SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1558

Postupci konstrukcije i optimizacije adicijskih lanaca

Josip Užarević

Zagreb, veljača 2018.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

Zahvaljujem se mentoru na otkrivanju vrlo zanimljive teme s područja računarske znanosti, koja me držala zaintrigiranim tokom cijelog diplomskog studija, na ogromnoj količini strpljenja i susretljivosti tokom izrade rada te otvaranju mogućnosti za daljnje znanstveno djelovanje i nakon studija.

Zahvaljujem se cijeloj svojoj obitelji te djevojci Josipi i njenim roditeljima na neizmjerno mnogo podrške koja mi je bila nužna kako bih uspješno završio studij te ga okrunio ovim radom.

SADRŽAJ

1. Uvod

Sigurnost suvremene kriptografije velikim se dijelom zasniva na nemogućnosti faktorizacije *velikog*[1] broja u razumnom vremenu. U cilju poboljšanja performansi postojećih kriptografskih algoritama, u vidu brzine izvođenja, procesorskih i memorijskih zahtjeva, te sklopovske implementacije istih, od velike važnosti su istraživanja na temu faktorizacije velikih brojeva. Uz pojam faktorizacije, usko je vezan i pojam potenciranja velikih brojeva, pri čemu se misli na efikasno potenciranje brojeva pomoću uzastopnih operacija zbrajanja i množenja, koje su efikasno sklopovski izvedene u svim modernim procesorskim arhitekturama.

Adicijski lanac predstavlja matematički aparat pomoću kojeg je moguće predstavljati brojeve uzastopnim zbrajanjem i množenjem manjih brojeva. Drugim riječima, počevši od broja 1, pomoću adicijskog lanca moguće je generirati svaki prirodni broj, bez obzira na njegovu veličinu. U poglavlju 2 razrađeni su adicijski lanci, njihova primjena te motivacija vezana za efikasno traženje kratkih adicijskih lanaca.

Budući da je traženje najkraćeg mogućeg adicijskog lanca za zadani broj NP-težak problem, za velike brojeve logično je pribjeći raznim heurističkim metodama kako bi se došlo do dovoljno dobrog rješenja u razumnom vremenu. Jedan od provjerenih heurističkih pristupa je i upotreba evolucijskih algoritama u svrhu traženja kratkih adicijskih lanaca. U poglavlju 3 opisani su evolucijski algoritmi, s naglaskom na upotrebu istih u traženju kratkih adicijskih lanaca.

Evolucijski algoritmi, ovisno o problemu i dostupnim računalnim resursima, odličan su izbor za rješavanje problema traženja kratkih adicijskih lanaca. Međutim, uz neke druge nedostatke, glavni nedostatak ovih algoritama je dulje vrijeme izvođenja. Zbog toga, razumno je pokušati upotrijebiti i determinističke metode koje će, uz znatno kraće vrijeme izvođenja i konzistentnost rezultata, davati dovoljno dobra rješenja. Deterministične metode koje su danas u upotrebi, kao i one novorazvijene u okviru rada, opisane su u poglavlju 4.

Kako i deterministički i stohastički algoritmi imaju svoje prednosti i mane, svakako

[1] Koriste se brojevi reda veličine 2^{100} pa i više.

se kao izvrsna opcija nudi kombinacija ova dva potpuno različita pristupa. U okviru poglavlja 5 obrađen je evolucijski algoritam potpomognut determinističkom metodom.

Nakon početnog ispitivanja smislenog skupa parametara kojima se kontrolira ponašanje obrađenih metoda, razrađen je plan izvođenja te je izvršeno ispitivanje nad reprezentativnim skupom brojeva za koje je potrebno naći kratke adicijske lance. Rezultati i analiza mjerenja izneseni su u poglavlju 6.

Posljednje, zaključno poglavlje 7 iznosi najvažnije zaključke vezane za istraživanje provedeno u okviru rada, kao i prednosti i nedostatke razmatranih metoda te moguća mjesta za poboljšanja razvijenih metoda i pristupa pri traženju kratkih adicijskih lanaca.

2. Konstrukcija adicijskih lanaca

U kriptografiji, naročito kad su u pitanju asimetrični kriptografski algoritmi poput RSA, od velikog značaja je potenciranje broja na neki veliki eksponent. Budući da je sama operacija potenciranja izuzetno računalno skupa, cilj je ubrzati ovu operaciju na način da se, što je više moguće, smanji broj osnovnih računalnih operacija.

2.1. Potenciranje

U svrhu razjašnjavanja problematike potenciranja na veliki eksponent, poslužit ćemo se konkretnim, jednostavnim primjerom: potrebno je izračunati vrijednost 2^{15}. Procesor računala mora potenciranje svesti na množenje, pa bi se naivnim pristupom ova vrijednost izračunavala na sljedeći način:

$$\begin{aligned} 2^{15} =& 2^1 \times 2^1 \times 2^1 \times 2^1 \times 2^1 \times \\ & 2^1 \times 2^1 \times 2^1 \times 2^1 \times 2^1 \times \\ & 2^1 \times 2^1 \times 2^1 \times 2^1 \times 2^1 \end{aligned} \tag{2.1}$$

pri čemu bi se operacija množenja obavila 14 puta. S druge strane, uz pretpostavku mogućnosti pamćenja međurezultata, do istog rezultata moguće je doći sljedećim postupkom:

$$\begin{aligned} 2^2 &= 2^1 \times 2^1 \\ 2^3 &= 2^1 \times 2^2 \\ 2^6 &= 2^3 \times 2^3 \\ 2^7 &= 2^1 \times 2^6 \\ 2^{14} &= 2^7 \times 2^7 \\ 2^{15} &= 2^1 \times 2^{14} \end{aligned} \tag{2.2}$$

te se u ovom slučaju obavlja svega 6 množenja. Kao što će kasnije biti objašnjeno, vrlo efikasno se do ovog rezultata dolazi tzv. *binarnom metodom eksponenciranja*.

Međutim, ova metoda također ne daje minimalan broj operacija množenja. Optimalan broj operacija množenja postiže se sljedećim izračunom:

$$\begin{aligned} 2^2 &= 2^1 \times 2^1 \\ 2^3 &= 2^1 \times 2^2 \\ 2^5 &= 2^2 \times 2^3 \\ 2^{10} &= 2^5 \times 2^5 \\ 2^{15} &= 2^{10} \times 2^{15} \end{aligned} \tag{2.3}$$

koji koristi 5 množenja.

Čak i pri ovakvim malenim eksponentima poput broja 15 primjećuje se velika razlika u broju množenja: od 14, preko 6 te naposljetku optimalnih 5 množenja, ovisno o metodi koja se koristi. Treći, optimalni izračun nije moguće deterministički odrediti, odnosno, ne postoji algoritam koji će efikasno za svaki zadani broj (u našem slučaju, eksponent) pronaći optimalan izračun u razumnom vremenu. Valja napomenuti kako se u praksi koriste brojevi daleko veći od ovdje upotrebljenog broja 15 - konkretno, koriste se brojevi reda veličine 2^{120} pa i veći. Dakle, vrše se izračuni npr. $a^{2^{127}}$.

Traženje optimalnog izračuna je NP-kompletan problem. Upravo zbog NP-kompletnosti problema, koriste se razne heurističke metode kako bi se došlo do optimalnog broja množenja u izračunu. Prilikom razmatranja ovog problema vrlo je zgodno potenciranje modelirati *adicijskim lancima* zbog jednostavnosti prikaza, računanja te naposljetku lake čitljivosti ovakvog modela.

2.2. Adicijski lanci

(Brauer, 1939) Adicijski lanac za zadani broj n je niz prirodnih brojeva A takav da vrijedi:

1. Prvi član niza A je 1.

$$a_0 = 1$$

2. Posljednji član niza A je n, a broj r naziva se *duljinom* lanca.

$$a_r = n$$

3. Niz A je rastući.

$$\forall i (a_{i+1} > a_i)$$

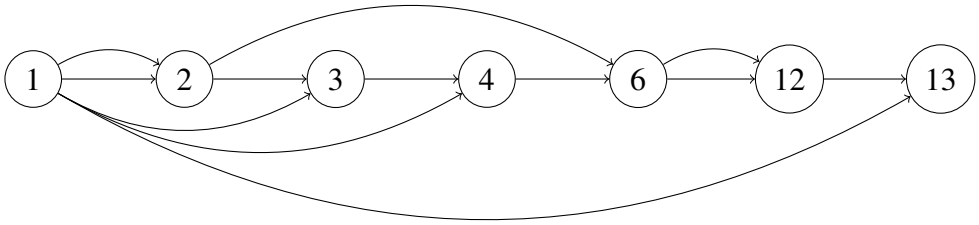
4. Svaki element niza A, osim prvog, može se napisati kao zbroj dva prethodnika.

$$\forall k \exists i \exists j (k > 0 \Rightarrow a_i + a_j = a_k)$$

Primjer jednog adicijskog lanca bio bi:

$$A(13) = \{1, 2, 3, 4, 6, 12, 13\} \tag{2.4}$$

Vidimo da u danom primjeru sve 4 tvrdnje vrijede: prvi broj je 1, zadnji broj je 13, niz je rastući te se svaki broj može prikazati kao zbroj neka dva prethodnika, pri čemu prethodnici smiju biti jednaki. Adicijski lanci se mogu zgodno prikazati pomoću usmjerenog grafa, a primjer 2.4 se prikazuje na sljedeći način:



Slika 2.1: Prikaz adicijskog lanca pomoću usmjerenog grafa

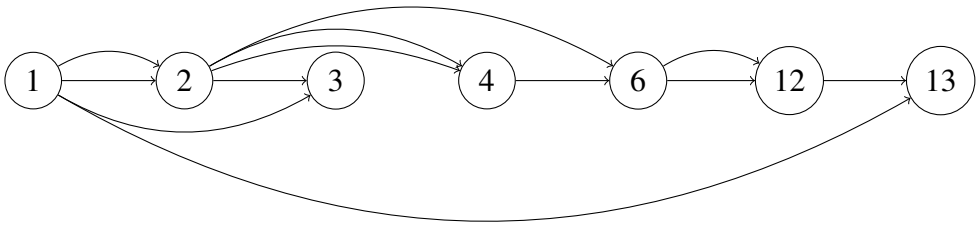
Pomoću prikaza jasno se vidi način na koji je dobiven svaki broj u lancu, odnosno, koji prethodnici prilikom zbrajanja daju željeni broj. Svaki broj u lancu ima točno dvije ulazne veze, odnosno točno dva pribrojnika, no broj može sudjelovati kao pribrojnik u dobivanju više brojeva. Tako npr. broj 1 sudjeluje u izračunu brojeva 2, 3, 4 i 13.

2.3. Svojstva adicijskih lanaca

Svi adicijski lanci imaju neka zajednička svojstva, od kojih su neka vrlo bitna prilikom izgradnje heuristike za traženje kratkih adicijskih lanaca. Bitnija svojstva su:

1. **Svaki adicijski lanac nužno počinje brojevima 1 i 2.** Budući da adicijski lanac uvijek počinje brojem 1, prvi idući broj koji je moguće dobiti korištenjem isključivo broja 1 je broj 2.

2. **Iz zadanih brojeva unutar adicijskog lanca nije jednoznačno određen način dobivanja elemenata adicijskog lanca.** Ukoliko su zadani samo elementi lanca, bez zadanih veza, moguće je da postoji više načina povezivanja elemenata. Tako npr. elementi iz lanca na slici 2.1 mogu biti povezani i na sljedeći način:



Slika 2.2: Drukčija povezanost elemenata lanca sa slike 2.1

Elementi lanca su jednaki, no u ovom slučaju, broj 4 se dobiva zbrajanjem 2 i 2, dok se u prvotnom primjeru dobiva zbrajanjem 3 i 1. Iz ovog primjera slijedi i važno, sljedeće svojstvo.

3. **Nisu nužno svi elementi lanca potrebni.** Na novodobivenom primjeru broj 3 je nepotreban, budući da ne sudjeluje u dobivanju nekog većeg broja preko kojeg se dolazi do konačnog broja. Dakle, izbacivanjem broja 3, lanac se skraćuje za 1 te i dalje ostaje ispravan. Valja primijetiti da je novodobiveni lanac, bez izbacivanja suvišnog broja 3 i dalje ispravan - svaki broj se može zapisati kao zbroj neka dva prethodnika pa tako i broj 3.

4. **Nije nužno da prvi prethodnik sudjeluje u dobivanju zbroja nekog broja**. U primjeru sa slike 2.2 broj 3 ne sudjeluje u dobivanju broja 4. Krivi zaključak bi bio da se broj 3 ionako treba izbaciti iz lanca - dovoljno je produljiti lanac do elementa 16 - tada bi broj 3 sudjelovao u dobivanju konačnog elementa 16, a broj 4 bi i dalje bio prikazan kao zbroj 2 i 2.

   U vezi s ovim zaključkom, adicijski lanac kod kojeg za svaki član vrijedi da u zbroju sudjeluje njegov prvi prethodnik, naziva se *Brauerov lanac* ili *star chain*.

2.4. Veza potenciranja i adicijskih lanaca

Kako prikazati potenciranje pomoću adicijskih lanaca? Ovo je moguće vrlo jednostavno izvesti ukoliko prilikom potenciranja uzimamo u obzir samo eksponent, bez baze koja se potencira. Sama baza nije niti bitna prilikom potenciranja - od interesa je samo postupak potenciranja na zadani broj, koji je onda primjenjiv na bilo koju bazu.

Kao primjer uzet ćemo izračun 2.2, koji se može zapisati i na sljedeći način:

$$2^{15} = 2^{((((1\times 2)+1)\times 2)+1)\times 2+1} \tag{2.5}$$

Primijetimo sljedeće:

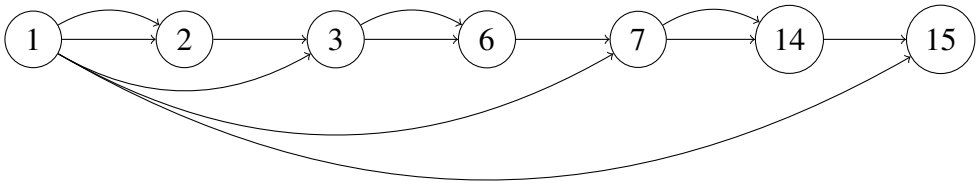
1. Kvadriranje potencije svodi se na množenje s 2 (udvostručenje) u domeni eksponenata:

$$\begin{aligned} 2^6 &= 2^3 \times 2^3 \\ &= \left(2^3\right)^2 \\ &= 2^{3+3} \\ &= 2^{3\times 2} \end{aligned}$$

2. Množenje potencija jednakih baza svodi se na zbrajanje u domeni eksponenata:

$$\begin{aligned} 2^7 &= 2^6 \times 2^1 \\ &= 2^{6+1} \end{aligned}$$

Upravo udvostručenja i zbrajanja eksponenata predstavljeni su u grafičkom prikazu adicijskog lanca vezama između elemenata, a sami elementi predstavljaju eksponente koje sudjeluju u izračunu. Shodno tome, izračun 2.5 prikazan je slikom:



Slika 2.3: Adicijski lanac izračuna 2.5

Svaki par usmjerenih veza koje ulaze u neki čvor predstavljaju zbrajanje, a udvostručenje je ustvari dvostruka veza od jednog elementa ka drugom (npr. od 3 prema 6.) Problem traženja optimalnog, odnosno što kraćeg izračuna, svodi se na traženje što kraćeg adicijskog lanca za zadani broj (eksponent). Ukupan broj udvostručenja i zbrajanja jednak je upravo duljini pripadajućeg adicijskog lanca.

Sada kada je ustanovljena veza potenciranja i adicijskih lanaca nadalje možemo razmatrati samo adicijske lance, kojima prikazujemo isključivo domenu eksponenata bez razmatranja baze.

2.5. Adicijsko-suptrakcijski lanci

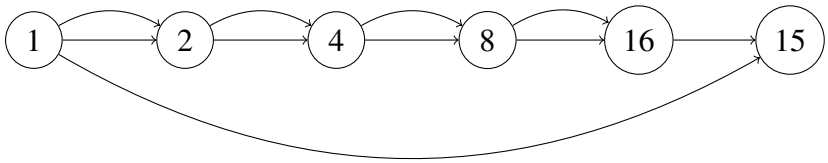
Postoje primjene u kojima se, osim operacije zbrajanja u domeni eksponenata, dozvoljava i operacija oduzimanja. Ovime se efektivno, uz množenje, dozvoljava i dijeljenje

međurezultata s potencijom. Ovaj dodatak definira generalizaciju adicijskih lanaca na adicijsko-supstrakcijske lance, koji se od adicijskih lanaca razlikuju u sljedećem:

1. **Svaki broj je zbroj ili razlika neka dva prethodnika.**

2. **Niz brojeva ne mora nužno biti rastući.** Ovo je direktna posljedica dozvoljavanja oduzimanja - prilikom svakog oduzimanja, sljedbenik će biti manji od prethodnika.

Primjer adicijsko-suzpstrakcijskog lanca za broj 15 prikazan je na slici 2.4. Veza od 1 prema 15 je oduzimajuća.



Slika 2.4: Adicijsko-supstrakcijski lanac broja 15

Ova vrsta lanaca se koristi ukoliko operacija dijeljenja nije računalno skupa, odnosno, usporediva je s množenjem.. Tada je moguće kod određenih brojeva doći do kraćih lanaca nego ako se dijeljenje zabrani. Ipak, uvođenjem dijeljenja znatno se povećava prostor stanja koji je potrebno pretraživati u cilju traženja što kraćih lanaca.

2.6. Problematika traženja kratkih adicijskih lanaca

Već kod ovih, dosada prikazanih, malenih eksponenata vidi se kako postoji mnoštvo različitih adicijskih lanaca za zadani eksponent. U primjerima 2.1 - 2.3 prikazana su zapravo tri bitna predstavnika adicijskih lanaca za eksponent 15:

1. Prvi slučaj je trivijalno rješenje, najgori slučaj u kojem je za eksponent n potrebno obaviti $n - 1$ zbrajanja u domeni eksponenta. U konkretnom primjeru, koristi se 14 zbrajanja te je duljina adicijskog lanca 14.

2. Drugi slučaj je relativno dobro rješenje dobiveno osnovnom deterministčkom metodom, binarnom metodom, opisanom kasnije, koja kao rješenje daje $\lfloor \log_2 n \rfloor$ udvostručenja te u najgorem slučaju $\lfloor \log_2 n \rfloor$ zbrajanja. U konkretnom primjeru, koristi se 3 udvostručenja i 3 zbrajanja te je duljina adicijskog lanca 6.

3. Treći slučaj je rješenje dobiveno ručnim izračunom[1], koje je ujedno i optimalno. Koristi se 3 udvostručenja te 2 zbrajanja, a duljina adicijskog lanca je 5. Ovo nije jedino optimalno rješenje za dani eksponent.

Isprva se čini kako optimalno rješenje nije puno drukčije, odnosno znatno bolje od rješenja dobivenog determinističkom metodom te da je korištena metoda *dovoljno dobra*. Međutim, podsjetimo se da se u praksi koriste brojevi reda veličine 2^{120} i veći, te je kod takvih brojeva razlika značajna, što će se kasnije i vidjeti u poglavlju 6.

Budući da je pronalazak optimalnog rješenja NP-kompletan problem, dolazimo do pitanja: Kako se što više približiti optimalnom rješenju? Na koji način je moguće pronaći dovoljno dobre rezultate s obzirom na uložene računalne resurse? Koje algoritme upotrijebiti ako je: (1) svrha pronaći najbolje rješenje za zadani eksponent koje će se ubuduće koristiti bez ponovnog računanja u odnosu na (2) traženje najboljeg rješenja uz zadano vremensko ograničenje (*online* izračun)? Ovisno o primjeni, koriste se i determinističke i stohastičke metode te njihove kombinacije. Reprezentativni primjeri postojećih, kao i vlastitih novorazvijenih metoda te dobiveni rezultati obrađeni su u poglavljima koja slijede.

[1]Do ovakvog rezultata se može doći ručno, ili nekakvom računalnom heuristikom. Međutim, kako je ranije spomenuto, ne postoji heuristika koja garantira optimalno rješenje za svaki broj.

3. Evolucijski algoritam za pronalaženje kratkih adicijskih lanaca

Evolucijski algoritmi predstavljaju skup metoda koje imaju sljedeća svojstva:

1. **Osnovna načelo rada je Darwinova teorija evolucije.** Rješenja koja je moguće dobiti metodom predstavljena su jedinkama, koja imaju određena svojstva (gene) te vremenom evoluiraju. Trenutni skup rješenja koji se razmatra je predstavljen populacijom, a svakom sljedećom generacijom, očekuje se sve kvalitetnija populacija. Kvaliteta jedinke ocjenjuje se funkcijom dobrote koja za zadanu jedinku daje podatak o njenoj kvaliteti. Svaka nova populacija se generira iz prethodne raznim genetskim operatorima te se, statistički gledano, sve više i više kvalitetnih gena prenosi u buduće generacije. Naposljetku, najbolja ikad pronađena jedinka, koja je vrlo često sadržana u posljednjoj generaciji, predstavlja konačno rješenje koje je metoda proizvela.

2. **Metoda stohastičkim putem dolazi do rješenja.** Stohastička priroda evolucijskih algoritama slijedi iz samog načela evolucije, budući da se početna populacija generira nasumično. Uz to, genetski operatori te odabir jedinki također sadrže nasumičnost, uz određene preferencije prema kvalitetnijim jedinkama. Posljedica stohastičnosti je nemogućnost višestrukog reproduciranja jednako provedenog postupka evolucije. Metoda, zbog svega navedenog, neće svakim pokretanjem kao rezultat dati jednako rješenje.

3. **Vrijeme izvođenja nije definirano.** Budući da je evolucija, kao kontinuirani proces, temelj evolucijskih algoritama, teoretski, metoda može raditi beskonačno dugo. Što se više generacija evaluira, veća je vjerojatnost pronalaska optimalnog rješenja. Međutim, poboljšanje je uglavnom najvidljivije početkom pokretanja metode, tokom prvih generacija, dok su kasnije promjene znatno sporije, a metoda je bliža konvergenciji. Zbog nepostojanja završetka evolucije, uvijek se

postavlja *kriterij zaustavljanja* npr. na maksimalan broj evaluacija jedinke, broj uzastopnih generacija sa stagnacijom, maksimalno vrijeme izvođenja i dr.

4. **Metoda ne garantira optimalno rješenje.** O zadanom problemu ovisi uspješnost metode u približavanju optimumu. Uz to, veliki faktor predstavlja i domena primjene u smislu koliko dobro rješenje je *dovoljno dobro* kako bi se evolucija prekinula te prihvatilo dobiveno rješenje. Uz to, gotovo uvijek je prilikom izvođenja metode u stvarnoj primjeni, u potpunosti nepoznato je li metoda pronašla upravo optimum.

5. **Radi statističke korektnosti, metoda se izvodi više puta.** Postoji velika mogućnost da je od samog početka evolucija osuđena na propast, ukoliko je početna, nasumično generirana populacija izrazito loša. Posljedica toga je zaglavljenje metode u lokalnom optimumu, naročito ako je funkcija dobrote izrazito kompleksna, ili je jedinka sačinjena od mnogo gena.

Uz druge metode, evolucijskim algoritmima pripada i *genetski algoritam*, koji je korišten u ovome radu zbog svoje primjenjivosti na dani problem.

3.1. Opće načelo rada genetskog algoritma

Genetski algoritam vjerojatno je najpoznatiji predstavnik evolucijskih algoritama. U najširem smislu, koristi se za problem optimizacije funkcije, pri čemu se stvarni problem modelira funkcijom dobrote koja se genetskim algoritmom optimira.

Algoritmom 1 na sljedećoj stranici prikazan je opći princip rada genetskog algoritma - svaka pojedina implementacija ove metode, bez obzira na primjenu, uz manje preinake može se svesti na prikazani način rada. Slijedi objašnjenje rada uz opis pojedinih funkcija unutar pseudokoda.

generirajPočetnuPopulaciju() Početkom pokretanja populacija je prazna. Stoga, potrebno je nasumično generirati jedinke koje će sačinjavati početnu populaciju. Ovo stvaranje populacije može biti i djelomično nasumično, uz dodavanje nekih predefiniranih jedinki kako bi se povećao prosjek dobrote populacije. Početna populacija je obično relativno loša budući da su jedinke (ili barem većina njih) nasumično generirane. Ipak, bitnije svojstvo koje je itekako poželjno je visoka raznolikost genetskog materijala, kako bi postojala veća vjerojatnost za stvaranje sve boljih jedinki.

Algoritam 1: Načelo rada općenitog genetskog algoritma

Ulaz: p_m - vjerojatnost mutacije, p_c - vjerojatnost križanja,
N - veličina populacije
Podatci: $f_{fitness} : jedinka \rightarrow broj$ - funkcija dobrote
Rezultat: Najbolja pronađena jedinka

```
populacija = generirajPočetnuPopulaciju();
dok !zadovoljenKriterijZaustavljanja() čini
    novaPopulacija = {};
    dok novaPopulacija.veličina < N čini
        roditelji = odaberiBudućeRoditelje(populacija);
        djeca = primijeniGenetskeOperatore(roditelji, p_m, p_c);
        evaluirajJedinke(djeca, f_fitness);
        novaPopulacija.dodaj(djeca);
    kraj
    populacija = novaPopulacija;
kraj
vrati najboljaJedinkaPopulacije(populacija)
```

zadovoljenKriterijZaustavljanja() Kako je ranije spomenuto, genetski algoritam je potrebno prisilno zaustaviti, kako se ne bi izvršavao u beskonačnoj petlji. Stoga se interno implementira kriterij zaustavljanja prema zadanom problemu te se uglavnom nakon početnog ispitivanja više ne podešava, odnosno, nije ulazni parametar u metodu.

odaberiBudućeRoditelje(populacija) Ova funkcija predstavlja operator *selekcije*, kojim je definirano na koji način se odabiru jedinke-roditelji. Implementacija selekcije je izuzetno bitna, budući da direktno uvjetuje koji će se geni prenijeti u sljedeću generaciju. Često se zasniva na polunasumičnom odabiru s preferencijom ka kvalitetnijim jedinkama, tako da se poveća vjerojatnost propagacije dobrih gena. S druge strane, odabir isključivo najboljih jedinki za roditelje lako dovodi do zaglavljenja u lokalnom optimumu, budući da je velika vjerojatnost da su najbolje jedinke međusobno genetski vrlo slične te se javlja manjak novog genetskog materijala.

primijeniGenetskeOperatore(roditelji, p_m, p_c) Nakon odabira roditelja (obično dva) potrebno je definirati način na koji će se njihovi geni prenijeti u sljedeću genera-

ciju. Osnovni razredi genetskih operatora su *križanje* i *mutacija*, a svaki od njih je moguće implementirati na mnoštvo različitih načina. Križanje se svodi na miješanje gena oba roditelja te stvaranje jedne ili više novih jedinki, koje imaju gene oba roditelja. U slučaju mutacije, geni se ne miješaju, nego se odabranoj jedinki (roditelju) nasumično modificira određeni gen, kako bi se u populaciju doveo novi, potencijalno bolji gen. Uz ove dvije operacije, moguće je vršiti i *replikaciju*, pri čemu se roditelji jednostavno kopiraju u novu generaciju, bez ikakve modifikacije gena.

Ulazni parametri u genetski algoritam, koji su ujedno i parametri za primjenu genetskih operatora su vjerojatnosti mutacije p_m i križanja p_c. Ova dva parametra najviše utječu na rad algoritma, budući da se njima direktno određuje omjer između propagacije dobrih gena pri križanju te dovođenju novih, moguće neviđenih gena prilikom mutacije. Ukoliko je udio križanja prevelik, populacija će postati previše jednolika, te će nastupiti stagnacija i zaglavljenje u lokalnom optimumu. S druge strane, ukoliko je udio mutacije prevelik, rad metode neće biti dovoljno usmjeren - metoda će se početi ponašati više kao nasumični odabir nego kao heuristika.

evaluirajJedinke(djeca, $f_{fitness}$) Kako bi heuristika imala smisla, potrebno je na neki način bodovati jedinke kako bi se mogla ustanoviti razlika između boljih i lošijih jedinki te shodno tome preferirati bolje jedinke. Funkcijom dobrote se evaluiraju novodobivene jedinke te im se u strukturu zapisuje dobrota. Ovisno o problemu, funkcija dobrote može biti računalno vrlo skupa, stoga se često i kao kriterij zaustavljanja postavlja upravo ukupan broj izvršavanja funkcije dobrote tokom rada algoritma. Bitno je naglasiti da, gdjegod da je postupak evaluacije smješten u algoritmu, nužno je da su jedinke unutar populacije nad kojom se vrši selekcija već evaluirane, kako bi se mogle preferirati bolje jedinke prilikom odabira.

najboljaJedinkaPopulacije(populacija) Postupak koji obuhvaća sve prethodne funkcije, osim stvaranja početne populacije, ponavlja se do kriterija zaustavljanja, a svako ponavljanje predstavlja sljedeću generaciju populacije. Zadovoljavanjem kriterija i izlaskom iz petlje, iz populacije se izdvaja najbolja pronađena jedinka te se vraća kao rezultat izvršavanja algoritma.

3.2. Prikaz jedinke i funkcija dobrote

Prikaz jedinke definira način na koji se jedinka pohranjuje u računalo. Ovo svojstvo se još naziva i *genotip*, budući da predstavlja genetsku kartu svake pojedine jedinke, za razliku od *fenotipa*, koji je manifestacija genotipa u stvarnom svijetu.

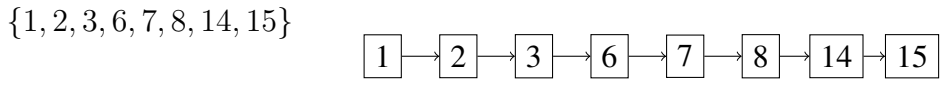
O prikazu jedinke direktno ovisi dizajn funkcije dobrote te implementacija genetskih operatora. Uz to, različiti načini prikaza međusobno imaju različite prednosti i nedostatke, a genetski algoritam može koristiti samo određeni, unaprijed zadani prikaz te pripadnu funkciju dobrote i genetske operatore. U okviru rada, razmatraju se dva, izuzetno različita prikaza te su objašnjeni u potpoglavljima koja slijede.

U okviru problema traženja kratkih adicijskih lanaca, jedinka je upravo adicijski lanac. Tako će populacija biti sastavljena od određene količine adicijskih lanaca, od kojih su neki bolji (kraći), a neki lošiji. U kontekstu genetskog algoritma, govorimo o uspješnijim jedinkama s većom dobrotom i manje uspješnim jedinkama s manjom dobrotom. Što je lanac kraći, dobrota mu je veća, pa se problem traženja kratkih adicijskih lanaca svodi na minimizaciju funkcije dobrote pri čemu je najintuitivnije funkciju dobrote direktno definirati kao duljinu lanca.

Uz različite genotipe, adicijski lanac će imati jednak fenotip - rastući niz brojeva koji zadovoljava svojstva definicije adicijskog lanca.

3.2.1. Vektorski prikaz jedinke

U okviru projekta ECF (Jakobović et al., 2017), koristi se vektorski prikaz jedinke - adicijski lanac prikazan je rastućim vektorom brojeva koji sačinjavaju lanac. Ovaj prikaz je izuzetno interpretativan, budući da je adicijski lanac direktno pohranjen u memoriju onako kakvim on jest - kao rastući niz prirodnih brojeva. Pri ovom prikazu, može se reći da je genotip jednak fenotipu, što se vidi na slici 3.1.



(a) Fenotip **(b)** Genotip

Slika 3.1: Fenotip i genotip vektorskog prikaza

Veze na grafu koji predstavlja genotip ne predstavljaju veze adicijskog lanca, nego prirodu vektora pohranjenog u računalo. Stoga, pri ovom prikazu ne čuvaju se podaci o povezanosti elemenata nego se dinamički izračunavaju ukoliko je to potrebno.

Nedostatak ovakvog prikaza je računalno zahtjevno popravljanje jedinki. Naime, prilikom primjene genetskog operatora na jedinku, velika je vjerojatnost da je lanac postao neispravan - najčešće u smislu da se neki od novodobivenih elemenata u lancu ne mogu dobiti kao zbroj nekih od postojećih elemenata u istom. Zbog toga je potrebno prilikom svake genetske operacije provesti provjeru novodobivenog lanca te eventualno i popravak, odnosno, izgradnju podlanca koji nedostaje za određeni element ili više njih. Ova situacija se dodatno pogoršava zbog činjenice da se u genotipu ne pamte veze između elemenata, stoga je provjera lanca relativno skupa. Ipak, izostankom upisanih veza među elementima, genetske operacije su računalno jeftinije te ne postoji potreba za stalnim održavanjem ispravnih veza među elementima.

3.2.2. Stablasti prikaz jedinke

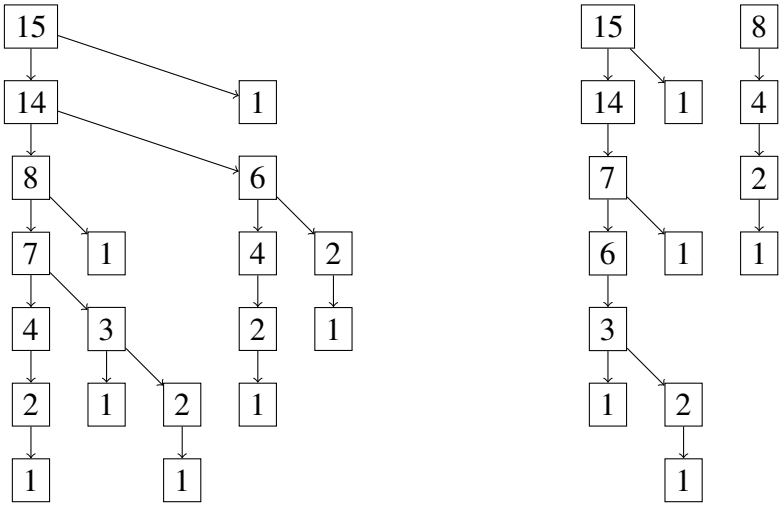
Kako bi se uklonio problem nezapisanih veza među elementima, u okviru ovog rada implementiran je novi genotip, koji predstavlja lance kao stablastu strukturu. Genotip i fenotip u ovom slučaju nisu jednaki, budući da definicija adicijskog lanca ne obuhvaća veze među elementima - nije bitno koji su elementi međusobno povezani dok god je lanac ispravan. Međutim, ovim prikazom je omogućeno razmatranje veza na razini kako genotipa tako i fenotipa. Pri prikazu fenotipa dovoljno je, umjesto niza brojeva, prikazivati stablo grafa zapisanog u genotipu. Na slici 3.2 prikazani su fenotip te neki od mogućih genotipa koji daju isti fenotip.

U stablastoj strukturi se vidi kako je svaki lanac predstavljen binarnim stablom, pri čemu čvorovi-djeca predstavljaju pribrojnike elementa-roditelja. Uz to, kako bi se uvelike pojednostavnila implementacija genetskih operacija, ustanovljeno je pravilo da je lijevo dijete uvijek veće ili jednake vrijednosti u odnosu na desno.

Nedostatak ovakvog prikaza je skupo održavanje veza među čvorovima pri modifikaciji jedinke. Osim toga, prilikom razmatranja jedinke, ona je na neki način ograničena upisanim vezama - kada ne bi bilo predefiniranih veza, bilo bi moguće optimirati neki podlanac izbacivanjem viška elemenata koji ne sudjeluju u stvaranju tog podlanca.

$$\{1, 2, 3, 6, 7, 8, 14, 15\}$$

(a) Fenotip



(b) Genotip 1 **(c)** Genotip 2

Slika 3.2: Fenotip i genotip stablastog prikaza

3.3. Prednosti i nedostaci genetskog algoritma

Prema svemu navedenom, genetski algoritam je vrlo istražen i široko korišten u računarskoj znanosti, upravo zbog svoje sveobuhvatnosti i mogućnosti traženja *dovoljno dobrih* rješenja za zadani problem. Stoga, prednosti korištenja ovakvog algoritma bi se mogle svesti na sljedeće:

1. **Mogućnost ulaganja vremenskog resursa u kvalitetu dobivenih rješenja.** Ukoliko postoji zadani problem za koji je potrebno pronaći što je moguće bolje rješenje problema kako bi se moglo kasnije redovito koristiti, isplati se uložiti procesorsko vrijeme za dobivanje što boljeg rezultata. Naprimjer, ako je iz nekog razloga potrebno sklopovski implementirati potenciranje na eksponent $2^{127} - 3$ da se odvija u što kraćem vremenskom roku, isplati se pokrenuti genetski algoritam da pretražuje rješenja neki dulji period, npr. mjesec ili dva. Nakon toga se dobiveno rješenje može sklopovski ugraditi te se kasnije korištenjem tog sklopa dobiva visoka brzina enkripcije ili dekripcije.

2. **Fleksibilnost algoritma za primjenu na raznim problemima.** Genetski algoritam je najintuitivnije implementirati kao radni okvir (engl. *framework*) koji se zatim koristi za rješavanje različitih problema, kao što je to slučaj kod ECF-a. Na

korisniku je da konkretan problem modelira pomoću prikaza jedinke, genetskih operatora te funkcije dobrote.

3. **Mogućnost istraživanja dobrih gena s ciljem boljeg razumijevanja problema.** Često je moguće analizom zadnje generacije ustanoviti koji su to geni koji su doveli do visokog stupnja dobrote jedinke. U slučaju adicijskih lanaca, može se dogoditi recimo da sve dobre jedinke sadrže neki ključni broj koji sudjeluje kao pribrojnik mnogih drugih elemenata lanca te je samim time lanac kraći, budući da je dobra iskoristivost tog ključnog broja.

4. **Mogućnost paralelizacije.** Jedan od načina paralelizacije bi bio dijeljenje populacije na podpopulacije koje nezavisno evoluiraju te se periodično agregiraju u jednu veliku populaciju radi miješanja gena, evaluacije i dr. Osim toga, može se paralelizirati sama evolucija jedne generacije - budući da populacija sadrži stotine jedinki, može se paralelno vršiti selekcija više jedinki odjednom te genetski manipulirati njima.

Uz ove i mnoge druge prednosti, nedostaci se mogu svesti na sljedeće:

1. **Nemogućnost garancije optimalnog rješenja.** U većini problema, traži se optimum funkcije dobrote čija je vrijednost u potpunosti nepoznata. U takvim situacijama, bez obzira na duljinu izvođenja algoritma, nikada nije jasno je li se došlo do optimalnog rješenja ili je algoritam zaglavio u lokalnom optimumu. Zbog toga je katkada teško procijeniti kako odrediti kriterij zaustavljanja, uz garanciju da algoritam, uz dodatan rad, neće doći do boljeg rješenja.

2. **Parametrizacija algoritma.** Genetski algoritam, slično kao algoritmi strojnog učenja, ima tzv. hiperparametre (veličina populacije te vjerojatnosti genetskih operacija) koje je potrebno namjestiti tako da algoritam pronalazi što bolja rješenja. Stoga je potrebno prije samog korištenja algoritma prvo istražiti ispitnim pokretanjima optimalnu kombinaciju parametara. Tek dobro postavljenim parametrima algoritam je u stanju evolucijom dolaziti do dobrih rješenja.

3. **Stohastička priroda.** Pronalaženje kvalitetnog rješenja je gotovo nemoguće reproducirati na jednak način na koji je odrađeno ranijim radom algoritma. U većini slučajeva, to nije niti bitno ako se traži dobro rješenje koje se pohranjuje i kasnije koristi bez ikakve daljnje interakcije s algoritmom. Ipak, zbog toga, genetski algoritam teško nalazi primjenu u sustavima gdje je važna konzistentnost u traženju rješenja.

Genetskim algoritmima, kao važnim predstavnicima stohastičkih metoda suprotnost predstavljaju determinističke metode, koje svojim karakteristikama predstavljaju upravo komplementaran način rješavanja istog problema. Sljedeće poglavlje razmatra determinističke metode, kako postojeće tako i novorazvijene u sklopu ovog rada.

4. Determinističke metode za pronalaženje kratkih adicijskih lanaca

Pri rješavanju bilo kojeg matematičkog problema, nakon formulacije, logično je osmisliti deterministički algoritam kojim je moguće doći do dovoljno dobrog rješenja. Tako su se i pri problemu traženja kratkih adicijskih lanaca prvotno osmislili algoritmi kojima je moguće u kratkom vremenu pronaći relativno kratke adicijske lance.

Uz postojeće i dobro poznate algoritme, u okviru ovog rada osmišljena je i nova metoda koja predstavlja proširenje postojećeg algoritma te daje znatno bolje rezultate od postojećih. Sljedeća potpoglavlja opisuju dvije poznate metode te spomenutu novorazvijenu metodu zajedno s parametrizacijom kojom se iz iste metode dobiva više različitih načina rada te samim time dolazi do različitih rezultata.

4.1. Binarna metoda

Binarna metoda je osnovni algoritam, intuitivno vrlo jasan, računalno nezahtjevan te se brzo izvodi. Ovaj algoritam ne dolazi do jako dobrih rješenja, no zgodan je uvod u deterministička optimizaciju adicijskih lanaca te se korištenjem ovog algoritma mogu uočiti neka vrlo važna svojstva adicijskih lanaca.

Intuitivno, rad ovog algoritma je vrlo sličan postupku pretvaranja binarnog broja u dekadski uzastopnim dijeljenjem s 2. Pseudokod 2 prikazuje način rada binarne metode. Zadani broj se uzastopno dijeli s 2 ako je paran, odnosno, oduzima se 1 ako je neparan. Novodobiveni broj se dodaje u lanac te se postupak ponavlja sve dok zadani broj ne poprimi vrijednost 1. Niz svih novodobivenih brojeva je upravo ispravan

adicijski lanac za početno zadani broj.

Algoritam 2: Načelo rada binarne metode

Ulaz: n - zadani broj za koji se traži adicijski lanac

Rezultat: Adicijski lanac A

$A \leftarrow \{n\}$;

dok $n > 1$ **čini**

  ako $n \mod 2 = 0$ **onda**

    $n \leftarrow n/2$;

  inače

    $n \leftarrow n - 1$;

  kraj

  $A.dodaj(n)$

kraj

vrati A

Slijedi prikaz rada algoritma na broju 54. Međurezultati su prikazani i u binarnom obliku, kako bi se vidjela jednostavnost izvedbe ovog algoritma ukoliko se radnje obavljaju na razini bitova. U ostalim algoritmima razmatranje na bitovnoj razini će biti od presudnog značaja, budući da se izvode kompliciranije operacije u odnosu na binarnu metodu.

$$
\begin{aligned}
n_0 = n = 54 \quad & n_0 = 110110_2 \quad && A_0 = \{54\} \\
n_1 = 54/2 = 27 \quad & n_1 = 11011_2 \quad && A_1 = \{27, 54\} \\
n_2 = 27 - 1 = 26 \quad & n_2 = 11010_2 \quad && A_2 = \{26, 27, 54\} \\
n_3 = 26/2 = 13 \quad & n_3 = 1101_2 \quad && A_3 = \{13, 26, 27, 54\} \\
n_4 = 13 - 1 = 12 \quad & n_4 = 1100_2 \quad && A_4 = \{12, 13, 26, 27, 54\} \\
n_5 = 12/2 = 6 \quad & n_5 = 110_2 \quad && A_5 = \{6, 12, 13, 26, 27, 54\} \\
n_6 = 6/2 = 3 \quad & n_6 = 11_2 \quad && A_6 = \{3, 6, 12, 13, 26, 27, 54\} \\
n_7 = 3 - 1 = 2 \quad & n_7 = 10_2 \quad && A_7 = \{2, 3, 6, 12, 13, 26, 27, 54\} \\
n_8 = 2/1 = 1 \quad & n_8 = 1_2 \quad && A = A_8 = \{1, 2, 3, 6, 12, 13, 26, 27, 54\}
\end{aligned}
$$

4.2. Metoda prozora

Osnova rada metode prozora je pronalaženje uzoraka u binarnom zapisu broja, čija je duljina unaprijed zadana parametrom veličine prozora. Ponavljajući uzorci se iskorištavaju kako bi se izgradio lanac sa što manje uvođenja novih elemenata te kako bi samim time imao i kraću duljinu.

Za zadanu duljinu prozora u bitovima w, u lanac se prvo dodaju svi neparni brojevi najveće duljine w bitova i broj 2, koji je nužno prisutan u svakom adicijskom lancu. Nakon toga se u bitovnom prikazu broja pronalaze sva pojavljivanja spomenutih unaprijed određenih brojeva. Jednostavnim množenjem s 2 te zbrajanjem s tim brojevima konačno se dolazi do traženog broja.

Slijedi prikaz rada algoritma na broju 54, uz veličinu prozora 2. Oznaka n_i predstavlja dio broja koji se u i-toj iteraciji razmatra, a w_i pronađeni uzorak. W je skup svih w-bitnih uzoraka:

1. Inicijalno, skup prozora W popunjava se svim neparnim brojevima do maksimalne veličine 2 bita te se dodaje i broj 2. Svi elementi skupa prozora ujedno se dodaju u lanac.
   $54_{10} = 110110_2, \quad A_0 = \{1, 10, 11\}, \quad W = \{1, 10, 11\}$

2. Prvi uočeni uzorak je 11 na mjestu 0. Već je u lancu pa se ne dodaje. Ujedno je prvi broj koji se razmatra upravo 11.
   $\underbrace{\overbrace{11}^{w_0}}_{n_0}0110 \quad A_0 = \{1, 10, 11\}$

3. Aktivni broj se množi s dva te se novodobiveni broj 110 dodaje u lanac. Iz skupa uzoraka, sljedeći uočeni broj je opet 11, označen s w_n. Aktivni broj se množi s dva sve dok aktivni broj ne obuhvati nađeni uzorak.
   $\underbrace{110}_{n_1}\overbrace{11}^{w_n}0, \quad A_1 = \{1, 10, 11, 110\}$

4. Svakim korakom se novodobiveni broj dodaje u lanac.
   $\underbrace{1101}_{n_2}10, \quad A_2 = \{1, 10, 11, 110, 1100\}$

5. Aktivni broj je konačno obuhvatio nađeni uzorak w_n.
   $\underbrace{11011}_{n_3}0, \quad A_3 = \{1, 10, 11, 110, 1100, 11000\}$

6. Aktivni broj se zbraja s pronađenim uzorkom te se dodaje u lanac i skup uzoraka.
   $\underbrace{110\overbrace{11}^{w_4}}_{n_4}0, \quad A_4 = \{1, 10, 11, 110, 1100, 11000, 11011\}$

$\underbrace{110110}_{n_5}, \quad A_5 = \{1, 10, 11, 110, 1100, 11000, 11011, 110110\}$

7. Aktivni broj jednak je zadanom broju te postupak završava s radom.
   $A = \{1, 2, 3, 6, 12, 24, 27, 54\}$

Valja primijetiti kako je ova metoda i na ovom manjem broju dala bolji rezultat u odnosu na binarnu metodu. Uz to, ovo je ujedno i optimalan adicijski lanac za broj 54. Ipak, pronalaženje optimalnog rješenja za veće brojeve u stvarnoj primjeni korištenjem ove metode gotovo je nemoguće. No, kako će se kasnije pokazati, ova metoda ipak daje znatno bolje rezultate od binarne metode, a razlika je tim veća što su veći brojevi koji se razmatraju.

4.3. Proširena metoda prozora

Pri radu s metodom prozora, naročito kad su u pitanju veći brojevi, uočeno je sljedeće:

1. **Premalena veličina prozora onemogućuje iskorištavanje velikih ponavljajućih uzoraka.** Ukoliko se razmatra npr. broj $1110001100111000 11_2$, uz veličinu prozora 3, mogu se iskoristiti uzorci 111 i 11. Međutim, na ovom broju bi bilo daleko bolje iskoristiti ponavljajući uzorak 11100011 te samim time dobiti kraći lanac.

2. **Prevelika veličina prozora dodaje ogromnu količinu suvišnih elemenata u lanac.** Ako se odabere npr. veličina prozora 8, kako bi se pokrio spomenuti veliki uzorak, u lanac se prvotno dodaju svi neparni brojevi manji od $2^8 = 256$, kojih ima $256/2 = 128$. Iz ovoga se vidi da je besmisleno uzimati preveliku veličinu prozora stoga se obično uzima veličina manja od 7.

Nova, proširena metoda razvijena u okviru ovog rada upravo rješava ovaj problem na vrlo elegantan način. Osnovna metoda se proširuje tako da se svaki novododani element lanca dodaje i u skup uzoraka koji se razmatraju. Ovime se početni skup uzoraka progresivno proširuje svim elementima lanca kako se on izgrađuje. Slijedi prikaz rada proširene metode na istom primjeru, broju 54:

1. Inicijalno, skup prozora W popunjava se svim neparnim brojevima do maksimalne veličine 2 bita. Svi elementi skupa prozora ujedno se dodaju u lanac.
   $54_{10} = 110110_2, \quad A_0 = W_0 = \{1, 10, 11\}$

2. Prvi uočeni uzorak je 11 na mjestu 0. Već je u lancu pa se ne dodaje. Ujedno je prvi broj koji se razmatra upravo 11.
   $\underbrace{\overbrace{11}^{w_0}}_{n_0}0110, \quad A_0 = W_0 = \{1, 10, 11\}$
3. Aktivni broj se množi s dva te se novodobiveni broj 110 dodaje u skup uzoraka i lanac. Iz skupa uzoraka, sljedeći uočeni broj je opet 110, označen s w_n. Aktivni broj se množi s dva sve dok aktivni broj ne obuhvati nađeni uzorak. Naglasak je na dodavanju broja 110 i u skup uzoraka - to je osnovna razlika ove metode u odnosu na osnovnu.
   $\underbrace{110}_{n_1}\overbrace{110}^{w_n}, \quad A_1 = W_1 = \{1, 10, 11, 110\}$
4. Svakim korakom se novodobiveni broj dodaje i u lanac i u skup uzoraka.
   $\underbrace{1101}_{n_2}10, \quad A_2 = W_2 = \{1, 10, 11, 110, 1100\}$
   $\underbrace{11011}_{n_3}0, \quad A_3 = W_3 = \{1, 10, 11, 110, 1100, 11000\}$
5. Aktivni broj je konačno obuhvatio nađeni uzorak w_n.
   $\underbrace{110110}_{n_4}, \quad A_4 = W_4 = \{1, 10, 11, 110, 1100, 11000, 110000\}$
6. Aktivni broj se zbraja s pronađenim uzorkom te se dodaje u lanac i skup uzoraka.
   $\underbrace{110\overbrace{110}^{w_5}}_{n_5}, \quad A_5 = W_5 = \{1, 10, 11, 110, 1100, 11000, 110000, 110110\}$
7. Aktivni broj jednak je zadanom broju te postupak završava s radom.
   $A = \{1, 2, 3, 6, 12, 24, 48, 54\}$

Duljina lanca je jednaka duljini lanca dobivenog osnovnom metodom prozora. Međutim, elementi koji ga čine su drukčiji, jer je proširena metoda u stanju primijetiti i veće uzorke. Ova prednost je naročito izražena kod većih brojeva, kod kojih su vjerojatniji dulji ponavljajući uzorci.

Ova, proširena metoda implementirana je tako da je moguće podesiti rad upotrebom sljedećih parametara:

Veličina prozora *w* Ovaj parametar je prisutan i u osnovnoj metodi prozora. Nužno je definirati veličinu prozora kako bi algoritam mogao inicijalizirati skup uzoraka koji se koriste prilikom pretrage. Brojevnog je tipa, a vrijednosti koje su smislene za korištenje su između 2 i 7.

Prošireni način rada *e* Ukoliko je uključen, koristi se opisana proširena metoda prozora. Ako je isključen, metoda se ponaša kao osnovna metoda. Uključivanjem

ovog parametra, stvara se početni skup uzoraka najveće duljine *w* bitova, a dalje se onda taj skup proširuje ostalim elementima lanca. Ovaj način rada je osmišljen tokom izrade ovog rada.

Optimiranje niza jedinica adicijskim lancima *ac* Prilikom rada na brojevima koji sadrže veliki broj uzastopnih bitovnih jedinica, uključivanje ovog parametra može doprinijeti pronalaženju kraćih lanaca nego kada je optimiranje isključeno, a funkcionalnost parametra je osmišljena i implementirana u okviru ovog rada. Način rada parametra opisan je u sljedećem potpoglavlju.

4.3.1. Parametar *ac*

Prilikom istraživanja rada raznih algoritama u okviru ovog rada, najzanimljivije je ponašanje algoritama na velikim brojevima, kakvi se koriste i u stvarnoj primjeni. Neki od brojeva koji se ispituju imaju svojstvo da sadrže veliki broj uzastopnih bitovnih jedinica. Jedan od takvih brojeva je i $2^{37} - 3$, čiji je binarni zapis:

$$11111111\ \ 11111111\ \ 11111111\ \ 11111111\ \ 11101$$

Optimiranje parametrom *ac* bit će objašnjeno na ovom primjeru, bez smanjivanja općenitosti. Na ovakvom broju cilj se sa što manje operacija izgraditi dio lanca koji sadrži jedinice. Prilikom izgradnje ovog podlanca potrebno je rukovoditi se sljedećim:

1. Neka je broj z_i takav da ima i bitova i to sve bitovne jedinice. Od broja z_j do broja z_k dolazi se pomoću $k - j$ množenja s 2, te nakon toga dodavanjem broja z_{k-j}. Npr. od broja 111 do broja 11111 dolazi se na sljedeći način:

   $$111 \rightarrow 1110 \rightarrow 11100 \rightarrow 11111$$

2. Kako bi se došlo od broja z_j do z_k, nužno je provesti $k - j$ množenja s 2. Međutim, broj dodavanja nekog broja z_l treba svesti na minimum. Npr. rješenje

   $$111 \rightarrow 1110 \rightarrow 11100 \rightarrow 111\underbrace{11}$$

   bolje je od rješenja

   $$111 \rightarrow 1110 \rightarrow 111\underbrace{1} \rightarrow 11110 \rightarrow 1111\underbrace{1}$$

3. Broj takvih zbrajanja direktno je vezan za količinu brojeva z_i koji se nalaze u lancu. Što je manje takvih brojeva, bit će i manje zbrajanja.

Prilikom ovih zapažanja, problem se može modelirati na sljedeći način: Potrebno je pronaći minimalan broj elemenata z_i pomoću kojih je moguće izgraditi cijeli podlanac z_k. Ako brojeve z_i transformiramo u njihove duljine binarnog prikaza i, problem se svodi upravo na traženje najkraćeg adicijskog lanca za zadanu bitovnu duljinu k. Način rada algoritma uz uključen parametar *ac* za broj 2039 može se opisati na sljedeći način:

1. **Za bitovni prikaz broja n razmatra se prvi niz jedinica z_i.**
   Konkretno, $n = 2039_{10} = 11111110111_2$ te se uzima z_7 kao prvi aktivni broj. Stoga, obrađuje se $\underbrace{1111111}_{z_7}0111$.

2. **Pomoću spremljene mape računa se optimalni adicijski lanac A_i za broj *i*.** U programu je pohranjena mapa svih optimalnih adicijskih lanaca do broja 620, čime se osigurava pouzdan rad parametra *ac* do nizova jedinica duljine 620 bitova. U ovom primjeru, traži se optimalni adicijski lanac za broj 7 te je on $1 \to 2 \to 3 \to 5 \to 7$

3. **Elementi lanca A_i transformiraju se natrag u bitovne brojeve z.** Konkretno, $A_i = \{1, 2, 3, 5, 7\}$ transformira se u

$$\begin{aligned} W_{z_k} &= \{z_1, z_2, z_3, z_5, z_7\} \\ &= \{1, 11, 111, 11111, 1111111\} \end{aligned}$$

4. **Podlanac z_k se gradi pomoću W_{z_k}.** U ovom slučaju, z_k se gradi prema:

$$\begin{aligned} &1 \to 10 \to \underbrace{1}_{z_1}\underbrace{1}_{z_1} \to 110 \to \underbrace{11}_{z_2}\underbrace{1}_{z_1} \to 1110 \\ &\to 11100 \to \underbrace{111}_{z_3}\underbrace{11}_{z_2} \to 111110 \to 1111100 \to \underbrace{11111}_{z_5}\underbrace{11}_{z_2} \end{aligned}$$

5. **Nakon izgradnje podlanca, zadnji aktivni broj množi se s 2 dok se ne dođe do novog niza jedinica.** Na ovom primjeru, slijedi postupak

$$1111111 \to 11111110$$

6. **Postupak se ponavlja za novi niz jedinica.** Konkretno, postupak se ponavlja za z_3, a u ovom slučaju z_3 je već izračunat te postoji u skupu uzoraka:

$$\begin{aligned} W_{z_3} &= \{z_1, z_2, z_3\} \\ &= \{1, 11, 111\} \end{aligned}$$

   te je stoga nastavak izgradnje:

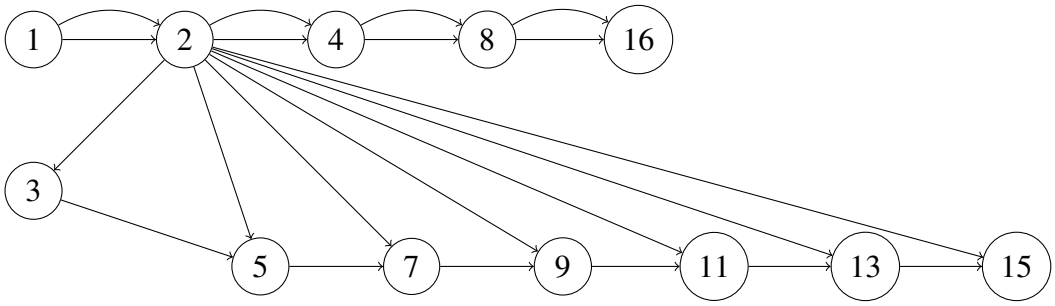
$$11111110 \to 111111100 \to 1111111000 \to 11111110000 \to 11111110111$$

Primjer iz prakse je broj $2^{37}-3$ koji sadrži 35 jedinica zaredom. Optimalni adicijski lanac za broj 35 je $1 \to 2 \to 4 \to 8 \to 9 \to 13 \to 26 \to 35$. Slijedi prikaz postupka izgradnje adicijskog lanca broja $2^{37} - 3$ s parametrom *ac*:

1	1
10	
$\underbrace{1}_{z_1}\underbrace{1}_{z_1}$	$1+1=2$
110	
1100	
$\underbrace{11}_{z_2}\underbrace{11}_{z_2}$	$2+2=4$
11110	
...	
11110000	
$\underbrace{1111}_{z_4}\underbrace{1111}_{z_4}$	$4+4=8$
111111110	
$\underbrace{11111111}_{z_8}\underbrace{1}_{z_1}$	$8+1=9$
1111111110	
...	
1111111110000	
$\underbrace{111111111}_{z_9}\underbrace{1111}_{z_4}$	$9+4=13$
11111111111110	
...	
1111111111111000000000000	
$\underbrace{1111111111111}_{z_9}\underbrace{1111111111111}_{z_9}$	$13+13=26$
11111111111111111111111110	
...	
1111111111111111111111111000000000	
$\underbrace{111111111111111111111111}_{z_26}\underbrace{111111111}_{z_9}$	$26+9=35$
11111111111111111111111111111111110	
11111111111111111111111111111111100	
11111111111111111111111111111111101	

4.3.2. Problem izbacivanja suvišnih elemenata

Prilikom upotrebe metode prozora, u bilo kojoj varijanti, dobiva se ispravan adicijski lanac. Međutim, za dobiveni lanac postoji mogućnost da su određeni elementi suvišni, a to je naročito izraženo korištenjem veće veličine prozora. Primjer na broju 16, uz veličinu prozora 4, zorno je prikazan na slici 4.1.



Slika 4.1: Problem suvišnih elemenata za $n = 16, w = 4$

Kako se vidi na slici, svi neparni elementi, odnosno, svi početni uzorci osim nužnih brojeva 1 i 2 mogu biti izbačeni iz lanca. U ovom, trivijalnom slučaju, takvih elemenata je čak i više od broja elemenata koji zaista čine lanac za broj 16. Na ovako malenim brojevima ovaj problem je jednostavno rješiv, no prilikom rada na velikim brojevima, nije trivijalno pronaći i izbaciti sve nepotrebne brojeve. Prilikom izbacivanja postoji opasnost da se izbaci neki broj koji je potreban za dobivanje nekog drugog broja te da se na taj način lanac prekine i postane neispravan.

U okviru rada se u cilju sprječavanja ovog problema koristi heuristika koja pri izbacivanju razmatra samo one brojeve koji se nalaze među početnim uzorcima, a koji ne doprinose izgradnji lanca. Ovaj postupak se izvodi na sljedeći način:

1. **Prilikom izgradnje lanca, svaki upotrebljeni element dodaje se u skup upotrebljenih elemenata.** Ovime se pamte svi elementi koji su sudjelovali u izgradnji lanca.

2. **Računa se razlika svih elemenata lanca i upotrebljenih elemenata.** Ova razlika će biti jednaka skupu neupotrebljenih elemenata iz početnog skupa uzoraka.

3. **Svi neupotrebljeni elementi izbacuju se iz lanca.** Ovime se lanac skraćuje za veličinu skupa neupotrebljenih elemenata.

4. **Za svaki element lanca bitovne duljine manje ili jednake veličini prozora provjerava se može li se izraziti kao zbroj neka dva manja elementa iz lanca.**

Izbacivanjem suvišnih elemenata postoji mogućnost da je lanac postao neispravan. Međutim, izbacivanje može oštetiti jedino elemente manje ili jednake najvećem elementu iz početnog skupa uzoraka.

5. **Za elemente koji nemaju pribrojnike u lancu, računaju se optimalni adicijski lanci.** Podlanac elementa čiji je pribrojnik izbačen izračunat će se pomoću fiksne mape optimalnih adicijskih lanaca. Kako je ranije spomenuto, fiksna mapa sadrži optimalne adicijske lance za sve brojeve manje od 620. Zbog toga, izbacivanje elemenata radit će pouzdano do veličine prozora 9 bita, jer je najveći element početnih uzoraka broj 511.

6. **Dobiveni podlanci dodaju se u adicijski lanac.** Dodavanjem izračunatih podlanaca u adicijski lanac, adicijski lanac postaje ispravan te ne sadrži suvišne elemente.

4.4. Prednosti i nedostaci deterministčkih metoda

Deterministčke metode, kao osnovna i glavna porodica algoritama za kojima se prvo poseže prilikom rješavanja bilo kojeg problema, kao i stohastičke metode, imaju niz prednosti i nedostaka. U odnosu na stohastički pristup, prednosti su sljedeće:

1. **Brzina izvođenja.** Osnovna prednost dobro osmišljene deterministčke heuristike definitivno je izuzetno kratko vrijeme izvođenja, naročito u usporedbi s evolucijskim pristupima. Dok se pri evolucijskom računanju, ovisno o željenoj kvaliteti rezultata, izvođenje mjeri u satima, pa čak i danima ili mjesecima, sve obrađene deterministčke metode se izvode u stotinkama sekunde.

2. **Jasna matematička podloga.** Deterministčke metode su nužno zasnovane na osmišljavanju heuristike, koja pak mora biti potkrijepljena matematičkim zakonitostima koje su sadržane u samom problemu. U stohastičkim pristupima je ovo djelomično točno: budući da se dio rada zasniva na nasumičnosti, nije moguće u potpunosti primijeniti željenu, matematički osnovanu heuristiku.

3. **Interpretativnost rada algoritma.** Pri radu deterministčke metode, algoritam se uvijek ponaša logično i predvidivo te su međurezultati očekivanih karakteristika. Radom evolucijskog algoritma često je nemoguće utvrditi zašto se razvoj rješenja odvija u nekom određenom smjeru te zašto su neki geni bolji od drugih.

Ove metode svakako imaju i svoje nedostatke, od kojih su važniji sljedeći:

1. **Nemogućnost poboljšanja rješenja.** Glavna mana ovih metoda je nemogućnost ulaganja više resursa u svrhu dobivanja boljeg rješenja. Heuristika, nakon što je osmišljena i implementirana, za jednak skup parametara uvijek će davati isti rezultat. Zbog toga je poboljšavanje postojeće heuristike relativno težak proces.

2. **Moguće je da ne postoji način približavanja optimumu.** Ovisno o prirodi problema, postoji mogućnost da ne postoji heuristika kojom bi se dobilo iole kvalitetno rješenje.

3. **Nemogućnost generalizacije metode.** Za razliku od sveobuhvatnog evolucijskog algoritma, deterministička metoda se mora izgraditi specifično za zadani problem, te nije primjenjiva u niti jednom drugom problemu.

Kako je i ranije spomenuto, prednosti i nedostaci deterministčkih i stohastičkih pristupa su prilično komplementarni te je stoga logično pokušati spojiti ova dva svijeta. Sljedeće poglavlje iznosi jedan od način realizacije metode koja istovremeno koriste i determinizam i stohastiku kako bi se pronašla što kvalitetnija rješenja.

5. Kombiniranje determinističkih metoda i evolucijskog algoritma

Genetski algoritam potrebno je inicijalizati početnom populacijom, koja će tokom rada algoritma dalje evoluirati i davati dobra rješenja. Osnovni pristup prilikom inicijalizacije je generirati nasumičnu populaciju, a prilikom nasumičnog stvaranja jedinki jedino je bitno da su novostvoreni lanci ispravni. Stoga, ovo nasumično generiranje ipak u sebi sadrži dio determinizma - u suprotnom bi populaciju činile neispravne jedinke za koje je izuzetno mala vjerojatnost da će postati dobri lanci.

Nasumičnim generiranjem jedinki osigurava se raznolikost gena unutar populacije, što je odlično svojstvo za kvalitetnu evoluciju. U slučaju kad su sve jedinke nasumično generirane, može se primijetiti sljedeće:

1. **Velika raznolikost gena uzrokuje početno brz napredak.** Genetski algoritam uglavnom pokazuje svojstvo da, pri početku evolucije, populacija znatno napreduje iz generacije u generaciju. Raznolikost se dodatno potiče većom vjerojatnosti mutacije, koja osigurava unos novih gena i povećanje raznolikosti. Ipak, nakon određenog broja generacija, kada je cjelokupna populacija evoluirana ka boljim rješenjima, napredak se asimptotski usporava, sve dok cjelokupna populacija ne zaglavi u optimumu, bilo lokalnom ili lokalnom.

2. **Početna populacija nema izuzetno kvalitetna rješenja.** Budući da su jedinke nasumično generirane, za očekivati je da je prosjek početne populacije izuzetno loš. Naime, mala je vjerojatnost da će se isključivo nasumičnim putem generirati neka jako kvalitetna jedinka, koja je blizu optimuma. Zbog ovog svojstva, potreban je veći broj generacija kako bi se ova, loša populacija dovela do nekih adekvatnih rješenja.

3. **Postoji mogućnost da algoritam ima loš početak.** Moguće je da se većina početnih jedinki generira vrlo blizu lokalnog optimuma. U tom slučaju, evolucija je osuđena na propast, jer će cjelokupna populacija konvergirati ka tom lokalnom

optimumu te će se dobiti izuzetno loša rješenja. Zbog toga, genetski algoritam se uglavnom više puta pokreće iznova, kako bi se mogućnost lošeg početka svela na minimum.

Uz pretpostavku da se početna popoulacija ne generira potpuno nasumično, nego da se npr. inicijalizira dobrim jedinkama, vrijedi sljedeće:

1. **Početna populacija je jako kvalitetna.** U početnoj populaciji nalazi se mnoštvo kvalitetnih jedinki, koje imaju dobar potencijal za daljnji razvoj. Kombinacijom dobrih jedinki operatorom križanja očekuje se dobivanje još boljih jedinki budući da su u križanje uključeni većinom kvalitetni geni.

2. **Postoji veliki rizik za zaglavljenje u lokalnom optimumu.** Budući da su sve jedinke kvalitetne, za pretpostaviti je da sadrže slične gene. Uz takvu pretpostavku, raznolikost genetskog materijala je premalena, te je smanjena mogućnost za daljnji razvoj jedinki pa je i konvergencija prilično spora. Uz to, moguće je da su početne jedinke blizu lokalnog optimuma pa algoritam zaglavljuje u lokalnom optimumu te ima beznadno malenu vjerojatnost da se približi globalnom optimumu. Ovo svojstvo je moguće donekle suzbiti većom vjerojatnosti mutacije kako bi se uvelo više novog genetskog materijala te samim time i otvorila mogućnost ka preusmjeravanju evolucije prema globalnom optimumu.

Genetski algoritam bi, prema svemu sudeći, bilo idealno inicijalizirati nasumično, uz dodatak nekoliko dobrih jedinki. U tom slučaju, nasumičnost bi osigurala raznolikost materijala i širinu evolucije, dok bi dobre jedinke unosile kvalitetan genetski algoritam. Takav pristup je primijenjen u okviru rada, a sljedeće potpoglavlje opisuje na koji način je realiziran.

5.1. Genetski algoritam djelomično potpomognut determinističkom heuristikom

Genetskom algoritmu, uz zadavanje parametara koji mu definiraju rad, moguće je i dio populacije inicijalizirati direktno zadavanjem jedinki. Uz fiksnu veličinu populacije N i S predefiniranih jedinki, algoritam nasumično stvara $(N - S)$ jedinki kako bi se početna populacija popunila do maksimalne veličine. Daljnji način rada algoritma je isti sa i bez direktnog zadavanja jedinki.

Direktno zadane jedinke stvorene su upravo novorazvijenom proširenom metodom prozora. Upotrebljene su sve kombinacije parametara te tri veličine prozora. Ovime je dobiveno 24 jedinke, kako je prikazano na tablici 5.1.

Tablica 5.1: Jedinke dobivene kombiniranjem svih parametara proširene metode prozora

o	e	ac	w		
			2	3	4
0	0	0	u_1	u_2	u_3
0	0	1	u_4	u_5	u_6
0	1	0	u_7	u_8	u_9
0	1	1	u_{10}	u_{11}	u_{12}
1	0	0	u_{13}	u_{14}	u_{15}
1	0	1	u_{16}	u_{17}	u_{18}
1	1	0	u_{19}	u_{20}	u_{21}
1	1	1	u_{22}	u_{23}	u_{24}

Svaki parametar ima određenu ulogu prilikom inicijalizacije te su zato upotrebljene sve kombinacije.

1. **Parametar *o*** treba biti uključen kako bi se izbacili suvišni elementi, stoga je logičan odabir uključiti ga. Međutim, generiranjem jedinki koje imaju i isključen ovaj parametar osigurava se veća genetska raznolikost, budući da ovakve jedinke sadrže sve elemente koji pripadaju početnim uzorcima.

2. **Parametar *e*** uključuje prošireni način rada metode prozora. Njegovim uključivanjem dobivaju se bolje jedinke, a isključivanjem se stvaraju jedinke s većom količinom zbrajanja manjih brojeva, do veličine prozora. Samim time, isključenim parametrom se dobivaju znatno lošije jedinke, no one imaju više elemenata koji su jako dobro povezani. Stoga, isključivanje ovog parametra također donosi bogatiji genetski materijal, koji je ipak drukčiji nego pri isključivanju parametra o.

3. **Parametar *ac*** omogućuje bolje rezultate za specifične brojeve s puno uzastopnih bitovnih jedinica. Stoga, njegovom uključenošću se pokrivaju takvi slučajevi, dok se isključenošću pokrivaju običniji brojevi bez tog svojstva.

4. **Veličina prozora *w*** U pravilu veća veličina prozora je u stanju bolje pronaći ponavljajuće uzorke te samim time doći i do kraćeg lanca. Ipak, ukoliko je

isključen parametar o, povećanjem veličine prozora povećava se i duljina lanca, jer ostaje mnoštvo elemenata koji su nepotrebni u izgradnji lanca.

Ovakvim kombiniranjem svih parametara postiže se i dobra genetska raznolikost i prisutnost jako kvalitetnih jedinki. Međuutjecaji parametara također dolaze do izražaja, stoga se očekuje da je taj skup od 24 jedinke vrlo bogat raznolikim genetskim materijalom.

⁂

Kako bi se smanjili nedostaci i determinističkih i stohastičkih metoda pokazan je način kako je moguće istovremeno upotrijebiti obje metode te samim time iskoristiti prednosti oba pristupa. Sljedeće poglavlje prikazuje rezultate dobivene upotrebom genetskog algoritma, determinističkom proširenom metodom prozora te daje li kombinacija dvaju pristupa bolja rješenja u odnosu na svaki pristup pojedinačno.

6. Rezultati

U cilju kvalitetne analize svih spomenutih algoritama te mogućnosti donošenja osnovanih zaključaka, potrebno je prvo izgraditi plan izvođenja, zajedno sa svim radnjama koje prethode samom izvođenju algoritama. Predradnje se svode na sljedeće:

1. **Odabir reprezentativnog skupa brojeva na kojem će se izvoditi mjerenja.** Skup brojeva treba biti takav da osigurava nepristranu analizu algoritama te da pokrije što je više moguće specifičnih slučajeva. Brojevi koji su korišteni tokom analize mogu se podijeliti u dvije skupine:

   (a) **Veliki brojevi.** Ovi brojevi su reda veličine $2^{37} \approx 10^{25}$ pa sve do $2^{107} \approx 10^{74}$. Težina problema pri radu s ovim brojevima je u velikom broju elemenata, a samim time i ogromnom prostoru pretraživanja. Genetski operatori na ovakvim brojevima imaju slabiji utjecaj nego na manjim te je konvergencija znatno usporena.

   (b) **Specifični brojevi.** Posebna svojstva optimalnih lanaca ovakvih brojeva otežavaju pronalazak dobrih rješenja. Recimo, za postizanje optimalnog lanca za specifičan broj potrebno je da neki elementi ne uključuju u zbroj svojeg direktnog prethodnika i nekog drugog prethodnika, nego da niti jedan pribrojnik nije direktan prethodnik. Uz pretpostavku da se svaki broj dobiva korištenjem jednog direktnog prethodnika, izuzima se ogroman prostor pretraživanja te samim time i potencijalna mogućnost dolaska do optimuma.

2. **Optimizacija hiperparametara genetskog algoritma.** Na određenom, reprezentativnom broju potrebno je pokrenuti genetski algoritam s raznim kombinacijama parametara, konkretno, veličinama populacije i vjerojatnostima mutacije. Iz rezultata pokretanja može se zaključiti koja je kombinacija parametara najbolja te najviše obećavajuća za ispitivanje na ostalim brojevima. Takva kombinacija parametara se fiskira te se kasnije više ne mijenja tokom izvođenja nad svim brojevima.

Nakon ovakve pripreme, određeni su parametri genetskog algoritma te brojevi na kojima će se vršiti ispitivanje. Sljedeći korak je ispitivanje algoritama:

1. **Pokretanje genetskog algoritma.** Genetski algoritam se pokreće na svakom odabranom broju 30 puta kako bi rezultati bili statistički korektni. Većim brojevima se postavlja kao kriterij zaustavljanja manji broj evaluacija, budući da je rad algoritma znatno sporiji.

2. **Pokretanje deterministčkih metoda.** Budući da se deterministčke metode izvršavaju vrlo brzo, nije potrebno ništa učiniti radi postizanja razumnog vremena izvođenja, kao što je to slučaj sa smanjivanjem broja evaluacija kod genetskog algoritma. Uz to, dovoljno je svaku metodu pokrenuti jednom na zadanom broju, jer će rezultat uvijek biti isti.

3. **Pokretanje genetskog algoritma potpomognutog deterministčkom heuristikom.** Genetski algoritam inicijaliziran deterministčkom metodom pokreće se s jednakim parametrima kao kod pokretanja osnovnog genetskog algoritma. Ovime su rezultati usporedivi te je moguće promatrati utjecaj deterministčke inicijalizacije populacije.

6.1. Genetski algoritam s binarnim prikazom

Prema planu izvođenja, genetski algoritam je prvo pokrenut na brojevima $2^{47} - 3$ i $2^{67} - 3$ s kombinacijama parametara $N \in \{100, 500, 1000\}$ te $p_m \in \{0.3, 0.5, 0.7, 0.9\}$, po 30 puta. Rezultati izvođenja prikazani su na slikama 6.1 i 6.2.

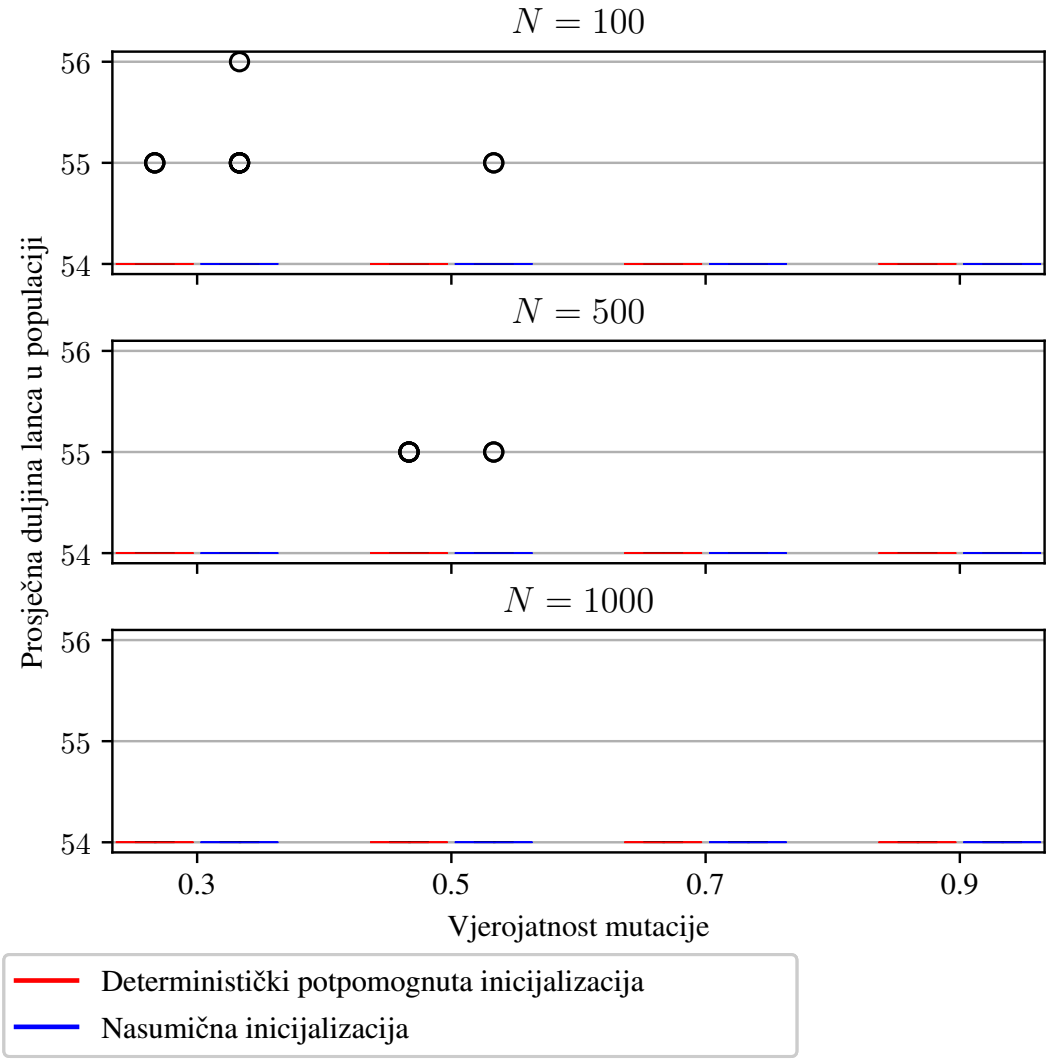
Iz grafa se može zaključiti:

1. **Povećanjem populacije smanjuje se disperzija.** Što je više jedinki u populaciji, algoritam stabilnije radi budući da su globalni utjecaji pojedinačnih mutacija slabiji.

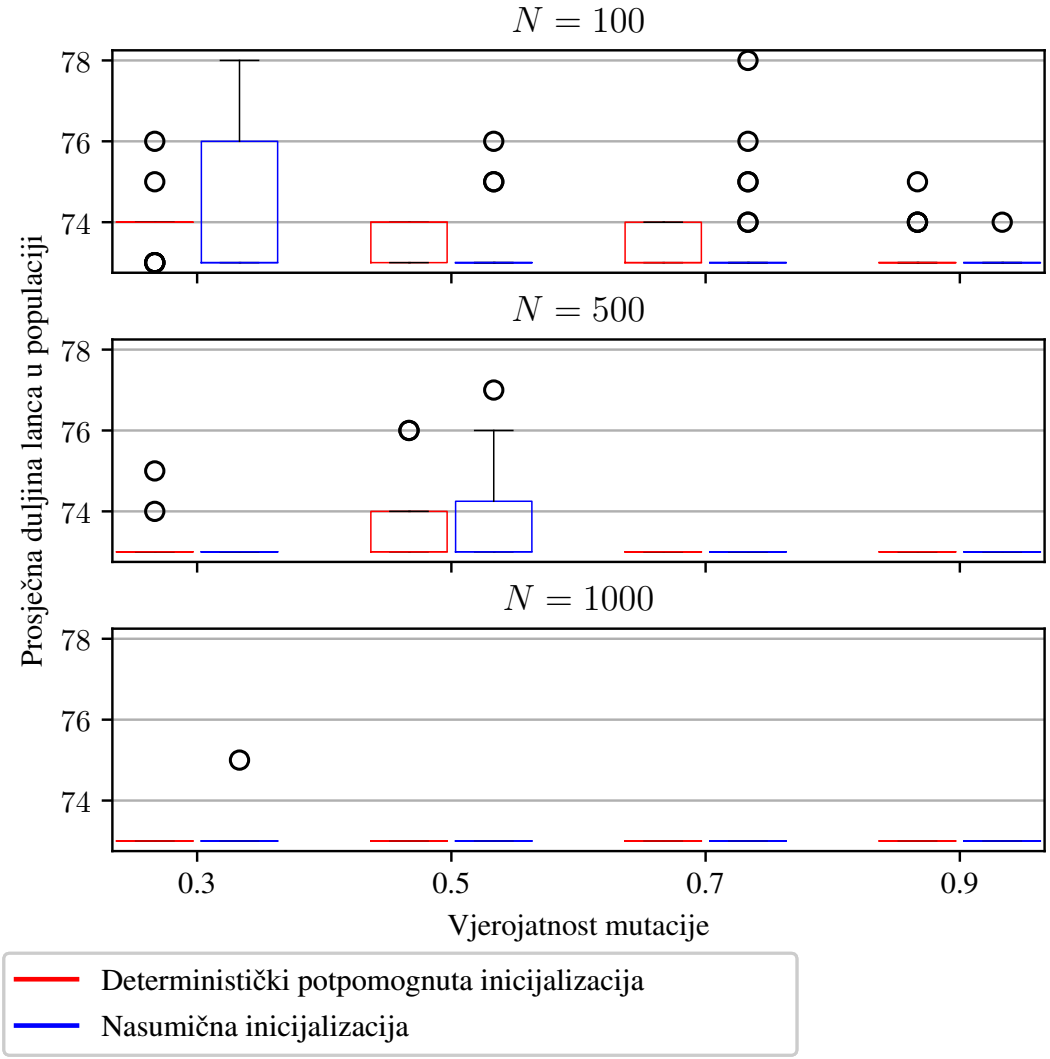
2. **U većini slučajeva, deterministički potpomognuta inicijalizacija daje bolje rezultate.** U samo dva od 48 slučajeva je lošija - u ostalima je jednaka ili bolja.

3. **Na većem broju veća je i disperzija.** Razlog ovome je veći prostor stanja koji je potrebno pretražiti uz jednak broj evaluacija. Algoritam nije u stanju dovoljno se približiti optimumu kako bi ušao u stabilno stanje.

Kao referentne vrijednosti parametara odabrani su $N = 500$ i $p_m = 0.3$ zbog dobre ravnoteže između kvalitete rješenja, brzine izvođenja te niske disperzije. Ova kombinacija pokazuje stabilnost kako pri radu osnovnog genetskog algoritma, tako i pri radu potpomognutog.



Slika 6.1: Prikaz najboljih rješenja kroz 30 pokretanja za $2^{47} - 3$ u odnosu na N i p_n



Slika 6.2: Prikaz najboljih rješenja kroz 30 pokretanja za $2^{67} - 3$ u odnosu na N i p_n

6.2. Determinističke metode

Budući da determinističke metode ne zahtijevaju nikakvu posebnu pripremu te se brzo izvode, za svaki broj se izvode svih 24 kombinacija parametara navedenih u tablici 5.1. Ovime je omogućen optimalan rad proširene metode prozora nad svakim brojem posebno - za neki broj će neka kombinacija parametara biti dobra.

6.3. Pregled rezultata

Brojevi nad kojima su svi algoritmi izvođeni prikazani su u tablici 6.1.

Tablica 6.1: Brojevi nad kojima su izvođeni svi algoritmi

Broj	Puni prikaz
$2^{37} - 3$	137438953469
$2^{47} - 3$	140737488355325
$2^{57} - 3$	144115188075855869
$2^{67} - 3$	147573952589676412925
$2^{77} - 3$	151115727451828646838269
$2^{87} - 3$	154742504910672534362390525
$2^{97} - 3$	158456325028528675187087900669
$2^{107} - 3$	162259276829213363391578010288125
$2^{117} - 3$	166153499473114484112975882535043069
$2^{127} - 3$	170141183460469231731687303715884105725
Specifični brojevi	191
	382
	12509
	375494703
	750989406

Tablica 6.2 prikazuje najbolji dobiveni rezultat za svaki algoritam. Masno otisnuti rezultati predstavljaju najbolje pronađeno rješenje za određeni broj.

U tablici 6.3 može se vidjeti rang svakog algoritma po brojevima nad kojima su se izvodili. Uz to, prikazan je prosječni rang svakog algoritma prema skupinama brojeva te konačan prosjek rangova. Ovime je pridjeljena jednaka težina rangova po skupinama brojeva.

Iz podataka se vidi kako je genetski algoritam, sa i bez determinističke inicijalizacije definitivan pobjednik te pronalazi najbolja rješenja na zadanim brojevima. Međutim, deterministička metoda, u svojim određenim varijantama, dijeli prvo mjesto s genetskim algoritmom u većini slučajeva. Dijeljenje prvog mjesta je naročito vidljivo prilikom rada na prvoj skupini brojeva, budući da je s uključenim dodatnim parametrima, deterministička metoda u stanju pronaći i iskoristiti duge ponavljajuće uzorke. U slučaju ovih brojeva, kao što je ranije spomenuto, to su dugi nizovi jedinica.

S drugom skupinom brojeva, deterministička metoda se nosi prilično dobro, ali

ipak manje uvjerljivo nego što je to slučaj s prvom skupinom. Razlog tome su posebna svojstva ovih brojeva, zbog kojih optimalan lanac nema jednostavnu strukturu te nije jednostavno na brz način pronaći optimalan lanac. Karakteristika ovih lanaca je detaljnije opisana na početku ovog poglavlja.

Tablica 6.2: Prikaz dobivenih rezultata algoritama po brojevima

Broj	Evolucijski algoritam		Determinističke metode						
	Broj evaluacija	Binarni prikaz	Binarna metoda	Metoda prozora	Proširena metoda prozora	$e = 1$	$ac = 1$	$e = ac = 1$	GA + det. inicijalizacija
$2^{37} - 3$	500000	**43**	71	51	44	44	44	44	**43**
$2^{47} - 3$	500000	**54**	91	63	55	**54**	**54**	**54**	**54**
$2^{57} - 3$	500000	**64**	111	76	67	**64**	65	65	**64**
$2^{67} - 3$	500000	**73**	131	88	78	74	74	74	**73**
$2^{77} - 3$	200000	**85**	151	101	90	**85**	**85**	**85**	**85**
$2^{87} - 3$	200000	**95**	171	113	103	**95**	**95**	**95**	**95**
$2^{97} - 3$	200000	**106**	191	126	117	**106**	**106**	**106**	**106**
$2^{107} - 3$	100000	**115**	211	138	123	116	116	116	**115**
191	500000	**11**	13	14	**11**	**11**	15	15	**11**
382	500000	**11**	14	16	12	12	16	16	**11**
12509	500000	18	20	21	**17**	**17**	18	18	**17**
375494703	500000	**35**	41	40	**35**	**35**	38	38	**35**
750989406	500000	**36**	42	40	**36**	**36**	39	39	**36**

Iz rezultata se vidi i prevladavajući utjecaj parametra *ac* na determinističkoj metodi. Naime, ukoliko je ovaj parametar uključen, svejedno je je li uz njega uključen i parametar *e* ili nije. Prilikom izvođenja algoritma uz parametar *ac*, algoritam prvenstveno traži duge uzorke jedinica, što je kod ovih brojeva većina samog broja, te zbog toga parametar e ne dolazi do izražaja.

Tablica 6.3: Prikaz rangova algoritama po brojevima

Broj		Determinističke metode						
	Evolucijski algoritam	Binarna metoda	Metoda prozora	Proširena metoda prozora	$e = 1$	$ac = 1$	$e = ac = 1$	GA + det. inicijalizacija
$2^{37} - 3$	1	4	3	2	2	2	2	1
$2^{47} - 3$	1	4	3	2	1	1	1	1
$2^{57} - 3$	1	5	4	3	1	2	2	1
$2^{67} - 3$	1	5	4	3	2	2	2	1
$2^{77} - 3$	1	4	3	2	1	1	1	1
$2^{87} - 3$	1	4	3	2	1	1	1	1
$2^{97} - 3$	1	4	3	2	1	1	1	1
$2^{107} - 3$	1	5	4	3	2	2	2	1
Prosječni rang	1	4.375	3.375	2.375	1.5	1.5	1.5	1
191	1	2	3	1	1	4	4	1
382	1	3	4	2	2	4	4	1
12509	2	3	4	1	1	2	2	1
375494703	1	4	3	1	1	2	2	1
750989406	1	4	3	1	1	2	2	1
Prosječni rang	1.25	3.5	3.5	1.25	1.25	2.5	2.5	1
Ukupni rang	1.125	3.9375	3.4375	1.8125	1.375	2	2	1

6.4. Proširena metoda u odnosu na postojeći genetski algoritam

Uz usporedbu proširene metode prozora s genetskim algoritmom podešenim i pokretanim u sklopu izrade rada, slijedi usporedba rezultata proširene metode s postojećim rezultatima (Picek et al., 2017). Objavljeni rezultati su dobiveni genetskim algoritmom jednake implementacije ispitane u okviru rada. Tablica 6.4 prikazuje rezultate dobivene postojećom metodom prozora, objavljenim genetskim algoritmom te novoostvarenom proširenom metodom prozora.

Brojevi su opet podijeljeni u dvije skupine: velike brojeve, koji su veći od prethodno korištenih velikih brojeva, te teške brojeve. Autori rada naglašavaju da je ovaj skup brojeva težak, jer svaki broj sadrži vrlo malo kombinacija koje osiguravaju optimalnost lanca, te je samim time potrebna dulja pretraga za pronalazak optimuma.

Iz tablice se vidi kako je proširena metoda prozora bolja za sve velike brojeve u odnosu na genetski algoritam. Najbolji rezultati su postignuti uključenim parametrom ac, koji dolazi još više do izražaja budući da ovi veliki brojevi sadrže još više jedinica od ranije upotrebljavanih velikih brojeva.

Tablica 6.4: Usporedba postojećih rezultata dobivenih genetskim algoritmom s rezultatima dobivenih novorazvijenom proširenom metodom prozora

Broj	Metoda prozora	Genetski algoritam	Proširena metoda prozora
$2^{165} - 21$	211	176	**174**
$2^{175} - 21$	223	187	**185**
$2^{185} - 21$	236	198	**195**
$2^{195} - 21$	248	210	**206**
$2^{205} - 21$	261	217	**215**
$2^{215} - 21$	273	228	**226**
$2^{225} - 21$	286	239	**236**
$2^{235} - 21$	298	250	**246**
$2^{245} - 21$	311	258	**255**
$2^{255} - 21$	323	269	**266**
17180843711	45	**42**	43
17181535967	46	**42**	44
17181824999	46	**42**	43
17181857663	46	**41**	42
17181878143	46	42	42
17181921023	46	42	42
17181425531	46	43	43
17181433703	46	**42**	43
17181750911	46	**42**	43
17181793151	46	**42**	44
17181963167	46	42	42
17182209983	46	**42**	44
17182210751	46	42	42
17182215157	46	**42**	44
17182219767	46	**43**	44
17182226303	46	**42**	44
17182318319	46	**42**	43

Kad su u pitanju teški brojevi, u svim slučajevima je genetski algoritam bolji od proširene metode prozora, a u najgorem slučaju su izjednačeni. Očito je kako je determinizmom vrlo teško obuhvatiti sve specifične slučajeve vezane za teške brojeve. Ipak, rezultati dobiveni deterministčkom metodom su vrlo blizu rezultatima dobivenih genetskim algoritmom.

7. Zaključak

Područje adicijskih lanaca je izuzetno zanimljivo - ovakav, naoko vrlo jednostavan matematički aparat, krije razne probleme čijim rješavanjem se dolazi do bitnih znanstvenih otkrića. Zanimljivo je promatrati i kako razni optimalni lanci za određene brojeve imaju različita svojstva te je vrlo teško osmisliti metodu koja će biti univerzalno vrlo uspješna. Na primjeru novorazvijene proširene metode prozora, vidi se kako se algoritam na dvije raličite skupine brojeva ponaša doista različito. Definitivno je prvo mjesto za poboljšanje prilagodba algoritma za rad s brojevima čiji optimalni lanci imaju elemente sa skokovitim prijelazima te samim time, ovakvim, trenutnim determinizmom ih nije moguće pronaći.

S druge strane, jasno se vidi širina koju posjeduju genetski algoritmi - oni su u stanju jednako kvalitetno raditi bez obzira na svojstva lanaca, budući da su ta svojstva prikrivena prikazom pomoću rastućeg niza brojeva, bez razmatranja i očuvanja međusobnih veza. Ipak, za većinu ispitanih brojeva, novorazvijena metoda se pokazala jednako dobrom kao i genetski algoritam, a vrijeme izvođenja je neusporedivo manje.

Kombinacija determinizma i stohastike pokazala se donekle uspješnom - najbolja pronađena rješenja nisu bila bolja u odnosu na osnovni genetski algoritam. Međutim, genetski algoritam je radio nešto stabilnije, odnosno, s manjom varijancom pronađenih rješenja. Stoga, ukoliko se za zadani problem koristi genetski algoritam, definitivno se isplati inicijalizirati dio populacije kvalitetnim rješenjima kako bi se algoritmu ipak dao početni pozitivni pomak.

Između determinističke i stohastičke heuristike valja odabrati ovisno o široj slici prilikom rješavanja problema. Ukoliko je potrebno pronaći optimalan adicijski lanac za neki broj, kako bi se taj lanac mogao u budućnosti koristiti bez ponovnog izračuna, tada se zasigurno isplati uložiti vrmeneski resurs te genetskim algoritmom doći vrlo blizu optimalnog rješenja. S druge strane, ako je priroda problema takva da je potrebno u kratkom vremenu dobiti adicijski lanac za broj koji je poznat isključivo u tom trenutku, te je promjenjiv u bliskoj budućnosti, novoostvarena deterministička metoda će zasigurno dati odličan rezultat.

LITERATURA

Alfred Theodor Brauer. On addition chains. *Bulletin of the American Mathematical Society*, 1939. URL `http://wwwhomes.uni-bielefeld.de/achim/Brauer_on_addition_chains_1939.pdf`.

Domagoj Jakobović, Stjepan Picek, et al. *Evolutionary Computation Framework*. Faculty of Electrical Engineering and Computing, 2017. http://ecf.zemris.fer.hr.

Stjepan Picek, Carlos A Coello Coello, Domagoj Jakobovic, i Nele Mentens. Finding short and implementation-friendly addition chains with evolutionary algorithms. *Journal of Heuristics*, 2017.

Postupci konstrukcije i optimizacije adicijskih lanaca

Sažetak

Uvod rada sadrži opći pregled sadržaja rada te povezuje odnose između pojedinih poglavlja. Sljedeće poglavlje razmatra problem potenciranja velikih brojeva te ga povezuje s pojmom adicijskog lanca. Uz to, opisuje se problematika traženja kratkih adicijskih lanaca te se predlažu dva nasuprotna načela rješavanja problema: deterministički i stohastički. Treće poglavlje opisuje opće načelo rada genetskog algoritma te specifičnosti vezane za adicijske lance, dok se četvrto poglavlje bavi determinitičkim metodama, postojećim te novorazvijenim u okviru rada. Peto poglavlje povezuje ova dva pristupa, a u šestom poglavlju su opisane priprema mjerenja, dobiveni rezultati te analiza rješenja. Naposljetku, u Zaključku su iznesena sva važna zapažanja izvedena tokom izrade programskog rješenja te analize rezultata.

Ključne riječi: adicijski lanac, optimizacija, eksponenciranje, genetski algoritam, deterministička metoda, binarna metoda, metoda prozora

Methods for construction and optimization of addition chains

Abstract

The introduction is a general overview of the thesis and interconnects individual chapters. The next chapter discusses the problem of exponentiation of large numbers and links it to the concept of the addition chain. Additionally, the problem of searching of short additional chains is described and two contrasting problem-solving principles are proposed: deterministic and stochastic. Chapter 3 describes the general principle of the genetic algorithm and the specifics associated with the addition chains, while the fourth chapter deals with deterministic methods, existing and newly developed within the thesis. The fifth chapter links these two approaches and in the sixth chapter describes the preparation of measurements, the obtained results as well as the analysis of results. Finally, all the important observations made during the design of the software solution and the results analysis are presented in the Conclusion.

Keywords: addition chain, optimization, exponentiation, genetic algorithm, deterministic method, binary method, window method