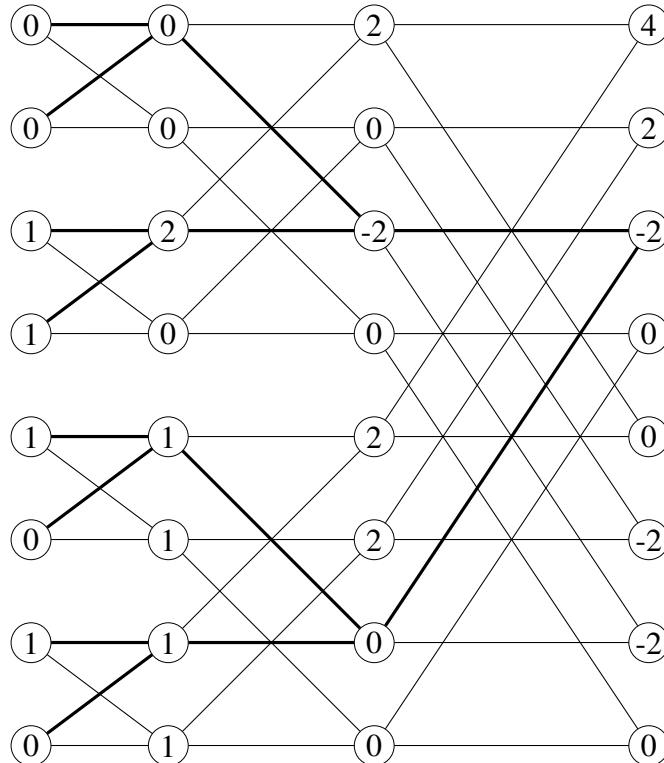
Svaki Walshov koeficijent predstavlja neočekivanu udaljenost (engl. *unexpected distance*) [23] Booleove funkcije  $f$  od pripadne linearne Booleove funkcije. Za svaki par Booleovih funkcija, očekivana vrijednost Hammingove udaljenosti iznosi pola od ukupnog broja bitova u tablici istinitosti, odnosno  $2^{n-1}$ . Neočekivana udaljenost definirana je kao razlika stvarne Hammingove udaljenosti dviju funkcija i njihove očekivane udaljenosti.

Na primjeru funkcije čija je tablica istinitosti 00111010, Walshov koeficijent za  $\vec{v} = 010$ , odgovara neočekivanoj udaljenosti od linearne funkcije  $f(\vec{x}) = x_1$ , čija je tablica istinitosti jednaka 00110011. Kako je riječ o funkcijama 3 varijable, očekivana udaljenost iznosi  $2^{3-1} = 4$ , dok je stvarna Hammingova udaljenost 2. Razlika te dvije vrijednosti daje  $-2$ , što je vrijednost neočekivane udaljenosti dane funkcije, kao i pripadni Walshov koeficijent.

Walshov spektar moguće je izračunati direktno korištenjem izraza (2.9), što je moguće izvesti u vremenskoj složenosti  $\mathcal{O}(n^2)$ . Isto je moguće izračunati u vremenskoj složenosti  $\mathcal{O}(n \log n)$  korištenjem brze Walshove transformacije (engl. *fast Walsh transform*). Taj se postupak provodi uzastopnom primjenom operatora koji za par brojeva  $(a, b)$ , računa novi par  $(a', b')$  prema izrazu:

$$(a', b') = (a + b, a - b). \quad (2.10)$$

Na slici 2.2 prikazan je cjelokupni postupak izračuna brze Walshove transformacije za prethodno prikazanu funkciju. Postupak se provodi u  $\log n$  koraka, a svaki je korak prikazan jednim stupcem u slici. Stupci na slici međusobno su povezani linijama koje prikazuju iz kojih brojeva je izračunat broj u pojedinom stupcu. Točnije, svaki broj je povezan s točno 2 broja iz prethodnog stupca, što ukazuje na to da je on doiven zbrojem ili razlikom ta dva broja, u skladu s operatorom (2.10). Prvi stupac



**Slika 2.2:** Primjer izračuna brze Walshove transformacije nad Booleovom funkcijom čija je tablica istinitosti 00111010

prikazuje Booleovu funkciju nad kojom se računa brza Walshova transformacija. U prvom koraku algoritma, tablica istinitosti se dijeli na grupe veličine 2, te se na svaku skupinu primjenjuje operator (2.10), čime se dobije rezultat prikazan u drugom stupcu slike. Sljedeći korak koristi grupe veličine 4 te se operator (2.10) primjenjuje redom na sve parove međusobno udaljene za 2. Općenito vrijedi da se u  $n$ -tom koraku brojevi grupiraju u grupe veličine  $2^n$ , a operator (2.10) se primjenjuje na one brojeve unutar grupe koji se nalaze na međusobnoj udaljenosti  $2^{n-1}$ . Postupak se ponavlja sve dok u zadnjem koraku ne preostane samo jedna grupa koja uključuje sve brojeve.

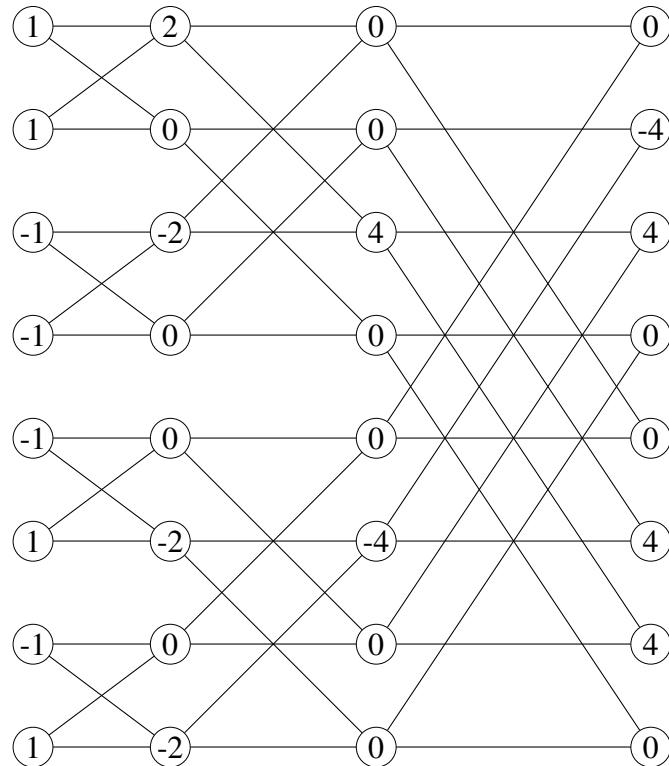
Iz opisanog postupka, kao i iz slike 2.2 moguće je primijetiti važno svojstvo, točnije kako svaki bit tablice istinitosti utječe na iznos pojedinog Walshovog koeficijenta. Navedeno svojstvo prikazano je podebljanom linijom na slici, čime su prikazani svi bitovi koji su uključeni u izračun trećeg Walshovog koeficijenta. Također se primjećuje kako vrijedi i obrat, odnosno da svaki pojedini bit utječe na sve vrijednosti Walshovog spektra.

Primjećuje se kako opisani postupak rezultira ispravnim Walshovim koeficijentima za cijeli Walshov spektar Booleove funkcije, osim za prvi koeficijent. Dobiven prvi koeficijent odgovara Hammingovoj težini Booleove funkcije, odnosno broju jedinica

u njezinoj tablici istinitosti. Ova činjenica predstavlja dodatno komplikiranje postupka izračuna vrijednosti željenih svojstava, s obzirom na to da je prvi broj potrebno obradivati na drugi način od ostatka spektra. U radu [23] taj je problem riješen tako da se na vrijednosti tablice istinitosti primijeni sljedeća transformacija:

$$x' = -1^x. \quad (2.11)$$

Na taj se način skup  $0, 1$  preslikava u skup  $1, -1$ . Postupak izračuna brze Walshove transformacije uz korištenje navedene transformacije ulaznih bitova prikazan je na slici 2.3. Novo dobiveni rezultati razlikuju se od prethodnih po predznaku i magnitudi. Us-



**Slika 2.3:** Primjer izračuna brze Walshove transformacije nad Booleovom funkcijom čija je tablica istinitosti  $00111010$  uz primjenu transformacije bitova

prkos tome, rezultati u prikladniji zbog prvog koeficijenta koji po svojstvima odgovara ostalima.

Pomoću Walshovog spektra računaju se ostala svojstva Booleove funkcije. Funkcija je balansirana ako je njezin prvi Walshov koeficijent jednak nuli [22]:

$$W_f(\vec{0}) = 0. \quad (2.12)$$

Nelinearnost  $N_f$  Booleove funkcije  $f$  određena je apsolutno najvećim Walshovim

koeficijentom prema izrazu [22]:

$$N_f = 2^{n-1} - \frac{1}{2} \max_{\vec{w} \in \mathbb{B}^n} |W_f(\vec{w})|. \quad (2.13)$$

Na primjeru funkcije prikazane u slici 2.3, absolutno najveći koeficijent iznosi 4, odnosno  $-4$  iz čega slijedi da je nelinearnost funkcije jednaka  $N_f = 2^{3-1} - \frac{1}{2}4$ , odnosno  $N_f = 2$ .

Za svaku Booleovu funkciju  $f$  s  $n$  varijabli vrijedi Parsevalova jednakost koja glasi:

$$\sum_{\vec{w} \in \mathbb{B}^n} (W_f(\vec{w}))^2 = 2^{2n}. \quad (2.14)$$

Iz toga slijedi da nelinearnost  $N_f$  Bent-funkcije s  $n$  varijabli iznosi:

$$N_f = 2^{n-1} - 2^{\frac{n}{2}-1}. \quad (2.15)$$

## 3. Pregled područja

S obzirom na već spomenutu važnost i primjenu Booleovih funkcija, postoji velik broj provedenih istraživanja o metodama njihova pronađenja. Booleove funkcije željenih svojstava moguće je stvoriti na tri konceptualno različita načina kao što je podijeljeno u radovima [20] i [21], a to su: nasumičnim generiranjem, algebarskim konstrukcijama ili heurističkom pretragom.

Nasumično generiranje predstavlja najjednostavniji i najbrži način stvaranja nove Booleove funkcije. Veliki nedostatak istoga je suboptimalnost rezultata, s obzirom na to da nasumične funkcije u većini slučajeva ne posjeduju loša svojstva.

S druge strane algebarska konstrukcija pruža mogućnost stvaranja funkcija poželjnih svojstava, bazirano na poznavanju funkcija manjeg broja varijabli koje zadovoljavaju određenja svojstva. Te se funkcije nadograđuju ili međusobno nadovezuju kako bi se dobila funkcija većeg broja varijabli, a također sa željenim svojstvima. Negativne strane algebarskih konstrukcija javljaju se u tome što unatoč dobrim rezultatima za funkcije određenih svojstava, ovi postupci pružaju mogućnost pronađenja samo određenih funkcija, te postoji mogućnost postojanja drugih, jednako dobrih ili boljih funkcija koje nije moguće dobiti na ovaj način. Dodatno loše svojstvo ovog postupka primjećuje se prilikom potrebe za stvaranjem većeg broja različitih funkcija, gdje se druge metode pokazuju prikladnijima. Konstrukcijski algoritmi nisu u fokusu ovog rada te nisu dodatno obrađeni, no široki pregled istih dan je u knjizi [4].

Treću skupinu algoritama predstavljaju heurističke metode pretrage. Ove metode pružaju mogućnost stvaranja velikog broja različitih funkcija pogodnih svojstava. Najčešći predstavnici ove skupine algoritama su algoritmi evolucijskog računanja, te postoje brojna istraživanja o efikasnosti pojedine vrste algoritma, kao i načina zapisa funkcije i evaluacije iste.

U radu [24] po prvi puta se spominje pojam bent-funkcija, odnosno funkcija koje za zadani broj varijabli imaju maksimalnu moguću nelinearnost, te su jednako udaljene od svake afine funkcije. U radu se također razmatra i Fourierova transformacija bent-funkcija, iz čega se dokazuje da bent-funkcije postoje samo za paran broj varijabli.

Mogućnost transformacije funkcije dodatno je istražena u [23] gdje se opisuje postupak brze Walshove transformacije (engl. ()Fast Walsh Transform). Objasnjeno je i tumačenje dobivenih Walshovih koeficijenata u vidu udaljenosti od njima pripadnim afinim funkcijama, iz čega slijedi mogućnost izračuna nelinearnosti funkcije na temelju tako dobivenih koeficijenata. Slično je prikazano i u radu [26], gdje se koristi Hadamardova matrica pomoću koje se računa Walshove transformacija. Prikazan je rekurzivni postupak izračuna matrice te pseudokôd izračuna transformacije funkcije i njene nelinearnosti.

Rad [13] baziran je na metodi uspona na vrh (engl. *hill climbing*). Korištena metoda kreće od nasumično generirane funkcije te za nju računa Walshove koeficijente. Kako bi se nelinearnost povećala, potrebno je smanjiti magnitudu koeficijenata najveće magnitude. To je ostvareno tako da se za svaki bit iz tablice istinitosti računa koji se koeficijenti mijenjaju promjenom tog bita te se na kraju odabire jedan od bitova koji smanjuje ukupnu nelinearnost. Ista metoda korištena je i u [12], gdje je dodatno proširena tako da umjesto promjene samo jednog bita računa promjene Walshovih koeficijenata za promjenu svakog para bitova. Na taj je način osigurano da se osim bent-funkcija mogu pronaći i balansirane Booleove funkcije visoke nelinearnosti. Rad [14] prikazuje usporedbu do tada najboljih poznatih rezultata s rezultatima dobivenim kombinacijom genetskog algoritma i metode uspona na vrh. Ista metoda dodatno je modificirana u radu [15] gdje se uvode slabi (engl. *weak requirements*) i jaki (engl. *strong requirements*) zahtjevi, gdje slabi zahtjevi garantiraju kako se trenutno rješenje neće pogoršati u sljedećoj iteraciji algoritma, dok jaki zahtjevi prepostavljaju poboljšanje rješenja.

Ponešto drugačiji pristup prikazan je u [16]. Predstavljen je algoritam koji na temelju Walshovih koeficijenata za danu Booleovu funkciju dolazi do najbliže bent-funkcije, a da pritom ispituje promjene nad samo polovicom bitova iz tablice istinitosti funkcije. Veliki nedostatak algoritma je mogućnost rada samo nad funkcijama koje imaju 4 varijable, te nemogućnost skaliranja istoga na funkcije većeg broja varijabli. [16] također ispituje mogućnosti iscrpne pretrage Booleovih funkcija korištenjem sklopa *FPGA*. Dobiveni rezultati ukazuju na visoko ubrzanje koje je moguće postići zahvaljujući paralelnom izračunu više funkcija korištenjem cjevovodne strukture (engl. *pipeline*). Unatoč dobivenom ubrzajućem pokazalo se kako nije moguće raditi iscrpnu pretragu za funkcije koje imaju više od 5 varijabli. Slični rezultati su dobiveni i u istraživanju [25], gdje autori također problemu pristupaju korištenjem sklopa *FPGA*. U radu se dodatno koristi i algebarski zapis Booleovih funkcija, za što je opisan postupak brze pretvorbe iz tablice istinitosti u algebarski zapis.

Autori u radu [2] pomoću simuliranog kaljenja (engl. *simulated annealing*) [9]

pretražuju funkcije koje zadovoljavaju više svojstava. Traženjem balansirane Booleove funkcije s 8 varijabli, dobiveno je rješenje nelinearnosti 116, što je jednako trenutno najboljem poznatom rješenju.

Genetski algoritam [8] također je često korišten pristup, te su napisani brojni radovi o mogućim operatorima križanja i mutacije, kao i o primjeni na problem pronalaska Booleovih funkcija visoke nelinearnosti. Rad [11] uspoređuje 3 različita operatora križanja primjenjiva na problem traženja balansiranih funkcija. [18] ispituje uspješnost genetskog algoritma korištenjem 3 različita operatora mutacije, kao i 3 različita operatorka križanja. U [22], autori testiraju utjecaj različitih funkcija dobrote te ispituju težinu pronalaska balansirane Booleove funkcije maksimalne nelinearnosti s 8 varijabli.

Pored navedenih, korišteno je i genetsko programiranje [10] u radovima [19], [17] i [27]. Autori u [19] dodatno koriste i kartezijsko genetsko programiranje te korištenjem skupa operatorka OR, XOR, AND, XNOR i AND kojemu je jedan od ulaza negiran ostvaruju bolje rezultate nego li genetskim algoritmom na problemu generiranja Skutija.

# 4. Pregled optimizacijskih algoritama

Računala se koriste za rješavanje velikog broja različitih problema. Ovisno o vrsti i težini problema razlikuju se i algoritmi rješavanja. Određene probleme moguće je riješiti egzaktnim postupcima koji u konačnom vremenu dolaze do traženog rješenja. Ponekad takvi postupci nisu poznati, ali je prostor mogućih stanja rješenja dovoljno mal da je do rješenja moguće doći iscrpnom pretragom svih stanja. Kako ni to nije uvijek moguće, postoji skupina algoritama koji se zovu metaheuristikama. Oni se primjenjuju na probleme u kojima je potrebno pronaći čim bolje, ali ne nužno i optimalno rješenje, a prostor stanja je takav da iscrpna pretraga nije moguća. Mnogi algoritmi iz ove skupine inspirirani su postupcima iz prirode, poput evolucije ili ponašanja skupine životinja i njihove međusobne suradnje (npr. mravi, ptice...). U nastavku poglavljia opisani su algoritmi korišteni u ovom radu, dok je detaljan pregled prirodom inspiriranih optimizacijskih algoritama moguće pronaći u knjizi [3].

## 4.1. Iterativni algoritam pretraživanja

Najjednostavniji korišteni optimizacijski algoritam je iterativni algoritam pretraživanja, čiji je pseudokôd dan u 4.1. Potrebno je odrediti susjedstvo nekog rješenja, nakon čega se pretraga svodi na odabir nasumičnog susjeda trenutnog rješenja. Postupak se ponavlja unaprijed određeni broj puta, ili dok nije zadovoljen uvjet zaustavljanja.

**Pseudokôd 4.1:** Iterativni algoritam pretraživanja

```
1 x = nasumično_rješenje
2 ponavlja n puta:
3     x = generiraj_susjeda(x)
4
5 vrati x
```

## 4.2. Metoda uspona na vrh

Iterativni algoritam pretraživanja bira nova rješenja potpuno nasumično, zbog čega je moguće da rješenje u sljedećem koraku bude lošije od prethodnog, kao i da algoritam ne pronalazi bolja rješenja. Taj je problem riješen metodom uspona na vrh tako da se novo rješenje prihvata samo onda kada je bolje od prethodnog rješenja. Pseudokôd opisanog algoritma prikazan je u 4.2.

**Pseudokôd 4.2:** Metoda uspona na vrh

```
1 x = nasumično_rješenje
2 ponavljaj n puta:
3     x' = generiraj_susjeda(x)
4     ako x' bolji_od x:
5         x = x'
6
7 vrati x
```

## 4.3. Simulirano kaljenje

Prethodni algoritam pruža poboljšanje u vidu toga da teži boljim rješenjima, no zato dolazi do drugih problema. Kako se novo rješenje prihvata samo ako je bolje od trenutnog rješenja, moguće je da algoritam pronađe lokalni optimum, odnosno rješenje koje je bolje od svih svojih susjeda, ali nije ujedno i najbolje rješenje koje postoji. Jedno moguće rješenje tog problema dano je algoritmom simuliranog kaljenja [9]. Osnovna ideja rada algoritma temelji se na principu kaljenja metala, što je postupak kojim se u metalurgiji postižu bolja svojstva metala. Prilikom kaljenja, metal se zagrijava na visoku temperaturu, koja se potom održava određeno vremensko razdoblje, nakon čega se metal polako i postepeno hlađi. Uz visoku temperaturu metala, atomi posjeduju visoku energiju, što povećava njihovu gibljivost, a zbog čega je vjerojatnost prelaska atoma u kristalnu konfiguraciju više energetske razine također visoka. Općenito vrijedi da je vjerojatnost prijelaza iz energetskog stanja  $E_1$  u više energetsko stanje  $E_2$  određena Boltzmannovom jednadžbom danom u izrazu (4.1)[3]:

$$P(\Delta E) = e^{\frac{-\Delta E}{k \cdot t}} \quad (4.1)$$

gdje je  $k$  Boltzmannova konstanta koja iznosi  $1.3806 \cdot 10^{-23} \frac{m^2 kg}{s^2 K}$ . Hlađenjem metala smanjuje se vjerojatnost prelaska atoma u više energetsko stanje te u konačnici atomi tvore pravilnu kristalnu strukturu niske energetske razine, zbog čega metal ima poželjna pozitivna svojstva.

Sličan princip primjenjuje se u optimizacijskom algoritmu simuliranog kaljenja. Potrebno je definirati funkciju kazne, odnosno funkciju koja svakom rješenju dodjeljuje brojčanu vrijednost tako da bolje rješenje ima manju vrijednost. To odgovara energetskoj razini kristalne strukture metala, koja također teži minimalnoj energetskoj razini. Potom se definira početna temperatura, koja se u kontekstu algoritma koristi samo za potrebe izračuna vjerojatnosti prijelaza u lošije stanje. Temperatura se postepeno smanjuje, a za svaku temperaturu se unaprijed određen broj puta generira novo rješenje iz susjedstva trenutnog rješenja, koje se prihvata kao trenutno s određenom vjerojatnošću, analogno izrazu (4.1). Pseudokôd algoritma dan je u 4.3.

#### Pseudokôd 4.3: Simulirano kaljenje

```

1 x = nasumično_rješenje
2 t = početna_temperatura
3
4 ponavljam_n_puta:
5   ponavljam_m_puta:
6     x' = generiraj_susjeda(x)
7     ako x' bolji_od x:
8       x = x'
9     inače:
10      p = exp((f(x) - f(x')) / t)
11      uz_vjerojatnost p:
12        x = x'
13      t = smanji_temperaturu(t)
14
15 vrati x

```

## 4.4. Genetski algoritam

Simulirano kaljenje dokazano konvergira u globalno najbolje rješenje uz uvjet da se unutarnja i vanjska petlja algoritma izvode beskonačno puta, što u praksi nije moguće. Kako bi se smanjila mogućnost dolaska u lokalni optimum pamti se veći broj rješenja, a najčešće korišteni algoritam koji koristi populaciju rješenja je genetski algoritam [3], čija motivacija potječe iz biološke evolucije. U pseudokôdu 4.4 prikazan je generacijski genetski algoritam koji radi tako da ima listu rješenja koja predstavljaju trenutnu populaciju, odnosno jednu generaciju algoritma. Kako bi se osiguralo da najbolje rješenje ostane prisutno i u sljedećoj generaciji, najboljih  $k$  rješenja kopira se u sljedeću generaciju, što se zove elitizam. Potom se stvaraju nova rješenja (jedinke) koje ulaze u novu generaciju, sve dok broj jedinki u novoj populaciji ne postane jednak

broju jedinki u početnoj populaciji. Nove jedinke stvaraju se tako da se odabiru dvije jedinke iz prethodne generacije. Pritom bolje jedinke imaju veću vjerojatnost biti odabrane, čime se postiže da algoritam teži boljim rješenjima. Odabrane jedinke (roditelji) se potom križaju, što je postupak u kojemu se kombinacijom dviju jedinki stvara nova jedinka. Novostvorena jedinka se potom mutira, odnosno nasumično mijenja, kako bi se u populaciji ostvarila genetska raznolikost. Postupak se ponavlja unaprijed određen broj generacija, ili do trenutka kada nove generacije prestaju biti bolje od prethodnih.

#### Pseudokôd 4.4: Genetski algoritam

```
1 x = nasumična_populacija
2
3 ponavljam n puta:
4     x' = prazna_populacija
5     dodaj najboljih k iz x u x'
6
7     dok x' < x:
8         a, b = odaberi_roditelje(x)
9         c = križaj(a, b)
10        mutiraj(c)
11        evaluiraj(c)
12        dodaj c u x'
13
14 vrati najbolji iz x
```

## 4.5. Genetsko programiranje

Genetsko programiranje [10] zapravo je samo podvrsta genetskog algoritma, gdje je kromosom, odnosno jedinka zapravo program koji predstavlja rješenje problema, ili čijim se izvođenjem dolazi do rješenja. Najčešće korištena struktura za prikaz rješenja je stablo. Čvorovi stabla predstavljaju operacije koje se izvode, dok su djeca pojedinog čvora operandi. Primjer jednog ovako definiranog stabla je zapis Booleove funkcije stablom operatora, što je prikazano na slici 2.1.

## 5. Analiza problema

Pronalazak visoko nelinearnih Booleovih funkcija je poznat problem na kojemu je provedeno mnogo istraživanja. Težina problema dolazi od veličine prostora stanja, koji raste super eksponencijalno s porastom broja varijabli. Općenito vrijedi da za  $n$  varijabli Booleova funkcija ima  $2^n$  elemenata u tablici istinitosti te postoji  $2^{2^n}$  različitih Booleovih funkcija. Eksponencijalni rast tablice istinitosti također predstavlja implementacijske probleme zbog memorijskih zahtjeva za zapis istoga, ali i vremenskih zahtjeva za računanje. Osim velikog prostora stanja, problem je i količina visoko nelinearnih Booleovih funkcija, odnosno to što je njihov udio u svim Booleovim funkcijama određenog broja varijabli sve manji s porastom broja varijabli. Tako primjerice za 4 varijabli postoji ukupno 65 536 Booleovih funkcija, od čega su 896 funkcije Bent-funkcije, što čini 1.3% svih funkcija. Sa 6 varijabli postoji ukupno  $1.84 \times 10^{19}$  različitih Booleovih funkcija, od čega su 5 425 430 528 funkcije Bent-funkcije, što čini svega  $2.94 \times 10^{-8}\%$  svih funkcija. Za 8 varijabli ovaj omjer postaje još manji te je samo  $8.155 \times 10^{-44}\%$  [16]. Za broj Bent-funkcija vrijedi ograda dana u izrazu (5.1) [4].

$$\left(2^{\frac{n}{2}}\right)! 2^{2^{\frac{n}{2}}} \leq \#bent \leq 2^{2^{n-1} - \frac{1}{2}(\frac{n}{2})}. \quad (5.1)$$

Ukupan broj Booleovih funkcija, kao i ograde i stvaran broj Bent-funkcija za određene brojeve varijabli dan je u tablici 5.1.

**Tablica 5.1:** Broj Booleovih funkcija i ograde za Bent-funkcije (podaci preuzeti iz [4])

$n$	donja ograda	broj Bent-funkcija	gornja ograda	broj Booleovih funkcija
2	8	8	8	8
4	384	896	2 048	65 536
6	$2^{23.3}$	$2^{32.3}$	$2^{38}$	$2^{64}$
8	$2^{95.6}$	$2^{106.3}$	$2^{129.2}$	$2^{256}$
10	$2^{262.2}$	?	$2^{612}$	$2^{1024}$

## 5.1. Mjere vrednovanja rješenja

Uz sve spomenute probleme prilikom pronalaska nelinearnih Booleovih funkcija, dodatan problem javlja se prilikom vrednovanja rješenja. Korišteni optimizacijski algoritmi temelje se na tome da svakom rješenju dodjeljuju vrijednost u skladu s time koliko je ono dobro ili loše te pomoći te informacije pronalaze nova, potencijalno bolja rješenja. Ovisno o vrsti mjere vrednovanja, one mogu biti ili mjere odnosno funkcije dobrote ili funkcije kazne. Funkcije dobrote rješenju pridjeljuju vrijednost tako da ono rješenje koje je bolje dobije veću vrijednost, dok funkcije kazne veću vrijednost dodjeljuju onom rješenju koje je lošije. Ovisno o tome koja vrsta funkcija vrednovanja se koristi, optimizacijski problem se postavlja kao maksimizacijski u slučaju funkcija dobrote, odnosno minimizacijski u slučaju funkcija kazne.

Najjednostavnija korištena mjera vrednovanja rješenja je upravo iznos nelinearnosti funkcije  $N_f$ . Navedena mjera pruža jednostavno tumačenje uspješnosti rješenja. Problem ovako definirane funkcije dobrote je u tome što velik broj različitih Booleovih funkcija posjeduje jednaku razinu nelinearnosti, zbog čega je postoje velika područja u kojima optimizacijski algoritmi nemaju informaciju o tome napreduju li ka boljem rješenju.

Sljedeća korištena mjera vrednovanja je nadogradnja prethodne tako da se osim nelinearnosti Booleove funkcije također vrednuje i iznos drugog po veličini Walshovog koeficijenta. Ideja ove mjere je iskoristiti svojstvo Walshovih koeficijenata da predstavljaju udaljenosti od pojedinih afnih Booleovih funkcija. Na taj se način od više funkcija jednakе nelinearnosti prednost daje onima čija je udaljenost od sljedeće najbliže afine funkcije maksimalna. Time je povećana vjerojatnost da povećavanje udaljenosti od trenutno najbliže afine funkcije rezultira i povećanjem nelinearnosti, jer neovisno o tome je li udaljenost od sljedeće najbliže afine funkcije povećana ili smanjena, ona je minimalna s obzirom na do tada pronađena rješenja. Dodatna motivacija za ovako definiranom mjerom vrednovanja je u tome što je poznato da Bent-funkcije imaju sve Walshove koeficijente jednakog iznosa, što znači da je prilikom traženja Bent funkcija cilj minimizirati sve Walshove koeficijente, gledano po njihovoј apsolutnoj vrijednosti.

Treća korištena mjera vrednovanja dana je izrazom (5.2), koja je korištena u mnogim radovima, poput: [22], [21], [2] i [20].

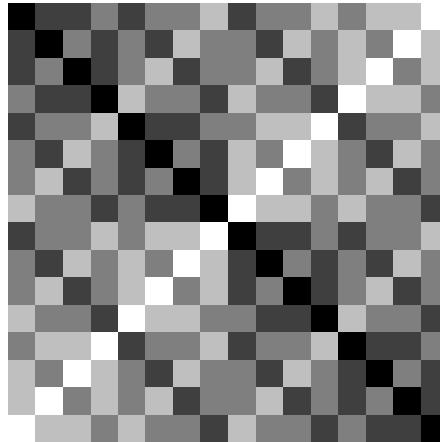
$$cost = \sum_{\vec{w} \in \mathbb{B}^n} \left| \left| W_f(\vec{w}) \right| - X \right|^R \quad (5.2)$$

Navedena mjera je funkcija kazne, koja se temelji na sličnom principu prethodne mjere vrednovanja, samo što se umjesto najveća dva koeficijenta Walshovog spektra koriste

svi koeficijenti. Funkcija posjeduje poželjna svojstva u vidu kažnjavanja velikih vrijednosti Walshovih koeficijenata, uz mogućnost utjecaja na način kažnjavanja kroz parametre  $X$  i  $R$ . Premda spomenuti parametri predstavljaju mogućnost podešavanja funkcije kazne, oni također predstavljaju i problem s obzirom na to da uvode potrebu za pomnim odabirom istih te ispitivanje njihovog utjecaja na rad optimizacijskih algoritama. Za potrebe ovog rada korištene su vrijednosti  $R = 3$  te  $X = 4$ , koje su se pokazale kao najuspješnije u dosadašnjim radovima [2].

## 5.2. Analiza susjedstva

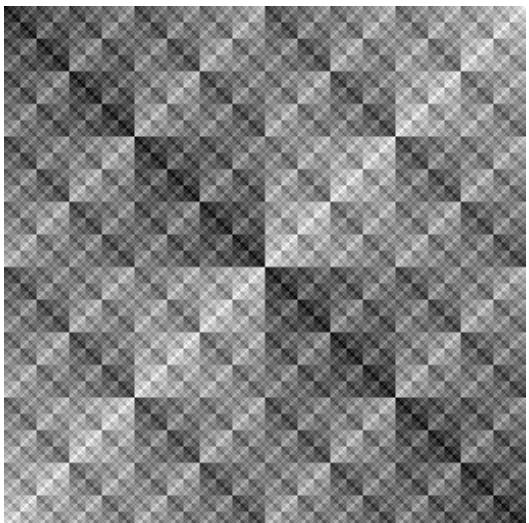
Prije nego li se primijene optimizacijski algoritmi, potrebno je proučiti problem, ne bi li se pronašle određene pravilnosti ili karakteristike rješenja, a na temelju čega se može unaprijediti rad optimizacijskih algoritama. S tim ciljem napravljena je vizualizacija susjedstva tako da je napravljena dvodimenzionalna matrica, gdje element  $a_{i,j}$  bojom odgovara Hammingovoj udaljenosti Booleovih funkcija  $f_i$  i  $f_j$ . Opisani prikaz za slučaj funkcija s dvije varijable dan je na slici 5.1. Konačna boja polja u



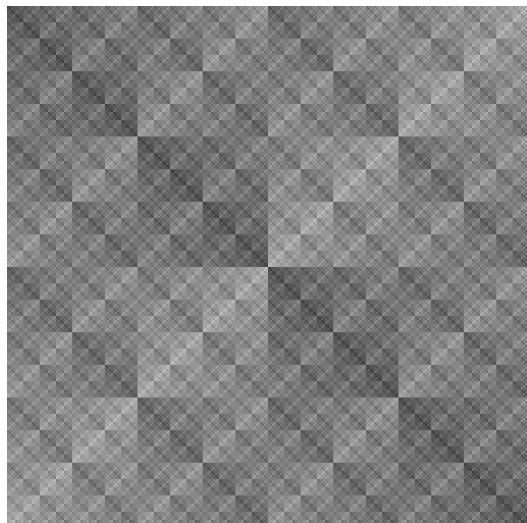
**Slika 5.1:** Prikaz susjedstva za funkcije dviju varijabli

matrici dobivena je skaliranjem Hammingovih udaljenosti na raspon od 0 do 255, te je tako dobiven broj korišten kao intenzitet sive boje. Tako je primjerice na slici mogće primjetiti kako je glavna dijagonala crna, što je posljedica toga da ona prikazuje udaljenost funkcija od samih sebe, a ta udaljenost iznosi 0, što rezultira crnom bojom. Osim navedene glavne dijagonale, primjećuje se pravilnost i na sporednoj dijagonali, koja je naime u potpunosti bijela, što je posljedica toga da su to funkcije na najvećoj mogućoj udaljenosti.

Na slici 5.2 grafički su prikazane udaljenosti funkcija tri varijable, dok slika 5.3



**Slika 5.2:** Prikaz susjedstva za funkcije tri varijable

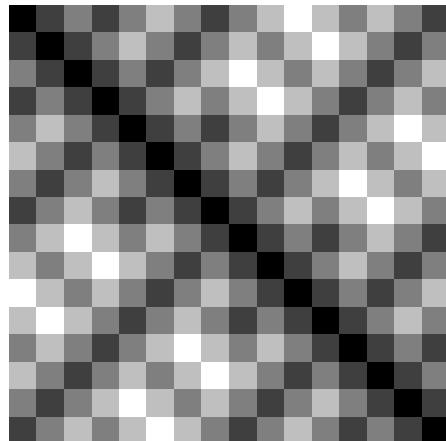


**Slika 5.3:** Prikaz susjedstva za funkcije četiri varijable

prikazuje udaljenosti za funkcije četiri varijable. Kako postoji ukupno 65 536 različitih Booleovih funkcija četiri varijable, nije moguće prikazati udaljenosti na prethodno opisano način jer bi tako dobivena slika zauzimala 4GB prostora. Kako bi dimenzije slike bile manje, umjesto da jedan slikovni element predstavlja udaljenost dviju funkcija, napravljeno je da jedan slikovni element predstavlja udaljenosti 8 funkcija međusobno. To je ostvareno tako da svaki element zapravo prikazuje aritmetičku sredinu onoga što bi prikazivalo područje od  $8 \times 8$  elemenata u slici pune razlučivosti.

Na prikazanim slikama moguće je primijetiti određene pravilnosti. Kao što je već spomenuto za sliku 5.1, i na primjerima funkcija s većim brojem varijabli vrijedi to da je glavna dijagonala crna, dok je sporedna dijagonala bijela. Osim toga, primjećuje se kako je sliku moguće podijeliti u kvadrate, tako je na primjer svaku sliku moguće podijeliti na 4 kvadrata, svaki od tih kvadrata također je moguće podijeliti na 4 daljnja kvadrata, i tako dalje. Premda navedeno može djelovati kao potencijalno korisno svojstvo, isto je posljedica redoslijeda kojim su zapisane Booleove funkcije. Na primjeru slike 5.2, postoji ukupno 256 različitih Booleovih funkcija. Prvih 128 funkcija ima 0 kao prvi bit tablice istinitosti, dok drugih 128 ima 1 na mjestu prvog bita. Ta promjena bita je razlog zbog kojeg je sliku moguće podijeliti na kvadrate upravo na mjestu gdje se mijenja vrijednost tog bita. Manje kvadrate unutar slike moguće je objasniti na isti način, naime bit na drugom mjestu u tablici istinitosti imat će vrijednost 0 za prve 64 funkcije, nakon čega će imati vrijednost 1 za sljedeće 64 funkcije, nakon čega će opet imati vrijednost 0 i tako dalje. Ista pravilnost vrijedi za sve bitove tablice istinitosti, jedino što se frekvencija promjene vrijednosti bita povećava s većim indeksima bita u

tablici istinitosti.

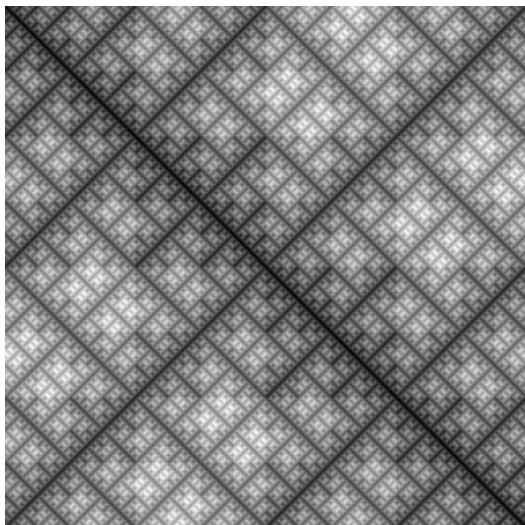


**Slika 5.4:** Prikaz susjedstva za funkcije dviju varijabli, za funkcije generirane korištenjem Grayevog kôda

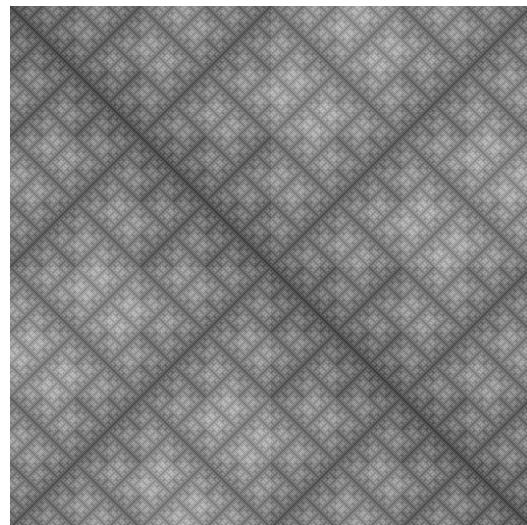
Kako su sve uočene pravilnosti u prethodnim slikama posljedica načina na koji su generirane Booleove funkcije, isproban je grafički prikaz u kojemu funkcije nisu zapisane leksikografskim redom u odnosu na njihove tablice istinitosti, već korištenjem Grayevog kôda. Grayev kôd je redoslijed zapisa Booleovih funkcija u kojemu se dvije susjedne funkcije uvijek razlikuju za točno jedan bit. Motivacija za prikazivanjem funkcija u ovom redoslijedu dolazi iz toga što su susjedne funkcije međusobno slične, zbog čega ne bi trebalo dolaziti do jednakih uzoraka udaljenosti kao u prethodnom zapisu. Primjer na funkciji dvije varijable prikazan je na slici 5.4.

Slika 5.5 prikazuje udaljenosti funkcija tri varijable, dok slika 5.6 prikazuje udaljenosti funkcija četiri varijable. Primjećuje se kako i u ovom redoslijedu zapisa funkcija postoje određene pravilnosti, točnije tamnija boja na dijagonalama. Kao i kod leksikografskog poretku, glavna dijagonala je najtamnija jer prikazuje udaljenost funkcije od sebe same te je vrijednost tog polja 0. Sporedna dijagonala je također izrazito tamna, što je ponovno uzrokovano redoslijedom generiranja funkcija. Naime, sporedna dijagonala prikazuje udaljenost dviju funkcija na međusobnoj udaljenosti 1. Sličan uzorak se ponavlja i na dijagonalama manjih kvadrata, dobivenih podjelom slike na četiri dijela. Kako ni u ovom slučaju grafički prikaz nije omogućio pronalazak pravilnosti u svojstvima funkcija, osim redoslijeda kojim su generirane, ovi pristupi nisu korišteni u nastavku rada.

Osim grafičkih prikaza susjedstva, korisno je pogledati i funkcije grupirane tako da je za Booleovu funkciju prikazan popis svih Bent-funkcija koje su joj najbliže. Motivacija tog prikaza je pronaći zajedničke dijelove najbližih Bent-funkcija te ustvrditi



**Slika 5.5:** Prikaz susjedstva za funkcije tri varijable, za funkcije generirane korištenjem Grayevog kôda



**Slika 5.6:** Prikaz susjedstva za funkcije četiri varijable, za funkcije generirane korištenjem Grayevog kôda

je li moguće odrediti koje bitove tablice istinitosti treba ili ne treba mijenjati kako bi se dobila funkcija veće nelinearnosti. Prikaz Booleovih funkcija i njima najbližih Bent-funkcija za funkcije dvije varijable dan je u tablici 5.2 U danoj tablici primjećuje se pravilnost. Točnije, svaka od prikazanih funkcija ima točno četiri Bent-funkcije koje su joj najbliže. Usporede li se funkcije međusobno, primjećuje se da svaka funkcija ima par, odnosno drugu funkciju takvu da prva funkcija ima četiri njoj najbliže Bent-funkcije, dok druga funkcija ima druge četiri Bent-funkcije koje su joj najbliže. Navedeni parovi funkcija su prikazani jedan ispod drugoga u tablici 5.2 te se primjećuje kako je jedna funkcija zapravo negacija druge funkcije. Isto se primjećuje i za

**Tablica 5.2:** Popis Booleovih funkcija dvije varijable i njima najbližih Bent-funkcija

0000	0001	0011	0001	0110	1110	0101	0001
0010		0010		0010		1101	
0100		1011		0100		0100	
1000		0111		0111		0111	
1111	1110	1100	1110	1001	0001	1010	1110
1101		1101		1101		0010	
1011		0100		1011		1011	
0111		1000		1000		1000	

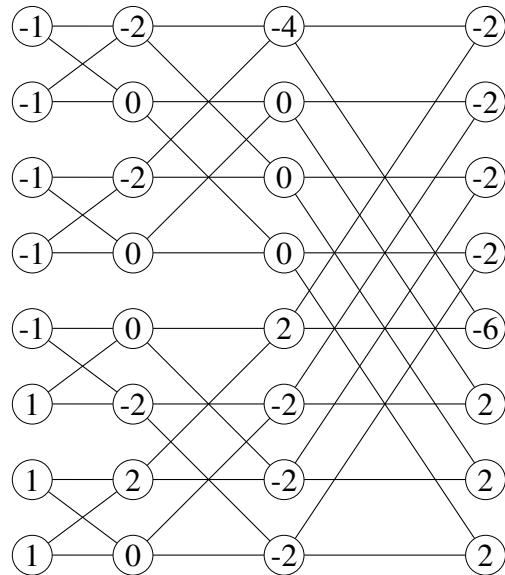
Bent-funkcije koje su najbliže navedenim funkcijama, gdje vrijedi da je Bent-funkcije najbliže prvoj funkciji moguće dobiti negiranjem Bent-funkcija najbližih drugoj funkciji. Opisano svojstvo proizlazi iz definicije afnih Booleovih funkcija, dane u izrazu (2.5). Ovisno o koeficijentu  $a_0$ , postoje dvije različite affine funkcije, jedna u kojoj je koeficijent  $a_0$  jednak nuli, druga u kojoj je jednak jedinici. Kako je koeficijent  $a_0$  povezan operatorom XOR, on zapravo uzrokuje da za svaku afinu funkciju  $f$ , postoji affina funkcija  $\bar{f}$  koja je jednaka negiranoj funkciji  $f$ . Upravo iz tog razloga slijedi svojstvo da za svaku grupu funkcije i njoj pripadnih Bent-funkcija, postoji pripadna grupa dobivena negiranjem funkcija prethodne grupe. Koristeći ovu činjenicu moguće je prostor pretrage smanjiti za pola, s obzirom na to da je dovoljno provjeriti jednu funkciju te ona ujedno daje informacije i za njoj komplementarnu funkciju.

Booleove funkcije dvije varijable korisne su za potrebe demonstracije jer ih je moguće sve prikazati, no ne pružaju vjeran uvid u problem, s obzirom na to da je točno pola funkcija affino, dok su preostale funkcije Bent-funkcije, s udaljenošću 1 od najbliže affine funkcije. Isti način grupiranja zato je primijenjen na Booleove funkcije četiri varijable. Primjećuje se kako u ovom slučaju funkcije imaju različit broj Bent-funkcija koje su im najbliže, no postoje određene pravilnosti.

Najviše susjednih Bent-funkcija imaju upravo affine funkcije te vrijedi da svaka affina funkcija ima točno 448 susjednih Bent-funkcija, što je pola od ukupnog broja Bent-funkcija četiri varijable. Kod afnih funkcija vrijedi da promjena bilo kojeg bita tablice istinitosti povećava nelinearnost za jedan, zbog čega ove funkcije nisu korisne za daljnje razmatranje. Za funkcije čija je nelinearnost jednaka jedan, moguće je primjetiti kako imaju uvijek jednak broj Bent-funkcija koje su im najbliže, točnije 168. Zanimljivo je primjetiti kako je ovdje moguće uočiti pravilnosti, tako primjerice za funkciju 01010101010100 vrijedi da promjena bilo kojeg bita vodi prema funkciji veće nelinearnosti, osim promjene zadnjeg bita, što dovodi do affine funkcije čija je nelinearnost jednaka nuli. Primjećuje se kako sve ostale funkcije također dolaze u pravilnim grupama, tako postoje funkcije koje imaju ukupno 64 najbliže Bent-funkcije, funkcije koje imaju 56, 16, 4 i 1 najbližu Bent-funkciju. Za svaku grupu definiranu na opisan način vrijedi da postoji skupina bitova zajednička svim Bent-funkcijama, što upućuje na postojanje skupina bitova unutar svake funkcije koju se ne smije mijenjati kako bi se postigla maksimalna moguća nelinearnost. Zanimljivo je uočiti postojanje funkcija koje su svojevrsni lokalni optimum. Na primjer, funkcija 010100000001010 ima nelinearnost 4, a promjenom bilo kojeg bita navedene funkcije dobiva se funkcija čija nelinearnost iznosi 3. Unatoč tome, navedena funkcija nije Bent-funkcija, odnosno postoje funkcije čija je nelinearnost veća.

Od iznimnog je interesa razviti postupak pronašlaska bitova koje treba te bitova koje ne treba mijenjati, kako bi se ubrzali postupci pretrage. U radovima [13] i [15] opisan je postupak temeljen na brzoj Walshovoj transformaciji. Opisani postupak zasniva se na tome da je jednom kad se izračunaju Walshovi koeficijenti za određenu Booleovu funkciju poznato koje koeficijente treba promijeniti i na koji način kako bi se ostvarila Booleova funkcija veće nelinearnosti. Jedno moguće rješenje je da se za svaku funkciju nastalu promjenom samo jednog bita tablice istinitosti ponovno računa cijeli postupak brze Walshove transformacije, što je moguće napraviti u vremenskoj složenosti  $\mathcal{O}(n \log n)$ , gdje je  $n$  broj bitova u tablici istinitosti. Kako je to potrebno napraviti za svaku moguću promjenu, odnosno za svaki bit tablice istinitosti, ukupna složenost iznosi  $\mathcal{O}(n^2 \log n)$ . Umjesto toga, autori predlažu postupak u kojem nakon promjene jednog bita tablice istinitosti nije potrebno provoditi cijelu Walshovu transformaciju, već samo ažurirati one brojeve koji se mijenjaju promjenom tog bita, što je moguće izvesti u vremenskoj složenosti  $\mathcal{O}(n)$ , čime je provjeru svih bitova moguće ostvariti u složenosti  $\mathcal{O}(n^2)$ .

Inspirirano opisanim postupkom, razmotreni su Walshovi koeficijenti Booleove funkcije, te način kako pomoći njih ustanoviti koje bitove je potrebno promijeniti. Na primjeru funkcije 00000111, Walshovi koeficijenti iznose  $-2, -2, -2, -2, -6, 2, 2, 2$ , što je prikazano na slici 5.7. Slika ujedno prikazuje i međurezultate prilikom raču-



**Slika 5.7:** Prikaz izračuna Walshovih koeficijenata za funkciju tri varijable

nanja koeficijenata, te su brojevi koji ovise jedan o drugome na slici povezani linijom. Walshovi koeficijenti računati su u tri koraka, gdje je svaki korak redom prikazan od lijeva prema desno. Istu transformaciju moguće je provesti i reverzno, od desna prema

lijevo, čime se određuje koji bit tablice istinitosti kako utječe na pojedini Walshov koeficijent. Kako bi se povećala nelinearnost funkcije, potrebno je smanjiti magnitudu najvećeg Walshovog koeficijenta, što je u ovom slučaju peti po redu koeficijent, odnosno  $-6$ .

Općenito vrijedi da za svaki koeficijent znamo kako promjena brojeva iz kojih je dobiven utječe na njega. Isto tako moguće je odrediti na koji način treba promijeniti određene brojeve iz međurezultata prethodnog koraka, kako bi se koeficijent promijenio na željeni način. Slučajevi u kojima je potrebno povećati broj, prikazani su na slici



**Slika 5.8:** Način na koji je potrebno promijeniti rezultate prethodnog koraka ako je potrebno povećati vrijednost trenutnog broja

5.8. Kako se koeficijenti računaju korištenjem operatora (2.10), potrebno je razlikovati dva slučaja, kada je koeficijent dobiven kao zbroj prethodna dva broja, što je prikazano slikom 5.8a te kada je koeficijent dobiven kao razlika prethodna dva broja, što je prikazano slikom 5.8b. U prvom slučaju, kako bi se povećao zbroj, potrebno je povećati barem jedan od pribrojnika. Kod oduzimanja, kako bi se povećala razlika, potrebno je ili povećati umanjenik, ili smanjiti umanjitelj, što odgovara prikazu na slici 5.8b.

Slučajevi u kojima je potrebno smanjiti broj, prikazani su na slici 5.9. Sukladno prethodnom primjeru, vrijedi da je potrebno smanjiti vrijednost bilo kojeg od pribrojnika kako bi se smanjila vrijednost zbroja, što je prikazano na slici 5.9a. Kako bi se smanjila vrijednost razlike, potrebno je ili smanjiti umanjenik, ili povećati umanjitelj, što je prikazano na slici 5.9b.

Svaki Walshov koeficijent moguće je ili smanjiti ili povećati, što su slučajevi koji su prikazani prethodnim primjerima. Valja primijetiti kako općenito za međurezultate postoji i treće moguće stanje, točnije stanje kontradikcije. To je stanje dobiveno tako da je za isti broj dobiveno da ga treba istovremeno i povećati i smanjiti kako bi se povećala nelinearnost funkcije. Posljedica toga je da vrijednost ovog broja ne smije biti promijenjena, jer bi i povećanje i smanjivanje vrijednosti rezultirali smanjenjem nelinearnosti. Načine na koje je potrebno mijenjati vrijednosti međurezultata u prethodnom koraku za slučajeve u kojima je međurezultat u trenutnom koraku u stanju



**Slika 5.9:** Način na koji je potrebno promijeniti rezultate prethodnog koraka ako je potrebno smanjiti vrijednost trenutnog broja

kontradikcije prikazani su na slici 5.10. Budući da se vrijednost trenutnog broja ne

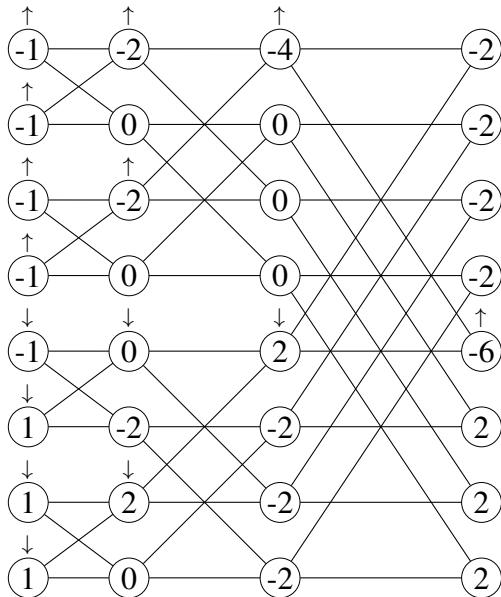


**Slika 5.10:** Način na koji je potrebno promijeniti rezultate prethodnog koraka ako je u trenutnom koraku pronađena kontradikcija

smije promijeniti, a da je moguće promijeniti vrijednost samo jednog od brojeva iz prethodnog koraka, slijedi da nije dozvoljeno mijenjati vrijednost niti jednog od brojeva iz prethodnog koraka, neovisno o tome radi li se o zbroju (prikazano slikom 5.10a) ili razlici (prikazano slikom 5.10b). U nedostatku poznatog izraza, u dalnjem tekstu je za opisani algoritam korišten naziv algoritam propagacije Walshovih koeficijenata unatrag. Korištenjem ovog algoritma moguće je pronaći Booleovu funkciju veće nelinearnosti u vremenskoj složenosti  $\mathcal{O}(n \log n)$ .

Navedena pravila primjenjena su na funkciju prikazanu u slici 5.7, tako da je za apsolutno najveći koeficijent (u ovom slučaju  $-6$ ) postavljeno da ga treba povećati te je to pravilo propagirano po koracima unatrag, čime je dobiveno na koji način je potrebno promijeniti svaki od bitova tablice istinitosti kako bi se povećala nelinearnost funkcije. Rezultati postupka prikazani su na slici 5.11. Primjećuje se kako je opisanim postupkom dobiveno da je peti bit potrebno smanjiti. Budući da navedeni bit ima vrijednost 0, nije ga moguće dodatno smanjiti, iz čega se zaključuje da se taj bit ne smije mijenjati. Sve ostale bitove moguće je promijeniti, što će rezultirati novom Booleovom funkcijom, veće nelinearnosti.

Isto je moguće primijeniti i na funkcije većeg broja varijabli. U nastavku su opisani



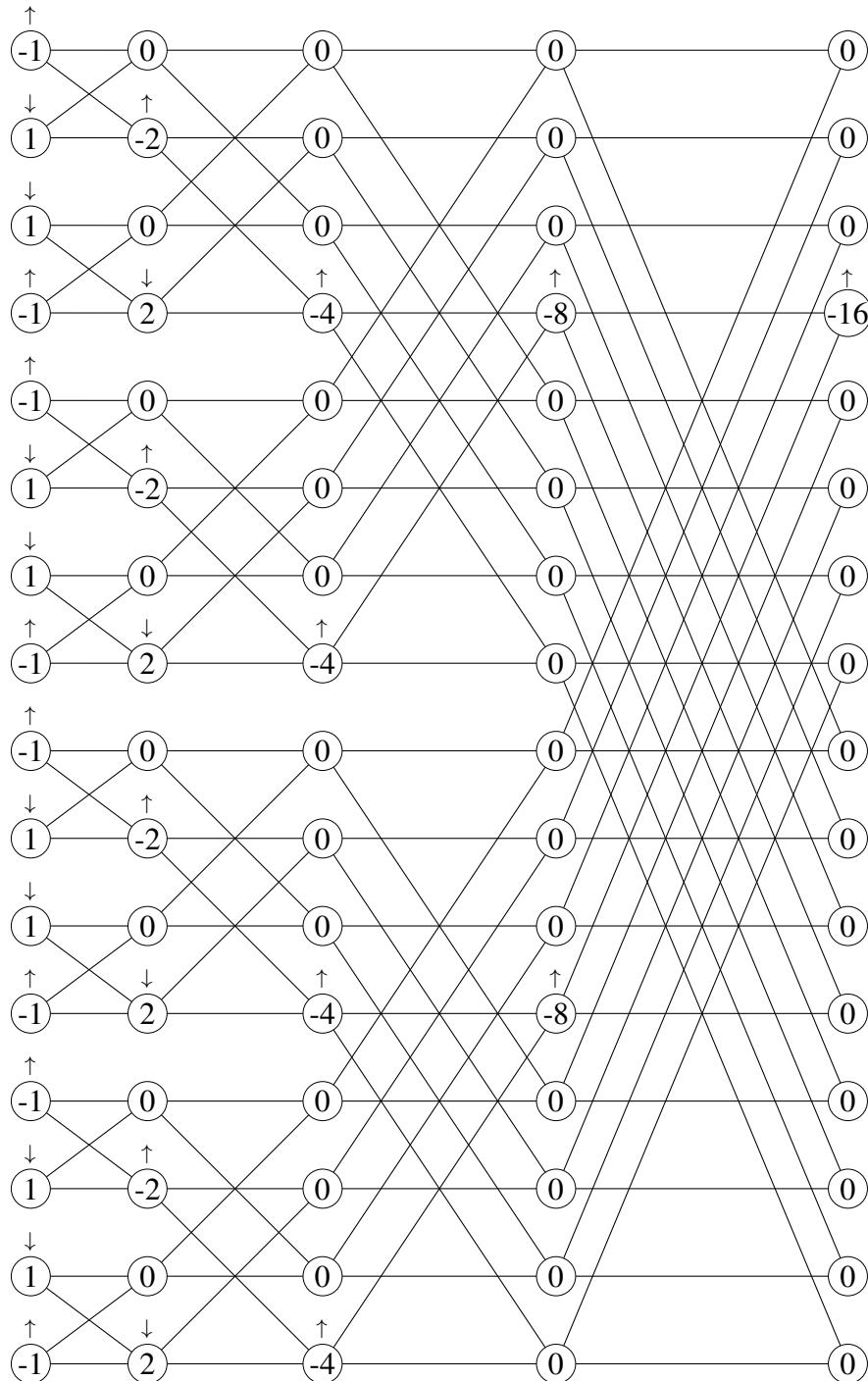
**Slika 5.11:** Prikaz izračuna načina promjene bitova tablice istinitosti za funkciju tri varijable

primjeri za Booleove funkcije četiri varijable. Slika 5.12 prikazuje funkciju 0110–011001100110 i njezine Walshove koeficijente. Navedena funkcija je afina te je njena nelinearnost jednaka 0, što se vidi iz toga što je najveći Walshov koeficijent jednak –16. Kako bi se smanjio absolutni iznos ovog koeficijenta, a time povećala nelinearnost funkcije, potrebno je napraviti promjenu koja će povećati koeficijent s –16 na –14, što je na slici označeno znakom  $\uparrow$  iznad koeficijenta. Prema prethodno opisanim pravilima, ta se promjena propagira unatrag, čime se određuje način na koji je potrebno promijeniti svaki od bitova. Primjećuje se kako je rezultat algoritma to da je moguće promijeniti bilo koji bit, što će uistinu povećati nelinearnost funkcije za 1.

Na primjeru funkcije nelinearnosti 1, koja je prikazana na slici 5.13, Walshov koeficijent najvećeg absolutnog iznosa je –14, te ga treba povećati na –12 kako bi se ostvarila funkcija nelinearnosti 2. Primjenom algoritma propagacije Walshovih koeficijenata unatrag nastaje situacija slična onoj u primjeru na slici 5.11, gdje jedna promjena ukazuje na to da je potrebno smanjiti vrijednost bita koji je trenutno postavljen na 0, iz čega se zaključuje da se taj bit ne smije mijenjati. Iz navedenoga, algoritam koristi preostalih 15 bitova kao moguće promjene.

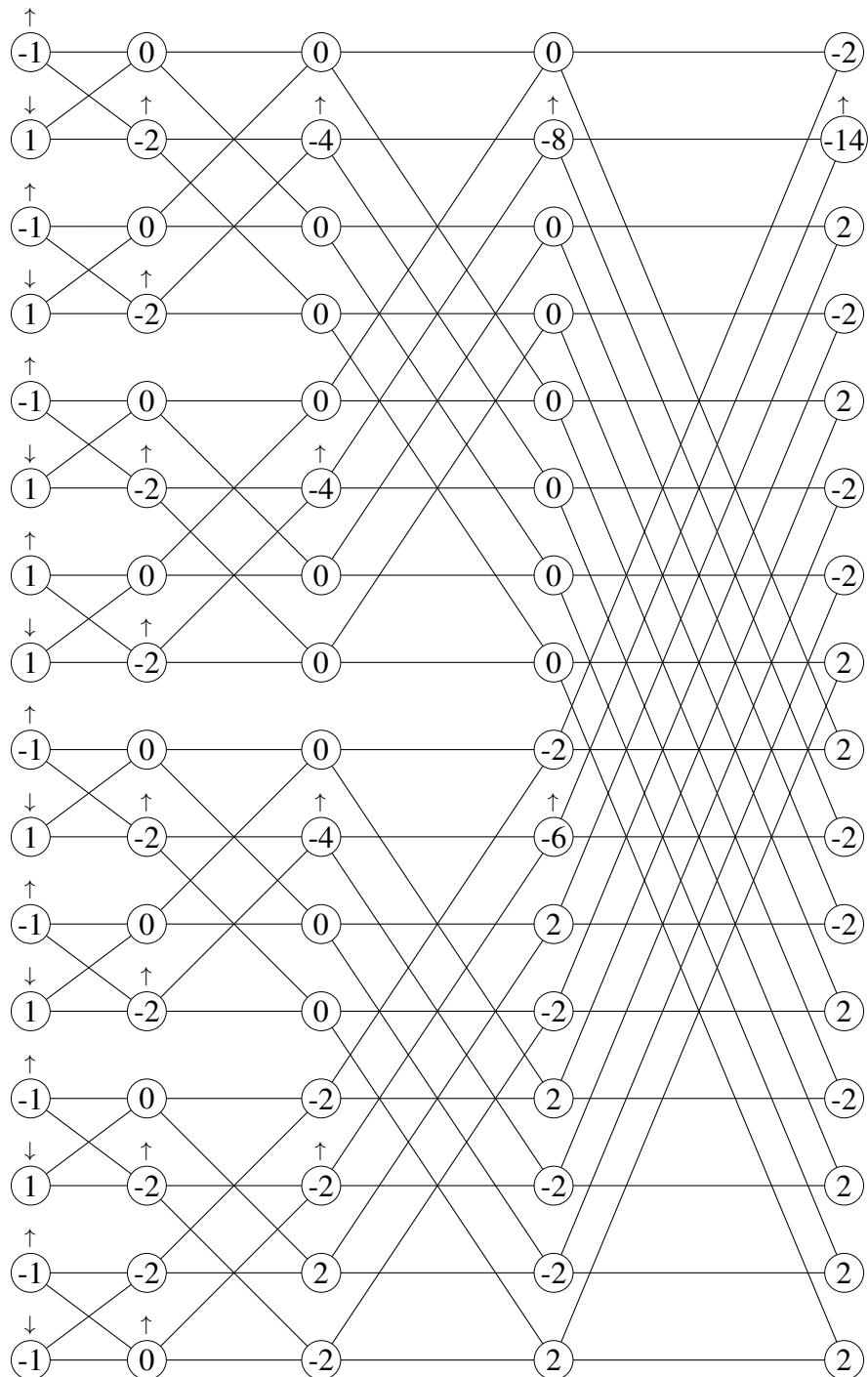
Slična situacija prikazana je i u slučaju funkcije nelinearnosti 2, prikazane na slici 5.14. Primjenom algoritma vidi se kako je potrebno povećati vrijednost bitova na pozicijama 13 i 15, što nije moguće, iz čega slijedi da se vrijednost tih bitova ne smije mijenjati.

Ponešto drugačija situacija prikazana je u slici 5.15. Prikazana je funkcije neli-



**Slika 5.12:** Prikaz izračuna načina promjene bitova tablice istinitosti za funkciju četiri varijable koja je šest promjena udaljena od najbliže Bent-funkcije

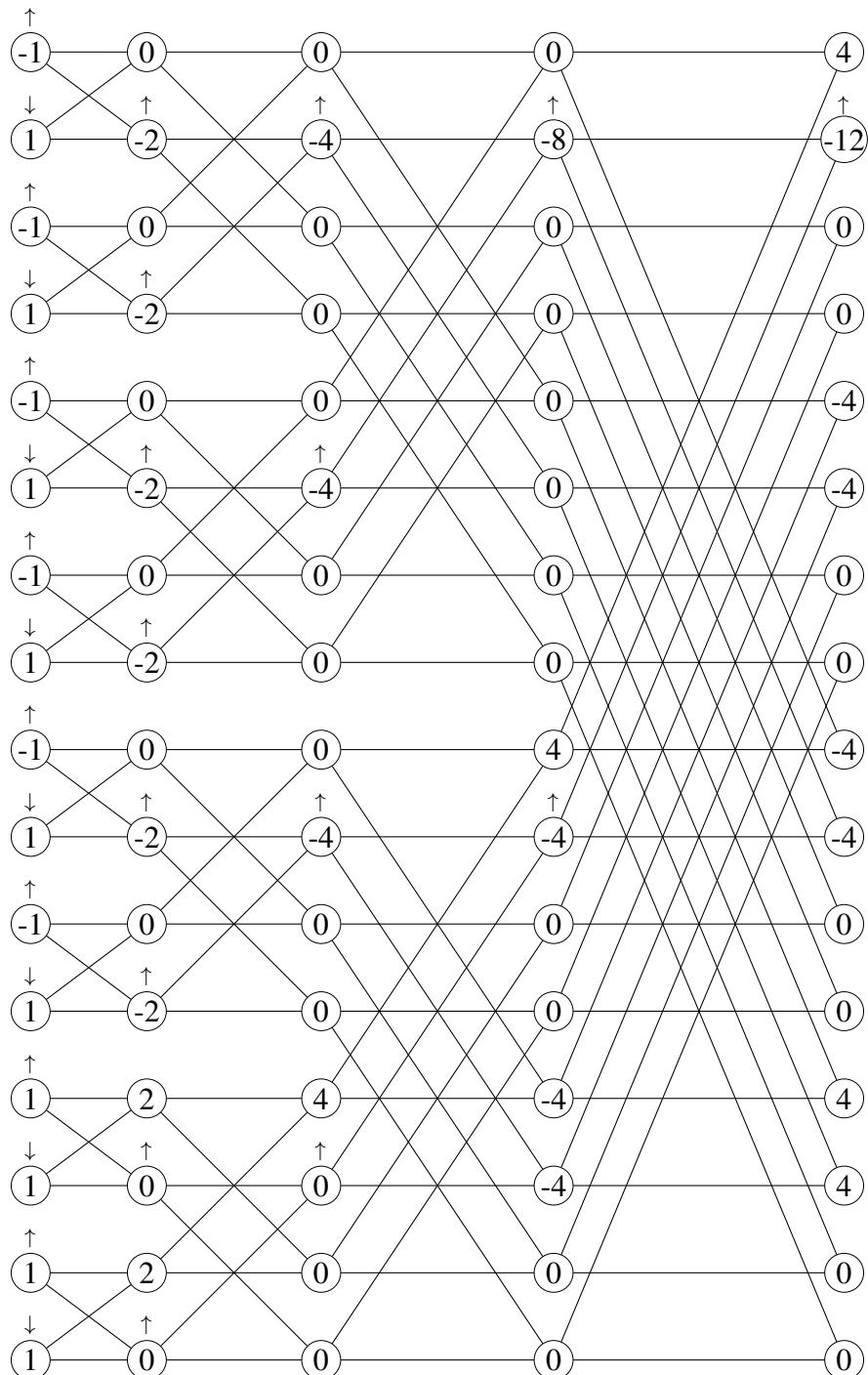
nearnosti 3, za koju je potrebno napraviti 3 promjene kako bi se ostvarila funkcija maksimalne nelinearnosti. Primjenom algoritma propagacije Walshovih koeficijenata unatrag, moguće je odrediti da promjene bitova na pozicijama 6, 9 i 11 ne vode prema traženom rješenju. Analizom svih Bent-funkcija koje su najbliže ovoj funkciji, pri-



**Slika 5.13:** Prikaz izračuna načina promjene bitova tablice istinitosti za funkciju četiri varijable koja je pet promjena udaljena od najbliže Bent-funkcije

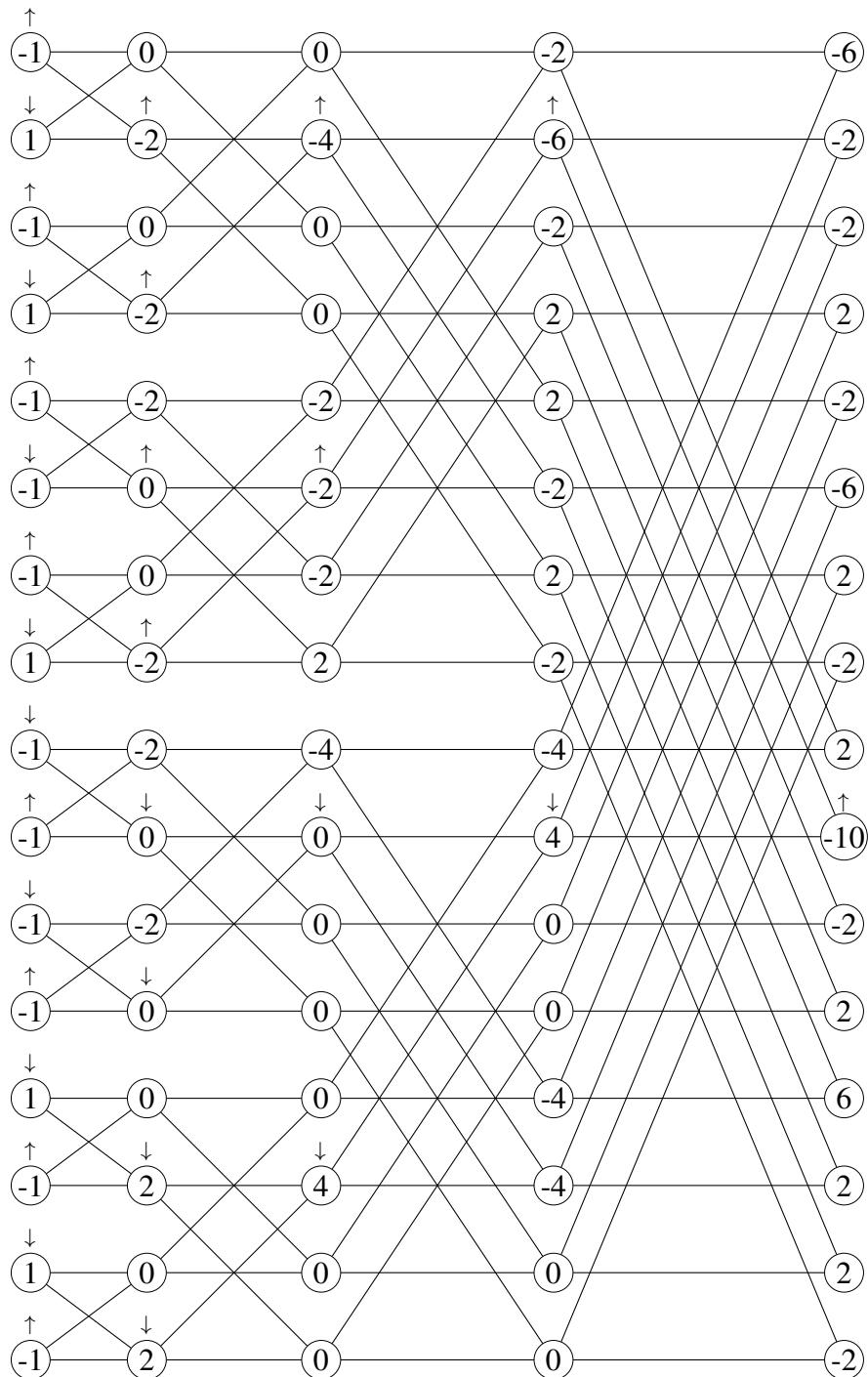
mjećuje se kako se bit na poziciji 8 također ne smije mijenjati, no algoritam to nije pronašao. Razlog tome je što algoritam pronalazi promjene koje vode u funkciju veće nelinearnosti, što može rezultirati i lokalnim optimumima.

Funkcija dobivena promjenom osmog bita, prikazana je u slici 5.16. Iz iznosa Wal-



**Slika 5.14:** Prikaz izračuna načina promjene bitova tablice istinitosti za funkciju četiri varijable koja je četiri promjene udaljena od najbliže Bent-funkcije

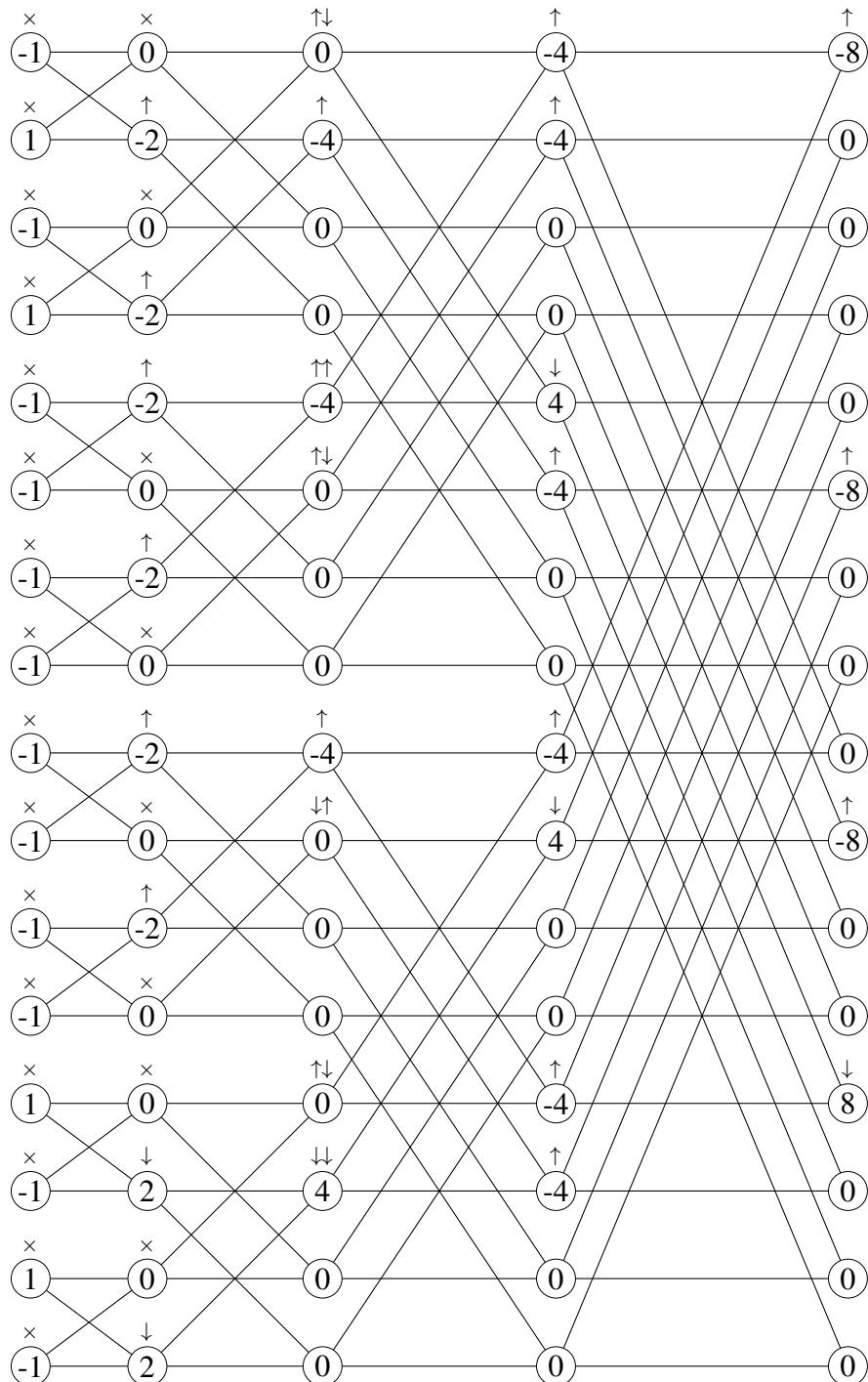
shovih koeficijenata primjećuje se da navedena funkcija posjeduje veću nelinearnost od funkcije sa slike 5.15, što je i bio cilj algoritma. Propagacijom promjena nad Walshovim koeficijentima unatrag, nastaju kontradikcije, čijom dalnjom propagacijom algoritam dolazi do zaključka da se niti jedan bit funkcije ne smije promjeniti. Una-



**Slika 5.15:** Prikaz izračuna načina promjene bitova tablice istinitosti za funkciju četiri varijable koja je tri promjene udaljena od najbliže Bent-funkcije

toč tome što funkcija nije Bent-funkcija, algoritam je došao do ispravnog zaključka, budući da promjena bilo kojeg bita smanjuje nelinearnost, ali i vodi prema globalnom optimumu.

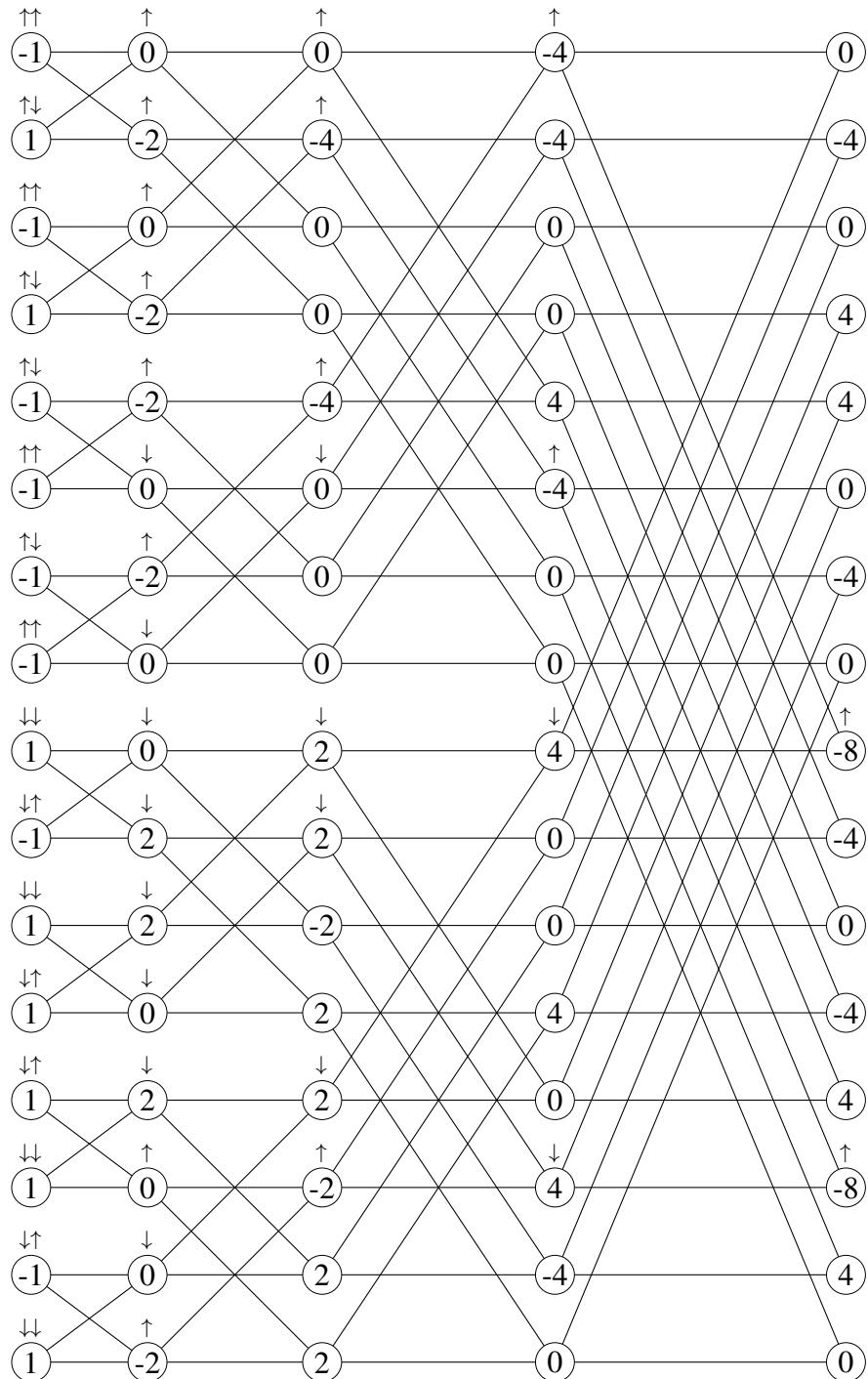
Slučaj s funkcijom nelinearnosti 4 prikazan je na slici 5.17. Algoritam je uspješno



**Slika 5.16:** Prikaz izračuna načina promjene bitova tablice istinitosti za funkciju četiri varijable koja je četiri promjene udaljena od najbliže Bent-funkcije, ali se nalazi u lokalnom optimumu

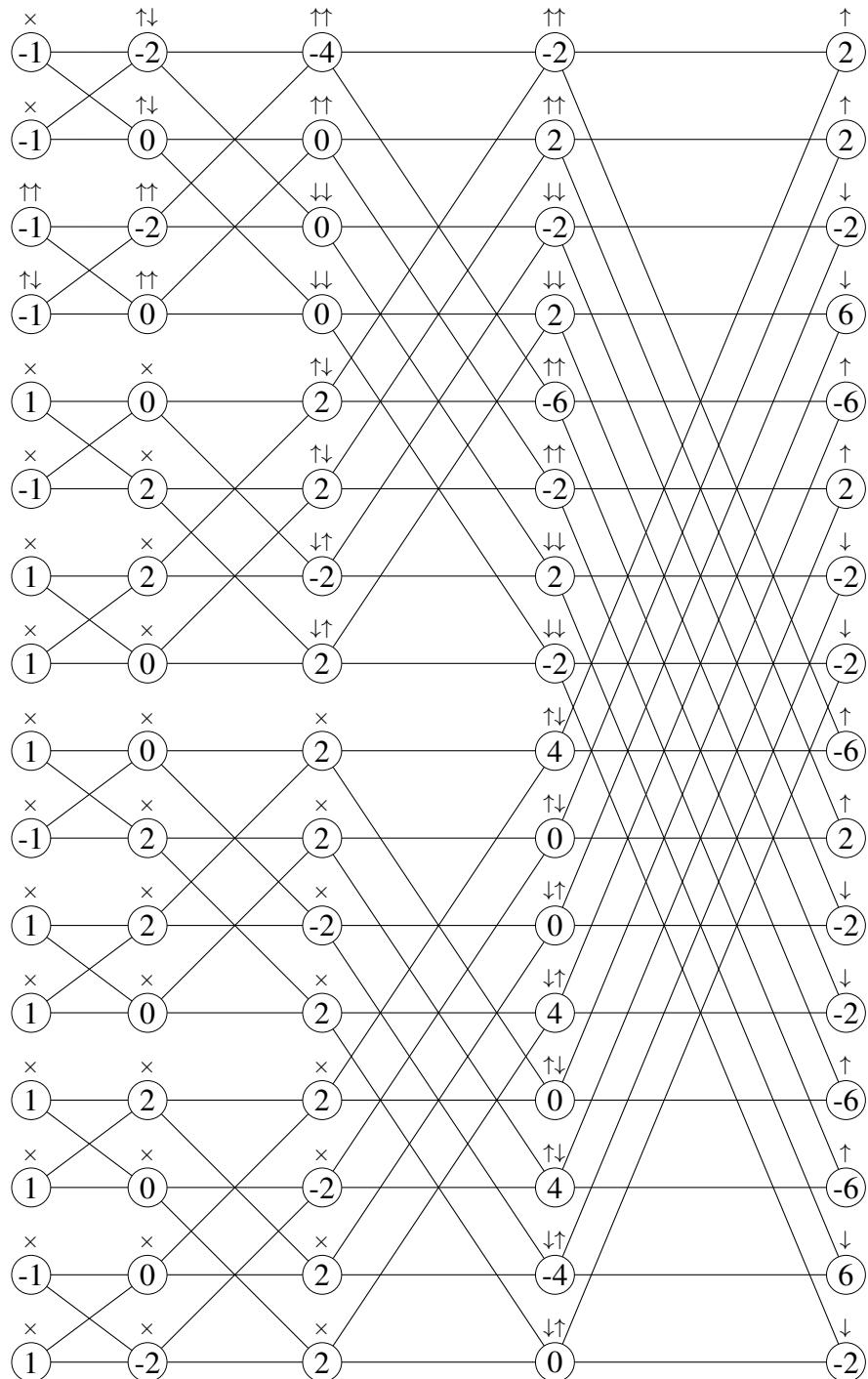
odredio 8 bitova koje se ne smije mijenjati, dok promjena u preostalih 8 bitova uistinu vodi prema globalnom optimumu.

Slika 5.18 prikazuje funkciju nelinearnosti 5, što znači da potencijalno postoji pro-



**Slika 5.17:** Prikaz izračuna načina promjene bitova tablice istinitosti za funkciju četiri varijable koja je dvije promjene udaljena od najbliže Bent-funkcije

mjena koja će ovu funkciju pretvoriti u Bent-funkciju. Kako je poznato da su svi Walshovi koeficijenti u Bent funkcijama jednaki, te za slučaj funkcija četiri varijable iznose 4 ili  $-4$ , moguće je odrediti na koji način je potrebno promijeniti svaki od koeficijenata. Uz tu informaciju moguće je pronaći mnogo kontradikcija u postupku



**Slika 5.18:** Prikaz izračuna načina promjene bitova tablice istinitosti za funkciju četiri varijable koja je jednu promjenu udaljena od najbliže Bent-funkcije

propagacije unatrag, što u konačnici rezultira pronašlaskom samo jednog bita kojega je moguće mijenjati, a promjena tog bita funkciju pretvara u Bent-funkciju.

U radu [13] autori također predlažu način za pronašlak balansiranih Booleovih funkcija veće nelinearnosti, što ostvaruju tako da za svaki par bitova iz tablice istinitosti

provjeravaju kako njihova promjena utječe na Walshove koeficijente, a samim time i na nelinearnost funkcije. Vremenska složenost navedenog postupka iznosi  $\mathcal{O}(n^3)$ .

Umjesto toga, u radu je korištena modifikacija algoritma propagacije Walshovih koeficijenata unatrag, tako da se propagacija provodi u dva koraka. Prvi korak jednak je kao i kod pronalaska Bent-funkcija, te je njegov cilj samo pronaći funkciju veće nelinearnosti. Drugi korak uvodi novo ograničenje, odnosno osim što se postavljuju ograničenja na smanjenje najvećih Walshovih koeficijenata, također se postavlja i ograničenje na prvi koeficijent, tako da ga se mijenja u nulu. Time je osigurano da nakon dva koraka algoritma, novo dobivena funkcija također posjeduje svojstvo balansiranosti, uz preduvjet da je početna funkcija također bila balansirana. Opisani algoritam ima vremensku složenost  $\mathcal{O}(n \log n)$ , isto kao i za pronalazak Bent-funkcija, no za razliku od algoritma predloženog u radu [13], moguće je da ovakva implementacija ne pronađe Balansiranu funkciju veće nelinearnosti, zbog toga što nisu ispitane sve moguće promjene nakon prvog koraka, već je nasumično odabrana jedna. Ako bi se ispitivale sve moguće promjene nakon prvog koraka, vremenska složenost iznosi  $\mathcal{O}(n^2 \log n)$ . Također valja primijetiti kako opisani algoritam premda ima bolju vremensku složenost, ima lošiju prostornu složenost s obzirom na to da osim tablice istinitosti mora pohranjivati i sve međurezultate prilikom izračuna Walshovih koeficijenata, zbog čega prostorna složenost iznosi  $\mathcal{O}(n \log n)$ , umjesto  $\mathcal{O}(n)$ , što je povećanje koje može biti značajno s obzirom na to da veličina tablice istinitosti raste eksponencijalno s porastom broja varijabli funkcije.

# 6. Implementacija i rezultati

Svi postupci opisani u ovom radu implementirani su u programskom jeziku Java. Izvorni kodovi su javno dostupni su na adresi <https://github.com/vkristijan/Evolving-Nonlinear-Functions>.

## 6.1. Iterativni algoritam pretraživanja

Kao što je navedeno u poglavlju o optimizacijskim algoritmima, iterativni algoritam pretraživanja jedan je od najjednostavnijih mogućih algoritama pretrage. Isti je isprobano uz korištenje tri različita načina generiranja susjednih funkcija: nasumičnom promjenom bita, korištenjem tabeliranih vrijednosti funkcija manjeg broja varijabli te algoritmom propagacije Walshovih koeficijenata unatrag. U tablici 6.1 prikazani su brojevi iteracija algoritma pretrage za svaku pojedinu definiciju susjedstva koji su bili potrebni za pronađak rješenja u 10 testiranja algoritma korištenjem nasumično generirane početne funkcije.

Kao što se vidi iz tablice, pretraga uz promjenu nasumičnog bita nije u stanju pronaći Bent-funkcije 6 varijabli unutar 1 000 000 000 iteracija pretrage, nakon čega je pretraga zaustavljena.

Korištenjem tabeliranih vrijednosti uspješno se dolazi do rješenja nakon prosječno 7 471 657 iteracija. Susjedstvo je ostvareno tako da je napravljen popis svih Booleovih funkcija četiri varijable te je za svaku od njih pohranjen popis bitova koje je potrebno promijeniti kako bi se dobila Bent-funkcija. Za funkciju većeg broja varijabli se potom odabire podskup duljine 16, što je duljina tablica istinitosti spremlijenih funkcija. Za tako odabran podskup se mijenja jedan bit, ovisno o promjenama koje je bilo potrebno napraviti u slučaju funkcije četiri varijable. Ideja ovog postupka pronalaska susjeda je iskoristiti znanje o funkcijama manjeg broja varijabli prilikom pronalaska funkcija većeg broja varijabli, tako da se prepostavlja ponavljanje određenih uzoraka.

Treći način određivanja susjedstva je korištenjem algoritma propagacije Walshovih koeficijenata unatrag, čime je rješenje pronađeno u prosječno 1 283 iteracija, što je

**Tablica 6.1:** Broj iteracija iterativnog algoritma pretrage za pronalazak Bent-funkcije 6 varijabli uz različite funkcije susjedstva, za deset nezavisnih pokretanja. Unutar svakog stupca vrijednosti su sortirane te je u zadnjem retku prikazana aritmetička sredina.

Nasumični bit	Tabelirane vrijednosti	Propagacija Walshovih koeficijenata unatrag
N/A	1 355 194	52
N/A	1 461 653	82
N/A	1 843 145	95
N/A	2 128 996	532
N/A	2 352 278	714
N/A	2 654 246	718
N/A	6 040 600	798
N/A	7 110 098	1 342
N/A	17 133 556	2 466
N/A	32 636 805	6 030
N/A	7 471 657	1 283

prema permutacijskom testu uz razinu signifikantnosti od  $\alpha = 0.05$  signifikantno bolje od tabeliranja vrijednosti za funkcije manjeg broja varijabli. Zanimljivo je istaknuti i podatak da je u otprilike 50% iteracija došlo do promjene nasumičnog bita jer je algoritam bio u lokalnom optimumu.

Niti jedan od opisana tri pristupa nije uspio pronaći Bent-funkciju za Booleove funkcije 8 varijabli.

## 6.2. Metoda uspona na vrh

Ova je metoda izrazito podložna lokalnim optimumima, s obzirom na to da jednom kada pronađe lokalni optimum nema mogućnosti za izlazak iz istoga. Ovisno o korištenoj funkciji vrednovanja rješenja, različit je postotak pokretanja algoritma u kojima pretraga završava u lokalnom optimumu. Konkretnije, za nasumično susjedstvo i tabelirano susjedstvo, niti jedan od 10 000 pokušaja pretrage nije pronašao globalni optimum korištenjem ukupne nelinearnosti, ili ukupne nelinearnosti i veličine sljedećeg po redu Walshovog koeficijenta kao mjeru uspješnosti. Algoritam propagacije Walshovih koeficijenata unatrag uspješno je pronašao globalni optimum u ukupno 2 od 10 000 pokušaja uz korištenje nelinearnosti kao mjere uspješnosti, te također 2

**Tablica 6.2:** Broj iteracija metode uspona na vrh za pronalazak Bent-funkcije 6 varijabli uz različite funkcije susjedstva i korištenje funkcije kazne (5.2). Prikazano je deset nezavisnih pokretanja te su unutar svakog stupca vrijednosti sortirane, a u zadnjem je retku prikazana aritmetička sredina.

Nasumični bit	Tabelirane vrijednosti	Propagacija Walshovih koeficijenata unatrag
73	44	14
99	61	16
174	83	18
189	98	22
200	126	22
200	295	25
213	414	32
222	414	39
279	940	41
390	1 684	44
204	416	27

od 10 000 uz korištenje nelinearnosti i sljedećeg po iznosu Walshovog koeficijenta. Ako se kao mjera uspješnosti koristi funkcija kazne iz izraza (5.2), pretraga rezultira globalnim optimumom u 56.5% slučajeva, kada se za susjedstvo koriste nasumične promjene. Uz korištenje iste mjere uspješnosti, ali tabeliranog susjedstva, pretraga postiže globalni optimum u 45.7% slučajeva, dok uz korištenje propagacije Walshovih koeficijenata pronalazi globalni optimum u 26.6% slučajeva. Primjećuje se kako velik broj pretraživanja završava u lokalnim optimumima. Također se primjećuje da informiranija susjedstva češće dolaze u lokalne optimume, što je posljedica toga što doprinose pohlepnom pretraživanju prema najboljem rješenju u blizini. Tablica 6.2 prikazuje brojeve iteracija koje su bile potrebne za pronalazak rješenja, u slučajevima kada je pronađeno globalno optimalno rješenje, ovisno o korištenoj definiciji susjedstva. U usporedbi s potrebnim brojevima iteracija iz tablice 6.1, primjećuje se značajno smanjenje potrebnog broja iteracija, ali zato velik broj pretraga nije postigao globalni optimum.

### 6.3. Simulirano kaljenje

Simulirano kaljenje donosi poboljšanje u odnosu na metodu uspona na vrh, utoliko što ne vrši pohlepnu pretragu, zahvaljujući čemu je manje skljono zapinjanju u lokalnim optimumima. Štoviše, matematički je dokazano da algoritam simuliranog kaljenja uvjek može pronaći globalni optimum, uz uvjet da se temperatura smanjuje u beskonačno malim pomacima te da se za svaku temperaturu provede beskonačno mnogo iteracija pretrage. Kako takvi uvjeti nisu mogući, potrebno je pažljivo odabratи početnu temperaturu te strategiju hlađenja. U okviru ovog rada implementirane su dvije strategije hlađenja; linearno hlađenje te geometrijsko hlađenje.

Linearno hlađenje za zadanu početnu i završnu temperaturu te broj smanjivanja temperature računa linearnu interpolaciju između početne i završne temperature tako da temperatura u koraku  $k$  odgovara izrazu (6.1).

$$t_k = t_{min} + k \cdot (t_{max} - t_{min}) / n \quad (6.1)$$

Geometrijsko hlađenje uvodi parametar  $\alpha$ , a temperatura se određuje prema izrazu (6.2).

$$t_k = \alpha^k \cdot t_{max} = \alpha \cdot t_{k-1} \quad (6.2)$$

Kako bi se temperatura smanjivala, mora vrijediti da je  $0 \leq \alpha \leq 1$ .

Testiranjem obje strategije, pokazalo se kako je za ovaj problem prikladnija strategija geometrijskog hlađenja u odnosu na linearno. Razlog tome je što se kod linearног hlađenja jednako mnogo vremena provodi na visokim, kao i na niskim temperaturama. To dovodi do toga da algoritam troši mnogo vremena na visokim temperaturama, kada je pretraga skoro pa nasumična, prije nego li dođe do niskih temperatura na kojima pretraga postaje sve više slična metodi uspona na vrh te algoritam konvergira prema nekom rješenju. S druge strane, kod geometrijskog hlađenja se temperatura puno brže mijenja kod visokih temperatura, dok promjene postaju sve manje kod nižih temperatura.

U tablici 6.3 prikazani su brojevi iteracija koje su bile potrebne kako bi algoritam simuliranog kaljenja pronašao Bent-funkcije 6 varijabli uz korištenje vrijednosti nelinearnosti funkcije kao metode vrednovanja rješenja. U usporedbi s brojem iteracija iterativnog algoritma prikazanim u tablici 6.1, primjećuje se kako je pretraga korištenjem promjene nasumičnog bita za generiranje susjedstva u stanju uspješno pronaći rješenje. Također se primjećuje i smanjenje potrebnog broja iteracija kada se kao susjedstvo koriste tabelirane vrijednosti. Kod susjedstva definiranog algoritmom propagacije Walshovih koeficijenata unatrag se primjećuje kako su brojevi iteracija slični

**Tablica 6.3:** Broj iteracija simuliranog kaljenja za pronalazak Bent-funkcije 6 varijabli uz različite funkcije susjedstva i korištenje vrijednosti nelinearnosti kao funkcije dobrote.

Prikazano je deset nezavisnih pokretanja te su unutar svakog stupca vrijednosti sortirane, a u zadnjem je retku prikazana aritmetička sredina.

Nasumični bit	Tabelirane vrijednosti	Propagacija Walshovih koeficijenata unatrag
2 751 055	107 891	235
2 910 629	180 152	285
3 070 496	499 936	741
3 176 652	975 913	832
3 227 534	1 142 061	1 023
3 287 251	1 159 724	1 133
3 460 184	1 348 438	1 452
3 528 421	1 389 500	1 759
3 734 571	1 499 543	2 245
3 755 913	1 526 981	4 258
3 290 721	983 014	1 396

onima u iterativnoj pretrazi. Uz početnu hipotezu o jednakom prosječnom broju iteracija za ta dva algoritma, nije moguće odbaciti hipotezu uz razinu signifikantnosti  $\alpha = 0.05$  korištenjem permutacijskog testa.

Kako algoritam simuliranog kaljenja vrši jednokriterijsku optimizaciju, nije moguće iskoristiti mjeru vrednovanja koja koristi ukupnu nelinearnost te iznos drugog po veličini Walshovog koeficijenta. Brojevi iteracija potrebnih za pronalazak rješenja korištenjem funkcije kazne iz izraza (5.2) prikazani su u tablici 6.4. Budući da je optimizacijski algoritam implementiran kao minimizacijski, korištena je recipročna vrijednost funkcije kazne kako bi se ista transformirala u funkciju dobrote. S obzirom na različite redove veličina funkcija dobrote u ovom i prethodnom slučaju, ovako dobivena funkcija dobrote pomnožena je s 10 000 kako bi postizala vrijednosti razmjerne nelinearnosti funkcije. Skaliranje vrijednosti funkcije dobrote ne utječe na učinkovitost iste, budući da je međusobni odnos rješenja ostao nepromijenjen, ali vrijednosti funkcija dobrote utječu na rad optimizacijskog algoritma prilikom izračuna vjerojatnosti prihvaćanja lošijeg rješenja. Osiguravanjem toga da obje funkcije dobrote imaju sličan raspon vrijednosti postignuti su jednaki uvjeti rada algoritma na jednakim temperaturama, čime je omogućena usporedba rezultata međusobno. Uz razinu značajnosti

**Tablica 6.4:** Broj iteracija simuliranog kaljenja za pronalazak Bent-funkcije 6 varijabli uz različite funkcije susjedstva i korištenje funkcije kazne (5.2). Prikazano je deset nezavisnih pokretanja te su unutar svakog stupca vrijednosti sortirane, a u zadnjem je retku prikazana aritmetička sredina.

Nasumični bit	Tabelirane vrijednosti	Propagacija Walshovih koeficijenata unatrag
1 224 570	133 333	107
1 380 446	177 920	149
1 401 639	364 795	341
1 415 769	500 271	571
1 599 351	571 739	872
1 637 653	579 728	1 047
1 690 626	689 224	1 113
1 744 255	774 338	1 147
1 775 540	820 657	1 878
1 815 088	1 109 313	5 133
1 568 494	571 132	1 236

od  $\alpha = 0.05$ , moguće je odbaciti hipotezu o jednakosti rada algoritma za ove dvije funkcije dobrote za slučajeve korištenja nasumičnog i tabeliranog susjedstva. Prilikom korištenja algoritma propagacije Walshovih koeficijenata unatrag nema statistički signifikantne razlike između korištenja jedne ili druge evaluacijske funkcije.

## 6.4. Genetski algoritam

Prednost genetskog algoritma u odnosu na simulirano kaljenje je postojanje populacije rješenja, čime se nastoji smanjiti mogućnost zaustavljanja u lokalnom optimumu. Prilikom implementacije algoritma, potrebno je definirati način zapisa rješenja, odnosno kromosom te metode selekcije, križanja i mutacije.

Kao zapis rješenja korišten je niz Booleovih vrijednosti koji predstavlja tablicu istinitosti Booleove funkcije, sukladno tome što je korišteno u prethodnim algoritmima.

Metoda selekcije koristi se prilikom odabira rješenja koja će biti korištena u križanju kako bi se stvorila nova rješenja za sljedeću generaciju. Korišten je algoritam turnirske selekcije [3], koji radi tako da nasumično odabere  $k$  jedinki iz populacije te ovisno o vrijednostima funkcije uspješnosti odabire dvije najbolje jedinke, od na-

sumično odabranog skupa. Ovisno o parametru  $k$ , moguće je postići veći ili manji selekcijski pritisak. Većim parametrom  $k$  više jedinki sudjeluje u turniru, zbog čega će bolje jedinke češće biti prisutne u nasumično odabranom skupu veličine  $k$ , zbog čega će iste također biti češće odabrane za križanje. Posljedica toga je veliki selekcijski pritisak, što može rezultirati gubitkom raznolikosti rješenja u populaciji i dolaskom u lokalne optimume. Korištenjem malih vrijednosti parametra  $k$ , dolazi do malog selekcijskog pritiska te loše jedinke imaju jednaku mogućnost biti odabrane kao i dobre jedinke, što kao posljedicu ima nasumičan odabir jedinki i divergenciju postupka.

Križanje je postupak u kojemu se od dva rješenja stvara novo rješenje koje je slično, no ne nužno jednako svakom od roditelja. U okviru ovog rada, korištena su dva postupka križanja, ovisno o tome je li cilj bio pronaći Bent-funkciju ili balansiranu Booleovu funkciju maksimalne nelinearnosti. U prvom slučaju, korišteno je križanje s jednom točkom prekida [3], koje radi tako da se odabere nasumična pozicija u kromosomu. Potom se kromosom djeteta stvara tako da se od početka kromosoma pa do odabrane pozicije kopiraju vrijednosti kromosoma prvog roditelja, dok se na ostale pozicije kopiraju vrijednosti drugog roditelja. Za pronalazak balansiranih funkcija korišteno je križanje koje ne narušava balansiranost funkcije. To je ostvareno tako da se najprije pronađu svi bitovi tablice istinitosti koji su zajednički u oba roditelja te se njihove vrijednosti prepišu u kromosom djeteta. Vrijednosti preostalih bitova određene su tako da se naizmjence postavljaju vrijednosti 0 i 1.

Mutacijom se u populaciju uvode dodatne raznolikosti kako bi se izbjegli lokalni optimumi. Kao i kod križanja, korišteni su različiti pristupi ovisno o tome traži li se Bent-funkcija ili balansirana funkcija maksimalne nelinearnosti. Kod pretrage Bent-funkcija korišteni su mutacije koje su prethodno korištene kao operatori za definiranje susjedstva, odnosno promjena nasumično odabranog bita, tabelirane vrijednosti te algoritam propagacije Walshovih koeficijenata unatrag. Za potrebe pronalaska balansiranih funkcija, mutacije su prilagođene tako da umjesto promjene jednog nasumičnog bita, mutacija pronalazi par bitova od kojih jedan ima vrijednost 0, a drugi vrijednost 1 te im potom zamjenjuje vrijednosti. Algoritam propagacije Walshovih koeficijenata unatrag prilagođen je na način koji je opisan u prethodnom poglavlju da umjesto jednog koraka provodi dva, gdje u prvom koraku nastoji samo povećati nelinearnost, dok u drugom koraku uz to postavlja dodatan uvjet na održavanje balansiranosti.

Genetski algoritam postizat će različite rezultate ovisno o korištenim hiperparametima algoritma. Kroz testiranje s različitim konfiguracijama istih, dobivene su sljedeće vrijednosti hiperparametara, koje su korištene prilikom generiranja rezultata prikazanih u nastavku. Za veličinu turnira prilikom selekcije odabrana je troturnirska selek-

**Tablica 6.5:** Broj generacija genetskog algoritma za pronalazak Bent-funkcije 6 varijabli uz različite funkcije mutacije i korištenje vrijednosti nelinearnosti kao funkcije dobrote.

Prikazano je deset nezavisnih pokretanja te su unutar svakog stupca vrijednosti sortirane, a u zadnjem je retku prikazana aritmetička sredina.

Nasumični bit	Tabelirane vrijednosti	Propagacija Walshovih koeficijenata unatrag
42	20	14
68	30	34
490	56	42
597	112	45
662	421	60
981	492	66
1 016	2 501	83
1 333	2 634	88
2 896	7 295	169
3 878	8 492	197
1 196	2 205	80

cija, dok je za veličinu populacije odabранo 75 jedinki. Također je korišten i elitizam veličine 1, što znači da se prilikom generiranja nove generacije najprije najbolje rješenje prethodne generacije kopira u novu populaciju, čime se osigurava da će najbolje rješenje nove generacije sigurno biti bolje ili jednakoj najboljem rješenju prethodne generacije. U tablici 6.5 prikazan je broj generacija potreban za pronalazak Bent-funkcije 6 varijabli, korištenjem nelinearnosti funkcije za funkciju dobrote. Korištenjem permutacijskog testa uz razinu signifikantnosti  $\alpha = 0.05$ , nije moguće odbaciti hipotezu o jednakosti genetskog algoritma koji koristi promjenu nasumičnog bita kao mutaciju i genetskog algoritma koji koristi tabelirane vrijednosti kao mutaciju. Za slučaj mutacije pomoću algoritma propagacije Walshovih koeficijenata unatrag, moguće je uz razinu signifikantnosti  $\alpha = 0.05$  odbaciti hipotezu o jednakosti broja generacija algoritma s brojem generacija prilikom korištenja drugih mutacijskih funkcija.

Brojevi generacija potrebnih za pronalazak rješenja korištenjem genetskog algoritma i evaluacijske funkcije koja koristi nelinearnost funkcije i iznos drugog po redu Walshovog koeficijenta prikazani su u tablici 6.6. Premda je prosječan broj generacija u ovom slučaju manji od onoga prikazanog u tablici 6.5, nema statistički signifikantne razlike prilikom usporedbe algoritma koji us istu funkciju mutacije koriti različite eva-

**Tablica 6.6:** Broj generacija genetskog algoritma za pronalazak Bent-funkcije 6 varijabli uz različite funkcije mutacije i korištenje vrijednosti nelinearnosti i sljedećeg po iznosu Walshovog koeficijenta kao funkcije dobrote. Prikazano je deset nezavisnih pokretanja te su unutar svakog stupca vrijednosti sortirane, a u zadnjem je retku prikazana aritmetička sredina.

Nasumični bit	Tabelirane vrijednosti	Propagacija Walshovih koeficijenata unatrag
54	30	27
102	41	28
107	48	28
116	54	45
127	60	50
128	61	51
139	69	72
170	426	90
1 433	1 414	174
2 037	1 680	229
441	388	79

luacijske funkcije.

U tablici 6.7 prikazani su potrebni brojevi generacija genetskog algoritma uz korištenje funkcije kazne definirane izrazom (5.2). Primjećuje se značajna razlika u broju generacija u odnosu na brojeve iteracija u tablicama 6.5 i 6.6. Također, uz razinu signifikantnosti  $\alpha = 0.05$  moguće je zaključiti kako je uz korištenje algoritma propagacije Walshovih koeficijenata unatrag potrebno u prosjeku manje generacija genetskog algoritma za pronalazak rješenja nego li korištenjem drugih ispitanih funkcija mutacije.

Kako se genetski algoritam pokazao kao najbolji optimizacijski algoritam od prethodno isprobanih, iskorišten je i na problemu pronalaska balansiranih Booleovih funkcija maksimalne nelinearnosti. Kako bi se postiglo da algoritam pronalazi balansirane Booleove funkcije, moguće je koristiti dva pristupa. Prvi je promijeniti mjeru vrednovanja rješenja, tako da kažnjava disbalans funkcije, kao što je to napravljeno u radovima [22] i [21]. Druga mogućnost je prilagoditi operatore tako da rade s balansiranim funkcijama, kao što je primjer u radovima [12] i [11]. Kao što je prethodno već opisano, u ovom radu odabran je pristup koji uz prilagodbu korištenih operatora osigurava svojstvo balansiranosti. Za potrebe toga najprije je generirana populacija balansiranih jedinki, tako da se za svaki bit tablice istinitosti nasumično odabire vrijednost te se

**Tablica 6.7:** Broj generacija genetskog algoritma za pronalazak Bent-funkcije 6 varijabli uz različite funkcije mutacije i korištenje funkcije kazne (5.2). Prikazano je deset nezavisnih pokretanja te su unutar svakog stupca vrijednosti sortirane, a u zadnjem je retku prikazana aritmetička sredina.

Nasumični bit	Tabelirane vrijednosti	Propagacija Walshovih koeficijenata unatrag
11	14	5
18	15	7
21	16	9
23	16	9
23	17	12
24	17	12
27	21	12
28	27	13
28	27	15
29	35	16
23	20	11

**Tablica 6.8:** Broj generacija genetskog algoritma za pronalazak balansirane Booleove funkcije 6 varijabli maksimalne nelinearnosti uz korištenje različitih funkcija mutacije. Prikazano je deset nezavisnih pokretanja te su unutar svakog stupca vrijednosti sortirane, a u zadnjem je retku prikazana aritmetička sredina.

<sup>1</sup> koristi ukupnu vrijednost nelinearnosti za evaluacijsku funkciju, <sup>2</sup> koristi funkciju kazne (5.2)

Nasumični par bitova <sup>1</sup>	Propagacija Walshovih koeficijenata unatrag <sup>1</sup>	Nasumični par bitova <sup>2</sup>	Propagacija Walshovih koeficijenata unatrag <sup>2</sup>
57	12	31	15
113	65	40	17
120	117	41	21
272	183	70	34
290	205	72	35
300	233	75	38
325	237	83	45
503	301	97	51
778	333	100	65
1 135	1 045	120	67
389	273	73	39

**Tablica 6.9:** Broj generacija genetskog algoritma za pronalazak balansirane Booleove funkcije 8 varijabli maksimalne nelinearnosti kroz deset nezavisnih pokretanja. Vrijednosti unutar stupca su sortirane, a u zadnjem retku je prikazana aritmetička sredina.

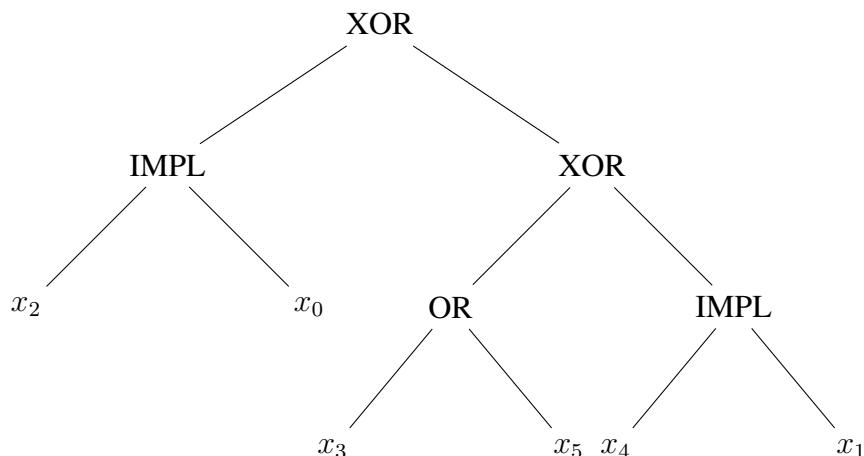
Propagacija Walshovih koeficijenata unatrag
1 549
1 561
2 164
3 133
6 656
7 770
10 498
20 753
20 988
20 988
9 606

broji koliko je puta odabrana koja vrijednost. U trenutku kad jedna vrijednost čini 50% tablice istinitosti, ostatak se popunjava drugom vrijednošću. Korišteni operatori križanja i mutacije opisani su prethodno, a rezultati uz korištenja istih na problem pronalaska balansiranih Booleovih funkcija 6 varijabli prikazani su u tablici 6.8. Primjećuje se kako sve prikazane metode uspješno dolaze do rješenja, odnosno Balansirane funkcije nelinearnosti 26. Također valja istaknuti kako je i u ovom slučaju korištenje funkcije kazne iz izraza (5.2) rezultiralo statistički signifikantno manjim prosječnim brojem generacija u odnosu na korištenje vrijednosti nelinearnosti. Isto vrijedi i za korištenje algoritma propagacije Walshovih koeficijenata unatrag, uz pomoć kojega je broj generacija algoritma manji nego li zamjenom nasumičnog para bitova.

Kako su najbolji rezultati postignuti korištenjem funkcije kazne (5.2) i mutacije pomoću propagacije Walshovih koeficijenata, taj je postupak primijenjen na problem pronalaska balansirane Booleove funkcije 8 varijabli. Valja istaknuti da je najniža poznata gornja ograda za ovaj problem nelinearnost 118, no najviša do sada postignuta nelinearnost iznosi 116. Kroz višestruka pokretanja algoritma uz različite konfiguracije hiperparametara, niti jednom nije pronađeno rješenje nelinearnosti 118. Iz tog razloga su u tablici 6.9 prikazani brojevi generacija koji su bili potrebni da bi algoritam pronašao rješenje nelinearnosti 116, što je ujedno i trenutno najbolje poznato rješenje.

## 6.5. Genetsko programiranje

Kao što je opisano u 4. poglavlju, genetsko programiranje zapravo je samo vrsta genetskog algoritma. Kao reprezentacija rješenja koristi se stablo operatora, koje tvori program čijim izvršavanjem nastaje krajnje rješenje. Taj je zapis prirodan za Booleove funkcije te odgovara onome iz primjera 2.1, gdje je izvršavanjem stabla moguće dobiti tablicu istinitosti nad kojom je potom moguće raditi evaluaciju kao i u prethodnim algoritmima. Općenito postoje dvije vrste čvorova u stablu operatora, funkcijski čvorovi i čvorovi koji imaju određenu vrijednost, poput konstante ili varijable. Glavna razlika ove dvije skupine čvorova je da funkcijski čvorovi imaju djecu te je njihova vrijednost funkcija vrijednosti djece, dok su čvorovi određene vrijednosti listovi u stablu. Za funkcijске čvorove korištene su sljedeće funkcije: AND, OR, XOR, negacija, implikacija i logička ekvivalencija. Kao čvorovi koji posjeduju određenu vrijednost, korištene su varijable koje odgovaraju svakoj od varijabli Booleove funkcije. Primjer jednog od dobivenih rješenja za pronačinak Bent-funkcije 6 varijabli prikazan je na slici 6.1.



**Slika 6.1:** Primjer Bent-funkcije 6 varijabli dobivene genetskim programiranjem

Početna populacija rješenja stvorena je slučajno generiranim stablima. Pritom su uvedeni hiperparametri za maksimalnu dozvoljenu dubinu stabla, kao i broj funkcijskih čvorova, kako bi se izbjegla duboka stabla koja su vremenski zahtjevna za izračun. Kao operator križanja, koristi se križanje opisano u [3], gdje se odabire slučajni čvor jednog roditelja te se on zamjenjuje slučajno odabranim čvorom drugog roditelja. Korišteni operator mutacije također je preuzet iz [3] te radi tako da odabire slučajan čvor stabla, briše ga i umjesto njega stvara novo, slučajno generirano podstablo, pridržavajući se pritom postavljenih uvjeta na maksimalnu dubinu i broj čvorova.

Opisana implementacija testirana je na pronačinak Bent-funkcija za Booleove funk-

**Tablica 6.10:** Broj generacija genetskog programiranja za različite metode vrednovanja rješenja kroz deset nezavisnih pokretanja. Unutar svakog stupca vrijednosti su sortirane te je u zadnjem retku prikazana aritmetička sredina.

<sup>1</sup> za Bent-funkciju 6 varijabli, <sup>2</sup> za Bent-funkciju 8 varijabli

Maksimalna nelinearnost <sup>1</sup>	Funkcija kazne (5.2) <sup>1</sup>	Maksimalna nelinearnost <sup>2</sup>	Funkcija kazne (5.2) <sup>2</sup>
2	4	5	15
5	6	7	21
7	6	8	27
8	10	9	45
8	11	11	51
9	11	12	61
10	12	13	63
12	16	13	65
12	21	14	69
18	29	15	77
9	13	11	49

cije 6 i 8 varijabli. Korištena je populacija veličine 100, gdje je svako stablo ograničeno na dubinu 7 i maksimalno 100 čvorova. Za operator selekcije korištena je trownirska selekcija. Brojevi generacija u kojima je algoritam pronašao rješenje prikazani su u tablici 6.10. Primjećuje se kako algoritam dolazi do optimalnog rezultata značajno brže nego li genetski algoritam, čiji rezultati su prikazani u tablicama 6.5 i 6.7. Također se primjećuje i to da u ovom slučaju korištenje funkcije kazne iz izraza (5.2) rezultira u prosjeku većim brojem generacija, dok je kod ostalih algoritama vrijedilo da se rješenje pronalazi brže korištenjem te mjere vrednovanja rješenja.

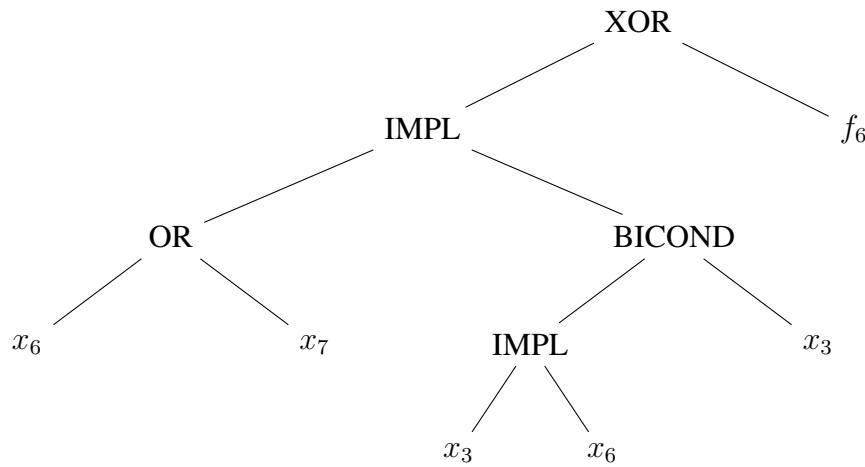
Kako se algoritam pokazao vrlo uspješnim na problemu pronalaska Bent-funkcija, testiran je za funkcije većeg broja varijabli, što je prikazano u tablici 6.11. Algoritam također uspijeva pronaći i Bent-funkcije 14 i 16 varijabli, no u tim slučajevima ne uspijeva pronaći optimalno rješenje prilikom svakog pokretanja već samo u određenom postotku. Unatoč tome što prosječan broj iteracija potrebnih za pronalazak rješenja nije znatno različit s porastom broja varijabli, vrijeme potrebno za izračun jedne generacije raste eksponencijalno s brojem varijabli, s obzirom na eksponencijalni porast tablice istinitosti funkcije, koja se koristi prilikom evaluacije rješenja.

Potaknuto idejom da postoji pravilnost kod nelinearnih funkcija koju je moguće iskoristiti za generiranje funkcija veće nelinearnosti, dodana je nova vrsta čvorova.

**Tablica 6.11:** Broj generacija genetskog programiranja za pronalazak Bent-funkcija različitog broja varijabli za deset nezavisnih pokretanja. Unutar svakog stupca vrijednosti su sortirane te je u zadnjem retku prikazana aritmetička sredina.

6 varijabli	8 varijabli	10 varijabli	12 varijabli
2	5	7	15
5	7	20	19
7	8	23	20
8	9	27	20
8	11	28	29
9	12	31	37
10	13	32	45
12	13	42	65
12	14	44	204
18	15	117	411
9	11	37	86

To su čvorovi koji predstavljaju Bent-funkcije nižeg broja varijabli, a koji se mogu koristiti prilikom pretrage funkcija većeg broja varijabli. Korištenjem do 10 različitih čvorova koji predstavljaju različite Bent-funkcije 6 varijabli, ustanovljeno je kako je pretraga funkcija 8 varijabli mnogo sporija u odnosu na slučaj u kojemu se ne koriste navedeni dodatni čvorovi. Unatoč tome zanimljivo je razmotriti jedno tako dobiveno



**Slika 6.2:** Primjer Bent-funkcije 8 varijabli dobivene genetskim programiranjem uz korištenje prethodno pronađenih rješenja

rješenje, što je prikazano na slici 6.2. Čvor označen sa  $f_6$  predstavlja Bent-funkciju 6

varijabli, koja je prikazana na slici 6.1. Razlog zbog kojega je ovaj primjer zanimljiv, bez obzira na to što zahtijeva mnogo veći broj generacija za pronađak rješenja je to što se pokazalo kako zamjena funkcije  $f'_6$  bilo kojom drugom Bent-funkcijom također daje Bent-funkciju 8 varijabli određenu operatorskim stablom sa slike 6.2.

## 7. Zaključak

U okviru ovog rada implementirano je i testirano nekoliko različitih pristupa za pronalazak Booleovih funkcija visoke nelinearnosti. Korišteni su sljedeći optimizacijski algoritmi: iterativni algoritam pretraživanja, metoda uspona na vrh, simulirano kaljenje, genetski algoritam i genetsko programiranje te se pokazalo kako je uz korištenje genetskog programiranja moguće pronaći Bent-funkcije u najmanjem broju koraka pretrage. Također je korišteno nekoliko različitih mjera vrednovanja rješenja, čime se pokazalo kako za većinu algoritama funkcija kazne iz izraza (5.2) pronalazi rješenje u najmanjem broju koraka, zahvaljujući najvećoj informiranosti funkcije. Isto nije bio slučaj kod genetskog programiranja, gdje se kao najuspješnija mjera vrednovanja pokazala funkcija dobrote koja poprima vrijednost nelinearnosti funkcije. Pored navedenog, predložena je nova funkcija za određivanje susjedstva, odnosno nova funkcija mutacije kod genetskog algoritma, slična onoj predloženoj u radu [13], uz poboljšanja u vidu vremenske složenosti. Predložena funkcija pokazala se uspješnom te je omogućila pronalazak rješenja u manjem broju koraka pretrage.

Potraga za balansiranom Booleovom funkcijom maksimalne nelinearnosti od 8 varijabli rezultirala je pronalaskom funkcija nelinearnosti 116, što je jednako do sada najbolje pronađenim rješenjima, ali ostavlja otvoreno pitanje o postojanju balansirane Booleove funkcije nelinearnosti 118.

Korištenjem genetskog programiranja ustanovljena je pravilnost prilikom generiranja rješenja uz korištenje postojećih Bent-funkcija. Točnije, zamijećeno je da jednom kad je pronađeno rješenje koje koristi Bent-funkciju manjeg broja varijabli, moguće je korištenu Bent-funkciju zamijeniti bilo kojom drugom Bent-funkcijom jednakog broja varijabli čime se ponovno ostvaruje Bent-funkcija. Navedeno svojstvo samo je eksperimentalno potvrđeno te je dokazivanje i daljnje korištenje istoga ostavljeno za daljnji rad.

# LITERATURA

- [1] Carlos Cid, Shinsaku Kiyomoto, i Jun Kurihara. The rakaposhi stream cipher. U *International Conference on Information and Communications Security*, stranice 32–46. Springer, 2009.
- [2] John A Clark, Jeremy L Jacob, Susan Stepney, Subhamoy Maitra, i William Millan. Evolving boolean functions satisfying multiple criteria. U *International Conference on Cryptology in India*, stranice 246–259. Springer, 2002.
- [3] Marko Čupić. Prirodnom inspirirani optimizacijski algoritmi. *Fakultet elektrotehnike i*, 2009.
- [4] Thomas W Cusick i Pantelimon Stanica. *Cryptographic Boolean Functions and Applications*. Academic Press, 2017.
- [5] Joan Daemen i Vincent Rijmen. Aes proposal: Rijndael. 1999.
- [6] B Gammel, Rainer Göttfert, i Oliver Kniffler. The achterbahn stream cipher. *Submission to eSTREAM*, 2005.
- [7] Jovan Dj Golić. Linear cryptanalysis of stream ciphers. U *International Workshop on Fast Software Encryption*, stranice 154–169. Springer, 1994.
- [8] John Henry Holland et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.
- [9] Scott Kirkpatrick, C Daniel Gelatt, i Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.
- [10] John R Koza i John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, svezak 1. MIT press, 1992.

- [11] Luca Manzoni, Luca Mariot, i Eva Tuba. Balanced crossover operators in genetic algorithms. *arXiv preprint arXiv:1904.10494*, 2019.
- [12] William Millan, Andrew Clark, i Ed Dawson. An effective genetic algorithm for finding highly nonlinear boolean functions. U *International Conference on Information and Communications Security*, stranice 149–158. Springer, 1997.
- [13] William Millan, Andrew Clark, i Ed Dawson. Smart hill climbing finds better boolean functions. U *Workshop on Selected Areas in Cryptology 1997, Workshop Record*, stranice 50–63. Citeseer, 1997.
- [14] William Millan, Andrew Clark, i Ed Dawson. Heuristic design of cryptographically strong balanced boolean functions. U *International Conference on the Theory and Applications of Cryptographic Techniques*, stranice 489–499. Springer, 1998.
- [15] William Millan, Andrew Clark, i Ed Dawson. Boolean function design using hill climbing methods. U *Australasian Conference on Information Security and Privacy*, stranice 1–11. Springer, 1999.
- [16] Timothy R O'Dowd. Discovery of bent functions using the fast walsh transform. Technical report, NAVAL POSTGRADUATE SCHOOL MONTEREY CA, 2010.
- [17] Stjepan Picek, Domagoj Jakobović, i Marin Golub. Evolving cryptographically sound boolean functions. U *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, stranice 191–192. ACM, 2013.
- [18] Stjepan Picek, Bariş Ege, Lejla Batina, Domagoj Jakobović, Łukasz Chmielewski, i Marin Golub. On using genetic algorithms for intrinsic side-channel resistance: the case of aes s-box. U *Proceedings of the First Workshop on Cryptography and Security in Computing Systems*, stranice 13–18. ACM, 2014.
- [19] Stjepan Picek, Julian F Miller, Domagoj Jakobović, i Lejla Batina. Cartesian genetic programming approach for generating substitution boxes of different sizes. U *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, stranice 1457–1458. ACM, 2015.
- [20] Stjepan Picek, Marko Čupić, i Leon Rotim. A new cost function for evolution of s-boxes. *Evolutionary computation*, 24(4):695–718, 2016.

- [21] Stjepan Picek, Domagoj Jakobović, Julian F Miller, Lejla Batina, i Marko Čupić. Cryptographic boolean functions: One output, many design criteria. *Applied Soft Computing*, 40:635–653, 2016.
- [22] Stjepan Picek, Roberto Santana, i Domagoj Jakobović. Maximal nonlinearity in balanced boolean functions with even number of inputs, revisited. U *2016 IEEE Congress on Evolutionary Computation (CEC)*, stranice 3222–3229. IEEE, 2016.
- [23] Terry Ritter. Measuring boolean function nonlinearity by walsh transform, jan 1998. URL <http://www.ciphersbyritter.com/ARTS/MEASNOLN.HTM>.
- [24] Oscar S Rothaus. On “bent” functions. *Journal of Combinatorial Theory, Series A*, 20(3):300–305, 1976.
- [25] Jennifer L Shafer, SW Schneider, Jon T Butler, i Pantelimon Stanica. Enumeration of bent boolean functions by reconfigurable computer. U *2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, stranice 265–272. IEEE, 2010.
- [26] Pedro Miguel Sosa. Calculating nonlinearity of boolean functions with walsh-hadamard transform. apr 2016.
- [27] Petr Tesař. A new method for generating high non-linearity s-boxes. *Radioengineering*, 19(1):23–26, 2010.

# **Pronalaženje Booleovih funkcija maksimalne nelinearnosti evolucijskim računanjem**

## **Sažetak**

Booleove funkcije sastavni su element kriptografskih algoritama. Kako bi se povećala otpornost na napade linearom kriptoanalizom, od posebnog je značaja svojstvo nelinearnosti Booleove funkcije. Booleove funkcije zadanog broja varijabli i maksimalne nelinearnosti nazivaju se Bent-funkcije, dok su sa stajališta primjene u kriptografskim algoritmima od posebnog interesa Booleove funkcije koje dodatno imaju i svojstvo balansiranosti.

U okviru ovog diplomskog rada proučeni su heuristički pristupi pronalaženja Booleovih funkcija maksimalne nelinearnosti te balansiranih Booleovih funkcija maksimalne nelinearnosti. Implementirani su i međusobno uspoređeni optimizacijski postupci temeljeni na simuliranom kaljenju, genetskom algoritmu i genetskom programiranju. Uspoređeni su i različiti načini prikazivanja Booleovih funkcija, poput zapisa u obliku tablice istinitosti te algebarskog zapisa, kao i razne funkcije izračuna dobrote rješenja u evolucijskim algoritmima. Dodatno je predložen i analiziran novi način pretrage traženih funkcija, temeljen na analizi Walsh koeficijenata funkcije.

**Ključne riječi:** Booleove funkcije, nelinearnost, heuristička optimizacija, evolucijski algoritmi

# **Evolutionary Computation Based Search for Maximal Nonlinearity Boolean Functions**

## **Abstract**

Boolean functions represent a crucial element for designing cryptographic algorithms. In order to resist against linear cryptanalysis attack, it is essential for Boolean functions to have high nonlinearity. Bent functions are Boolean functions with maximal possible nonlinearity for a given number of variables. For the usage in cryptographic algorithms it is additionally important for functions to be balanced.

This thesis is based upon researching heuristic search methods for maximal nonlinear Boolean functions and maximal nonlinear balanced Boolean functions. Solutions based on simulated annealing, genetic algorithms and genetic programming are implemented and compared. The impact of different function representations, such as truth tables and algebraic notation is also analyzed, together with various fitness functions. Finally, a new method based on the analysis of Walsh coefficients for finding nonlinear functions is proposed.

**Keywords:** Boolean Functions, Nonlinearity, Heuristic Optimization, Evolutionary Algorithms