KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
SISTEMINĖS ANALIZĖS KATEDRA

Mindaugas Smolinskas

Dvejetainės informacijos kodavimo taikant bazinius skaidinius analizė: teoriniai ir praktiniai aspektai

Magistro darbas

Darbo vadovas
doc. dr. J. Valantinas

Kaunas, 2005

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
SISTEMINĖS ANALIZĖS KATEDRA

TVIRTINU
Katedros vedėjas
prof. habil. dr. R. Barauskas
2005 01 11

Dvejetainės informacijos kodavimo taikant bazinius skaidinius analizė: teoriniai ir praktiniai aspektai

Magistro darbas

Vadovas

doc. dr. J. Valantinas
2005 01 11

Recenzentas

doc. dr. K. Plukas
2005 01 11

Atliko
IFM-8/1 gr. stud.
M. Smolinskas
2005 01 11

Kaunas, 2005

ANOTACIJA

Darbe pateikta dvejetainės informacijos fragmentų glaudinimo, taikant dekompoziciją suma moduliu 2, metodika, leidžianti sumažinti skaičiavimų apimtis. Įvesta skaidomumo poaibį identifikuojančios funkcijos sąvoka, ištirtas ir realizuotas euristinis tokių sekų paieškos algoritmas. Pateikti spartūs algoritmai ir jų realizacijos: duomenų sekų srautui glaudžiai koduoti naudojant iš anksto sudarytą skaidinių lentelę, ir jiems dekoduoti realiuoju laiku. Eksperimentiškai parodyta, kad glaudinimas naudojant dekompoziciją suma moduliu 2 gali būti realizuotas ir atskirais atvejais duoda glaudinimo efektą įprastos konfigūracijos sistemoje.

SUMMARY

A new approach (designed to lower computational complexity) to compression of finite binary data, based on the application of “exclusive-or” operation, is presented in the paper. The new concept of an identifying sequence, associated with a particular decomposition scheme, is introduced. A new heuristic algorithm for calculation of the identifying sequences has been developed. Two robust algorithms – for compressing and streaming the sequences (using a priori compiled decomposition tables), and for real time decoding of the compressed streams – were proposed and implemented. The experimental results confirmed that the developed approach gave a data compression effect on an ordinary computer system.

TURINYS

1. ĮVADAS

Duomenų apimtys ir kompaktiškas jų saugojimas nuo skaitmeninių mašinų sukūrimo iki dabar išlieka aktualia problema. Kinta duomenų pobūdis, techninės galimybės. Vis daugiau bendrosios paskirties duomenų perkeliama į skaitmenines laikmenas. Natūralu, kad kintant duomenų pobūdžiui ir techninėms saugyklų galimybėms, turi vystytis ir glaudinimo priemonės, kurių tikslas – sumažinti perteklinių duomenų kiekį prieš juos įrašant į pastoviąsias saugyklas ar perduodant ryšio priemonėmis. Neretai, ypač atsiradus naujo pobūdžio duomenų (kaip erdviniai rastriniai vaizdai ar kintamos struktūros duomenys), jiems saugoti parenkami įprasti formatai, neatitinkantys duomenų specifikos, ir turintys daug perteklinės informacijos arba mažos vertės informacijai išnaudojantys didžiąją dalį laikmenos. Dažnai universalūs duomenų glaudinimo metodai šių problemų neišsprendžia pakankamai efektyviai.

Nors yra sukurta ir plačiai naudojama daug tarpusavyje nesuderinamų glaudinimo algoritmų, tačiau jų taikymo sritys iš esmės yra bendros, ir, atsižvelgiant į duomenų taikymo sričių gausybę, gana siauros. Antrajame skyriuje apžvelgiami esami populiariausi duomenų glaudinimo algoritmai. Nepraradiminiai diskretieji algoritmai – grupavimas (PackBits, Run-Length Encoding), Huffmano kodavimas (Huffman, 1952; CCITT Group 3 and 4, 1985), Lempel-Ziv-Welch (Lempel, Ziv, 1977; Welch, 1984; Compuserve, 1990), aritmetinis entropinis (Rissanen, 1979; Langdon, 1984; Witten, 1987; Abrahamson, 1989), – iš esmės tinkami optimizuoti tekstinius ar išvestinius tekstinius duomenis (kaip spaudos formatai, kompiliuotas programinis kodas, skaičiavimų ataskaitos). Praradiminiai algoritmai (MPEG Audio Layer 3, MPEG-4, fraktaliniai, bangelių) siaurai orientuoti į konkrečios srities duomenis, kaip tolygiai kintantis vaizdas ar garsas, kurie (dažniausiai dėl diskrečiųjų įrenginių specifikos) neša daug žmogui nejuntamos informacijos. Kaip sėkmingo skirtingų glaudinimo technologijų derinimo pavyzdžiai pristatomi JPEG (JPEG, 1991), JBIG (JPEG, 2002), StuffIt JPEG (Allume Systems, Inc., 2004), DjVu (Boutton et al., 2000) standartai. Tačiau jie, nors ir kiek efektyvesni duomenų glaudumo požiūriu, iš esmės aprėpia tik dar siauresnes taikymo sritis.

Tarp pastebimų glaudinimo algoritmų vystymosi tendencijų, labiausiai akcentuojamas tikslinių didėjantis duomenų struktūrų sudėtingumas, mažėjantis metodų deterministiškumas, ir augantys poreikiai skaičiavimų laikui ir tarpinės atminties kiekiui. Ypatingas dėmesys skiriamas metodų asimetriškumo tendencijoms.

Kaip alternatyvą ir galimą tolesnę universaliųjų (pagal poreikį specializuojamų konkrečiai sričiai) glaudinimo algoritmų evoliucijos kryptį pristatome dvejetainės informacijos (dvejetainių sekų) spektrinį skaidymą, naudojant sumos moduliu 2 operaciją. Iš kompiuterių ir buities prietaisų pramonės, matome, kad giminingi metodai plačiai taikomi praradiminiams duomenų (ypač daugiaterpės informacijos) glaudinimo algoritmams.

Remdamiesi klasikiniu matematiniu aparatu, pateikiame turimus duomenis apie glaudinimui palankių duomenų sekų kiekį ir spektrinio skaidymo deterministinio algoritmo sudėtingumą. Nors metodai, paremti vien šiomis priemonėmis, negali varžytis efektyvumu su esamais entropinio glaudinimo algoritmais, bet jie nėra nepritaikomi, siekiant gauti bent minimalų išlošį glaudinant dažniausiai praktikoje pasitaikančių struktūrų duomenų sekas.

Siūlome keletą naujų priemonių (plačiau aprašytų trečiajame skyriuje), leidžiančių padidinti spektrinio nepraradiminio duomenų glaudinimo metodo efektyvumą. Tarp jų – duomenų (dvejetainių sekų) segmentavimo galimybės ir kriterijai sekos segmentams parinkti. Naudojantis apibrėžtaisiais skaidomumo požymiais, siūlome grupuoti į poaibius fiksuoto ilgio fragmentus pagal jų optimaliuosius skaidinius. Pagal toliau pateiktą poaibį identifikuojančios sekos (dvejetainės funkcijos) apibrėžimą, ir jais besiremiančius reikalavimus, sukurtas deterministinis godusis algoritmas vienareikšmiai rasti bet kurios sekos identifikuojančiąją funkciją.

Kiekybiškai apytiksliai įvertintos fiksuoto ilgio sekų identifikuojančiųjų funkcijų lentelių apimtys, kuomet tiriamųjų fragmentų ilgiai neviršija 64 bitų. Iš jų matome, kad deterministinis identifikuojančiosios funkcijos algoritmas praktiškai nepritaikomas dideliam fragmentų skaičiui apdoroti. Tuo tikslu sukurta keletas (skirtingo efektyvumo) euristinių algoritmų, leidžiančių preliminariai sumažinti tiriamų pseudoidentifikuojačiųjų (vienareikšmiai identifikuojančiųjų poaibį algoritmo atžvilgiu) funkcijų aibę. Šiuo metodu galima žymiai sumažinti skaičiavimų apimtis, panaudojant baigtinio dydžio paskirstytąją fragmentų pseudoidentifikuojančiųjų funkcijų glaudinių duomenų bazę.

Pateikiamas algoritmas yra gana abstraktus, neorientuotas į konkretų efektyvų praktinį taikymą. Tačiau metodo programinę realizaciją praplėtus, ir pagal poreikius modifikavus atskiras euristikas, galima tikėtis tolesnių efektyvių darbų scenarijų, remiantis pateikta metodika glaudinti dvejetainių srautus, ir galimais glaudžiųjų duomenų srautiniais formatais. Glaudžiųjų duomenų srauto formatu galima efektyviai koduoti ir dekoduoti glaudžiąsias duomenų sekas be didelių laikmenos nuostolių, jei sekos nepavyksta suglaudinti.

Ypatingas dėmesys skiriamas realizacijos optimizavimui: visi algoritmai aprašomi orientuojantis į vektorinius ar paskirstytuosius skaičiavimus. Kiek įmanoma labiau remiamasi aparatinės aritmetikos galimybėmis.

Aprašytieji spartūs euristiniai algoritmai leidžia praktikoje išbandyti ir detaliau ištirti faktorizacijos etapus, kurių ankstesnės realizacijos negalėjo būti pritaikytos dėl skaičiavimų apimties. Šių algoritmų pagalba įgyvendinamos kai kurios būtinos sąlygos toliau vystyti glaudinimo, naudojant sumos moduliu 2 operaciją, metodiką.

2. DUOMENŲ GLAUDINIMO PROBLEMA IR JOS SPRENDIMAS

2.1. Glaudinimo algoritmų efektyvumo kriterijai

Duomenų entropija. Tarkime, *A* yra atsitiktinis įvykis, pasirodantis su tikimybe *P*(*A*). Tuomet *informacija*, kurią suteikia įvykis *A*, apibrėžiama kaip

$$I(A) = \log_x \frac{1}{P(A)} = -\log_x P(A); \quad x>1 \qquad (2.1.1)$$

Kuomet tikimybė *P*(*A*) didelė (artimai vienetui), įvykio teikiama informacija *I*(*A*) maža. Jei *P*(*A*)=1, tai *i*(*A*)=0. Mažai tikėtini įvykiai teikia didelį kiekį informacijos, o labiau tikėtini – mažą.

Informacijos teorijoje procesas, sukeliantis atsitiktinius įvykius A_j, vadinamas *šaltiniu*. Tarkime, kad visų šaltinio sukeliamų nepriklausomųjų įvykių A_j tikimybės yra $P(A_j)$. Šaltinio *entropija* tuomet vadinama tokia vidutinė informacija:

$$H = \sum_j P(A_j) I(A_j) \qquad (2.1.2)$$

Paprasčiausias būdas užkoduoti įvykių teikiamą informaciją – suteikti jiems vienodo ilgio identifikatorius (pavyzdžiui, natūraliuosius skaičius iš intervalo [1,*n*], kur *n* – skirtingų įvykių skaičius). Šiuo atveju kiekvienam įvykiui užkoduoti reikia $\log_2 n$ bitų, o *m* ilgio įvykių sekai – $m \cdot \log_2 n$ bitų.

Tačiau priklausomai nuo įvykių eiliškumo, jų tarpusavio priklausomybės ir kitų veiksnių, dauguma atvejų įmanoma rasti tokį kodą, kuris įvykių seką (statistiniu atveju) išreikštų mažesniu bitų skaičiumi. Literatūroje [7] minimas pavyzdys (Klaid Sayood, 1996), kai analizuojama seka:

1 2 1 2 3 3 3 3 1 2 3 3 3 3 1 2 3 3 1 2

Jei sekos struktūra (įvykių eiliškumas, išsidėstymas ir pan.) nėra svarbi, tuomet turime kodą, sudarytą iš simbolių „1“, „2“ ir „3“, kurių tikimybės yra:

P(„1“)=5/20=0,25
P(„2“)=5/20=0,25
P(„3“)=10/20=0,5

Stebimoji šaltinio entropija šiuo atveju lygi

$$-(0{,}25 \log_2(1/4) + 0{,}25 \log_2(1/4) + 0{,}5 \log_2(1/2)) = 1.5 \text{ (bitų simboliui).}$$

Tačiau, jei atsižvelgsime į struktūrą ir imsime įvykius poromis, aptiksime, kad seka sudaryta iš posekių „12“ ir „33“. Tuomet:

$P(\text{„12“}) = 5/10 = 0{,}5$
$P(\text{„33“}) = 5/10 = 0{,}5$

Stebimoji šaltinio entropija šiuo atveju lygi

$$-(0{,}5 \log_2(1/2) + 0{,}5 \log_2(1/2)) = 1 \text{ (bitas simboliui).}$$

Taigi pirmuoju atveju sekai užkoduoti naudojame $1{,}5 \cdot 20 = 30$ bitų, o antruoju – tik 10. Kitaip tariant, šio šaltinio teikiamą informaciją, užkoduotą pirmuoju būdu galime suglaudinti $30/10 = 3$ kartus, arba sutaupyti $(1 - 0{,}3) \cdot 100\% = 70\%$ duomenų laikmenos ar perdavimo kanalo pralaidumo.

Šis glaudinimo kriterijus ypač svarbus, apdorojant archyvuojamų ar didelių apimčių siauros paskirties duomenų masyvus.

Duomenų srauto entropijos analize taip pat paremti Huffmano ir aritmetinis kodavimas.

Glaudinimo universalumas įvairaus pobūdžio duomenims. Kai kurie glaudinimo metodai yra specializuoti tik tam tikro pobūdžio duomenims apdoroti. Pavyzdžiui, PackBits algoritmas gerai tinka paveikslėliams, kuriuose kiekvienas taškas išreiškiamas vienu bitu, ir kartojasi reguliarūs šablonai. Tačiau su kitos (nepriimtinos) struktūros duomenimis šių metodų generuojamų kodų informacijos entropija neretai yra didesnė už paties šaltinio entropiją. Pavyzdžiui, PackBits formatu užkoduoti paveikslėliai gali būti suglaudinti iki 128 kartų, tačiau netinkami duomenys dažnai užkoduojami dvigubai didesniu bitų kiekiu nei pradiniai, kadangi kiekvienam pradinio kodo fragmentui aprašyti pridedamas algoritmo tarnybinis metakodas. Kita vertus, entropiniai kodavimo algoritmai, su didelės apimties duomenų masyvais, efektyvumu nežymiai nusileidžia PackBits kodavimui, tačiau jų perteklinių dekodavimo įrašų apimties santykis su pradinių duomenų apimtimis artėja prie 0.

Dažnai įmanoma euristiškai atskirti duomenų masyvus, ir jiems pritaikyti kodavimo metodus. Tačiau ir „palankių“ duomenų glaudinimo efektyvumas vėlgi priklauso tiek nuo konkretaus metodo, tiek nuo duomenų „palankumo“ glaudinimui.

PackBits algoritmo atveju, galima nesunkiai apskaičiuoti baigtinę glaudinamumo išraišką visiems įmanomiems duomenų rinkiniams, kadangi šis algoritmas apdoroja tarpusavyje nepriklausomus apibrėžto ilgio duomenų posekius, ir kiekvienu atveju nesunkiai apskaičiuojamas glaudžiausiojo įmanomo posekio kodo ilgis. Tačiau sudėtingesniems metodams tokią analizę atlikti kebloka, kadangi nėra baigtinio atvejų skaičiaus, ir dažniausiai nėra suformuluota tam tinkamų apytikslių statistinių metodų. Kita vertus, dauguma šiuolaikinių algoritmų yra euristiniai, tad įtakos turi ir skaičiavimų laikas, ir darbinės atminties kiekis.

Dėl šių priežasčių dažnai taikomi eksperimentiniai glaudinimo efektyvumo įvertinimo metodai (pavyzdžiui, lyginant atskirų duomenų masyvų glaudinimo efektyvumą skirtingais metodais ir vienodomis sąlygomis).

Kodavimo ir dekodavimo algoritmų sudėtingumas. Paprasčiausi deterministiniai algoritmai, galima sakyti, turi apibrėžtą sudėtingumą visais atvejais. Pavyzdžiui, grupavimo metodų sudėtingumas dažniausiai yra O(n). Sudėtingesnius metodus realizuojantys algoritmai dažniausiai yra godieji tiek atminties, tiek laiko atžvilgiu. Paprastai, tokie algoritmai pritaikomi dirbti su fiksuoto ilgio posekiais, apribojant euristikos apimtis. Tuomet jų sudėtingumą galima laikyti O(n), kadangi skaičiavimų sąnaudos beveik tiesiškai priklauso nuo glaudinamų duomenų apimties. Kita vertus, euristikos apribojimai kai kuriais atvejais turi lemiamos įtakos koduotų duomenų entropijai (tarkim, fraktalinio arba praradiminio glaudinimo atveju).

Todėl šis (sudėtingumo) kriterijus aktualesnis analizuojant metodų teorines galimybes. Neretai, realizuojant algoritmą, tenka jį įvertinti numatant apdorojamų posekių ilgius, išankstinių lentelių apimtis ir techninės įrangos galimybes.

Priklausomai nuo įrangos architektūros, kai kada efektyviau duomenis saugoti glaudžiuoju kodu (pavyzdžiui, jei jie laikomi mažos spartos laikmenoje). Šiuo atveju labai svarbu parinkti metodą ir jo parametrus, kad fragmento paieška, kodavimas ir dekodavimas nesueikvotų daugiau sistemos išteklių nei sutaupoma juos koduojant.

Kai kurių metodų svarbus privalumas – kodavimo ir dekodavimo sudėtingumų asimetrija. Realiojo laiko duomenų kaupimo sistemose kodavimo (įrašymo) sąnaudos aktualesnės nei dekodavimo (skaitymo). Kuriant tiražuojamus arba retai perrašomus, bet dažnai naudojamus išteklius (kaip daugiaterpiai duomenys, aparatinių valdiklių programos), žymiai svarbesnė duomenų dekodavimo sparta.

Tinkamumas duomenų srautams koduoti ir dekoduoti. Kai kurie metodai (LZW, aritmetinis, Huffmano) glaudžiųjų duomenų metainformaciją įrašo duomenų sekos pradžioje. Kitų metodų (JPEG, bangelių, fraktaliniai metodai; ištisinis archyvavimas, dauguma šifruojančiųjų algoritmų) glaudžiųjų sekų *n*-tasis fragmentas gali būti sėkmingai dekoduotas tik turint dekoduotąjį (*n*–*k*)-tąjį fragmentą.

Tokios kodo savybės gali pakelti tiek kodavimo, tiek dekodavimo efektyvumą, tačiau jos apriboja metodų taikymą konkrečiose srityse (pavyzdžiui, laisvosios kreipties laikmenoms, kaip versijų archyvai ir valdiklių programinė įranga, arba realiojo laiko duomenų srautams kaip nuotolinės konferencijos).

Dauguma nesrautinių kodavimo algoritmų į glaudžiuosius duomenis įterpia sinchronizavimo metaduomenis, tačiau tai mažina metodo efektyvumą, o duomenų struktūrą daro sudėtingesnę.

2.2. Glaudinimo metodų vystymosi tendencijos

Grupavimo metodai. Paprasčiausi glaudinimo metodai, anksčiau plačiai taikyti kompiuterių technikoje yra rastrinių failų formatai: l bito taškui PackBits (Apple, Inc., 1982), ir Run-Length Encoding (Microsoft, Inc., 1983). Bendra jų idėja: keletą vienodų duomenų blokų užkoduoti skaitliuku ir reikšme.

Pavyzdžiui, PackBits formatu, po vieną baitas skiriamas pasikartojimų skaičiui, ir vieną-pasikartojančiam fragmentui [10]. RLE formatu reikšmės saugomos adaptyviuoju kodu: pirmajame baite saugoma reikšmė nuo 0 iki 3, jei toliau eina 1-4 baitai be pasikartojimų, ir 4-255, jei atitinkamą skaičių kartų reikia kartoti toliau einantį baitą [11].

Kaip pavyzdį paimkime duomenų masyvą:

X=[2,2,2,2,2,2,2,2,2,1,1,1,6,6,6,6,6,6,4,4,4]

Šiam masyvui taikome RLE („Run Length Encoding“) algoritmą: pirmasis masyvo simbolis (skaičius) yra lyginamas su antruoju; jeigu jie vienodi, į palyginimą įtraukiamas trečiasis skaičius ir t.t. Skaitliukas fiksuoja pirmojo simbolio (skaičiaus) pasikartojimų iš

eilės kiekį. Panašiai analizuojami likę duomenų masyvo simboliai. Suspaustas masyvas, mūsų atveju, bus toks:

Y = [9,2,3,1,6,6,3,4].

Atkuriant masyvą, suspausta informacija nuskaitoma poromis: pasikartojimų kiekis ir pats simbolis (skaičius), užrašant jį tiek kartų, kiek rodo prieš jį esantis skaičius. Atkurtas duomenų masyvas bus toks:

$\tilde{X}$ =[2,2,2,2,2,2,2,2,2,1,1,1,6,6,6,6,6,6,4,4,4],

t. y. gautasis masyvas yra identiškas pradiniam masyvui. Jeigu pradinius duomenis pažymėsime X, o $\tilde{X}$ – po suspaudimo atkurtus duomenis, tai suspaudimą be informacijos praradimo galima nusakyti taip: $X \equiv \tilde{X}$.

Nesunku apskaičiuoti informacijos suspaudimo koeficientą α. Iš tikrųjų, jeigu vieno skaičiaus (simbolio) kodavimui (prieš ir po suspaudimo) imamas tas pats bitų skaičius, tai

$$\alpha = \frac{|Y|}{|X|} = \frac{21}{8} \approx 2{,}6\,.$$

Ši duomenų apdorojimo procedūra pakankamai efektyviai taikoma grafinių vaizdų kodavimui.

Yra keletas šios procedūros modifikacijų, skirtų specialios paskirties duomenims (diskretieji duomenų spektrai, kompiuterinės grafikos vaizdai). Apie jas čia plačiau nekalbėsime.

Šie algoritmai gerai tinka konkretaus taikymo paveikslėliams spausti. Jų kodavimas labai spartus, reikalauja nedaug papildomos atminties. Juos dėl paprastumo galima realizuoti aparatiškai. Tačiau jie tinkami tik siaurai sričiai: jų neįmanoma praplėsti, bei jie visiškai netinkami kito pobūdžio informacijai spausti: PackBits geriausiu atveju sutaupo iki 127/128 pradinio kodo apimties, tačiau blogiausiu atveju (jei nė vienas baitas nesikartoja), sueikvoja dvigubai daugiau atminties nei reikėtų nesuglaudintiems duomenims. Tuo tarpu RLE formatas analogiškomis situacijomis sutaupytų iki 125/126 laikmenos dydžio, tuo tarpu blogiausiu atveju prarasdamas tik 1/5.

Žodyniniai metodai. Šie metodai kūrėjus domina seniausiai, pagrindas jiems sukurtas dar 1952-aisiais (D. Huffman). Tačiau iki šiol dauguma bendrosios paskirties glaudinimo algoritmų taiko žodyninius metodus. Dažniausiai paplitusi modifikacija, taikoma tiek archyvavimo

programose (ZIP, GZIP, ARJ, RAR), tiek grafiniuose formatuose (GIF89a/87a, JPEG, JBIG), paremta LZW kodavimu [10].

Jų esmė – pirmuosius pasitaikiusius kodus aprašyti parankiausiomis (trumpiausiomis) sekomis, tolesniems paliekant ilgesnes. Dažnai tam naudojamas Huffmano kodas[8].

LZW suspaudimo algoritmą trumpai galima apibūdinti taip: nuosekliai nuskaitant duomenis, kuriama kodų vertimo lentelė, sudaryta iš nuorodų. Kiekviena nauja nuoroda yra įtraukiama į kodų lentelę, o jos nurodoma reikšmė – į taip vadinamą išvedimo srautą. Jeigu, toliau nuskaitant duomenis, sutinkama nuoroda, kuri jau buvo aprašyta kodų lentelėje, ji įtraukiama į išvedimo srautą, o kodų lentelėje formuojama nauja nuoroda, apimanti daugiau simbolių. Realizuojant šį procesą, pasireiškia duomenų suspaudimo efektas.

Atkuriant duomenis, nuskaitomas suspaustas masyvas bei kuriama kodų vertimo lentelė. Pilna vertimo lentelė sudaroma, nuskaičius visą suspaustą masyvą. Pradiniai duomenys atkuriami, suspaustame sraute nuorodas pakeičiant atitinkamomis reikšmėmis.

Kaip pavyzdį panagrinėkime duomenų masyvą [12]:

/WED/WE/WEE/WEB (15 simbolių).

Suspaudimo metu kuriama kodų lentelė (l lentelė).

Nuskaitomi duomenys	Suspausti duomenys	Kodai
/W	/	256 =/W
E	W	257 = WE
D	E	258 = ED
/	D	259 = D/
WE	256	260 = /WE
/	E	261 =E/
WEE	260	262 = /WEE
/W	261	263 = E/W
EB	257	264 = WEB
<END>	B	

1 lentelė. LZW kodų lentelė

Taigi, vietoj pradinio duomenų masyvo gauname:

/WED<256>E<260><261><257>B (10 simbolių).

Duomenų atkūrimas pavaizduotas 2 lentelėje.

Nuskaitomi duomenys	Atkurti duomenys	Kodai
/W	/	256 = /W
E	W	257 = WE
D	E	258 = ED
256	/W	259 = D/
E	E	260 = /WE
260	/WE	261 =E/
261	E/	262 = /WEE
257	WE	263 = E/W
B	B	264 = WEB

2 lentelė. Koduotųjų LZW duomenų atkūrimas

Šis metodas taip ilgai populiarus dėl to, kad jis yra gana universalus. Blogiausiuoju atveju, kai duomenų srautas yra labai didelės apimties, jo papildomos metainformacijos apimties santykis su originalių duomenų apimtimi artėja prie 0. Tačiau labai palankiais atvejais (kuomet duomenų apimtis maža, o signalų pasiskirstymas labai netolygus), šis metodas sutaupo iki 95-97% pradinio duomenų kiekio.

Aritmetiniai metodai. Tai taip pat nepraradiminių metodų klasė, artima žodyniniams metodams [12]. Tačiau jie analizuoja fiksuoto ilgio duomenų fragmentų (šaltinio signalo įvykių) statistinį pasiskirstymą, tuomet suteikia jiems realiuosius kodus nuo 0 iki l pagal jų tikimybes: taip, kad išskirtasis intervalas sutaptų su kodo (įvykio) pasirodymo tikimybe. Vėliau visa įvykių seka koduojama kaip realioji kintamo ilgio reikšmė.

Pavyzdžiui, masyvo [BILL GATES] elementams būtų suteikti intervalai [13]:

Tarpas – 0 - 1/10
A – 1/10 - 2/10
B – 2/10 - 3/10
E – 3/10 - 4/10
G – 4/10 - 5/10
I – 5/10 - 6/10
L – 6/10 - 8/10
S – 8/10 - 9/10
T – 9/10 - 1

Toliau šia koduote aprašomas skaičius, išreiškiantis visą seką. Kiekvienos iteracijos metu koduojama vis mažėjančiame skaičių intervale, kuris prieš pirmąjį simbolį yra nuo 0 iki l:

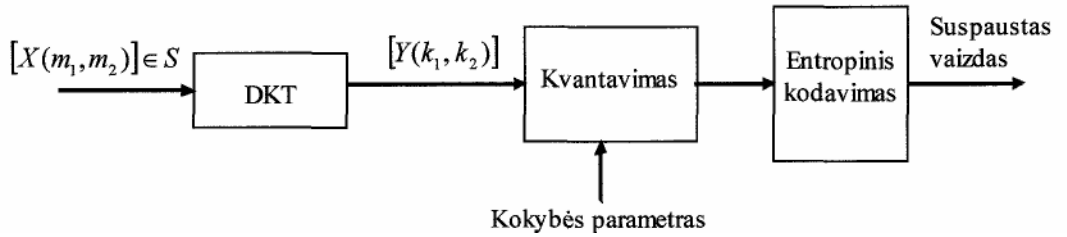
	0.0	1.0
B	0.2	0.3
I	0.25	0.26
L	0.256	0.258
L	0.2572	0.2576
Tarpas	0.25720	0.25724
G	0.257216	0.257220
A	0.2572164	0.2572168
T	0.25721676	0.2572168
E	0.257216772	0.257216776
S	0.2572167752	0.2572167756

Tokiu būdu gautas skaičius 0.2572167752 šioje koduotėje reiškia visą koduojamą seką.

Šis kodavimas yra panašaus efektyvumo kaip Huffmano kodai, kai kodų intervalai panašūs. Tačiau jis žymiai efektyvesnis, kai jie skiriasi. Bendrosios šių algoritmų charakteristikos panašios, tik šio metodo sudėtingumas didesnis, be to, prieš koduojant seką, reikia bent apytiksliai žinoti jos elementų tikimybes. Taigi šis algoritmas nepritaikomas srautiniam kodavimui, kai duomenų kiekis nežinomas, didelis, kai jų didelis vėlinimas arba kai iš anksto nežinomas jų pasiskirstymas.

Praradiminiai metodai: spektrinės analizės taikymas. Daugiaterpiams duomenims (aukštos raiškos paveikslėliams, garsams ir pan.) koduoti dažnai taikoma spektrinė analizė, kurios esmė – suskaidžius seką fragmentais, juos analizuoti, lyg kiekvienas jų sutaptų su realiosios periodinės funkcijos vienu periodu (tuomet atliekama Furjė ar panaši analizė) arba kiekvienas smulkesnis fragmentas laikomas stambesniojo patikslinimu (bangelių arba fraktaliniai metodai). Yra ir kitų panašių praradiminių metodų, kaip diskrečioji kosinusinė transformacija (DCT), dispersiniai kriterijai, hiperbolinis filtravimas ir pan.. Visų jų esmė – kad signalas (pradiniai duomenys) interpretuojamas kaip tam tikros struktūros funkcija, kurią pakankamai gerai aprašo pirmieji keli ją atitinkančios eilutės koeficientai, o likusius galima atmesti [7].

Kai kurie metodai (kaip MP3 garsams, JPEG spalvotiems, o JBIG – bespalviams paveikslėliams koduoti) taiko abi šias metodikas kombinuotai, ir taip pasiekia dar geresnį išlošį. Kaip pavyzdį pateiksime trumpą JPEG algoritmo aprašą [15]:



1 pav. Bendra JPEG suspaudimo metodo schema

1. JPEG standartas yra skirtas nespalvotiems (su pilka intensyvumo skale) vaizdams apdoroti. Todėl spalvoti vaizdai yra preliminariai pertvarkomi, išskaidant juos į spalvines komponentes, kurios vėliau apdorojamos atskirai. Ryškumo komponentė, kuri yra identiška originalo pilkų atspalvių vaizdui, yra atskiriama nuo spalvų intensyvumų (RGB paletės skaidymas parodytas 1 pav.). Tai daroma, norint išsaugoti ryškumo komponentę, kuriai žmogaus akis yra ypač jautri.

2. Apdorojamos tik spalvinės komponentės (ryškumo komponentė neliečiama). Pirmiausia jos yra suspaudžiamos horizontaliai santykiu 2:1, o vertikaliai – santykiu 2:1 arba 1:1. Tai padaroma gretimų taškų reikšmes pakeitus jų vidurkiu. Jau po šio etapo vaizdo dydis sumažėja dvigubai arba trečdaliu, o kokybė beveik nenukenčia.

3. Nuo šio žingsnio vienodai apdorojami tiek pilkų atspalvių, tiek spalvoti vaizdai. Jie yra dalinami į 8x8 dydžio blokus, ir kiekvienam iš jų taikoma diskrečioji kosinusinė transformacija (DKT). Tokiu būdu, vaizde sukaupta informacija perskirstoma pagal dažnį.

4. Gauti 64 vaizdo bloko spektro elementai kvantuojami, panaudojant specialias tam tikslui skirtas kvantavimo matricas. Rezultatai suapvalinami iki sveikųjų skaičių. Būtent šiame etape prarandama dalis vaizde sukauptos informacijos, bet tuo pačiu pasireiškia ir vaizdo suspaudimo efektas.

5. Gauti tarpiniai duomenys apdorojami, taikant entropinį kodavimą su aritmetiniais (arba Huffmano) kodais. Pastarasis kodavimo būdas duomenis spaudžia be informacijos praradimo.

6. Prie gautų duomenų prijungiama antraštė, į kurią įtraukiamos kvantavimo konstantos bei aritmetinių (arba Huffmano) kodų lentelės.

Atkuriant vaizdą, atliekami tie patys žingsniai, tik atvirkščia tvarka.

2.3. Dvejetainės informacijos (sekų) spektrinė analizė kaip glaudinimo priemonė

Diskrečiųjų dvejetainių sekų spektrai kaip sekų faktorizacijos priemonė. Spektrinės analizės metodai, pritaikyti daugiaterpiams duomenims glaudinti, pasiteisino, ir dabar jie yra beveik visų praktikoje taikomų daugiaterpių duomenų srautų glaudinimo metodų pagrindas. Tačiau panašūs metodai kol kas nėra taikomi diskrečiųjų duomenų srautams. Skirtingai nei analoginės prigimties daugiaterpiuose duomenyse, diskrečiuosiuose (dvejetainiuose) srautuose (bendruoju atveju) neįmanoma išskirti „nejuntamų“ paklaidų, kurias būtų galima įvesti. Be to, klasikinės tolydžiųjų funkcijų periodiškumo analizės priemonės netinka analizuoti diskretiesiems signalams.

Kita vertus, diskretieji srautai gali būti interpretuojami kaip *signalinės funkcijos* $f : \mathbf{N} \to \mathbf{B}$; čia $\mathbf{B}=\{0;1\}$ – dvejetainių (loginių, Būlio) reikšmių, aibė. Fiksuoto ilgio seką taip pat galima išreikšti bitų seka (numeruojant nuo 0):

$$B = [b_0 b_1 ... b_i ... b_{n-1}], b_i \in \mathbf{B}; \qquad (2.3.1)$$

Dėl analizės patogumo, paprastai laikoma, kad sekos ilgis $|B| = 2^n, n \in \mathbf{N}$, o sekos *i*-tasis narys sutampa su *n*-matės *dvejetainės funkcijos* $f^{<n>}$ reikšme, kurią nusako funkcijos *teisingumo lentelė* $\tau\tau\, f$ [5]:

$$\begin{aligned} & f^{<n>}(x_1, x_2, ..., x_i, ..., x_n) = f^{<n>} : \mathbf{B}^n \to \mathbf{B}, \\ & f^{<n>}(x_1, x_2, ..., x_i, ..., x_n) = \tau\tau\, f^{<n>}(2^{n-1} x_1 + ... + 2^{n-i} x_i + ... + x_{n-1}), \\ & b_i = \tau\tau\, f(i), \\ & |\tau\tau\, f^{<n>}| = |B| = 2^n \end{aligned} \qquad (2.3.2)$$

Tokias funkcijas patogu analizuoti diskrečiosios Būlio aritmetikos priemonėmis. Dėl simetriškumo ir nelyginumo[2], paprastai, diskrečiajai spektrinei analizei taikoma sumos moduliu 2 operacija [1].

Loginės funkcijos gali būti išskleistos į baigtinio skaičiaus periodinių signalinių funkcijų kompoziciją naudojant sumą moduliu 2. Gautieji periodinių funkcijų koeficientai vadinami *signalinės funkcijos spektru*. Yra sukurtas ir praktiškai išbandytas optimizuotas algoritmas [Vala01], kurio pagalba (naudojant diskretųjį funkcijos spektrą), randama optimali funkcijos faktorizacija į dvi funkcijas.

Dvejetainės sekos glaudinimas naudojant faktorizaciją. Dvejetainę seką (signalinės funkcijos teisingumo lentelę) siekiama suglaudinti, ją atitinkančią loginę funkciją išreiškiant

kaip dviejų ar daugiau mažesnių matmenų (komponuojančiųjų) funkcijų kompoziciją, tam naudojant sumos moduliu 2 operaciją:

$$f_1^{<n>}(X) = f_2^{<l>}(Y) \oplus f_3^{<m>}(Z) \oplus 0^{<n>};$$
$$Y, Z \subset X = \{x_i\}; m, l, n \in \mathbf{N}; m, l < n; \qquad (2.3.3)$$

čia $0^{<n>}$ – n kintamųjų nulinė funkcija.

Pilnai išskaidoma dvejetaine seka vadinama tokia seka, kurią atitinkanti loginė funkcija $f^{<n>} \neq 0^{<n>}$ turi skaidinį pagal (2.3.3), ir $Y \bigcap Z = \varnothing$.

Dalinai išskaidoma dvejetaine seka vadinama tokia pilnai neišskaidoma seka, kurią atitinkanti loginė funkcija turi skaidinį pagal (2.3.3), kad $2^{|Y|}+2^{|Z|} \leq 2^{|X|}$ ir $Y \bigcap Z \neq \varnothing$.

Pavyzdžiui, 16 bitų ilgio duomenų seka

$B_1 = \tau\tau(f_1(x_1,x_2,x_3,x_4)) = [1001011001011010]$,

turi dalinį skaidinį:

$$f_1 = x_3 \oplus \bar{x}_4 \oplus x_1\bar{x}_3 = f_2(x_3, x_4) \oplus f_3(x_1, x_3) \oplus 0^{<4>},$$
$$\tau\tau(f_2(x_3,x_4)) = [1001];$$
$$\tau\tau(f_3(x_1,x_3)) = [0101];$$
$$\tau\tau\ 0^{<4>} = [0000000000000000].$$

Šios sekos komponuojančiųjų funkcijų teisingumo lentelėms užrašyti (neskaitant informacijos, skirtos kompozicijos parametrams) pakanka 4+4=8 bitų. Kompozicijos parametrų duomenims reikalingas atminties kiekis yra sąlyginai didelis, kai faktorizuojamos „mažos“ funkcijos, tačiau esant ilgesnėms „palankioms“ glaudinimui pradinėms sekoms, galima tikėtis išlošio.

Kodavimo ir dekodavimo spektrinio faktorizavimo metodu sudėtingumas. Gautųjų fragmentų suliejimą į pradinę seką įmanoma realizuoti aparatiškai, be to, jis gali būti atliekamas aparatiškai, vienu procesoriaus taktu (lygiagretinant skaičiavimus).

Tačiau, sekai faktorizuoti (glaudinti) iki šiol nebuvo pasiūlyta metodo, geresnio nei godusis. Spektrinis faktorizavimo algoritmas yra pakankamai spartus, jei sekos ilgis neviršija 2^6 bitų, tačiau jis netinkamas ilgesnėms sekoms faktorizuoti, kadangi sekos neskaidomumo nustatymo algoritmo sudėtingumas yra $O(n!^2)$ [3].

Dvejetainės sekos faktorizavimo prielaidos. Skirtingai nei visi aptartieji nepraradiminiai metodai, tiriamasis faktorizacijos metodas neparemtas fragmentų pasikartojimu. Tiriamuoju atveju, be papildomų euristinių priemonių, kiekvienas fragmentas yra lokalizuojamas ir analizuojamas nepriklausomai nuo gretimų fragmentų (panašia strategija remtasi, kuriant pirminius praradiminius spektrinius glaudinimo metodus). Tai leidžia teigti, kad kuriamas metodas vienodai efektyviai glaudintų duomenis, nepriklausomai nuo jų glaudinimo žodyniniais ar aritmetiniais algoritmais efektyvumo.

Nustatyta keletas svarbesnių būtinųjų sekos dalinio ir pilno skaidomumo požymių (jų formuluotės ir įrodymai pateikti tolimesniuose skyreliuose). Remiantis jais, bei atlikus preliminarų perrinkimo eksperimentą, buvo apskaičiuotas efektyviai skaidomų sekų skaičius atskirose ribotų ilgių sekų aibėse (1 lentelė.).

n	Sekų skaičius, 2^{2^n}	Skaidomų sekų skaičius
2	16	6
3	256	84
4	65536	8486
5	$4.29 \cdot 10^9$	$1.85 \cdot 10^7$

2 lentelė. Efektyviai skaidomos neapdorotos 2^n bitų ilgio sekos

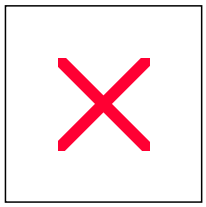
Iš turimų rezultatų galima teigti, kad vien faktorizacijos išraiškų įrašymas, bendruoju atveju, teigiamo glaudinimo efekto neduotų. Nors, galima būtų tikėtis, kad dauguma žodyninių algoritmų efektyviai glaudintų panašų kiekį atsitiktinių duomenų sekų, tačiau esamos informacijos kodavimo priemonių savybės lemia geresnį jų praktinį efektyvumą.

3. DVEJETAINĖS INFORMACIJOS (SEKŲ) SPEKTRINIŲ SKAIDINIŲ LENTELIŲ PRAKTINIO PANAUDOJIMO ANALIZĖ

3.1. Sekų ekvivalentumas jų skaidomumo atžvilgiu

Geometrinės dvejetainių sekų interpretacijos. Nėra tikslaus algoritmo, kurio sudėtingumas būtų mažesnis nei $O(n!^2)$ ir kuris leistų optimalias komponuojančiąsias funkcijas (apribojant komponuojančiųjų funkcijų skaičių iki 2) [4]. Taigi be modifikacijų šis metodas praktiškai nepritaikomas, kai $n>7$. Tačiau esant mažiems n, perteklinės informacijos, reikalingos pradinei sekai atstatyti, kiekis yra per didelis, kad būtų galima tikėtis efektyvaus realių duomenų glaudinimo. Todėl iškeltas tikslas kiek įmanoma sumažinti analizuojamų sekų aibę ir apibendrinti esmines jų savybes.

Dvejetainę loginę (ilgio 2^n) bitų ilgio seką sutapatinus su n-tojo laipsnio logine funkcija, jos grafiką galima pavaizduoti n-mačiu hiperkubu (1 pav.).



2 pav. Funkcijos $f(x,y,z)$=[11001001] geometrinė interpretacija

Sekos *ašimi* x_i vadinsime ją atitinkančios funkcijos argumentą x_i.

Sekos $A^{<2^n>}$ *tašku* $b_1b_2...b_i...b_n$ vadinsime jos bitą, kurį atitinka funkcijos reikšmė $f_A(b_1,b_2,...,b_i,... ,b_n)$.

Atstumu tarp sekos A taškų $b_1b_2...b_i...b_n$ ir $b'_1b'_2...b'_i...b'_n$ vadinsime reikšmę:

$$\mathbf{dist}(b_1b_2...b_i...b_n,\ b'_1b'_2...b'_i...b'_n)=\sum_{i=1}^{n} b_i \oplus b'_i \qquad (3.1.1)$$

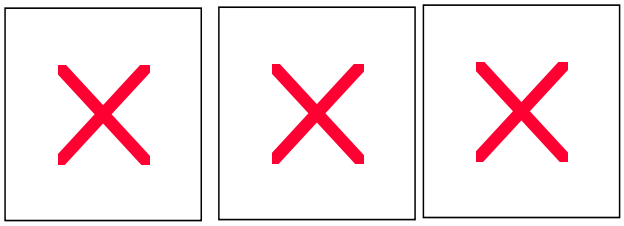
Sekų ekvivalentumas skaidomumo atžvilgiu. Siekiant sumažinti skaičiavimų apimtis ir tuo pačiu padidinti maksimalų analizuojamų sekų ilgį, naudojamas preliminaraus fiksuoto ilgio sekų klasifikavimo ir apdorojimo metodas, paremtas *bazinėmis* (*identifikuojančiosiomis*) sekomis. Bazinės sekos randamos naudojantis *ekvivalenčiąja skaidomumo atžvilgiu transformacija*.

Toliau pateikiame sekų ekvivalentiškumo skaidomumo atžvilgiu apibrėžimus. Jiems tinka visos matematinio ekvivalentumo operatoriaus savybės (tranzityvumas, komutatyvumas, asociatyvumas), todėl atskirų atvejų apibrėžimus galima taikyti nebūtinai dviems minimaliai skirtingoms sekoms. Pavyzdžiui, kai kalbama apie sekų ekvivalentumą dviejų ašių sukeitimo atžvilgiu, taip pat turimas omenyje ekvivalentumas trijų ar daugiau ašių sukeitimo atžvilgiu.

Ekvivalenčiosiomis inversijos atžvilgiu sekomis vadinamos vienodo ilgio sekos $B=[b_0...b_i...b_{n-1}]$ ir $\overline{B}=[\overline{b}_0...\overline{b}_i...\overline{b}_{n-1}]$. Jei seka B yra dalinai (arba pilnai) išskaidoma, tai ir jai ekvivalenti inversijos atžvilgiu seka $\overline{B}$ yra atitinkamai dalinai (pilnai) išskaidoma. Jei seka B yra neišskaidoma, tai ir jai ekvivalenti inversijos atžvilgiu seka $\overline{B}$ yra neišskaidoma. Jei seką B atitinkanti funkcija f_B turi skaidinį $f_B=f_1\oplus...\oplus f_i\oplus...\oplus f_n$, tai jai ekvivalenčią seką atitinkanti funkcija turi skaidinį $f_{\overline{B}}=\bar{f}_B=f_1\oplus...\oplus f_i\oplus...\oplus\bar{f}_{n-1}$ [6].

Ekvivalenčiosiomis kintamojo x_i inversijos atžvilgiu arba *ekvivalenčiosiomis ašies x_i inversijos atžvilgiu sekomis* vadinamos vienodo ilgio sekos B_a ir B_b, kurias atitinka loginės funkcijos $f_{Ba}(x_1,x_2,...,x_i,...,x_n)=f_{Bb}(x_1,x_2,...,\bar{x}_i,...,x_n)$. Ekvivalenčiųjų kintamojo x_i atžvilgiu sekų skaidinių atitinkami nariai yra lygūs, jei jų reikšmės nepriklauso nuo kintamojo x_i ir yra ekvivalentūs kintamojo x_i inversijos atžvilgiu priešingu atveju.

Ekvivalenčiosiomis kintamųjų (ašių) x_i ir x_j sukeitimo atžvilgiu sekomis vadinamos vienodo ilgio sekos B_a ir B_b, kurias atitinka loginės funkcijos $f_{Ba}(x_1,x_2,...,x_i,x_j,...,x_n)=f_{Bb}(x_1,x_2,...,x_j,x_i,...,x_n)$. Ekvivalenčiųjų kintamojo x_i atžvilgiu sekų skaidinių atitinkami nariai yra lygūs, jei jų reikšmės nepriklauso nuo kintamųjų x_i arba x_j, ir yra ekvivalentūs kintamųjų sukeitimo atžvilgiu priešingu atveju.



a) Originali seka b) Kintamojo *x* inversija c) Kintamųjų *x* ir *y* sukeitimas

3 pav. Ekvivalenčiųjų transformacijų pavyzdžių geometrinė interpretacija

Ekvivalenčiosiomis skaidomumo atžvilgiu sekomis vadinamos vienodo ilgio sekos B_a ir B_b, kurios yra ekvivalenčiosios inversijos, kintamojo inversijos arba kintamųjų sukeitimo atžvilgiu.

Ekvivalenčiąja skaidomumo atžvilgiu transformacija vadinama aritmetinių ir loginių veiksmų seka, kurių pagalba seka B_a pakeičiama jai ekvivalentiška skaidomumo atžvilgiu seka B_b. Transformaciją galima vienaprasmiškai išreikšti kaip sveikąjį teigiamą skaičių, kuriame užkoduoti sukeičiamųjų bei invertuojamųjų ašių numeriai bei sekos invertavimo požymis. Nustatyta, kad bet kuri ekvivalenčioji skaidomumo transformacija $\mathbf{trans}:\mathbf{B}^{2^n}\times\mathbf{N}\to\mathbf{B}^{2^n}$ gali būti išreikšta baigtiniu skaičiumi loginių ir postūmio veiksmų bei realizuotas algoritmas šiems veiksmams užkoduoti į transformacijos kodą, taip pat įvykdyti transformaciją (blogiausio atvejo realizacijos sudėtingumas $O(n\cdot\log n)$) bei apskaičiuoti atvirkštinės transformacijos kodą [5].

Taigi, *turint sekos B_a skaidinį ir ekvivalentiškosios transformacijos* **trans**$(B_b, c_{ba})=B_a$ *kurios pagalba seka B_b transformuojama į B_a, kodą c_{ba}, baigtiniu loginių ir postūmio operacijų skaičiumi randama:*

a) sekos B_b skaidinys, ne blogesnis nei sekos B_a skaidinys;

b) transformacijos, sekos B_a skaidinio narius transformuojančios į sekos B_b skaidinio sekos narius, kodas c_{ab}.

Norint rasti bet kurios 2^n bitų ilgio sekos skaidinį, pakanka ištirti po vieną kiekvieno ekvivalenčiųjų sekų poaibio seką.

Poaibius identifikuojančiosios sekos. Kiekvienas ekvivalentiškųjų skaidomumo atžvilgiu sekų poaibis gali būti identifikuojamas pagal iš anksto susitartą kriterijų vienprasmiškai parenkama seka iš šio poaibio.

Identifikuojančiosios funkcijos kriterijumi $B_a\succ B_b$ vadinama laisvai pasirenkama realioji funkcija $f(B_a,B_b)$, kuri įgyja reikšmę, didesnę nei 1, jei B_a geriau atitinka kriterijų nei B_b, mažesnę nei 1 priešingu atveju, arba lygi 1, kai B_a ir B_b sutampa. Jei *kriterijų realizuojančiu algoritmu* neįmanoma B_a ir B_b rasti ekvivalenčiosios transformacijos kodo c_{ab}, tuomet kriterijaus reikšmė neapibrėžta. Skaitinio sekos ekvivalento (leksikografiniu) kriterijumi

vadinama funkcija, lygi aritmetiniam B_a ir B_b skaitinių ekvivalentų santykiui arba 1, jei jų vertės lygios nuliui.

Kadangi algoritmas, realizuojantis tikslią leksikografiškai didžiausios ekvivalenčiosios skaidomumo atžvilgiu funkcijos paiešką, yra sudėtingas ir kai kuriais atvejais net artimas pilnojo perrinkimo algoritmui [5], tai tikslinga suformuoti euristinį kriterijų, kuris būtų spartus, tačiau suskaidytų sekų aibę į kiek įmanoma mažiau poaibių, kurie (ne algoritmo atžvilgiu) sudarytų ekvivalenčiųjų sekų poaibius. Tačiau šis algoritmas turėtų atskirti skirtingų ekvivalenčiųjų sekų poaibius. Vėliau, turint rezultatus, sugeneruotus poaibius galima dar kartą apdoroti tikslesniu algoritmu.

Identifikuojančiąja seka kriterijaus f atžvilgiu vadinama tokia seka B_{baz}, kad kriterijų realizuojančio algoritmo atžvilgiu:

$$\forall B_i, B_i \neq B_{baz} : \exists c_{baz,i} : \mathbf{trans}(B_{baz}, c) = B_i : B_{baz} \succ B_i. \qquad (3.1.2)$$

Sekos B ekvivalentiškųjų skaidomumo atžvilgiu sekų poaibį identifikuojančią seką toliau žymėsime **id** B. Ekvivalentiškumo kriterijų k realizuojantį algoritmą (funkciją, transformuojančią seką B į **id**$_k$ B) toliau žymėsime **id**$_k$.

3.2. Kai kurios sekų skaidomumo būtinosios sąlygos

Tyrinėjant dvejetainių sekų faktorizavimo, naudojant sumą moduliu 2, galimybes (pirmuosiuose darbuose – fiksuoto ilgio sekas perrenkant rankiniu būdu, vėliau šį procesą automatizuojant ir ieškant anomalijų), eksperimentiškai išskirta, o vėliau analitiškai pagrįsta keletas sekos skaidomumo būtinųjų požymių. Būtent jų pagalba buvo formuluojamos funkcijų ekvivalentumo skaidomumo atžvilgiu taisyklės. Taip pat, taikant identifikuojančiųjų funkcijų paiešką, dažniausiai šie požymiai (jei jie tiesiogiai nepasireiškia pradinėje sekoje) aptinkami iteracinio proceso metu.

Be to, jie leidžia kelti optimistinių prielaidų, kalbant apie skaidomų ir neskaidomų sekų informacijos entropiją atskiruose sekų poaibiuose, taigi ir apie papildomas glaudžiojo kodavimo galimybes (kaip minėta ankstesniuose skyreliuose, tiriamoji metodika orientuota į sekas, kurioms netinka klasikiniai entropiniai metodai, kaip fragmentų pasikartojimų ar netolygaus informacijos šaltinio įvykių pasiskirstymo analizė).

Toliau pateikiame keletą akivaizdžių neskaidomumo požymių pavyzdžių.

Nelyginė sekos bitų suma. Nesunkiai įrodoma, kad seka $B=[b_1b_2...b_i...b_n]$ išskaidoma tik tada, kai

$$b_1 \oplus b_2 \oplus ... \oplus b_i \oplus ... \oplus b_n = 0. \qquad (3.2.1)$$

Šio požymio netenkina 50% visų įmanomų sekų. Tačiau jį nustatyti labai paprasta (4-ajame skyriuje aprašytas $O(\log_2 n)$ sudėtingumo algoritmas). Kita vertus, tokie paprasti požymiai leidžia vystyti euristinius ar adaptyviuosius sekų glaudinimo algoritmus bei efektyviau užkoduoti sekas glaudžiajame kode. Vienas iš šio požymio ypatybių – sąlyginai paprasta jo lokalizacija.

Pavyzdžiui, jei seka neišskaidoma pagal šį požymį, tai lygiai vienas iš jos posekių B_0 ir B_1, sudarytų iš funkcijos $f(x_1, x_1, \dots x_i, \dots x_n)$, bitų, kai x_i=const (atitinkamai, kai x_i=0, ir kai x_i=1), turi šį požymį su bet kuriuo kintamuoju x_i kai $i \in [1,n]$. Realizavus efektyvų kintamo ilgio fragmentų kodavimą, šį (nelyginės bitų sumos) požymį nesunku pritaikyti kaip kriterijų fragmento ilgiui parinkti (ir tuo būdu lokalizuoti neskaidomumo požymį optimalaus ilgio fragmente).

„Veidrodinės" sekos. Seka neišskaidoma tuomet, kai ją atitinka dvejetainė funkcija

$$f^{<n>}(X) = \begin{cases} 1, & X = A, \\ 1, & X = \overline{A}, \\ 0, & \text{kitais atvejais.} \end{cases} \qquad (3.2.3)$$

Čia $n>2$, $A \in \mathbf{B}^n$ – bet kuri iš galimų X reikšmių. Šį požymį turinčios funkcijos gali būti faktorizuotos į nenulinių dalinių funkcijų rinkinį, iš kurių bent dvi bus tos pačios eilės, kaip ir pradinė funkcija. Ši savybė tinka tik kai $n>2$.

Taip pat neišskaidomi tokių vienodos eilės funkcijų dariniai $f_1^{<n>}(X) \oplus f_2^{<n>}(X) \oplus ...$, kuriems neįmanoma rasti tokių posekių kintamojo x_i=const atžvilgiu (kaip nelyginumo požymio atveju), kad jų kompozicija būtų išskaidoma. Negana to, funkcija yra neišskaidoma, jei šį požymį turi lygiai vienas jos posekis kintamųjų x_i, x_j ...=const atžvilgiu. Panašūs sekos neskaidomumo požymiai (su papildomomis sąlygomis) išvedami, kai seka turi keletą tokių posekių.

3.3 Optimalaus glaudinamų fragmentų ilgio parinkimas

Norint pasiekti didžiausią glaudinimo efektyvumą, labai svarbu parinkti tinkamo ilgio apdorojamųjų sekų fragmentus. Parinkus per mažo ilgio fragmentus, sumažėja jų apdorojimo galimybių, bei smarkiai išauga metaduomenų ir glaudžiųjų duomenų apimčių santykis. Taip pat mažų sekų faktorizacija neturi prasmės. Kita vertus, paprastai fragmento glaudinimo algoritmai yra labai imlūs sistemos ištekliams.

Siūlomo metodo atveju, kai fragmentai yra fiksuoto ilgio, svarbiausias jo parinkimo aspektas yra techninės sistemos galimybės, iš kurių labiausiai ribojančios yra pastoviųjų saugyklų talpa (reikalinga didelės apimties skaidinių lentelei saugoti) ir skaičiavimų laikas (reikalingas identifikuojančiosioms sekoms rasti ir joms transformuoti).

Tarkime, kad identifikuojančiųjų skaidinių lentelė sudaryta iš tokių įrašų (realizuojant programiškai, paprastai pasirenkama kiek sudėtingesnė struktūra, tačiau lentelių apimtys dėl to keičiasi palyginti nežymiai; taip pat čia neatsižvelgiame į galimą kodų optimizaciją):

Tarsime, kad tokiam n kintamųjų identifikuojančiąja funkciją atitinkančiam įrašui saugoti 2^{n+1} bitų (bet kuriuo atveju, skaidinys užrašomas mažesniu skaičiumi bitų, nei skaidomoji funkcija). Laikysime, kad identifikuojančiosios funkcijos randamos be klaidų, todėl lentelėje nėra perteklinių ar pasikartojančių įrašų. Imsime atvejį, kuomet lentelėje neskaidomų funkcijų nesaugome.

Esant fiksuotam funkcijos kintamųjų (sekos ašių) skaičiui n, įmanoma sugeneruoti 2^{2^n} skirtingų funkcijų (sekų), kurioms užrašyti reikia 2^{2^n+n+1}bitų. Jei tarsime, kad visos funkcijos yra identifikuojančiosios (įmanoma parinkti tokį neefektyvų kriterijų), į lentelę rašysime visas sekas (arba laikysime, kad jos visos skaidomos), ir kad lentelei saugoti galime paskirti iki l terabaito (2^{43} bitų), tuomet galėsime saugoti pilnąją atvejų lentelę, kai $n<6$, o maksimalus fragmento ilgis bus $2^5=32$ bitai. Tokia lentelė užimtų 64 GB saugyklos.

Viename ekvivalenčiųjų funkcijų poaibyje negali būti daugiau funkcijų, nei jų įmanoma sugeneruoti su visais skirtingais transformacijos kodais ($2^{n+1}\cdot n!$). Tarkime, kad vidutiniame poaibyje yra apie $(2^{n+1}\cdot n!)^{0,5}$ funkcijų. Taip pat – kad iš 2^{2^n} funkcijų tik 2^{2^n-1} yra skaidomos. Tuomet su n=6 lentelės apimtis būtų $2^{n+1+2^{n-1}}/(2^{n+1}\cdot n!)^{0,5}\approx 2{,}218\cdot 10^9$ bitų, arba 264,4 MB. Kadangi tokia lentelė (kaip parodyta 4 skyriuje) būtų naudojama tik duomenims glaudinti, tai ji neturėtų būti laikoma per didele. Taigi kuriamą metodą orientuosime į 64 bitų fragmentus. Ketvirtajame skyriuje detaliai aprašomas pakankamai efektyvūs algoritmai (orientuoti į vektorinius skaičiavimus bei aparatinę dvejetainę aritmetiką) apdoroti tokio ilgio sekas, todėl čia jų sudėtingumų atskirai nedetalizuosime, ir priimsime, kad skaičiavimų apimtys yra tinkamos atlikti tolesniems tyrimams.

4. DVEJETAINIŲ SEKŲ GLAUDINIMO, NAUDOJANT IDENTIFIKUOJANČIĄSIAS SEKAS, ALGORITMAS

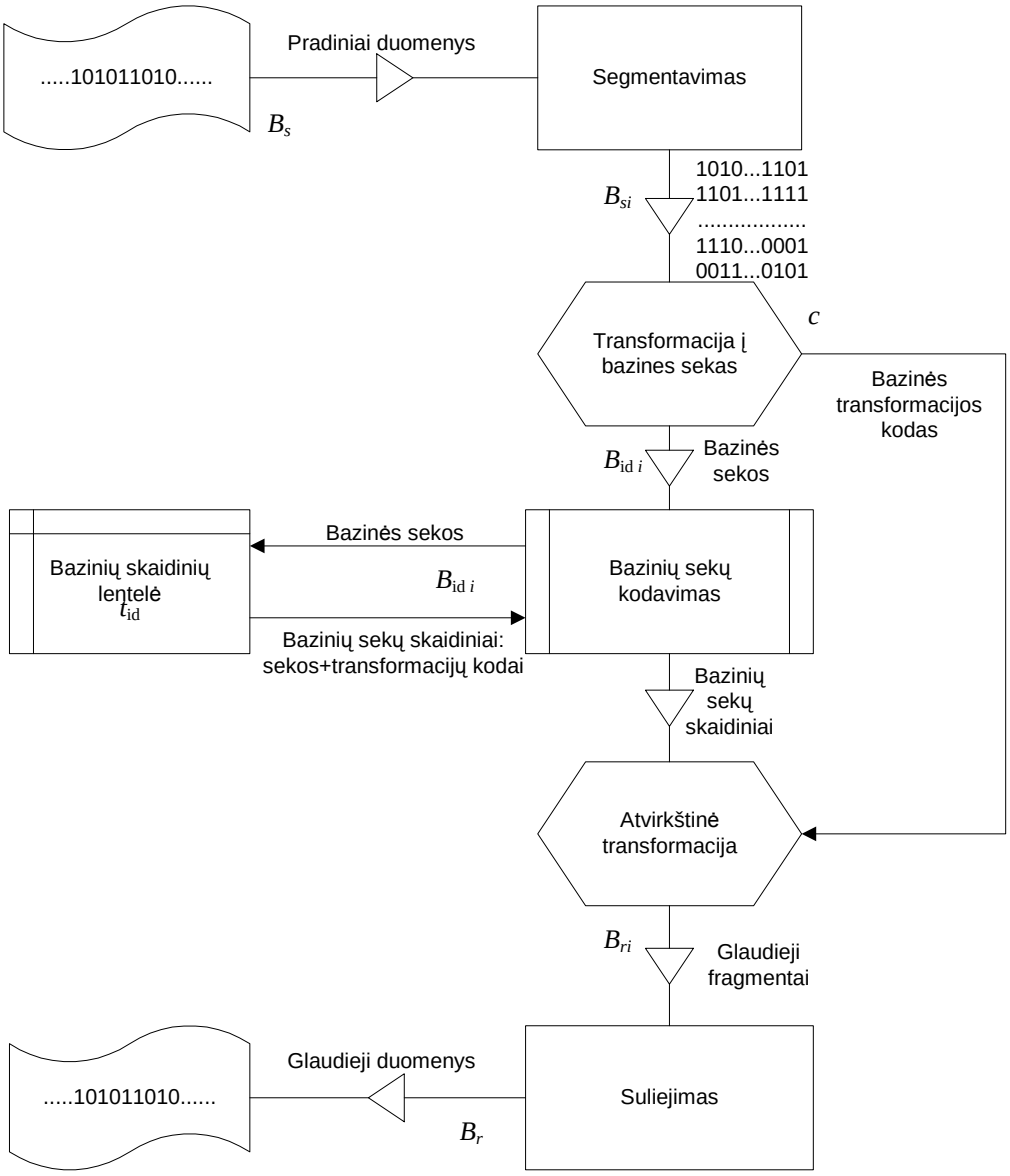
4.1. Bendrasis dvejetainių sekų kodavimo ir dekodavimo algoritmas naudojant identifikuojančiąsias funkcijas

Tiriamasis metodas nėra pritaikytas galutiniam naudojimui praktikoje. Tokio galutinio algoritmo sudarymas reikalautų platesnės analizės nei leidžia šio darbo apimtis. Tačiau aptarta pagrindinė kliūtis realizuoti šią metodiką – skaičiavimų apimtis – ir būdai ją sumažinti. Norint šį metodą pritaikyti kasdieniam naudojimui, reikėtų plačiau ištirti glaudžiuosius išskaidytų sekų kodus, sudarytųjų identifikuojančiųjų sekų poaibių savybes ir daug kitų aspektų. Preliminarūs bandymai rodo, kad identifikuojančiosios sekos nėra pasiskirsčiusios tolygiai, skirtinguose ekvivalenčiųjų sekų poaibiuose sekų skaičius skirtingas. Tik išnaudojant šias savybes galima tikėtis efektyvaus glaudinimo.

Šiame skyriuje pateikiamam metodui nekeliami glaudinimo efektyvumo reikalavimai. Siūloma bendroji tokių algoritmų schema, kuri ateityje galėtų būti taikoma kaip bazė efektyvaus glaudinimo algoritmams.

Toliau bus minima identifikuojančiųjų funkcijų optimaliųjų skaidinių lentelė, kuri pilnai sudaryta atvejams iki n=6. Ji paremta Solinger DB tiesine optimizuotąja lentele, kurios elemento paieškos laikas – iki 10 peržiūros iteracijų (1 iteracija „sekų medžio šaknies“ parinkimui ir 9 – funkcijos paieškai balansuotajame dvejetainiame medyje).

Kodavimo algoritmo struktūra. Sekų kodavimo (glaudinimo) algoritmas pradinio dvejetainių duomenų šaltinio informacijos srautą koduoja glaudesniu kodu. Pradinius algoritmo duomenis sudaro tiesinis šaltinio duomenų srautas (signalinės funkcijos teisingumo lentelė) $B_s{=}\tau\tau\, f_s$; $f_s{:}\mathbf{N}{\to}\mathbf{B}$ ir bazinių (identifikuojančiųjų) sekų iki n bitų ilgio lentelė $t_{\mathrm{id}}{:}\mathbf{B}^n{\to}\mathbf{B}^n$. Laikoma, kad lentelė optimizuota taip, kad sekos paieška joje atliekama per O(n) laiką. Taip pat turima funkcija, realizuojanti lentelės identifikuojančiųjų funkcijų kriterijų $k(B_{\mathrm{id}\,i}, B_i) = B_{\mathrm{id}\,i} \succ_{\mathrm{k}} B_i$, ir šį kriterijų maksimizuojanti funkcija $\mathbf{id}_k$: $B_{\mathrm{id}\,i} = \mathbf{id}_k(B_i)$. Aptariant bendrąją algoritmo schemą, laikoma, kad funkcijos k ir $\mathbf{id}_k$ aprašomos atskirai (tokių funkcijų atskirų realizacijų aprašymas pateiktas 3 skyriuje), o bendroji algoritmo schema nuo jų vidinės realizacijos nepriklauso.



3 pav. Bendrasis kodavimo algoritmas

Duomenų srautas f_s skaidomas fragmentais B_{si}:

$$B_s = B_{s0} \,||\, B_{s1} \,||\, ... \,||\, B_{si} \,||\, ... \,||\, B_{sm}$$

kur m – iš anksto nežinomas fragmentų skaičius.

Kiekvienam fragmentui B_{si} randama sekų poaibį identifikuojanti seka $B_{\mathrm{id}\,i}$, transformacijos kodas c: **trans**(B_{si}, c) = $B_{\mathrm{id}\,i}$ ir atvirkštinės transformacijos kodas c': trans($B_{\mathrm{id}\,i}$, c') = B_{si} . Iš lentelės t_{id} parenkamas optimalusis $B_{\mathrm{id}\,i}$ skaidinys. Atlikus atvirkštinę transformaciją , randamas B_{si} optimalusis skaidinys B_{ri}:

$$B_{ri} = \mathbf{trans}(\, t(B_{\mathrm{id}\,i}),\, c') = \mathbf{trans}(\, t(\, \mathbf{trans}(B_{si},\, c)\,),\, c')$$

Algoritmo rezultatus sudaro duomenų srautas:

$$B_r = \tau\tau\, f_r = B_{r0} \,||\, B_{r1} \,||\, ... \,||\, B_{ri} \,||\, ... \,||\, B_{rm}$$

Bendroji algoritmo schema pateikta 3 pav.

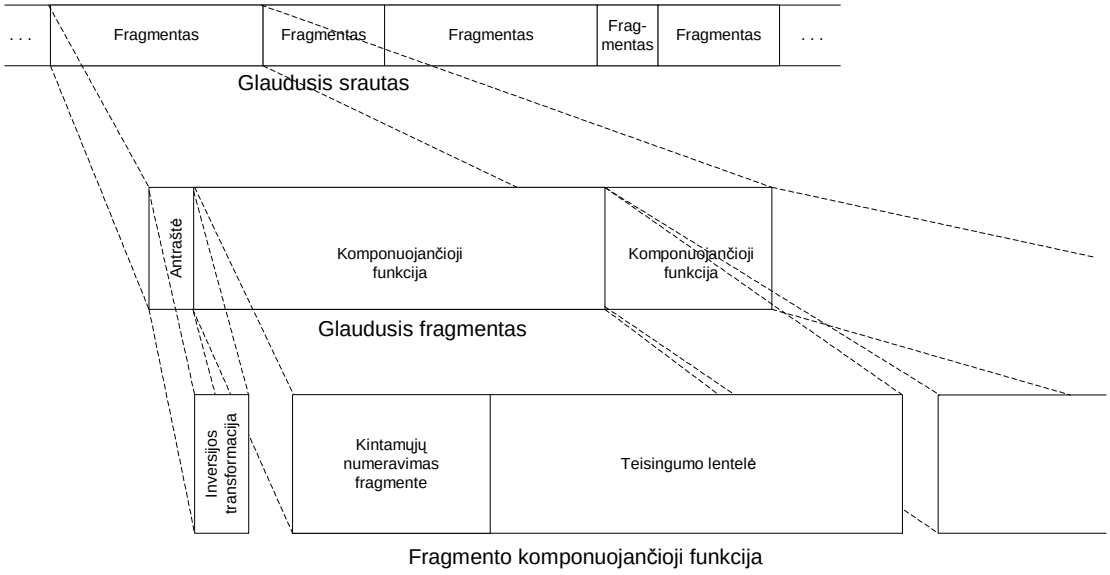
Skaidinių srauto kodavimas. Kaip efektyviai užkoduoti gautuosius fragmentų skaidinius, priklauso nuo konkretaus metodo taikymo ir parinktosios euristikos. Todėl pateiksime metodą, kuris iliustruoja bendruosius glaudžiųjų duomenų srauto formavimo principus, tačiau nėra efektyvus entropijos ir skaičiavimų apimties atžvilgiu. Taip pat nedetalizuosime atskirų koduočių parinkimo motyvų.

Norint vienareikšmiai atkurti pradinį (šaltinio) duomenų srautą, apdorotajame kode reikia išsaugoti šią informaciją:

- šaltinio duomenų fragmentų ribas arba ilgius,
- fragmento komponuojančiųjų funkcijų ribas arba ilgius,
- minimizuotus komponuojančiųjų funkcijų transformacijų kodus fragmento erdvėje.

Apsiribosime paprasčiausiu atveju, kuomet šaltinio duomenys fragmentai yra fiksuoto ilgio (64 bitų). Praktiniams uždaviniams spręsti toks segmentavimo būdas netinka, kadangi 64 bitų sekų faktorizacija didelio glaudinimo efekto neduoda, tačiau optimalių fragmentų parinkimas reikalautų papildomos analizės, kuri išeitų už šio darbo ribų. Kiek detaliau ši problema aptarta 3.4 skyrelyje.

Taip pat tarsime, kad kiekvienas fragmentas faktorizuojamas į dvi komponuojančiąsias funkcijas. Kiekviena komponuojančioji funkcija įrašoma daliniu transformacijos kodu ir dalinai transformuotąja teisingumo lentele. Dalinį transformacijos kodą sudaro adaptyviojo kintamųjų numeravimo kodas. Funkcijos inversijos transformacija įrašoma segmento (komponuojančiųjų funkcijų grupės) antraštėje, o komponuojančiosios funkcijos transformuojamos taip, kad jų pirmasis bitas būtų lygus 1, kurį galime atmesti iš visų komponuojančiųjų funkcijų teisingumo lentelių ar panaudoti kitai metainformacijai saugoti. Bendra glaudžiojo fragmento schema pateikta 4 pav..



4 pav. Suglaudintas duomenų srautas

Svarbiausia, ir daugiausia perteklinių metaduomenų sauganti glaudžiojo srauto dalis yra kintamųjų numeravimo kodas atskiruose fragmentuose. Juo aprašomas atskirų fragmentų ilgis, galima papildoma informacija bei komponuojančiųjų funkcijų tarpusavio sąryšis. Papildomai neoptimizavus šio kodo, dviems fragmentams jo ilgis būtų $\log_2(2 \cdot n! \cdot \log_2 n)$ bitų. 64 bitų ilgio fragmentams tai sudarytų 14,08 bitų, taigi, atsižvelgus į tam tikras „nerealias“ kombinacijas (atmetus vienodus rezultatus duodančias kombinacijas ir pan.), kiekvienas fragmentas turėtų 14 bitų ($14/64 \cdot 100\% = 21{,}875\%$) perteklinių metaduomenų. Tokio ar didesnio išlošio, glaudinant 64 bitų ilgio atsitiktinius fragmentus, be papildomų optimizacijos ar euristinių priemonių tikėtis neverta.

Kompozicijos metaduomenų kodo optimizacija. Norint sumažinti perteklinių duomenų kiekį, tikslinga kiek įmanoma daugiau jų užkoduoti komponuojančiųjų funkcijų teisingumo lentelėse. Todėl tariame, kad kintamųjų tarpusavio eiliškumas komponuojančioje funkcijoje nesikeis (t.y., teisingumo funkcijos apdorotos taip, kad atskiros komponuojančiosios funkcijos kintamųjų eiliškumas išliks toks pats, kaip ir pradinio fragmento funkcijoje). Tai taip pat leidžia metaduomenyse saugoti ne pilnuosius ašių numerius, o jų pokyčius (kitaip tariant, numerius tarp likusiųjų aukštesnės eilės pradinės funkcijos ašių).

Norint toje pačioje struktūroje įrašyti fragmento ilgį, tikslinga n+1-ąją ašį pažymėti specialiu metakodu (pavyzdžiui, 0). Taip pat galima daryti prielaidą, kad komponuo-

jančiosios funkcijos surikiuotos pagal pirmąją atskirai aprašomą ašį pradinėje funkcijoje. Kitaip tariant, jei pirmoji komponuojančioji funkcija turi ašis x_2,x_3,x_4, tai iš to seka, kad antroji komponuojančioji funkcija neturi ašies x_1, ir atitinkamai neturi bent vienos iš ašių x_2,x_3,x_4. Kaip galima šias reikšmes glaudžiai užkoduoti, parodyta 5 lentelėje.

Pirmoji dalinė funkcija: $f_1(x_2,x_3,x_4)$		
Ašis x_i	Galimas kodų skaičius	Kodas
x_2	$\|x_1 - x_6, 0\| = 7$	2
x_3	$\|x_3 - x_6,0\| = 5$	1
x_4	$\|x_4 - x_6,0\| = 4$	1
0	$\|x_5,x_6,0\| = 3$	0
Iš viso:	$7\cdot5\cdot4\cdot3=420$	$((0\cdot4+1)\cdot5+1)\cdot7+2=44$

Antroji dalinė funkcija: $f_1(x_2,x_3,x_5)$		
Ašis x_i	Galimas kodų skaičius	Kodas
x_2	$\|x_2 - x_6, 0\| = 6$	1
x_3	$\|x_3 - x_6,0\| = 5$	2
x_5	$\|x_5,x_6,0\| = 3$	1
0	$\|x_6,0\| = 2$	0
Iš viso	$6\cdot5\cdot3\cdot2=180$	$((0\cdot3+1)\cdot5+2)\cdot6+1=43$

5 lentelė. Dviejų glaudinių kodavimas.

Kaip matyti iš pateikto pavyzdžio, toks dalinai adaptyvus kodavimas šiuo atveju reikalauja $\log_2(420\cdot180)=17$ bitų. Vadinasi, visas 64 bitų funkcijos glaudinys užims 1+17+7+7=32 bitus. Tačiau šiuo atveju efektyviau būtų galima koduoti tiesiog reikiamas ašis pažymint bitų kauke: [011100] || [*11*10] (žvaigždutėmis pažymėtos negalimos ašys). Toks kodas užima 10 bitų, o viso fragmento glaudinys – 1+10+7+7=25 bitus. Pirmasis kodavimas efektyvesnis, kai sekos išskaidomos labai gerai, antrasis – kai daug neišskaidomų sekų.

Kita vertus, antruoju būdu (bitų kauke) gauname ilgą metakodą tuomet, kai funkcija neskaidoma (o tokių funkcijų bendruoju atveju yra dauguma). Todėl siūlome kombinuotą kodavimo būdą. Kadangi laikoma, kad funkcijos surikiuotos jų ašių kaukės skaitinės vertės mažėjimo tvarka, tai tikimybė, kad pirmoji ašis yra x_1 arba x_2, labai didelė. Taigi tikslinga pirmąjį arba du pirmuosius kaukės bitus koduoti, pavyzdžiui, taip:

00 – seka neskaidoma, toliau koduojama jos teisingumo lentelė;
01 – seka skaidoma: pirmoji skaidinio ašis nėra x_1, toliau ašių x_2– x_n bitų kaukė;
10 – seka skaidoma; pirmoji ašis yra x_1, bet antroji nėra x_2, toliau ašių x_3– x_n kaukė;
11 – seka skaidoma; pirmosios ašys yra x_1 *ir* x_2, toliau kauke koduojamos ašys x_3– x_n.

Toks kodavimas generuoja 2/64·100%=3,125% perteklinių metaduomenų, kuomet seka neišskaidoma (64 bitų fiksuoto ilgio segmentavimo atveju) ir ne daugiau nei 1 bitą perteklinės informacijos, kai seka išskaidoma. Taip pat šiuo būdu galima užrašyti kintamą ilgį dalinių funkcijų. Pavyzdyje pateiktos funkcijos atveju, glaudusis jos kodas sudarytų 1+(2+4)+4+7+7=25 bitus. Tačiau, atsižvelgiant į reikalavimus arba atlikus daugiau tyrimų, šis kodas dar gali būti tobulinamas.

Kodavimo algoritmo sudėtingumo įverčiai. Pateiktasis algoritmas susideda iš šių etapų:

1. Duomenų srauto segmentavimas. Tiriamas tik fiksuoto ilgio segmentavimas, nors (kaip buvo parodyta 3 skyriuje), turint spartų euristinį bazinės (identifikuojančiosios) funkcijos paieškos algoritmą ir išvysčius euristines sekos skaidomumo nustatymo priemones, būtų tikslinga taikyti adaptyvųjį segmentavimo metodą. Tiriamasis metodas yra O(1) sudėtingumo fiksuoto ilgio segmentui.

2. Identifikuojančiosios funkcijos paieška. Identifikuojančioji funkcija randama invertuojant funkcijos reikšmes, sukeičiant ir invertuojant ašis. Kadangi n kintamųjų funkcija gali turėti iki $2 \cdot 2^n \cdot n!$ ekvivalenčiųjų skaidomumo atžvilgiu transformacijų, tai deterministinio pilnojo perrinkimo algoritmo sudėtingumas – $O(2^{n+1} \cdot n!)$. Euristinis algoritmas, aprašytas 4.3 skyrelyje, susideda iš šių etapų:

    a. Inversijos nustatymas. Euristinis algoritmas vektorinei skaičiavimo mašinai realizuojamas kaip operacija **pop**, kurios sudėtingumas – $O(n)$.

    b. Ašių inversijos nustatymas (centroido parinkimas). Pateikiami du algoritmai: spartus „kaimynų“ skaičiaus sumavimo (sudėtingumas $O(2^n \cdot n)$, atmintis $O(2^{2n})$ vektorinei mašinai) ir lėtesnis „matomumo“ algoritmas ($O(2^{2n})$ ir $O(n \cdot 2^n)$).

    c. Ašių eiliškumo nustatymas. Pateikiamas „izoliuotų kriterijų“ algoritmas ($O(kn \cdot \log n)$, $O(1)$; k – kriterijų skaičius).

   Euristinis identifikuojančiosios funkcijos paieškos algoritmas yra atitinkamai $O(2^n \cdot n)$ arba $O(2^{2n})$ priklausomai nuo atskiriems etapams numatytų kriterijų. Gali būti naudojamas kombinuotas metodas: jei sparčiaisiais algoritmais apskaičiuota

identifikuojančioji seka lentelėje nerandama, tai ji perskaičiuojama sudėtingesniu metodu.

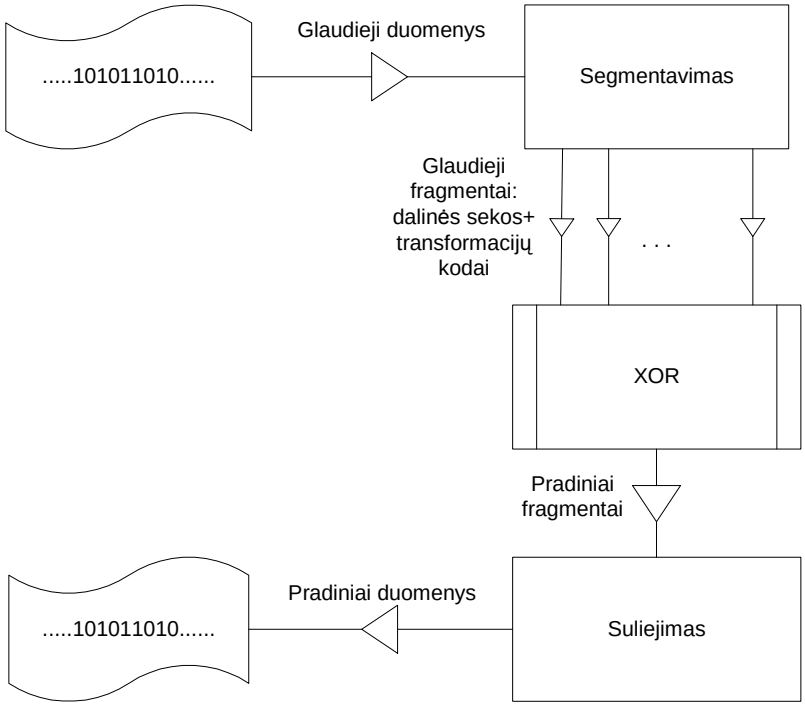
3. Identifikuojančiosios funkcijos paieška lentelėje. Naudojant Solinger DB lenteles, paieška atliekama per O(n) laiką. Lentelės apimtis (nustatyta eksperimentiškai) – apie $2^{2^n - n^{1,5}}$. Kai n=5, duomenų bazės apimtis yra 19,254 MB. Ši duomenų bazės dydžio problema gali būti sprendžiama kintamo ilgio segmentų euristinės analizės metodais.

4. Fragmentų įrašymas. Kiekvienas fragmentas dalinai transformuojamas (sukeičiant kintamuosius ir, jei reikia, invertuojant funkciją) iš bazinės sekos erdvės į pradinės sekos erdvę (O(n) sudėtingumas). Generuojami optimizuoti fragmentų ašių sukeitimo kodai (O(n)). Gautasis kodas suliejamas ir įrašomas į rezultatų srautą.

Apskaičiuotasis sekos glaudinimo algoritmo sudėtingumas: laikas – $O(2^n \cdot n)$ arba $O(2^{2n})$, priklausomai nuo parinktos euristikos, darbinės atminties kiekis – $O(2^{2n})$ ir $O(n \cdot 2^n)$, identifikuojančiųjų funkcijų lentelių (naudojamų tik skaitymui) apimtis – apie $2^{2^n - n^{1,5}}$ (nustatyta eksperimentiškai; 19,254 MB, kai n=5).

Vystant šį metodą toliau, daugiausia dėmesio turėtų būti skiriama euristinei fragmentų analizei (faktorizavimui, naudojant žemesnės eilės funkcijas, ir identifikuojančiųjų funkcijų paieškai).

Dekodavimo algoritmo struktūra. Glaudžiųjų sekų dekodavimo algoritmas analizuoja duomenų srautą, kurį sudaro sekos, faktorizuotos naudojant sumos moduliu 2 operaciją. Kadangi formuojant glaudųjį srautą nebuvo atsižvelgta į papildomas sinchronizavimo galimybes, tai laikome, kad srautas dekoduojamas tik iš anksto žinant pirmojo dekoduojamo fragmento pradžios adresą sraute. Jei vystant kodavimo algoritmą būtų pasiekta, kad suglaudintoji duomenų seka perteklinės informacijos neneštų, tuomet euristiškai ar kitaip aptikti eilinio fragmento pradžios adresą taptų neįmanoma.

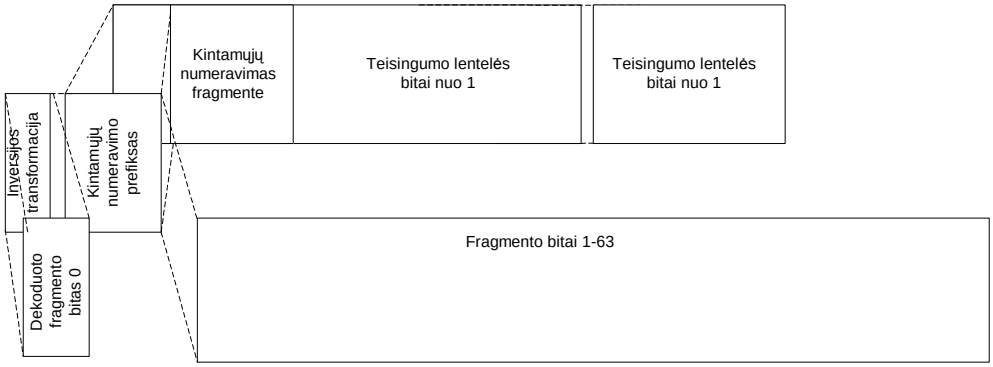


5 pav. Bendrojo sekų dekodavimo algoritmo struktūra

Glaudžiojo duomenų srauto dekodavimas – žymiai paprastesnė operacija nei kodavimas (3, 5 pav.). Todėl pateikiame tik konkrečios realizacijos (kaip aprašyta aukščiau) dekodavimo procesą, nesigilindami į galimas modifikacijas.

Algoritmas išskiria atskirą fragmentą iš glaudžiųjų duomenų srauto, jį analizuodamas nuosekliai, viena kryptimi, skaitydamas po iš anksto žinomą bitų skaičių. Vienu metu saugoma informacija tik apie vieną dekoduojamą fragmentą (jei kitokių reikalavimų nekeliama atskiru atveju). Glaudžiojo srauto struktūra bendruoju atveju pavaizduota 6 pav.

Kiekvienas glaudusis fragmentas pradedamas galutinės funkcijos inversijos bitu, kuris įsimenamas (5 pav.) ir įrašomas į dekoduotųjų duomenų (rezultatų) srautą. Tuomet skaitomas ir iššifruojamas 2 bitų ilgio (kaip aprašyta kodavimo algoritmo aprašyme) kintamųjų numeravimo prefiksas. Jei jis lygus 00, tai iš koduotųjų duomenų srauto į rezultatų srautą perrašomi tolesnieji 63 bitai.



6 pav. Dekoduojamojo fragmento struktūra

Kitais atvejais suformuojamos 6 bitų ilgio kintamųjų (ašių) kaukės ir atitinkamo ilgio dalinių funkcijų teisingumo lentelės. Trumpas naudojamų priemonių aprašymas pateiktas 4.3 skyrelyje.

```
Pirmosios kaukės ir teisingumo lentelės formavimo algoritmas.
Naudojamos procedūros:
  read(n) - įveda n bitų iš duomenų srauto,
  pop(n)  - apskaičiuoja kintamojo n bitų sumą.

Grąžinamas rezultatas:
  mask_1 - pirmosios dalinės funkcijos ašių kaukė.
  tt_1   - pirmosios dalinės funkcijos teisingumo lentelė.

{interpretuojam prefiksą}
elem   := 5;      { kaukių ilgis yra 5+1=6 bitai}
mask_1 := prefix;
prefix >>= 1;

mask_1 ∧= ¬prefix;

elem   -= prefix;

{įvedam likusią kaukės dalį}
mask_1 := mask_1<<elem ∨ read(elem);

{įrašom 1 į teisingumo lentelės priekį}
pop_m  := 1 << pop(mask_1);
tt_1   := 1 << pop_m;

{šioje vietoje įrašyti kitų funkcijų kaukių skaitymą}

{skaitom likusią teisingumo lentelės dalį}
tt_1   ∨= read( pop_m-1 );

{praplečiam tt_1 iki 64 bitų}
kol pop_m<64:
  tt_1  ∨=  tt_1 << pop_m;
  pop_m <<= 1;
.
```

Atlikus transformaciją kaip įrašyta kaukėje, gaunama dalinė funkcija, transformuota į pradinės (nesuglaudintos) sekos erdvę. Analogiškai randama antroji dalinė funkcija. Jų

sumos moduliu 2 inversija (jei inversijos transformacijos kodas lygus 1) lygi pradinei funkcijai prieš glaudinimą.

Dekodavimo algoritmo įverčiai. Aprašytojo algoritmo realizacija vektorinei mašinai yra O($k \cdot n$) sudėtingumo (čia k–fragmentų skaičius duomenų sraute, n–funkcijos eilė; realizuotame algoritme n=6). Naudojamas darbinės atminties O(1) kiekis. Identifikuojančiųjų skaidinių lentelė nenaudojama, todėl tiriamas metodas tenkina asimetriškumo kriterijų. Nors duomenims glaudinti reikia galingos technikos ar net paskirstytųjų išteklių, tačiau jų išskleidimas realizuojamas atominėmis operacijomis, kurios pritaikomos net paprasčiausiems integruotiesiems įrenginiams kaip mobilieji telefonai, kišeniniai kompiuteriai ir pan..

4.2. Spartus ekvivalenčiojo sekų transformavimo metodas

Sekos kaip vektorinės skaičiavimo mašinos registrų duomenys. Kaip parodyta ankstesniuose skyriuose, pagrindinė kliūtis praktiškai taikyti faktorizaciją moduliu 2 dideliems duomenų srautams apdoroti yra metodo algoritmų sudėtingumas. Taigi tikslinga elementariąsias operacijas su duomenų fragmentais realizuoti kiek įmanoma išnaudojant aparatūros technines galimybes. Vienas iš būdų tai atlikti – lygiagretinti skaičiavimus, siekiant, kad elementariosios fragmentų apdorojimo operacijos būtų atominės (atliekamos per O(1) laiką, baigtiniu procesoriaus taktų skaičiumi).

Kadangi plinta asmeniniai kompiuteriai su 64 bitų operacijų rinkiniais, o šiuolaikiniai kompiliatoriai sugeba šias operacijas sparčiai emuliuoti 32 bitų procesorių sistemose, toliau laikysime, kad dvejetainė seka iki 64 bitų (8 baitų) yra atitinkamo ilgio beženklis sveikasis skaičius, su kuriuo aparatūra geba atlikti šias operacijas:

$\neg x$ arba $\bar{x}$	visų bitų inversija
$x \wedge y$ arba $x\&y$	atitinkamų bitų konjunkcija („ir“)
$x \vee y$ arba $x \vert y$	atitinkamų bitų disjunkcija („arba“)
$x \oplus y$	atitinkamų bitų suma moduliu 2 („griežtasis arba“)
$x << y$	beženklis postūmis kairėn (sveikoji daugyba iš 2^y ignoruojant perpildymą)
$x >> y$	beženklis postūmis dešinėn (sveikoji dalyba iš 2^y)
$x <<^r y$	ciklinis postūmis kairėn
$x >>^r y$	ciklinis postūmis dešinėn

Taip pat laikysime, kad apibrėžtos aritmetinės operacijos su sveikaisiais skaičiais be perpildymo (+,–,·,/), aritmetinio lyginimo operacijos (=,<,>), priskyrimo operacija (:=) ir jas atitinkančios pirmojo operando modifikavimo operacijos (+=, –=, ⊕=, <<= ir pan.).

Apibendrinant priimsime, kad aprašytieji algoritmai gali būti realizuoti bet kokio 2^n ilgio registrų procesoriui, kuriame šios operacijos laikytinos atominėmis. Toks tikslinis procesorius neprivalo būti realizuotas aparatiškai. Tai gali būti programinė emuliacija mažesnių registrų procesoriumi, daugiaprocesorinė sistema arba vektorinių skaičiavimų aparatinė įranga.

Atominės elementariųjų ekvivalenčiojo transformavimo operacijų realizacijos. Kuriant programas praktiniams eksperimentams, išvystyti ir realizuoti spartūs ekvivalenčiosios sekų transformacijos metodai bitų sumai moduliu 2 ir bitų sumai nustatyti, taip pat posekiams išrinkti, ir dviems funkcijos kintamiesiems sukeisti. Taip pat realizuotas spartus sekos gretimų bitų praretinimo algoritmas, naudojamas skaičiuojant viršūnių gretimumų ir atstumų masyvus.

Funkcijų šeimos $f_i(x_1, x_2, \ldots, x_i, \ldots x_n)=x_i$ teisingumo lentelės (kintamųjų kaukės) $\tau\tau$ f_i = **mask** *i* apskaičiuojamos naudojant dalybos operaciją:

```
mask_1 := ¬0;
mask_i := mask_1/( 1<<(i-1) +1).
```

Kadangi šiame algoritme naudojama dalyba, apskaičiuojama naudojant *n* atimčių, bei dvi papildomos atimties operacijos, tai reikalingų konstantų masyvas *masks* sugeneruojamas iš anksto, kompiliavimo metu.

Sekos $B=[b_0 \ldots b_i \ldots b_{n-1}]$ bitų suma moduliu 2 **odd** *B* apskaičiuojama pagal O(log *n*) sudėtingumo algoritmą:

```
odd_B := B;
shift := n; {postūmis}
kol (shift >>= 1) > 0, odd_B ⊕= odd_B>>shift;
odd_B ∧= 1.
```

Pagal analogiškos struktūros algoritmą, naudojant kintamųjų kaukes, apskaičiuojama sekos $B=[b_0 \ldots b_i \ldots b_{n-1}]$ bitų suma (užpildymas, angl. *population*) **pop** *B*:

```
pop_B := B - (B >> 1) ∧ MASKS[1];
pop_B := (pop_B & MASKS[2]) + ((pop_B >> 2) ∧ MASKS[2]);

shift := 8;
mask  := 3;
kol shift < n:
     pop_B += pop_B >> shift;
     pop_B ∧= MASKS[mask];

     shift <<= 1;
     mask  +=  1;

pop_B ∧= (n << 1) - 1.
```

Kadangi šioje algoritmo realizacijoje yra perteklinių skaičiavimų, jį tikslinga išskleisti ir optimizuoti konkrečiai architektūrai. Algoritmas, pritaikytas 64 bitų architektūrai, įvykdomas apie 230% sparčiau, tarpiniams duomenims naudojant tik vieną registrą (tai aktualu vektoriniuose skaičiavimuose):

```
pop_B := B - (B >> 1)
pop_B ∧= MASKS[1];
pop_B ∧= pop_B ∧ MASKS[2];
pop_B += ((pop_B >> 2) ∧ MASKS[2]);
pop_B += pop_B >> 4;
pop_B ∧= MASKS[3];

pop_B += pop_B >> 8;
pop_B += pop_B >> 16;
pop_B += pop_B >> 32;
pop_B ∧= 127.
```

Šis algoritmas gerai vektorizuojamas arba lygiagretinamas, todėl, lyginant su įprasta jo realizacija masyvų pagalba, gauti žymiai geresni skaičiavimo laiko įverčiai (6 lent.).

Sekos ilgis *n*	Skaičiavimo įranga	Tiesioginio alg. vykdymo laikas (ns)	Optimizuoto alg. vykdymo laikas (ns)
64	AMD Duron 700 MHz	7,24	2,42
	Intel Celeron M 1.2GHz	4,12	0,74
	Intel Pentium 4 2.6 GHz/HTT	1,37	0,142
128	AMD Duron 700 MHz	12,6	2,63
	Intel Celeron M 1.2GHz	7,4	0,76
	Intel Pentium 4 2.6 GHz/HTT	2,1	0,148
1024	AMD Duron 700 MHz	113,1	3,12
	Intel Celeron M 1.2GHz	83,4	0,82
	Intel Pentium 4 2.6 GHz/HTT	18,6	0,161

6 lentelė. Optimizuotos **pop** funkcijos algoritmo vykdymo laiko priklausomybė nuo atsitiktinės sekos ilgio ir procesoriaus architektūros

Analogiškai optimizuotas sekos viršūnių gretimumų skaičiavimo algoritmas. Naudojant postūmio, konjunkcijos ir sudėties operacijas, per O(*d* log *n*) laiką apskaičiuojama, kiek vienetinių viršūnių (sekos bitų, lygių 1), nutolusių atstumais nuo 1 iki *d*, turi kiekviena vienetinė viršūnė.

Dviejų funkcijos kintamųjų (ašių) sukeitimas realizuotas naudojant išrinkimo pagal ašies kaukę *masks*[*i*] operaciją ir du postūmius:

```
Argumentai: B-pradinė seka, v1 ir v2 – funkcijos kintamųjų eilės
numeriai, log_n – funkcijos matmuo.

Jei v1>v2:
   v1⊕=v2; v2⊕=v1; v1⊕=v2; {jei reikia, sukeičiam indeksus}

v1 := log_n-v1;
v2 := log_n-v2;
mask1 := MASKS[v1];
mask2 := MASKS[v2];

{išrenkam nejudančias sekos dalis}
trans_B_v1_2 := B ∧ ¬(mask1⊕mask2);

{kiek pozicijų pasislinks judančios sekos dalys}
shift := (1 << v1-1) - (1 << v2-1);

{išrenkam, paslenkam ir įstatom judančias dalis}
trans_B_v1_2 ∨= (B ∧ mask1 ∧ ¬mask2) << shift;
trans_B_v1_2 ∨= (B ∧ mask2 ∧ ¬mask1) >> shift.
```

Analogiškai realizuojama kintamojo inversijos operacija. Lyginant su aritmetiškai optimizuotomis matricininėmis bitų perstatymo realizacijomis, šis metodas yra žymiai spartesnis (aritmetiškai neoptimizuotas algoritmas yra O(n^3 log n) sudėtingumo, o ši realizacija O(1) sudėtingumo vektorinei mašinai arba O(n) sudėtingumo programinei vektorizavimo emuliacijai. Be to, jis reikalauja O(n) tarpinės atminties, kai tuo tarpu matricinei realizacijai reikia O($2^{2n \log n}$).

Aptartosios funkcijos iteratyviai naudojamos žemiausiame sukurtojo euristinio sekų ekvivalentiškumo kriterijaus funkcijos lygmenyje, todėl pasiektas išlošis žymiai pagerina pilnosios sekų aibės perrinkimo ir sekų analizės realiuoju laiku galimybes. Tai gali būti ypač aktualu tolesniuose tyrimuose, nagrinėjant ilgesnes sekas ar euristiškai tyrinėjant jų skaidomumą (pavyzdžiui, siekiant realizuoti praradiminį glaudinimą ar dinaminį duomenų srauto fragmentavimą adaptyvaus ilgio posekiais).

Ekvivalenčiojo sekos transformavimo suvedimas į skaliarų rikiavimo uždavinį Ekvivalenčioji sekos transformacija aprašoma kompaktišku kodu, kurį sudaro sekos inversijos (iki 1 bito), ašių inversijų (iki n bitų) ir ašių eiliškumo kodas (iki $1+\log_2 n!$ bitų). Transformavimo algoritmą sudaro atitinkami etapai. Sekos inversija (vektorinių skaičiavimu atveju) yra atominė operacija. Kintamųjų inversija realizuota tiesioginiu O(n) sudėtingumo algoritmu. Tuo tarpu ašių pernumeravimo operaciją tiesioginio algoritmo atveju sudaro O($n^2 \log n$) sudėtingumo matricos sudarymo ir O($n^2 \log n$) sudėtingumo reikšmių išskleidimo operacijos.

Ašių sukeitimo uždavinys, turint O(1) sudėtingumo dviejų ašių sukeitimo algoritmą, gali būti suvestas į masyvo narių perindeksavimo uždavinį (O(n)) su O(n) skaliarine atmintimi:

```
Duota: B – pradinė seka, log_n – sekos ašių skaičius, POS[log_n]
– norimi ašių eilės numeriai.

{įsimenam esamas ašių padėtis masyve asiu_nr}
Su visais i=(1,n): asiu_nr[i] := i;

trans_B := B;
i := log_n;

kol i>1:
   j := POS[i];
   mask_i := MASKS[i];

   {sukeičiam dvi ašis}
   trans_B := trans_B_v1_2( trans_B, j, i, log_n);
   asiu_nr[j] := asiu_nr[i];  {įsimenam naują ašies padėtį}

   i -= 1;
.
```

Testuotasis programinis kodas dar papildomai optimizuotas: ištiesintas kreipinys į funkciją *trans*(B, v_1, v_2, *log_n*) ir minimizuotas kreipinių į masyvus skaičius.

4.3. Euristinis identifikuojančiosios sekos parinkimo algoritmas

Kaip minėta 3 skyriuje, žinomas deterministinis identifikuojančiosios sekos parinkimo algoritmas blogiausiu atveju pilnai perrenka visą ekvivalenčiųjų funkcijų poaibį. Kadangi atskirais atvejais tokių funkcijų viename poaibyje gali pasitaikyti iki $2^{n+1} \cdot n!$ (pagal funkcijų ekvivalentumo apibrėžimus), tai toks algoritmas laikytinas godžiuoju ir netaikytinas praktikoje. Tuo tikslu sukurtas euristinis identifikuojančiosios sekos paieškos algoritmas, negarantuojantis

vienareikšmio identifikuojančiosios funkcijos rezultato viename ekvivalenčiųjų funkcijų poaibyje. Tačiau atskiros *pseudoidentifikuojančiosios* funkcijos šiam algoritmui naudojamų kriterijų atžvilgiu priklauso atskiriems, uždariems poaibiams.

Visas pseudoidentifikuojančiosios funkcijos paieškos algoritmas susideda iš trijų nepriklausomų etapų. Kiekvienu etapu siekiama euristiškai priartėti prie leksikografiškai (teisingumo lentelės skaitiniu ekvivalentu) didžiausios poaibį identifikuojančios funkcijos.

Sekos inversijos etapas. Parinkta paprasta euristika, negarantuojanti artėjimo į maksimaliąją ekvivalenčiąją funkciją. Laikoma, kad „geresnė“ funkcija yra ta, kurios **pop** $\tau\tau f$ yra didesnis. Kitaip tariant, jei **pop** $\tau\tau^{<n>}<n/2$, seka invertuojama. Jei **pop** $\tau\tau f^{<n>}=n/2$, seka neinvertuojama, kadangi jų skaidiniai, išskyrus vieną komponuojančiąja funkciją, sutampa.

Kintamųjų inversijos etapas. Kintamųjų inversijos operacija, maksimizuojant sekos skaitinį ekvivalentą, iš esmės lemia, kuris funkcijos taškas perkeliamas į koordinačių pradžią(teisingumo lentelėje tampa nuliniu bitu). Tikslas – rasti tokį tašką, nuo kurio pradedant, laisvai parinkus ašių numeravimą, būtų galima rasti didžiausią vienetinių bitų seką. Toliau tokį funkcijos tašką (sekos bitą) vadinsime *centroidu.*

Sukurtas ir išbandytas „kaimynų skaičiavimo“ algoritmas, paremtas bitų aritmetika ir vektoriniais skaičiavimais. Jo bendra struktūra:

1. Nukopijuojame invertuotą pradinę seką kaip bitų kaukę *necentroidai.* Jei **pop** *necentroidai*=0, baigiame: seka yra identifikuojančioji. Šią sąlygą galima optimizuoti viena palyginimo operacija: *necentroidai*=0.

2. Nukopijuojame invertuotą pradinę seką kaip masyvą *nekaimynai*. Nustatom „kaimyniškumą“ *dist*=1.

3. Kartojame, kol bitų kaukėje liks vienas *necentroidas* arba *dist* pasieks 2^n. Patikrinti, ar bitų kaukėje yra lygiai vienas bitas, galima, atlikus palyginimą *(necentroidai*—1) ∧ *necentroidai*=0.

    a. Praplečiam masyvą *nekaimynai,* kad jame ties kiekvienu sekos bitu tilptų jo kaimyninių vienetinių bitų suma: *dist* kartų tarp bet kurių dviejų bitų įterpiame po tuščią bitą (O(*dist*·*n*) sudėtingumas).

    b. Kiekvienai ašiai: atliekam masyvo nekaimynai ciklinį postūmį per $2\cdot dist-1$, $4\cdot dist-2$, ..., $2^n\cdot dist-2^n-1$ bitų ir sudedam su *nekaimynai*. Masyve *nekaimynai* turime $\log_2 dist$ keliais pasiekiamų nekaimynų skaičių.

c. Padvigubinam: *dist*:=*dist* << 1.

d. Pašalinam *necentroidus*: $O(2^n)$ sudėtingumo ciklu randam daugiausiai *nekaimynų* turinčius *necentroidus*. Likusius pašalinam.

4. *necentroidų* kaukėje likus vieninteliam bitui, centroidas yra tas bitas, kurio koordinatės yra priešingos *necentroidui*. Centroido koordinatės ir yra ieškomasis kintamųjų inversijos kodas.

Taip pat išbandytas sudėtingesnis algoritmas, kurio esmė – ne skaičiuoti „kaimynus“, bet kaupti žinomų „tolimų kaimynų“ tiesioginius kaimynus bitų kaukėje – 2^{2n} bitų dydžio matricoje. Šio algoritmo sudėtingumas ir naudojama atmintis yra $O(2^{2n})$ ir $O(2^{2n})$. Tačiau jų patikimumas taip pat skiriasi: n=5 atveju, pirmasis algoritmas netiksliai nustatė 2,43% centroidų, kai tuo tarpu antrasis – tik 0,09%.

Kintamųjų sukeitimo etapas. Nustačius sekos ir kintamųjų inversijos kodus, kintamųjų sukeitimas gali būti suvedamas į kintamųjų rikiavimo uždavinį. Pagrindinė problema – nustatyti ašių prioriteto kriterijus. Kai n<5, tinka „sukeitimo kriterijus“, kuris įvertina, padidėja, ar sumažėja sekos reikšmė sukeitus du kintamuosius. Jei n ne didesnis nei 4, likusių kintamųjų padėtys neturi įtakos tokiam bet kurių dviejų ašių palyginimui. Tačiau kai n=5, kintamuosius tenka įvertinti po tris, kai n=6 – po 4 ir t.t. Tai labai apsunkina kriterijaus formuluotę.

Tačiau identifikuojančioji funkcija nebūtinai turi atitikti akivaizdžiai suvokiamus kriterijaus maksimizavimo požymius. Kaip viena iš galimų išeičių, pasirinkti euristiniai kriterijai, leidžiantys suteikti kintamiesiems prioritetus, neatsižvelgiant į jų sąveiką su kitais kintamaisiais.

Tokiais kriterijais gali būti bet kokia dvejetainė transformacija, kuriai įtakos turi tik vienetinių funkcijos taškų išsidėstymas priešingose ašies pusėse bei elementarūs skaitiniai įverčiai, nepriklausantys nuo kitų ašių išsidėstymo. Tokie įverčiai leidžia ženkliai sumažinti analizuojamų transformacijų skaičių. Pavyzdžiui, n=5 atveju, pakanka pagal kurį nors kriterijų sumažinti ar pakelti bet kurio kintamojo prioritetą, ir likusi uždavinio dalis gali būti išspręsta kaip rikiavimo pagal poros „sukeitimo kriterijų“ uždavinys.

Tokie kriterijai yra:

– Loginiai asimetrijos kriterijai. Tarkime, kad yra loginė funkcija *k:* $\mathbf{B}^2 \to \mathbf{B}$. Kintamasis x_i tuo „geresnis“, kuo didesnė suma $\sum_{\vec{x} \in X \setminus x_i} k(f(\{\vec{x},0\}), f(\{\vec{x},1\}))$. k gali būti bet kuri iš 16 galimų dvejetainių funkcijų.

– Skaitiniai asimetrijos kriterijai. Tokio kriterijaus pavyzdys – funkcija

$$k(x_i) = |\, \mathbf{pop}(\tau\tau f(\{X \setminus x_i, 0\})) - \mathbf{pop}(\tau\tau f(\{X \setminus x_i, 1\}))\, | \,.$$

Kokius konkrečius kriterijus pasirinkti, priklauso nuo daugybės sąlygų. Tačiau bandymai parodė, kad su n=5, įvedus vieną skaitinį asimetrijos kriterijų, kaip pateikta pavyzdyje aukščiau, ir tris loginius asimetrijos kriterijus („x_i“, „suma moduliu 2“, „implikacija“), netiksliai aptiktų identifikuojančiųjų funkcijų sumažėjo nuo 1,45% iki 0,021%.

5. EKSPERIMENTINIS REZULTATŲ ĮVERTINIMAS

Tyrimo sritis yra palyginti nauja, todėl nėra sukaupta pakankamai duomenų ar metodikos glaudinimo, naudojant sumos moduliu 2, tyrimams. Darbo eigoje, teko realizuoti keletą pagalbinių priemonių ir atlikti keletą eksperimentų, iš kurių rezultatų sprendėme apie vėlesnių darbo etapų kryptis ir perspektyvas.

Rankiniu būdu apskaičiuotos visos identifikuojančiosios funkcijos iki n=3 ir dalis funkcijų kai n=4. Realizuotas pilnojo perrinkimo algoritmas, kurį vykdant patikslintos apskaičiuotosios reikšmės, ir pilnai suformuota identifikuojančiųjų funkcijų lentelė, kai n=4.

Vėliau buvo bandomi skirtingi euristiniai identifikuojančiųjų funkcijų paieškos algoritmai. Jų rezultatai palyginti su gautaisiais perrinkimo būdu. Ištaisius algoritmų netikslumus, apskaičiuota pseudoidentifikuojančiųjų funkcijų lentelė, kai n=5, kurios vėliau buvo peržiūrėtos pilnojo perrinkimo algoritmu. Gautieji šių bandymų rezultatai pateikti 7 lentelėje.

Kintamųjų skaičius *n*	Funkcijų skaičius 2	Perrinktų funkcijų dalis, %	Euristinis algoritmas		Perrinkimas	
			Skaičiavimų laikas	Rasta pseudo *id* funkcijų	Skaičiavimų laikas	Rasta *id* sekų
2	16	100	30 ns	4	12 ms	4
3	256	100	463 ns	14	195 ms	14
4	65536	100	2811ms.	304	845 sek	280
5	4,29-1 0^9	100	324 min. 12sek.	2110703	—	17844826

7 lent. Sekų perrinkimas

Kaip matome iš rezultatų, vienos funkcijos identifikuojančiai ai funkcijai rasti euristiniu algoritmu sugaištamos kelios ns. Bazinių funkcijų lentelė taip pat nėra didelė.

Kadangi nėra sudaryta 64 bitų ilgio optimaliųjų skaidinių lentelės, testai buvo atliekami naudojant „padirbtus“ optimaliuosius skaidinius su atsitiktinai užpildytais skirtingais skaidiniais dviem dalinėmis funkcijomis. Ištestuoti įvairūs kodavimo/dekodavimo variantai. Koduojant buvo randama identifikuojančioji funkcija ir kreipiamasi į identifikuojančiųjų sekų lentelę, kuri visa buvo patalpinta darbinėje atmintyje, gautieji fragmentai transformuojami pagal transformavimo kodą ir įrašomi glaudžiąja forma į duomenų srautą. Srautai buvo imituojami statiniais masyvais sistemos darbinėje atmintyje. Kiekvienas testas

buvo vykdomas ciklišk ai, po 10 min. Nustatyta vidutinė jų vykdymo trukmė pateikta 8 lentelėje.

Srauto ilgis, KB	Skaičiavimo įranga	Vidutinė kodavimo trukmė, sek	Vidutinė dekodavimo trukmė, sek
64	AMD Duron 700 MHz	0,243	0,0283
	Intel Celeron M 1.2GHz	0,094	0,0124
	Intel Pentium 4 2.6 GHz/HTT	0,021	0,0075
1024	AMD Duron 700 MHz	3,720	0,6334
	Intel Celeron M 1.2GHz	1,423	0,2612
	Intel Pentium 4 2.6 GHz/HTT	0,215	0,0143
10240	AMD Duron 700 MHz	35,114	6,6207
	Intel Celeron M 1.2GHz	12,412	2,5940
	Intel Pentium 4 2.6 GHz/HTT	1,913	0,2133

8 lentelė. Sekų kodavimoir dekodavimo laikai skirtingo galingumo kompiuteriais

Kaip matome iš rezultatų, kodavimo ir dekodavimo trukmė beveik tiesiškai priklauso nuo apdorojamų srautų ilgio. Nuokrypius galima būtų paaiškinti skirtingų architektūrų ypatumais apdorojant didelius duomenų kiekius (įtakos galėjo turėti kešavimas, atmintinės fragmentacija, skirtingų operacijų sistemų laiko skirstymo procesams strategija).

Negana to, gautieji laiko įverčiai rodo, kad operacijos atliekamos labai sparčiai, taigi pasirinktoji strategija pasiteisino.

Paskutinio testavimo metu visi pradiniai duomenų srautai nepasikeitė juos užkodavus ir dekodavus, o su turėtaisiais parinktais duomenimis visų jų vidutinis išlošis užkodavus glaudžiuoju formatu, buvo apie 3%. Šis skaičius neatspindi realios situacijos, kadangi generuojant atsitiktinių skaidinių lentelę nebuvo atsižvelgta į realių skaidinių praktinį pasiskirstymą.

6. IŠVADOS

Darbe aprašytas naujas metodas diskrečiųjų duomenų srautų fragmentams glaudžiai koduoti, naudojant dvejetainių sekų dekompoziciją suma moduliu 2. Parodyta, kad jo realizacijoje, net kai tikslingai neatsižvelgiama į koduojamų duomenų šaltinio entropinius ar statistinius parametrus, įmanomas duomenų glaudinimo efektas.

Sprendžiamas bendrasis uždavinys yra *NP* klasės tiek laiko, tiek atminties sąnaudų atveju, o jo glaudinimo efektas nežymus. Siekiant sukurti prielaidas spartesniems euristiniams metodams vystyti, ir priemones metodą adaptuoti konkretaus taikymo duomenims, suformuluoti elementarūs sekų dekompozicijos požymiai, nepriklausantys nuo duomenų šaltinio entropinių ar statistinių savybių. Jais remiantis, įvesta ekvivalentumo skaidomumo atžvilgiu sąvoka ir apibrėžtos elementarios transformacijos, nekeičiančios sekos skaidomumo savybių. Remiantis šiomis transformacijomis, galima atlikti konkretaus taikymo sekų analizę, pavyzdžiui, parinkti „gerai“ glaudinamas sekas, su leidžiama paklaida ar informacijos pertekliškumu artimas konkretaus taikymo neskaidomoms ar „prastai“ skaidomoms sekoms, taip pat – identifikuoti neskaidomus ar „prastai“ skaidomus posekius.

Atskirais atvejais apskaičiuotas fiksuoto ilgio ekvivalenčiųjų sekų uždarų poaibių skaidomumo atžvilgiu skaičius. Eksperimentiškai parodyta, kad tokių poaibių skaičius yra mažesnis nei galimų fiksuoto ilgio sekų skaičius, ir auga žymiai lėčiau nei sekų skaičius, didinant fiksuotą sekų ilgį. Tuo remiantis, galima sudaryti išankstines „geriausių“ skaidinių lenteles trumpoms sekoms (eksperimentiškai parodyta, kad tokia pilnoji lentelė galėtų būti taikoma sekoms iki 32 bitų ilgio analizuoti įprastos konfigūracijos sistemose, ir sekoms iki 64 bitų – serverinėse ar korporatyvinėse sistemose). Turint tokias lenteles, sekos optimaliojo skaidymo uždavinys suvedamas į poaibio identifikavimo uždavinį.

Kadangi poaibio identifikavimo uždavinys yra *NP* klasės laiko atžvilgiu, išvystytos kelios strategijos euristiniam ekvivalenčiųjų sekų grupavimui. Eksperimentiškai parodyta, kad per polinominį laiką sekas galima sugrupuoti į poaibius, kurių skaičius nežymiai didesnis (t.y., tos pačios eilės) kaip ir tikslus ekvivalenčiųjų sekų poaibių skaičius. Esant reikalui, gautųjų poaibių identifikuojančiosios sekos gali būti papildomai peržiūrėtos tiksliu algoritmu. Taip per laiką, artimą polinominiam, poaibio identifikavimo uždavinys (tuo pačiu – ir sekos optimaliojo glaudinio radimo uždavinys) išsprendžiamas norimu tikslumu iš anksto ištirtam ekvivalenčiųjų sekų poaibių rinkiniui.

Rezultatai rodo, kad įmanoma sėkmingai analizuoti ir glaudinti duomenų sekas, kurioms netinka klasikinė šaltinio entropija paremta analizė. Tolesni darbai galėtų remtis bendruoju darbe aprašytu metodu ir pateiktomis sekų analizės bei optimizavimo priemonėmis. Jie galėtų būti taikomi praktikoje, pavyzdžiui: 1 bito rastrinių duomenų praradiminiam glaudinimui; kaip papildomas glaudinimo etapas esamuose blokinio glaudinimo algoritmuose (kaip JPEG); anomalijų paieškai duomenų srautuose, kuriems analizuoti netinka entropiniai ir aritmetiniai metodai.

LITERATŪRA

1. Valantinas J. *Diskrečiosios transformacijos: mokymo priemonė.* Kaunas, 1992.
2. Lassaigne R., Rougemont M. *Logika ir algoritmu sudėtingumas.* Žara, V, 1999.
3. Reingold M., Nievergelt J., Deo N. *Combinatorial Algorithms. Theory and Practice.* Prentice-Hall, Inc., 1977, ISBN 0-13-152447-X
4. Žilinskas A. *Matematinis programavimas.* VDU leidykla, Kaunas, 2000. ISBN 9986-501-51-2
5. Warren Henry S., Jr. *Hacker's Delight.* Addison-Wesley, 1995, ISBN 0-201-91465-4
6. Donald E. Knuth *The Art Of Computer Programming Pre-Fascicle 2B. Generating All Permutations.* Addison-Wesley, 2003, Internete: http://www-cs-faculty.stanford.edu/~knuth/fasc2b.ps.gz
7. Welstead S. *Fractal and Wavelet Image Compression Techniques.* SPIE, New York, 2003. ISBN 5-89392-079-1
8. Blelloch G.E. *Introduction to Data Compression.* Carnegie Mellon University, 2001. Internete: http://www-2.cs.cmu.edu/afs/cs/project/pscico-guyb/realworld/www/compression.pdf
9. Shannon C. E. *Reprinted with corrections from The Bell System TechnicalJournal,* Vol. 27, pp. 379-423, 623-656, July, October, 1948. Internete: http://cm.bell-labs.com/cm/ms/what/shannonday/shannonl948.pdf
10. Born, G. *Referenzhandbuch Dateiformate. Grafik, Text, Datenbanken, Tabellenkalkulation.* Addison-Wesley, 1995, 472 p. ISBN 5-87685-023-3
11. *Encyclopedia of graphics file formats.* Internete: http://netghost.narod.ru/gff/graphics/book/index.htm
12. Storer, James A., *Data Compression: Methods and Theory,* Computer Science Press, Rockville, MD, 1988. ASIN: 0716781565
13. Nelson M., *Arithmetic Coding + Statistical Modeling = Data Compression Part 1 – Arithmetic Coding,* Dr. Dobb's Journal February, 1991. Internete: http://dogma.net/markn/articles/arith/partl.htm