KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Jevgenij Makanin

VIENAS IŠ ALGORITMINĖS KOMPOZICIJOS BŪDŲ

Magistro darbas

Vadovas

dr. M. Kavaliauskas

KAUNAS, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

TVIRTINU
doc. dr. N. Listopadskis
2013 06 07

VIENAS IŠ ALGORITMINĖS KOMPOZICIJOS BŪDŲ

Taikomosios matematikos magistro baigiamasis darbas

Vadovas
dr. M. Kavaliauskas
2013 06 07

Recenzentas
Dr. R. Alzbutas
2013 06 07

Atliko
FMMM0 gr. stud.
J. Makanin
2013 06 07

KAUNAS, 2013

KVALIFIKACINĖ KOMISIJA

Pirmininkas: Leonas Saulis, profesorius (VGTU)

Sekretorius: Eimutis Valakevičius, docentas (KTU)

Nariai: Jonas Valantinas, profesorius (KTU)

Vytautas Janilionis, docentas (KTU)

Vidmantas Povilas Pekarskas, profesorius (KTU)

Rimantas Rudzkis, habil. dr., vyriausiasis analitikas (DnB NORD Bankas)

Zenonas Navickas, profesorius (KTU)

Arūnas Barauskas, dr., vice-prezidentas projektams (UAB „Baltic Amadeus“)

Santrauka

Darbe nagrinėjamas muzikinio kūrinio struktūros generavimo metodo sudarymas, tuo tikslu aprašomos pagrindinės kūrinio sudedamosios dalys, sudaroma schema, nurodanti, kurios dalys ir kaip jos yra įtraukiamos į bendrą kūrinį, kaip jos yra tarpusavyje susijusios, ir kaip galima įtakoti jų charakteristikas (t.y. bandoma sudaryti schemą, generuojančią kūrinio "griaučius", ant kurių galima uždėti garsus ir gauti kūrinį). Schemos tikslas yra suteikti galimybę paprastai naudoti natų generavimo metodus ir turėti galimybę juos išbandyti, nesijaudinant dėl struktūros. Pasirinkti keli parametrai struktūros charakteristikų išreiškimui, kurių pagalba galima gauti platų galimų struktūrų pasirinkimą. Darbe nenagrinėjami natų generavimo metodai ar harmonizacijos metodai, bet norima sudaryti "platformą" patogiam šių metodų panaudojimui.

Kai kurių schemos dalių realizavimui nagrinėjamas skaičių skaidymo uždavinys, taip pat nagrinėjamas skaidinių skaičiaus apskaičiavimas. Sudaromi keli skaidinių sudarymo būdai - visų skaidinių sudarymas ir atsitiktinių skaidinių sudarymas. Išnagrinėjamos metodų pagalba gaunamų skaidinių gavimo tikimybės. Sudaromi rekurentiniai sąryšiai skaičiaus skaidinių statistinių charakteristikų gavimui, gaunami skaidinių skirstinių pagal didžiausią ir mažiausią dalį sudarymo metodai. Ištiriamas siūlomų metodų sudėtingumas laiko prasme ir realizacijos būdai.

TURINYS

PAVEIKSLŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

ĮVADAS

Norima sudaryti kompozicijos - kūrinio struktūros generatorių.

Algoritminės kompozicijos knygose [1],[2] siūlomi metodai generuoja vieno instrumento natas arba generuoja neatsižvelgiant į jokią struktūrą, tai labiau panašu į natų parinkimą, arba judėjimų sudarymus. Metodai nepasižymi struktūros sudarymu, užbaigtumu, kokio nors vyksmo išreiškimu. T.y. metodai pradeda nuo natų link kūrinio.

Todėl kompozicijos struktūros sudarymui siūlomas - nagrinėjamas metodas, kuris gali naudoti judėjimų ar vyksmų generavimo algoritmus (iš knygų arba ne iš knygų), struktūra pasižyminčio kūrinio sudarymui. T.y. norima pradėti nuo struktūros (skeleto) ir eiti link natų.

Norima sudaryti schemą, į kurią "pajungiant" natų generavimo metodus gali būti generuojami muzikiniai kūriniai. Pavyzdžiui, sugeneruojant pasirinktu metodu (nors ir visiškai atsitiktinai) kakią nors kompozicijos atkarpą, ją galima keisti, kartoti, modifikuoti ir tai apibendrinti į didesnes dalis, kurios ir sudarytų struktūrą.

Ar metodas netampa "imitaciniu" dėl to, kad suteikiama struktūra? Tikimasi, kad ne, nes struktūra priklauso nuo parametrų ir atitinkamiems parametrams galima padaryti visiškai atsitiktinę struktūrą. Parametrų pagalba bandoma "susiaurinti" galimybes norimos struktūros gavimui. Be to, ar sugeneruotas kūrinys imituoja ką nors ar ne, priklauso labiau nuo natų generavimo metodų (nes atpažįstami motyvai, bet ne bendra struktūra), kurie nenagrinėjami. Vienoda struktūra gali apibūdinti skirtingus kūrinius.

Kūrinio struktūros generavimo darbų daug nėra, yra kelios algoritminės kompozicijos knygos, bet kaip paminėta, jose siūlomi metodai nepasižymi struktūros sudarymu. Matytos kompozicijos knygos, nagrinėjančios struktūrą, kompozicijos automatizavimo nenagrinėja.

Darbe apibrėžtos kelios kūrinio sudedamos dalys, kurios, manoma, yra esminės. Įvedama bendro intensyvumo sąvoka, instrumentų intensyvumo sąvoka, kūrinio vyksmų, judėjimų ir taktų sąvokos. Sudaroma schema, parodanti įvestu dalių vaidmenį ir ryšį tarpusavyje. Pateikiamas intensyvumo sudarymo metodo pavyzdys. Išnagrinėjamas instrumentų intensyvumo sudarymas ir priklausomybė nuo parametrų. Sudaromi skaičių skaidymo algoritmai, ir nagrinėjami galimi skaidinių sudarymo būdai, skaidinių charakteristikos ir operacijos su skaidiniais. Pateikiami būdai vyksmų sudarymui, judėjimų ir taktų generavimui.

1 TEORINĖ DALIS

1.1 Muzikinis kūrinys

Muzikinio kūrinio esminės sudedamosios dalys [4] yra :

Ritmas - vyksmas, pasižymintis reguliaria, dėsninga svaresnių ir silpnesnių įvykių seka.
- ritmas yra tai, pagal ką (atitinkamai ko) galimą patogiai judėti.

Melodija - įvairaus dažnio, trukmės ir garsumo muzikinių garsų ir jų intervalinių santykių visuma, išreiškianti muzikinę mintį.
- seka muzikinių garsų, kuriuos klausytojas suvokia kaip vieningą "objektą" (esybę).
- dainiga garsų seka.

Bosas - žemo dažnio garsai, užpildantys ir jungiantys muzikinę "erdvę".

Harmonija - kelių natų skambėjimas vienu metu (kelių vienalaikių natų naudojimas).
- santykis, tvarka, sąryšis tarp natų.

Tai yra "horizontalios" kūrinio sudedamosios dalys, nes visos išskirtos dalys vyksta (esa) tuo pačiu laiku, visą laiko intervalą. Taip pat galima paminėti tembrą [5] :

Tembras - garso "spalva", "kokybė", arba spektras ir jo kitimas laike.

Kita esminė kūrinio dalis, esanti ne tiek jo sudedamoji dalis, kiek kūrinio apibūdinimas, yra jo forma :

Forma - kūrinio (kompozicijos) struktūra (planas), nusakanti (atspindinti), kaip kūrinio dalys, vyksmai ir kitos muzikinio kūrinio sudedamosios sudaro vieningą kūrinį.

Kūrinio formą, struktūrą galima nagrinėti skirtinguose lygiuose : atskiros dalys - vyksmai, kurie sudaro visą kūrinį; vyksmų struktūra, sudaryta iš "judėjimų", atspindinti vyksmo charakterį - kilimą, judėjimą žemyn, pastovumą, pasikartojamumą; skirtingų instrumentų vyksmai ir jų ryšys tarpusavyje, ir t.t.

1.2 Muzikinio kūrinio struktūros sudarymas

Šiame skyriuje aprašoma, kaip buvo pasirinkta sudaryti kūrinio struktūrą.

Kompozicija išskaidyta į atskiras lygiagrečias laiko atžvilgiu dalis ("horizontalias" dalis), kurios atspindi muzikos sudedamąsias dalis. Tai gali būti *instrumentai* (mušamieji, klavišiniai, styginiai ir t.t.), *instrumentų charakteristikos* (garsumas, dažnis, efektai ar kt.), *kūrinio charakteristikos* (bendras intensyvumas, vyksmų (dalių) pasikeitimas). Gali būti bendros ar esminės sudedamosios dalys, tokios kaip ritmas, melodija, bosas ir harmonija. Galima analogija yra įmonėje dirbantys žmonės, kiekvienas atliekantis savo darbus, kurie gali būti susiję tarpusavyje arba nesusiję, lygiagretūs laike ir turintys planą. Planas yra vertikali charakteristika, t.y. atspindi darbo struktūrą.

Taip pat kompozicija ar jos ("horizontalios") dalys yra išskaidyti į atskiras dalis laiko atžvilgiu - blokus, kurie gali atspindėti atskiras kūrinio dalis - vyksmus, judėjimus, taktus ("vertikalios" dalys, laiko intervalai). Analogija yra su darbo planu ir skirtingais plano lygiais, t.y. "dienos mastu, savaitės, sezono arba metų mastu" arba "savaitės mastu, mėnesio, sezono, metų mastu".

***Intensyvumas* (bendras)**

Kompozicijos generavimui iš pradžių sudaromas jos bendras "vaizdas", t.y. sudaromas bendras *intensyvumas*. *Intensyvumas* yra laiko funkcija, vaizduojanti laiko intervalą į realiųjų skaičių intervalą [0,1], t.y. $[0,T] \rightarrow [0,1]$. *Intensyvumas* gali atspindėti ritmo "tankį" (kiek natų yra laiko intervale), garsumą, natų aukštį ar staigumą, "balsų" skaičių (harmoniją) ir t.t. (intensyvumą galima interpretuoti skirtingai).

Intensyvumas yra bendras visam kūriniui tuo, kad kaip jis bebūtų interpretuojamas, jis taikomas visiems kūrinio metodams, kurie priklauso nuo jo. Į intensyvumą galima žiūrėti kaip į automato įeinamus duomenis (duomenis, valdančius "mechanizmą", sistemą, generuojančią kūrinį).

Intensyvumas pagrinde padeda išgauti bendrą judėjimo tendenciją ar tendencijas, kas tuo pačiu gali sudaryti atskiras kūrinio dalis.

Ką reiškia ar išreiškia intensyvumas yra labiau susitarimo klausimas, kaip ir klausimas "kas yra muzika ?", todėl *intensyvumą* galima perinterpretuoti, esant kitiems tikslams ar kitaip išreiškiant ar suprantant esamus tikslus.

Instrumentų intensyvumas

Toliau kūrinys skaidomas į *instrumentus* (instrumentai nebūtinai yra muzikos instrumentai, tai gali būti "balsų" skaičius, instrumentų skaičius, efektai ir t.t.,- bendru atveju galima interpretuoti skirtingai, tai galima laikyti "horizontalios" kūrinio charakteristikos atvaizdavimu). Visiems *instrumentams* galioja vienodas *intensyvumas*, bet kiekvienam instrumentui suteikiamas *individualus intensyvumas,* kuris apibūdinamas *sklaida, kintamumu, kartotinumu ir sinchroniškumu*.

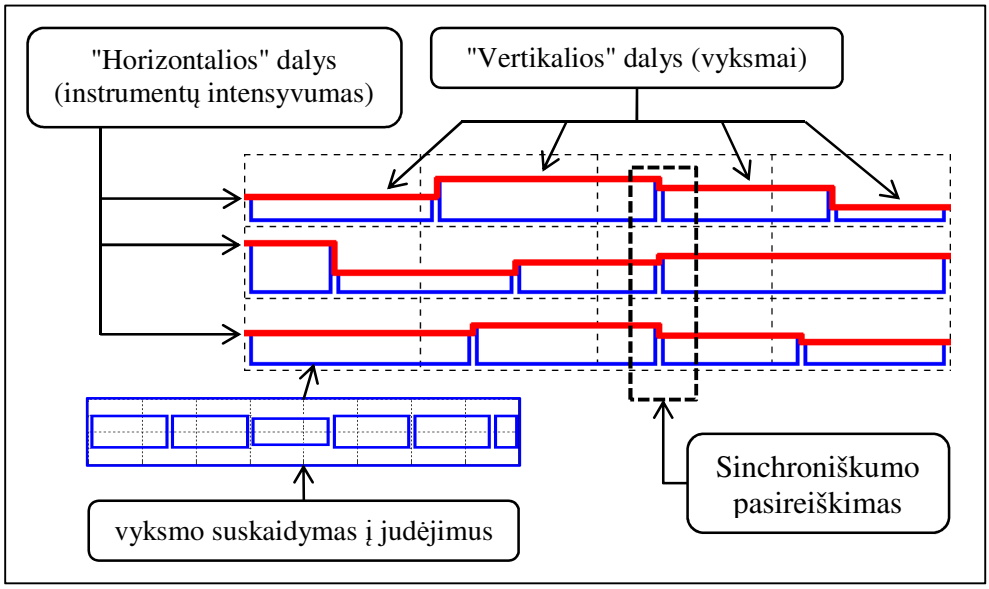
Sklaida - atspindi individualaus intensyvumo galimą kitimo diapazoną,

Kintamumas - atspindi instrumento individualaus intensyvumo pasikeitimų skaičių (instrumento "dalyvavimo" pasikeitimus ar instrumento vyksmus).

Kartotinumas - atspindi pasikartojančią "manierą", atskirų dalių pasikartojimą.

Sinchroniškumas - atspindi instrumentų sinchroniškumą, t.y. jų vyksmų sutapimą laiko prasme (individualių intensyvumų vienalaikius pasikeitimus).

Skirtingų instrumentų intensyvumai susiejami sinchroniškumo rodikliu, taip pat jie turi bendrą *intensyvumą,* kas irgi kažkuria prasme sieja juos. (Supratimo pagerinimui pav. 1.2.1)



pav. 1.1 Kūrinio struktūra

Kūrinio blokai - vyksmai

Kūrinio blokai - vyksmai - tai kūrinio dalys ("vertikalios"), kurios yra suvokiamos kaip vieningos dalys, jungiančios "judėjimus" į vyksmus (kaip savaitės ar mėnesiai sudaro sezoną), išreiškiančios kokį nors kitimą ar jo nebuvimą ir pasižyminčios vieningumu (gali būti "judėjimo" vystymas, modifikavimas arba jo atkartojimas, arba atkartojimas ir keitimas). Kūrinio vyksmai sudaromi instrumentų intensyvumo pagalba ir yra skirtingi skirtingiems instrumentams, bet gali būti sinchronizuoti sinchroniškumo rodiklio pagalba. Vyksmas yra kaip sezonas palyginus su metų laikotarpio planu, vyksmai sudaro kūrinį kaip sezonai metus. (Supratimo pagerinimui pav. 1.2.1)

Judėjimai vyksmuose

Judėjimai - tai *vyksmų* dalys, kurių pagalba galima sudaryti vyksmus, pasižyminčius kartotinumu (t.y. judėjimo atkartojimas), vystymu ar kitimu. Judėjimai sudaromi iš taktų. Judėjimas yra kaip savaitė pagal analogiją su metų laikotarpio planu, judėjimai sudaro vyksmą - sezoną pagal analogiją.

Taktai

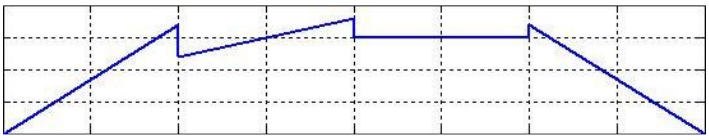
Taktai - tai mažiausias struktūrinis vienetas (pagal analogiją su savaite ir sezonu tai *diena*). Taktai sudaryti iš natų. Jie gali būti skirtingo dydžio ir skirtingo ritmo. Ritmo tankis išreiškiamas natų skaičiumi takte; kuo daugiau natų takte, tuo jos (vidutiniškai) trumpesnės, nes takto ilgis vienodas.

Apibendrinant

Taktai (maži blokeliai, nusakantys tam tikrą mažą motyvą (dieną) kuris gali kartotis, kisti, ar skirtis) organizuojami į judėjimus - didesnes dalis (savaites), judėjimai taip pat gali kartotis, kisti ar būti skirtingi, ir yra organizuojami į vyksmus (mėnesis ar sezonas), vyksmai sudaro kūrinį (sezonas ar metai).

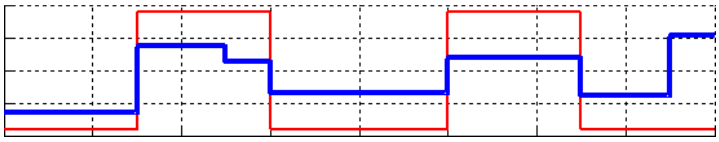
1.3 Bendra schema

1. Intensyvumo generavimas, bendros kūrinio formos sudarymui ir/arba tendencijų nustatymui.
Intensyvumo pavyzdys pav.1.3.1

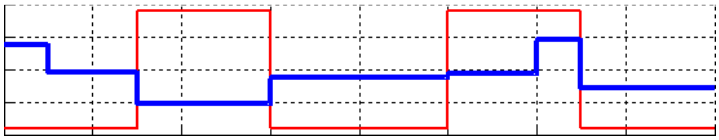


pav. 1.2 Kūrinio intensyvumo pavyzdys

2. Instrumentų *individualaus* (dalyvavimo) *intensyvumo* generavimas, instrumentų dalyvavimo kūrinyje nusakymui ir vyksmų sudarymui. (Galima keisti vidutinį intensyvumą, intensyvumo sklaidą, pokyčių skaičių ir sinchroniškumą).
Instrumentų *individualių* (dalyvavimo) *intensyvumų* ir *sinchroniškumo* pavyzdžiai pav. 1.3.2, 1.3.3, (storesne linija pavaizduoti *instrumentų intensyvumai*, plonesne - *sinchroniškumas*).

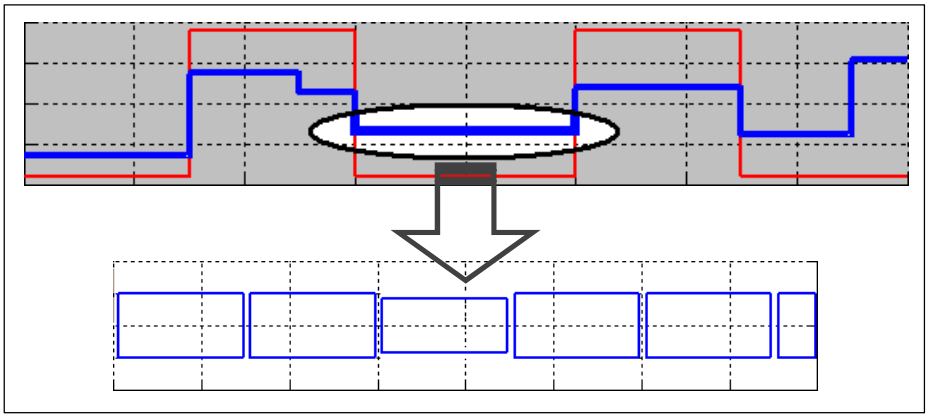


pav. 1.3 Instrumentų individualaus intensyvumo pavyzdys



pav. 1.4 Instrumentų individualaus intensyvumo pavyzdys

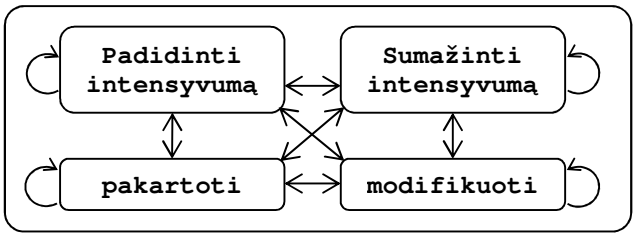
3. Instrumentų dalyvavimo blokų (atspindinčių vyksmus ir gautų individualaus intensyvumo pagalba) skaidymas į besikartojančias ir skirtingas dalis judėjimų generavimui, vyksmų sudarymui (vyksmai gali būti sudaryti iš besikartojančių, nesikartojančių ar modifikuojamų judėjimų).
Bloko suskaidymo pavyzdys pav. 1.3.4.



pav. 1.5 Dalyvavimo blokų skaidymo pavyzdys

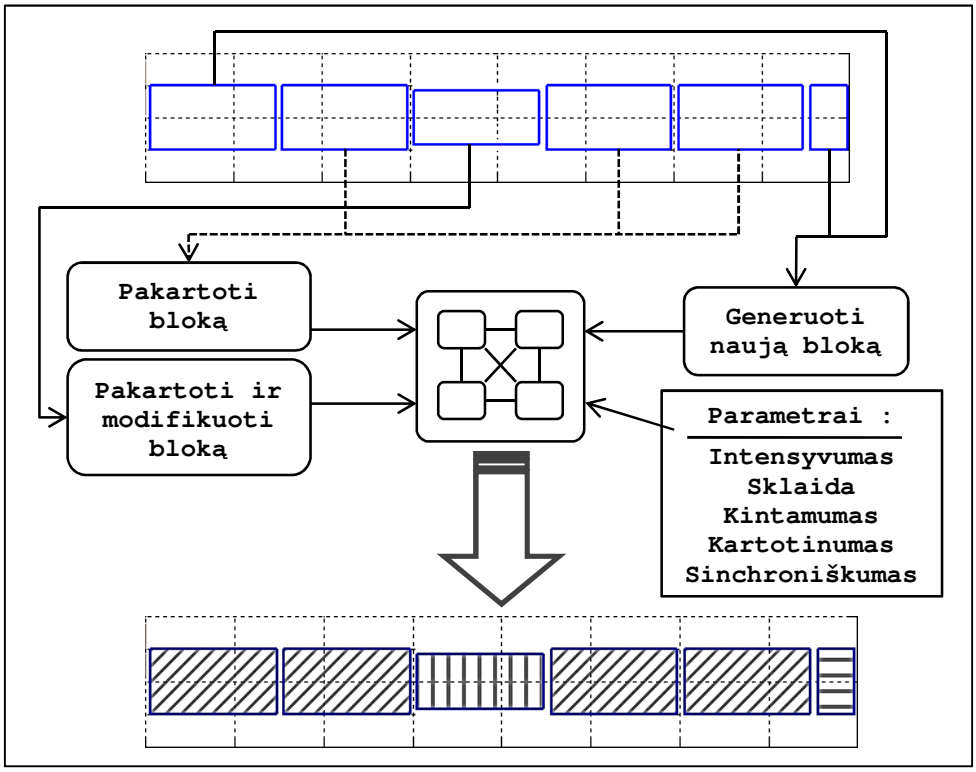
4. Taktų generavimas judėjimų sudarymui. Gali būti besivystantis judėjimas, modifikuojant taktus, arba galima generuoti tiesiog atsitiktinius taktus ir vadinti tai judėjimu.

Taktai gali būti generuojami keliais būdais. Jei intensyvumas interpretuojamas kaip natų skaičius takte ir judėjimų užpildymas taktais numato taktų modifikacijos metodą, tada turi būti galimybė taktus kažkaip pakeisti atitinkamai intensyvumo pokyčių, taip pat atitinkamai kitų parametrų (jei tokie yra), pavyzdžiui, atitinkamai sklaidos, kintamumo ar sinchroniškumo. Besivystančio takto generavimo schemos pavyzdys pav. 1.3.5. (vaizduojami galimi veiksmai su taktais ir galimi veiksmų pasikeitimai, nuosekliai gaunami taktai sudaro judėjimą)



pav. 1.6 Galimi taktų generavimo veiksmai ir jų tarpusavio ryšys

5. Išskaidytų instrumentų blokų - vyksmų - užpildymas judėjimais, t.y. vyksmų sudarymas.



pav. 1.7 Vyksmų sudarymas - užpildymas judėjimais

1.4 Bendri parametrai

Nagrinėjant - sudarant bendrą generavimo schemą buvo nutarta schemos parametrais pasirinkti : *sklaidą, kintamumą, kartotinumą, sinchroniškumą* (visi parametrai gali turėti įtaką skirtinguose schemos lygiuose, ir vienas parametras gali turėti skirtingas įtakas skirtinguose lygiuose). Taip pat ne tiek schemos veikimą, kiek rezultato mastelį, nusakantis parametras - *intensyvumo dalies dydis taktais*.

1.4.1 Intensyvumas

Intensyvumas atspindi bendrą kūrinio "formą" ar vystymosi tendenciją, "vaizdą", todėl nėra kažkokio "tikslaus sprendimo" jo generavimui. Generavimo procedūra priklauso nuo to, ką norima gauti, ar kūrinys turi turėti kažkokio "prasmingumo" ar tradiciškumo (pavyzdžiui, intensyvumas didėja, po to kinta ir pabaigoje mažėja), ar tai yra nesvarbu. Kai kuriais atvejais galbūt būtų patogiau tiesiog nupiešti intensyvumo funkciją koordinačių ašyse ir transformuoti ją į funkciją tiesiogine žodžio prasme.

Intensyvumas gali priklausyti nuo *sklaidos, kintamumo, kartotinumo*. *Sinchroniškumas* neturėtų turėti tiesioginės įtakos intensyvumui (nes sinchroniškumas pasireiškia instrumentų ryšiu tarpusavyje, todėl intensyvumas neturėtų būti įtakojamas sinchroniškumo parametro šiuo atveju siūlomoje schemoje), bet schemoje yra galimybės perinterpretavimui.

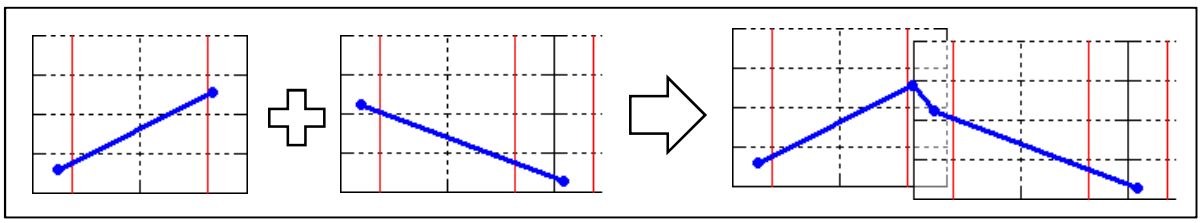
Funkciją, pasižyminčią kokiais nors "suvokiamais" ar "apčiuopiamais" vyksmais (intensyvumas kyla, leidžiasi, nekinta, ir kad tai būtų suvokiama, viskas vyksta pastebimu greičiu), galima generuoti taip:

Instrumento dalyvavimo rodiklis - $\boldsymbol{Id} \in [0, 1]$, išreiškia vidutinę generuojamo intensyvumo reikšmę. Sklaida - $\boldsymbol{S} \in [0, 1]$ yra generuojamų skaičių diapazonas į abi puses, t.y. diapazonas = $[\boldsymbol{Id}-\boldsymbol{S}, \boldsymbol{Id}+\boldsymbol{S}] \cap [0,1]$.

1. Priklausomai nuo *sklaidos* $\boldsymbol{S}$, generuojami 2 taškai: pirmas - iš intervalo $[0, \boldsymbol{S}/2]$, antras - iš intervalo $[1-\boldsymbol{S}/2, 1+\boldsymbol{S}/2]$. (Skirstinys gali būti tolygus). Šie taškai bus transformuoti į laiko momentus.

2. Generuojami 2 taškai iš intervalo [0, 1], kurie yra intensyvumo reikšmės atitinkamais laiko momentais.

3. Procedūra kartojama kiek norima kartų, naujus laiko momentus pridedant prie paskutiniojo laiko momento. Visi laiko momentai padauginami iš *intensyvumo dalies dydžio taktais*. Gauti taškai interpoliuojami laužte (tiesėmis tarp dviejų gretimų taškų). Procedūros pavyzdys "vaizdžiai" - pav.1.4.1.



pav. 1.8 Intensyvumo sudarymo pavyzdys

1.4.2 Instrumentų intensyvumas

Instrumentų intensyvumas turi nurodyti instrumento dalyvavimo reikšmę (aktyvumo reikšmę) kūrinyje ir instrumento dalyvavimo kūrinyje pasikeitimus.

Pasikeitimų skaičius nusakomas *kintamumo* $\boldsymbol{K} \in [0, 1]$ parametru (pasirenkamas iš anksto, vienas iš kūrinį nusakančių rodiklių). Kaip susieti *kintamumo* reikšmę su pasikeitimų skaičiumi ?

Kūrinio ilgį taktais pažymėsime **N** (šiame skyriuje).

Jeigu *kintamumo* reikšmė būtų proporcinga pokyčių skaičiui, tada *kintamumo* reikšmei esant 1, kiekvienas taktas būtų kaip naujas vyksmas su skirtingomis instrumento intensyvumo reikšmėmis ir vis naujais vyksmais. Sumažinus *kintamumą* iki 0.9 pasikeitimų vis tiek būtų beveik kiekvieną taktą, ir retkarčiais kas du taktai, kas nesukelia kintamumo mažėjimo jausmo.

Kintamumas turėtų mažėti tokia tvarka : kinta kas taktą, kas du taktus, kas keturis ir t.t., tai reiškia, kad :

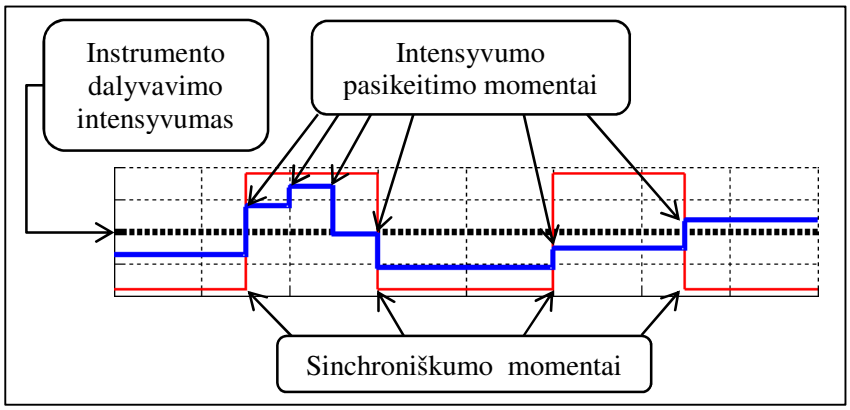
$$\boldsymbol{K} = (\log_2 (\textit{pokyčių skaičius}) / \log_2(\mathbf{N})) \in [0, 1]$$

iš ko seka, kad : $\textit{pokyčių skaičius} = \textit{suapvalinti iki sveiko skaičiaus}\ (\mathbf{N}^{\mathbf{K}})$

Taip gauname, kad esant **N**=32, **K**=0.2, *pokyčių skaičius* - $p\_k = 2$; **K**=0.4, $p\_k = 4$; **K**=0.6, $p\_k = 8$; **K**=0.8, $p\_k = 16$.

Taip keičiant parametrą visame jo diapazone galima lengviau gauti skirtingus kūrinio suskaidymus į vyksmus. Jeigu susiejimas būtų tiesinis, tada intervale [0.5, 1] gaunamas pokyčių skaičius būtų [16, 32] ir bet kuris skaičius iš šio intervalo reiškia labai didelį kintamumą, vyksmai kinta jei ne kiekviena taktą, tai kas antrą (vidutiniškai). Žinant *pokyčių skaičių* p_k, galima generuoti p_k pasikeitimo įvykimo laiko reikšmių ir p_k+1 atsitiktinių dydžių iš $[\boldsymbol{Id-S}, \boldsymbol{Id+S}] \cap [0,1]$ (nes reikalinga reikšmė laiko momentu 0).

Sinchroniškumą tarp kelių instrumentų galimą realizuoti generuojant atskirą intensyvumą, kurio pokyčių skaičius yra sinchroniškumo reikšmė padauginta iš instrumentų pokyčių skaičius. Ir generuojant instrumentų intensyvumą, atitinkamą dalį laiko momentų panaudoti iš sugeneruoto sinchroniškumui intensyvumo (kaip parodyta pav.1.4.2).



pav. 1.9 Instrumento intensyvumo ir vyksmų sudarymo pavyzdys

1.4.3 Taktų generavimas

Taktai gali būti generuojami skaidant skaičių į norimą dalių skaičių - natų skaičių. Skaidomas skaičius turi atitikti maksimalų natų skaičių takte. Skaidinio dalių skaičius turi atitikti galutinį intensyvumą atitinkamu laiko momentu ***t***.

Galutinis intensyvumas (t) = Bendras intensyvumas (t) · Instrumento intensyvumas (t) .

Skaidymas gali būti atsitiktinis ar atsitiktinis su apribojimais - tokiu atveju, pasirenkant ne skaidinio dalis, bet skaidinio dalines sumas (atitinkama tvarką), iš karto gaunamos takto natų pozicijos.

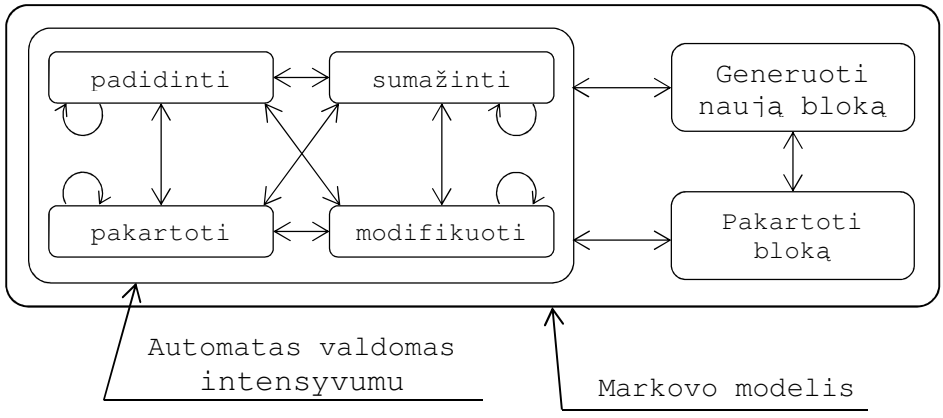
Generuojant visus galimus skaidinius ir pasirenkant iš jų norimą skaidinį atitinkamai parametrų reikia nepamiršti, kad skaidinys sugeneruojamas kaip skaidinio dalių rinkinys, surikiuotas didėjimo tvarka.

Dėl to visi galimi taktai negaunami. Tuo tikslu reikia skaidinį "sumaišyti", t.y. iš sugeneruoto skaidinio sudaryti naują, kurio dalys yra tos pačios, bet jų tvarka gali skirtis. Tai galima padaryti atsitiktinai renkantis skaidinio dalis ir surašant jas iš eiles, taip gaunant naują skaidinį, sudarytą iš tų pačių dalių, bet surikiuotą kita tvarka.

Taip pat reikalingi taktų modifikavimo metodai, t.y. natos ištrynimo iš takto ir natos perskėlimo takte metodai. Turint taktą, žinomos visų natų pozicijos (natų ilgiai, supaprastinimui pasirenkami iki kitos natos arba iki pabaigos. t.y. vienu metu viename takte visada yra viena nata). Todėl norint ištrinti natą jos atitinkama pozicija pašalinama iš sąrašo. Norint perskelti (įdėti naują) natą, pasirenkama dar viena pozicija. Norint modifikuoti taktą (t.y. šiek tiek jį pakeisti, bet nekeisti natų skaičiaus), galima ištrinti kokį nors skaičių natų iš takto, ir po to papildyti taktą tuo pačiu natų skaičiumi, arba papildyti ir po to ištrinti.

1.4.4 Judėjimų sudarymas

Judėjimai sudaromi iš taktų, atitinkamai kintamumo, sklaidos, kartotinumo, intensyvumo



pav. 1.10 Judėjimų sudarymo iš taktų schemos pavyzdys

1.4.5 Kūrinio bloko - vyksmo sudarymas

Žinant bloko ilgį **N** (gautą iš instrumento intensyvumo), **N** - išskaidomas į intervalus, atitinkamai pasirinkto kartotinumo rodiklio, vienodi blokai užpildomi vienodais judėjimais, skirtingi blokai skirtingais, vyksmo sudarymo schema pavaizduota pav.1.3.6 psl 13.

Skaidinio kartotinumo įvertinimas

Kartotinumas skaičiaus skaidiniui gali būti įvertintas skirtingai dėl to, kad galima skirti skirtingą dėmesį skirtingiems aspektams. Gana paprastas būdas įvertinti kartotinumą yra apskaičiuoti skaidinio kartotinas dalis, t.y. skaičiai, kurie kartojasi, sudedami. Pavyzdžiui 8-to skaidinys (3,3,1,1) turi dvi kartotinas dalis 3+3=6 ir 1+1=2. Akivaizdu, kad skaidinys (3,3,1,1) yra mažiau kartotinas nei skaidinys (2,2,2,2), kurio kartotina dalis yra 8. Dėl to skaidinio kartotinų dalių negalima sudėti, bet reikia kaip nors sumažinti, galima trumpesnes dalis dalinti iš didėjančio skaičiaus dalies ilgiui trumpėjant. Taip pat skaidinys (1,1,1,1, 1,1,1,1) yra labiau kartotinas nei (2,2,2,2), dėl to verta vertinti skaidinio dydį. Bet verta vertinti viso skaidinio dydį, o ne jo kartotinos dalies dydį.

Jei kartotina dalis skaidinyje laikoma tik ilgiausia dalis, tada siūlomas įvertinimas yra

Ilgiausios kartotinos dalies ilgis - i_d

Skaidinio dydis - s_d (skaičių, sudarančių skaidinį, skaičius)

Kartotinumas = $i\_d \cdot log_2(s\_d)$

Jei "reikia" kelių kartotinų dalių skaidinyje, bet tokių skaidinių kartotinumas vis tiek laikomas mažesniu už skaidinius su viena besikartojančia dalimi, tada kartotinas dalis reikia sudėti su mažėjančiais svoriais, pavyzdžiui, gali būti harmoninė seka arba laipsninė.

Kartotinų dalių suma - k_d (suma su atitinkamais svoriais)

Skaidinio dydis - s_d (skaičių, sudarančių skaidinį, skaičius)

Kartotinumas = $k\_d \cdot log_2(s\_d)$

2 TIRIAMOJI DALIS

2.1 Skaičiaus skaidinys

Natūralaus skaičiaus skaidinys - tai natūralių skaičių, kurių suma yra lygi duotajam skaičiui, rinkinys, pavyzdžiui 3 ir 2 yra vienas iš penketo skaidinių, t.y. 5 = 3 + 2.

2.2 Skaičiaus skaidinių skaičius

Nagrinėjat natūralųjį skaičių, jo skaidinį sudaro už jį mažesni arba jam lygūs teigiami natūralūs skaičiai. Skaičiaus skaidinių skaičius išreiškiamas rekurentine formule (2.2.1) [WM] arba gali būti įvertintas apytiksliai naudojant asimptotinį įvertinimą (2.2.2) [WM].

$$p(N) = \sum_{k=1}^{N}(-1)^{k+1}\left[P\left(N - \frac{k\cdot(3k-1)}{2}\right) + P\left(N - \frac{k\cdot(3k+1)}{2}\right)\right] \qquad (2.2.1)$$
$$p(0) = 1, \;\; p(-k) = 0, \;\; kai \;\; k \in \mathbb{N}$$

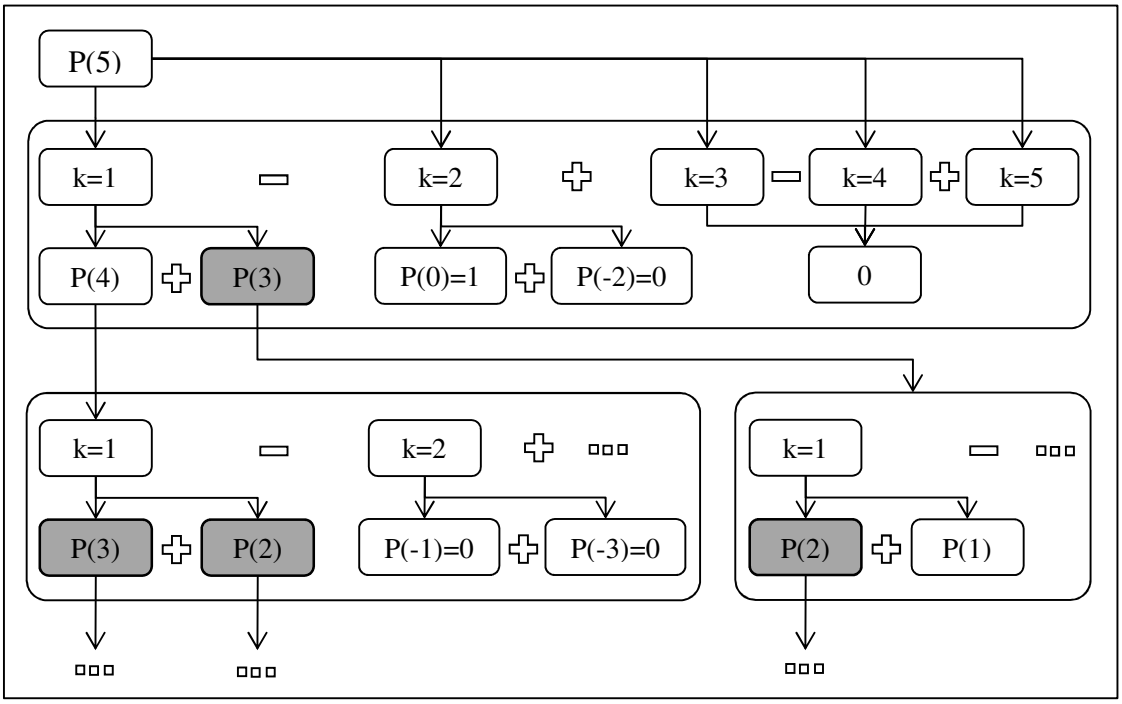
$$p(N) \approx \frac{1}{4N\sqrt{3}} \cdot e^{\left(\pi\sqrt{\frac{2N}{3}}\right)}, kai \;\; N \to \infty \qquad (2.2.2)$$

Skaičiuojant skaičiaus skaidinių skaičių ir tiesiogiai naudojant rekurentinę formulę (2.2.1) (t.y. pradedant skaičiuoti nuo norimo N ir apskaičiuojant visas reikiamas vertes), daug kartų apskaičiuojama tas pats (t.y. atliekama daug nereikalingų - kartotinų skaičiavimų), kaip parodyta pav.2.2.1 (funkcijos *p*(5) iškvietimo medis, tamsesni langeliai žymi kartotinus veiksmus). Dėl to reikia skaičiuoti "iš apačios", t.y. skaičiuoti *p*(*n*) visiems *n* nuo 1 iki N, ir naudoti jau apskaičiuotas reikšmes, vietoj to, kad skaičiuotume jas iš naujo.

Rekurentinės formulės (2.2.1) skaičiavimo laiko sudėtingumas analiziškai netirtas, nes jos sudėtingumo didelis augimas (pvz., eksponentinis) matomas ir taip, bet skaičiavimo laikas pateikiamas kai kurioms reikšmėms lentelė 2.2.1 (panagrinėta praktiškai). Iš lentelės matoma, kad keičiant argumentą 2-ais vienetais laikas trigubėja (artėja į trigubėjimą), kas reiškia, kad funkcijos skaičiavimo laiko sudėtingumas auga eksponentiškai. T.y. skaičiavimo laiko įvertinimo funkciją žymėsime T(*f*), *f* - vertinama funkcija, tada T(*p*(*a*+2)) = 3T(*p*(*a*)) ir taip pat :

T(*p*(*a*+2*k*)) = 3^k ·T(*p*(*a*)), kas ir yra eksponentinis augimas.

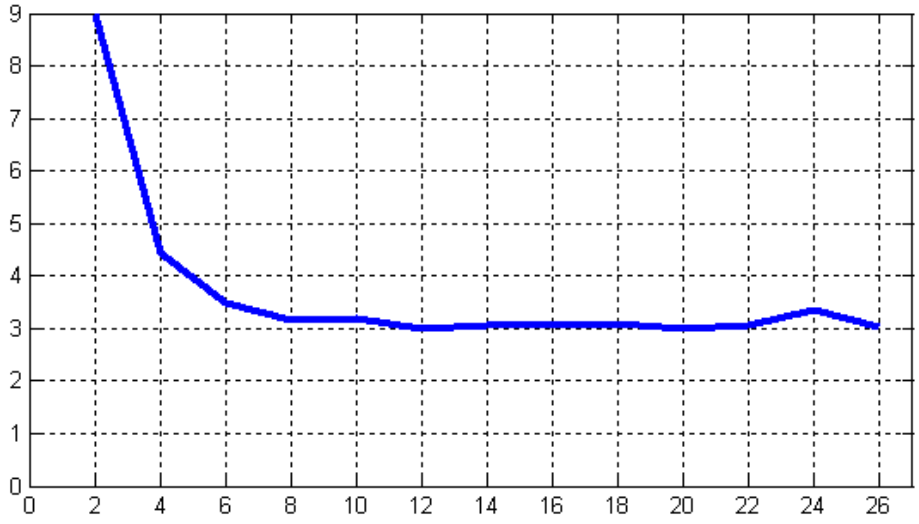
Laiko, reikalingo skaičiavimams, santykinis pokytis, lyginant su prieš tai skaičiuota reikšme, pavaizduotas pav. 2.2.2. Laiko, reikalingo skaičiavimams, grafikas pavaizduotas pav. 2.2.3, vertikali ašis yra logaritminė, ir grafikas labai panašus į tiesę, kas taip pat liudija apie algoritmo eksponentinį sudėtingumą (laiko prasme).



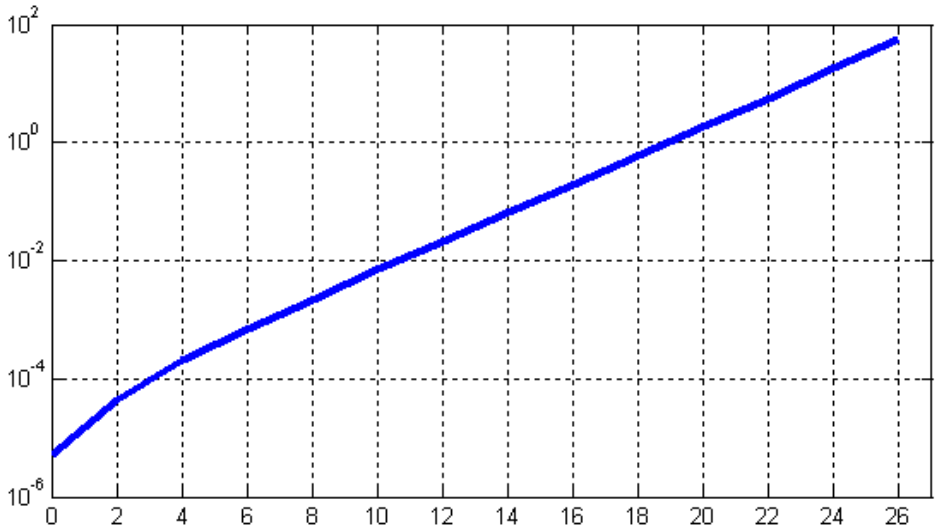
pav. 2.1 Funkcijos $p(5)$ iškvietimo medis

Funkcija	Laikas s.	Skaičiavimų kartai	Vieno skaičiavimo laikas s.	Laiko augimas (kartais)
$p(0)$	5	10^6	0.000005	
$p(2)$	4.5	10^5	0.000045	9.00
$p(4)$	20	10^5	0.0002	4.44
$p(6)$	7	10^4	0.0007	3.50
$p(8)$	11	5000	0.0022	3.14
$p(10)$	14	2000	0.007	3.18
$p(12)$	21	1000	0.021	3.00
$p(14)$	32	500	0.064	3.04
$p(16)$	39	200	0.195	3.04
$p(18)$	30	50	0.60	3.07
$p(20)$	18	10	1.80	3.00
$p(22)$	27.5	5	5.50	3.05
$p(24)$	18.5	1	18.50	3.36
$p(26)$	56	1	56.0	3.02

Lentelė 2.2.1 Funkcijos skaičiavimo laikas



pav. 2.2 Laiko, reikalingo skaičiavimams, santykinis pokytis



pav. 2.3 Laikas, reikalingas skaičiavimams

Dėl sudėtingo skaidinių skaičiaus apskaičiavimo rekurentiniu būdu reikia skaičiuoti kitaip. Rekurentiškai skaičiuojama ilgai, nes atliekama daug kartotinų skaičiavimų. Kad kartotinų skaičiavimų nebūtų, reikia išsaugoti visas jau naudotas funkcijų reikšmes. Tai padaryti paprasta skaičiuojant funkciją nuo pat "pradžių", t.y. skaičiuojant $p(n)$, kai n kinta nuo 1 iki N, išsaugant rezultatus, ir naudojant juos skaičiuojant tolesnes $p(n)$ reikšmes. Tokio skaičiavimo realizavimui reikalingi du ciklai, vienas - kai k kinta nuo 1 iki N, kitas - pirmo ciklo viduje, kai n kinta nuo 1 iki k. Cikluose atliekami veiksmai turi vienodą sudėtingumą laiko prasme, kas reiškia, kad skaičiavimo sudėtingumas proporcingas N^2. T.y. iš viso atliekamų ciklais žingsnių yra skaičių nuo 1 iki N suma, kuri proporcinga N^2.

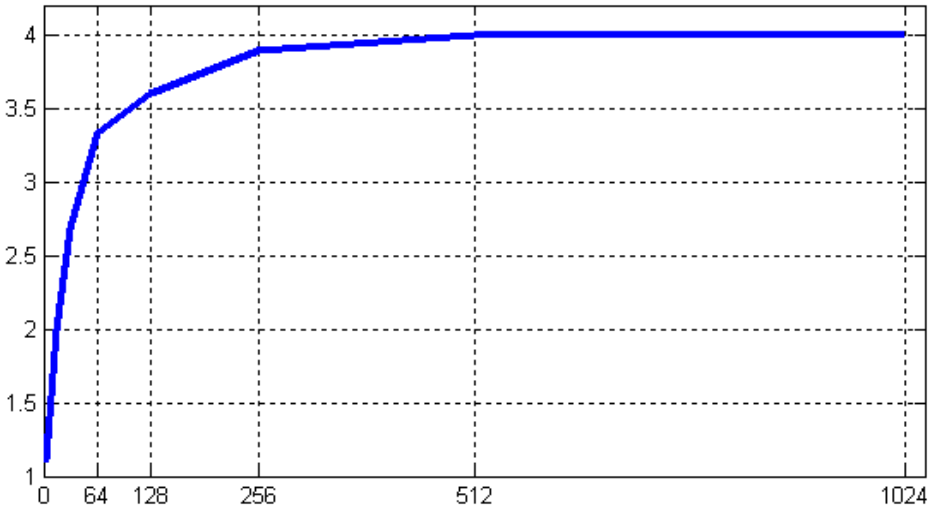
Skaičiuojant "iš apačios", funkcijos skaičiavimo laiko sudėtingumas yra kvadratinis, ką galima pamatyti iš Lentelė 2.2.2 Funkcijos skaičiavimo laikas. Lentelėje funkcijos argumentas dvigubėja, o sunaudotas skaičiavimams laikas keturgubėja (artėja į keturgubėjimą, kas matoma pav. 2.2.4). T.y. skaičiavimo laiko įvertinimo funkciją žymėsime T(*f*), *f* - vertinama funkcija, tada $T(p(2a)) = 4T(p(a))$ ir taip pat :

$$\mathbf{T(\ p(2^k \cdot a)\ ) =}\ 4^k \cdot T(p(a)) = (2^2)^k \cdot T(p(a)) = \mathbf{(2^k)^2 \cdot T(\ p(a)\ )}$$

kas ir yra kvadratinis augimas.

Funkcija	Laikas s.	Skaičiavimų kartai	Vieno skaičiavimo laikas s.	Laiko augimas (kartais)
P(0)	7	10^6	0.000007	
P(2)	9	10^6	0.000009	1.28
P(4)	10	10^6	0.000010	1.11
P(8)	14	10^6	0.000014	1.40
P(16)	28	10^6	0.000028	2.00
P(32)	7.5	10^5	0.000075	2.67
P(64)	25	10^5	0.000250	3.33
P(128)	9	10^4	0.00090	3.60
P(256)	35	10^4	0.0035	3.88
P(512)	14	10^3	0.014	4.00

Lentelė 2.2.2 Funkcijos skaičiavimo laikas



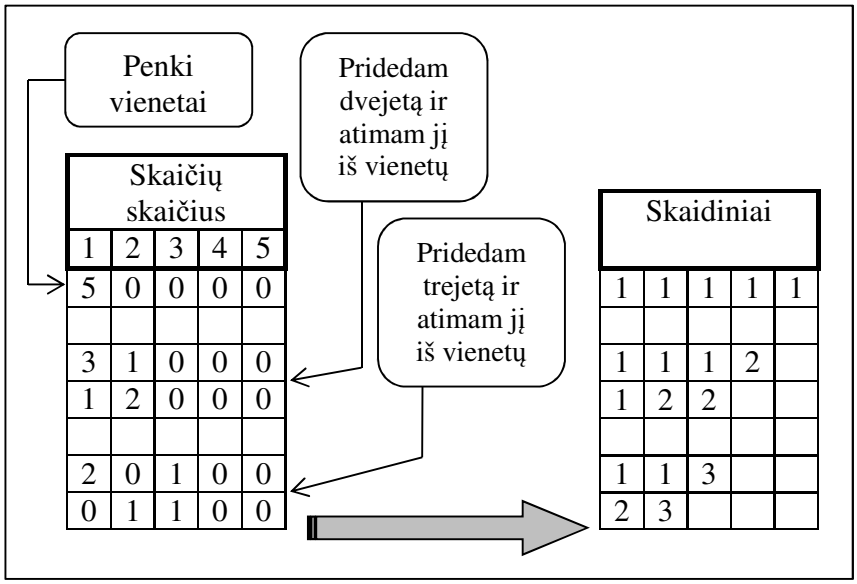
pav. 2.4 Laiko, reikalingo skaičiavimams, santykinis pokytis

Priedai : `partition_function_seq.m` - skaidinių skaičiaus apskaičiavimo metodas, kurio rezultatas yra skaičių nuo 1 iki N (N - parametras) skaidinių skaičius (kvadratinio sudėtingumo metodas).

2.3 Visų galimų skaičiaus skaidinių generavimas

Kiekvieną skaidinį galima surikiuoti didėjimo ar mažėjimo tvarka, ir skirtingi skaidiniai išlieka skirtingi, nes skaidinys nepriklauso nuo suskaidymo tvarkos (nes suma komutatyvi). Todėl galima generuoti skaidinius tiesiog generuojant juos leksikografine tvarka. Pradedant nuo (1), po to (1 1) ir t.t. Bet ne visi tokie skaidiniai yra tinkami. Paprasčiau generuoti nagrinėjant ne skaičiaus skaidinį, bet skaičiaus skaidinio dažnių lentelę, pavyzdžiui, penketą sudaro *vienas* dvejetas ir *vienas* trejetas.

Pradedant nuo skaidinio, sudaryto vien iš vienetų, kitus skaidinius galima generuoti pridedant po vieną sekantį skaičių (pradedant nuo dvejeto, kitas skaičius 3, po to 4) ir atimant atitinkamą skaičių vienetų prie kiekvieno iki tol sugeneruoto skaidinio (kelių 5-to skaidymo žingsnių pavyzdys pav.2.3.1). Gali atrodyti, kad taip gaunami ne visi skaidiniai, nes kai kurių skaidinių gavimui reikia iš turimo skaidinio atimti ne vienetus, bet kitus skaičius, bet tokiais atvejais yra kitas skaidinys, kuriame vietoje skaičių, kuriuos norima atimti, yra vienetai. Pavyzdžiui, skaidant 8, iš skaidinio (2 2 4) negalime gauti skaidinio (4 4), nes skaidinyje (2 2 4) nėra vienetų, bet yra kitas 8 skaidinys (1 1 1 1 4), iš kurio galima gauti skaidinį (4 4), pridedant 4 ir atimant keturis vienetus. Tokiu būdu gaunami visi galimi skaidiniai. Dėl to, kad pradedama nuo vienintelio skaidinio, sudaryto vien iš vienetų, ir pridedant naujas dalis į skaidinį jos atimamos tik iš vienetų, kiekvienas skaidinys sudaromas vienintelį kartą.



pav. 2.5 Skaičiaus skaidinių generavimo pavyzdys

Skaidinių skaičius duotajam **N** apytiksliai išreiškiamas asimptotine formule (2.2.2). Vieno skaidinio sudarymo laikas gali būti laikomas pastoviu ir lygiu **N**. Jeigu neatsižvelgti į laiką, skiriamą skaidinio patikrinimui, algoritmo sudėtingumas gaunamas sudauginus skaidinių skaičių ir jų dydį (2.3.3). Šiuo sudėtingumo įvertinimu nesiekiama tiksliai įvertinti algoritmo sunaudojamo laiko, bet siekiama tik parodyti, kad sudėtingumas yra eksponentinis (kaip ir skaidinių skaičius), kas reiškia, kad ir "ne teoriniais" (pavyzdžiui 1000-is turi $2.406 \cdot 10^{31}$ skaidinių) atvejais skaičiaus visų skaidinių sudarymui nepakaks jokių resursų.

$$p(N) \cdot N \approx \frac{1}{4\sqrt{3}} \cdot e^{\left(\pi\sqrt{\frac{2N}{3}}\right)}, kai \;\; N \to \infty \qquad (2.3.3)$$

Visų galimų skaičiaus skaidinių generavimo algoritmas :

`Skaidiniai` - eilutėse - skaičiaus skaidiniai, atvaizduojami kaip skaičių skaidinyje skaičius,

`N` - skaidomas skaičius,

`IS` - į skaidinį įtraukiamas skaičius,

`ss` - senesni skaidiniai,

`PS` - paskutinio senesnio skaidinio numeris,

`EI` - einamas indeksas.

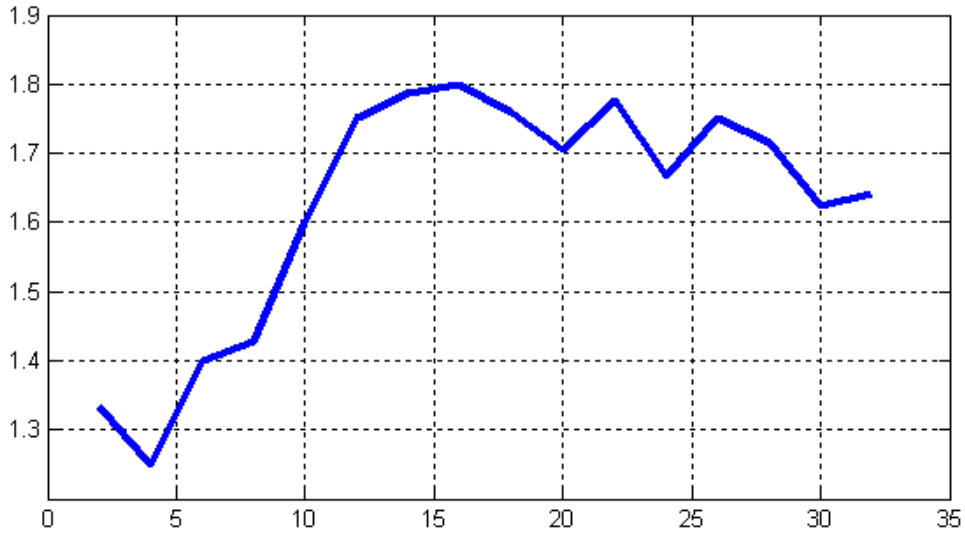
```
Skaidiniai(1,1:N) = 0;
Skaidiniai(1,1)   = N;
PS = 1;
EI = 2;

for  IS = 2 : N

    for  k = 1 : N/IS
        for  ss = 1 : PS
            if Skaidiniai (ss, 1) >= k*IS
                Skaidiniai (EI, :)  = Skaidiniai (ss, :);
                Skaidiniai (EI, 1)  = Skaidiniai (EI, 1) - k*IS;
                Skaidiniai (EI, IS) = k;

                EI = EI + 1;
            end
        end
    end

    PS = EI - 1;

end
```

Priedai : `number_partitioning.m` - metodas, sudarantis visus skaičiaus skaidinius; rezultatas - skaičiaus (parametro) skaidiniai, išreikšti skaičių pasitaikymo skaidinyje skaičiumi.

Algoritmo sudėtingumas laiko prasme panašus į eksponentinį, nes laikas didėja panašiu santykiu (algoritmui "nusistovėjus" - pradedant nuo N≈12), bet santykis didinant N mažėja, kas atspindi kvadratinę šaknį iš argumento eksponentės rodiklyje, gautame įvertinime. Lentelė 2.3.1 aprašo algoritmo tyrimo rezultatus, pav.2.3.2 vaizduoja laiko augimą.

Skaidomas skaičius N	Laikas s.	Skaičiavimų kartai	Vieno skaičiavimo laikas s.	Laiko augimas (kartais)
0	1.5	10^5	0.000015	
2	2	10^5	0.000020	1.33
4	2.5	10^5	0.000025	1.25
6	3	10^5	0.000030	1.40
8	5	10^5	0.000050	1.43
10	8	10^5	0.000080	1.60
12	7	50 000	0.00014	1.75
14	7.5	30 000	0.00025	1.79
16	9	20 000	0.00045	1.80
18	9.5	12 000	0.00079	1.76
20	13.5	10 000	0.0013	1.70
22	12	5 000	0.0024	1.78
24	8	2 000	0.004	1.67
26	7	1 000	0.007	1.75
28	12	1 000	0.012	1.71
30	19.5	1 000	0.019	1.62
32	32	1 000	0.032	1.64

Lentelė 2.3.1 Algoritmo skaičiavimo laiko tyrimo rezultatai



pav. 2.6 Laiko, reikalingo skaičiavimams, santykinis pokytis

2.4 Visiškai atsitiktinis skaidymas

Turi būti žinomas skaidinio dydis, t.y. sudarančiųjų skaidinį skaičių skaičius.

Žinant skaidinio dydį SD, galima generuoti SD tolygiai pasiskirsčiusių natūralių *nepasikartojančių* skaičių, mažesnių už N - skaidomą skaičių. Šie skaičiai parodo skaidinio dalines sumas, esant atitinkamai skaidinio dalių tvarkai. Dėl to galima rasti pačius skaidinio skaičius. Mažiausias sugeneruotas skaičius yra vienas iš skaidinio skaičių, toliau, surikiavus sugeneruotus skaičius ir apskaičiavus skirtumus tarp gretimų skaičių, galima gauti dar SD-2 skaidinio skaičių, ir iš skaidomo skaičiaus N atėmus didžiausią sugeneruotą skaičių gaunamas paskutinis skaidinio skaičius.

Be to, generuojami skaičiai nebūtinai turi būti pasiskirstę tolygiai, svarbu, kad galima būtų sugeneruoti bet kokį skaičių nuo 1 iki N (bendru atveju).

Generuojant tokiu būdu nėra skaidinių, kurie negalėtų būti sugeneruoti, nes turint bet kokį skaidinį (atitinkamo dydžio), surikiuotą bet kokia tvarka, galima paskaičiuoti šio skaidinio dalines sumas, kurios yra teigiamos ir mažesnės už skaidomą skaičių ir dėl to gali būti sugeneruotos.

Generuojant tokiu būdu skirtingų skaidinių pasirodymo tikimybės nėra vienodos, ir kuo didesnis skaidomas skaičius, tuo didesnis skirtumas. Taip yra, nes kai kurie skaidiniai gali būti surikiuoti skirtinga tvarka ir nuo to pasikeičia jų dalinės sumos, t.y. jie gali būti sugeneruoti keliais skirtingais būdais. Kai kurie kiti skaidiniai nepriklauso nuo jų surikiavimo ir gali būti sugeneruoti vieninteliu būdu (pavyzdžiui 8-to skaidinys (2,2,2,2)).

Turint skaičiaus **N** skaidinį ($d_1,d_1,...,d_1$, $d_2,d_2,...,d_2$, $d_k,d_k,...,d_k$), kurio dydis (dalių skaičius) yra **K** ir kuriame skaidinio dalis d_i pasikartoja n_i *(i=1,2,...,k)* kartų, skaidinį galima skirtingai surikiuoti *n_t* skaičių kartų, kuris yra aprašomas formule (2.4.4).

$$n\_t = \frac{K!}{n_1! \cdot n_2! \cdot \ldots \cdot n_k!} \qquad (2.4.4)$$

Kai visos skaidinio dalys vienodos (pavyzdžiui, 8-to skaidinys (2,2,2,2)), gauname, kad skaidinys gali būti sugeneruotas (*4!/4! = 1*) vieninteliu būdu. Kitas atvejis, pavyzdžiui, 8-to skaidinys (1,1,1,1,2,2) gali būti sugeneruotas (*6!/(4!·2!) = 15*) 15 skirtingų būdų.

Skaidant skaičių, naudojant tolygųjį skirstinį (kai kiekvieno skaičiaus pasirinkimo tikimybė vienoda) tikimybė sugeneruoti pasirinktą skaidinį yra n_t / *p*(N).

Kai kurių skaidinių dažnis auga katastrofiškai palyginus su kitais, ir net nedideliems N skirtumas tarp tikimybių labai didelis (pavyzdyje su 8-tu tikimybė sugeneruoti skaidinį skiriasi 15 kartų).

Kaip greitai ir paprastai sugeneruoti nepasikartojančius atsitiktinius natūralius skaičius, mažesnius už N ? (tariant, kad galime sugeneruoti atsitiktinius natūralius skaičius, mažesnius už N)

Vienas iš būdų

Turint vektorių su elementais nuo 1 iki N-1, generuoti natūrinį atsitiktinį skaičių ***r*** nuo 1 iki N-1 ir pasirinkti ***r*** - tąjį vektoriaus narį (***r*** - tasis narys tampa generuojamos sekos nariu), tada sukeisti ***r*** - tąjį vektoriaus narį su N-1 nariu vietomis ir generuoti natūrinį atsitiktinį skaičių ***r*** nuo 1 iki N-2, rinkti ***r*** - tąjį vektoriaus narį (kuris negali kartotis, nes išrinktas narys neberenkamas) ir sukeisti išrinktą narį su N-2 vektoriaus nariu. Tęsti procedūrą tol, kol bus išrinktas reikiamas skaičius narių.

Turint sugeneruotą skaičių rinkinį (seką) ir norint jį papildyti, nesunku tai padaryti generuojant atsitiktinį skaičių iš atitinkamo intervalo ir pasirenkant atitinkamoje pozicijoje esantį skaičių. Bet "grąžinti" norimą skaičių esantį sekoje (t.y. ištrinti skaičių iš sekos) į galimų pasirinkimui skaičių sąrašą yra sudėtingiau, reikia rasti grąžinamą skaičių, sukeisti jį vietomis su anksčiausiai vektoriuje panaudotu skaičiumi, ir padidinti pasirenkamų pozicijų skaičių vienetu. Taip pat toks metodas nepatogus norint iš pradžių rinktis atitinkamus skaičius (t.y. norint apriboti galimus pasirinkimui skaičius), ir po to likusius.

Kitas būdas

Turint vektorių su elementais nuo 1 iki N-1, surašytais i atitinkamas pozicijas nuo 1 iki N-1, pasirinktus elementus žymėti specialiu simboliu (pavyzdžiui -1). Renkant ***r*** - tąjį narį, pažymėtieji skaičiai neskaičiuojami, pavyzdžiui, vektoriuje (1 -1 3) antras elementas būtų 3. Tokiu būdu galima greitai "grąžinti" skaičių (ištrinti skaičių iš sekos).

Skaidomo skaičiaus galimų pasirinkimui skaičių apribojimas - "tvarkos" padidinimui.

Renkantis atsitiktinius skaičius, atspindinčius dalines sumas, galima apriboti galimų pasirinkimui skaičių skaičių. Pavyzdžiui, skaidant 8-tą į 3 dalis (pasirenkant 2 skaičius) galima rinktis tik iš lyginių skaičių (2,4,6). Toks skaidymas gali būti naudingas norint, kad skirtingi skaidiniai turėtų bendrą "ritmą".

Kad būtų greita ir paprasta rinktis tinkamus skaičius, papildyti sugeneruotų skaičių seką ar ištrinti narį iš sekos, su sąlyga, kad renkami ar trinami skaičiai turi būti pasirenkami "su apribojimais", galimų skaičių vektorių reikia surikiuoti "patogesniu būdu". Pavyzdžiui, skaidant 8, norint, kad pirmas skaičius būtų 4 (padalinti 8 per pusę), po to 2 arba 6 ir po to 1,3,5 arba 7, patogu surašyti skaičius tokia tvarka (4, 2, 6, 1, 3, 5, 7). Tokiu būdu lengva generuoti atsitiktinius skaičius taip, kad iš pradžių būtų pasirenkamas 4, po to 2 arba 6, ir t.t.

2.5 Atsitiktinis skaidinio generavimas su vienodomis skaidinių pasirodymo tikimybėmis

Iš skyriaus 2.4 *Visiškai atsitiktinis skaidymas* galima pastebėti, kad generuojant skaičiaus skaidinius skirtingų skaidinių tikimybės skiriasi, nes kai kurie skaidiniai gali būti sugeneruoti skirtingais būdais (pavyzdžiui, skaidant 8-tą į 5 ir 3, galima gauti (5,3) arba (3,5)). Jei skaidinio tvarka nesvarbi, arba norima gauti skirtingus skaidinius su vienodom tikimybėm, ir tik po to skaidinį "sumaišyti" (perrikiuoti skaidinio dalis), reikia pašalinti skirtingas skaidinio sudarymo galimybes, t.y. kiekvienas skaidinys turi būti sudaromas vieninteliu būdu. Tai galima padaryti generuojant surikiuotus kokia nors tvarka skaidinius (pavyzdžiui, didėjimo ar mažėjimo tvarka).

Skyriuje 2.4 Visiškai atsitiktinis skaidymas siūlomas metodas tiesiogine prasme skaido skaičių, taip gaunamos atitinkamai surikiuoto skaidinio dalinės sumos ir iš jų apskaičiuojamos skaidinio dalys. Kaip skaidyti skaičių taip, kad iš dalinių sumų būtų gaunami didėjimo ar mažėjimo tvarka surikiuoti skaidiniai, neaišku, lengviau generuoti pačias skaidinio dalis ir generuoti jas atitinkama tvarka.

Dėl to pasirinkta generuoti ne skaidant skaičių, bet generuojant skaidinio dalis. Kad skaidinys būtų sudaromas vieninteliu būdu, dalys turi būti generuojamos didėjimo (mažėjimo) tvarka. Pirmas generuojamas skaičius turi būti didžiausias (mažiausias), kitas generuojamas skaičius - mažesnis arba lygus (didesnis arba lygus) ir t.t. Taip generuojant kiekvienas skaidinys bus gaunamas vieninteliu būdu. Bet jei generuoti tiesiog mažėjančių skaičių seką, kiekvieną skaičių renkantis pagal tolygųjį skirstinį, skirtingi skaidiniai gaunami su skirtingomis tikimybėmis. Pavyzdžiui, skaidant 8-tą, tikimybė gauti skaidinį (8) tokia pati kaip ir tikimybė gauti skaidinį su didžiausia dalimi 3, o tokių skaidinių yra 5: (3,1,1,1,1,1), (3,1,1,1,2), (3,1,2,2), (3,1,1,3), (3,2,3).

Norint generuoti skaidinius ne su vienodomis tikimybėmis kiekvienam skaidiniui, bet, pavyzdžiui, su vienodomis tikimybėmis didžiausiai (mažiausiai) skaidinio daliai, galima naudoti tolygųjį skirstinį ir jo pagalba sudarinėti mažėjančias (didėjančias) skaičių sekas.

Bet kad generuotume bet kokį skaidinį su vienoda tikimybe, reikia žinoti skaidinių pasiskirstymą pagal didžiausią (mažiausią) jų dalį. T.y. argumentas atspindi skirstinius su didžiausia (mažiausia) dalimi, skirstinio reikšmė atspindi tokių skirstinių dažnį tarp visų duoto skaičiaus skaidinių.

2.5.1 Skaičiaus skaidinių skirstinio pagal didžiausią (mažiausią) dalį gavimas

Žinant, kiek skaičius turi skaidinių (iš formulės (1)), galima apskaičiuoti, kiek yra skaidinių, kuriuose didžiausia (mažiausia) dalis yra pasirinktas skaičius.

Skaičiaus **N** skaidinių skaičių $p(N)$ žymėsime **|N|.**

Skaidinių su didžiausia dalimi skaičiavimas :

Jei didžiausia skaidinio dalis **K** yra didesnė arba lygi nei pusė skaidomo skaičiaus **N**/2, tai visos likusios dalys privalo būti mažesnės už didžiausią dalį, ir didžiausią dalį atitinka **|N-K|** skaidinių.

Jei didžiausia skaidinio dalis **K** yra mažesnė nei pusė skaidomo skaičiaus **N**/2, tai **|N-K|** skaidinių įtrauks į save ir tuos skaidinius, kuriuose yra dalis didesnė už pasirinktą didžiausią skaidinio dalį (pavyzdžiui, skaidant 8-tą su didžiausia dalimi 3, |5| įtraukia ir tokius skaidinius kaip (5), (4,1), kurie netinkami, nes jų didžiausia dalis yra didesnė nei 3). Dėl to iš **|N-K|** reikia atimti skaičiaus N-K skaidinių skaičių, kurių didžiausia dalis yra didesnė nei pasirinkta. Taip gaunama rekurentinė formulė (2.5.5), kurios reikšmė duotiems N ir K yra skaičiaus N skaidinių skaičius, kurio didžiausia dalis yra K.

$$max(N,K) = |N-K| - \sum_{i=K+1}^{N-K} max(N-K,i) \qquad (2.5.5)$$

Skaidinių su mažiausia dalimi skaičiavimas :

Taip pat galima išvesti skaidinių su mažiausia dalimi skaičių. Minimali skaidinio dalis K negali būti didesnė nei pusė skaidomo skaičiaus **N**/2. Kai minimali skaidinio dalis yra mažesnė arba lygi **N**/2, visos kitos skaidinio dalys sudaro skaičiaus **N-K** skaidinį, todėl visų galimų skaidinių, kuriuose yra skaičius **K,** skaičius yra **|N-K|**, bet tarp šių skaidinių taip pat yra ir netinkamų - tokių, kuriuose yra dalis, mažesnė nei pasirinkta minimali dalis, dėl to tokius skaidinius reikia pašalinti (neskaičiuoti jų). Taip gaunama rekurentinė formulė (2.5.6)

$$min(N,K) = |N-K| - \sum_{i=1}^{K-1} min(N-K,K-i) \qquad (2.5.6)$$

Pastabos :

Iš sudarytų rekurentinių formulių galima apskaičiuoti skaidinių su didžiausia (mažiausia) dalimi skaičių ir iš apskaičiuotų skaičių galima gauti skaidinių skirstinį (skaidinių pasitaikymo dažnius) pagal didžiausią (mažiausią) skaidinio dalį. Apie apskaičiuotų rezultatų teisingumą liudija (bet negarantuoja) (2.5.7) ar (2.5.8) sąryšis (nesudėtingas patikrinimas skaitiškai).

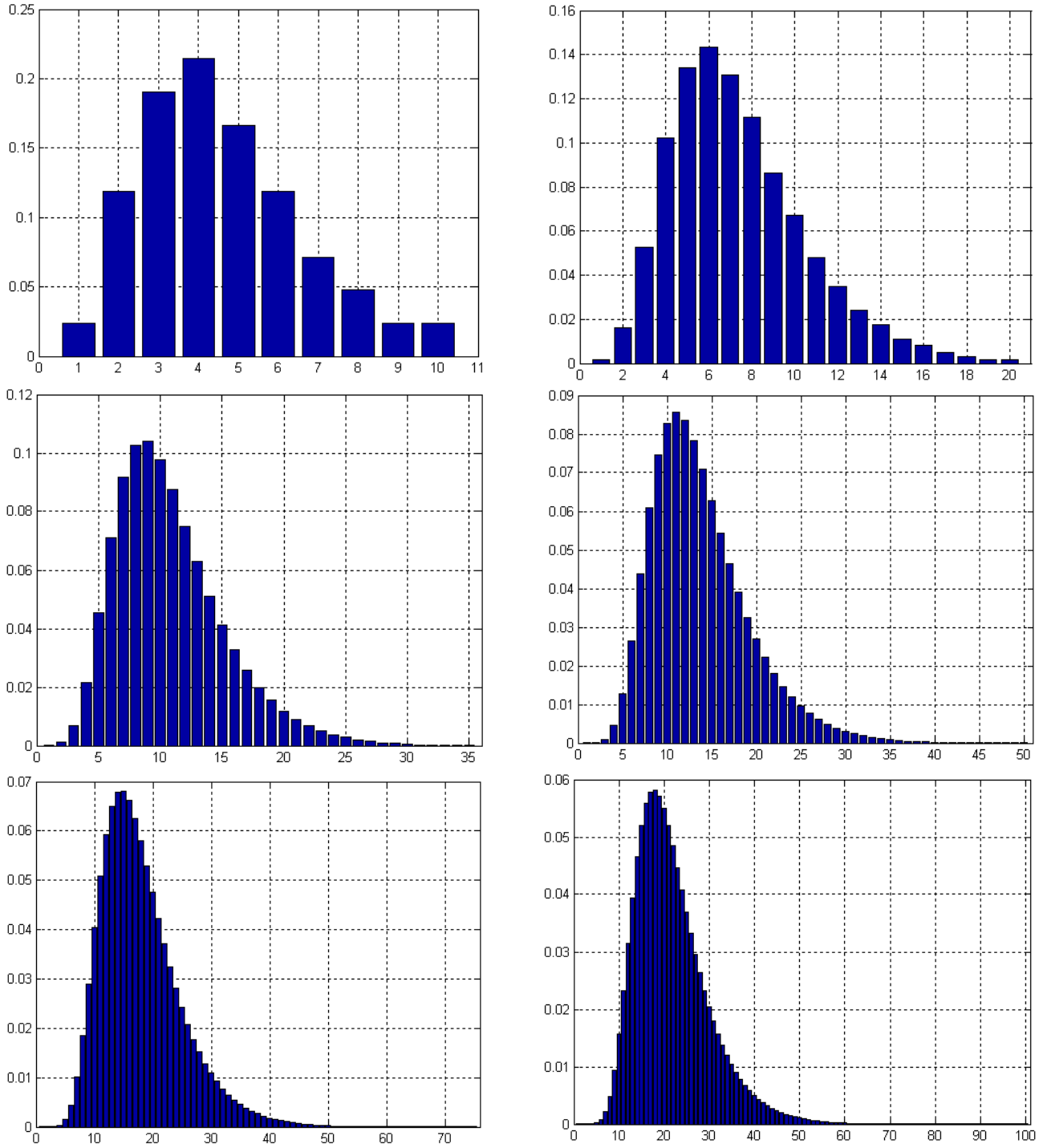
$$\sum_{i=1}^{N} max(N,i) = p(N) \qquad (2.5.7)$$

$$\sum_{i=1}^{N} min(N,i) = p(N) \qquad (2.5.8)$$

Skaičiuojant rekurentiškai (2.5.5) ir (2.5.6), kartotinai skaičiuojamos tos pačios funkcijos (kaip ir skaičiuojant (2.2.1) aprašytą skyrelyje *Skaičiaus skaidinių skaičius*), kuo galima įsitikinti pabraižius funkcijų iškvietimo medį (jis yra analogiškas pav.). Dėl to reikia skaičiuoti "nuo apačios", t.y. visiems skaičiams nuo 1 iki **N,** ir vietoje rekurentinio skaičiavimo tiesiog imti prieš tai apskaičiuotas reikšmes.
Skaičiuojant "nuo apačios" gaunama NxN matrica, kurios elementai *n,k* yra *max*(*n,k*) ir iš tokios matricos lengva gauti pasiskirstymo tankius visiems $n \le N$. Metodų sudėtingumas yra $O(N^3)$.

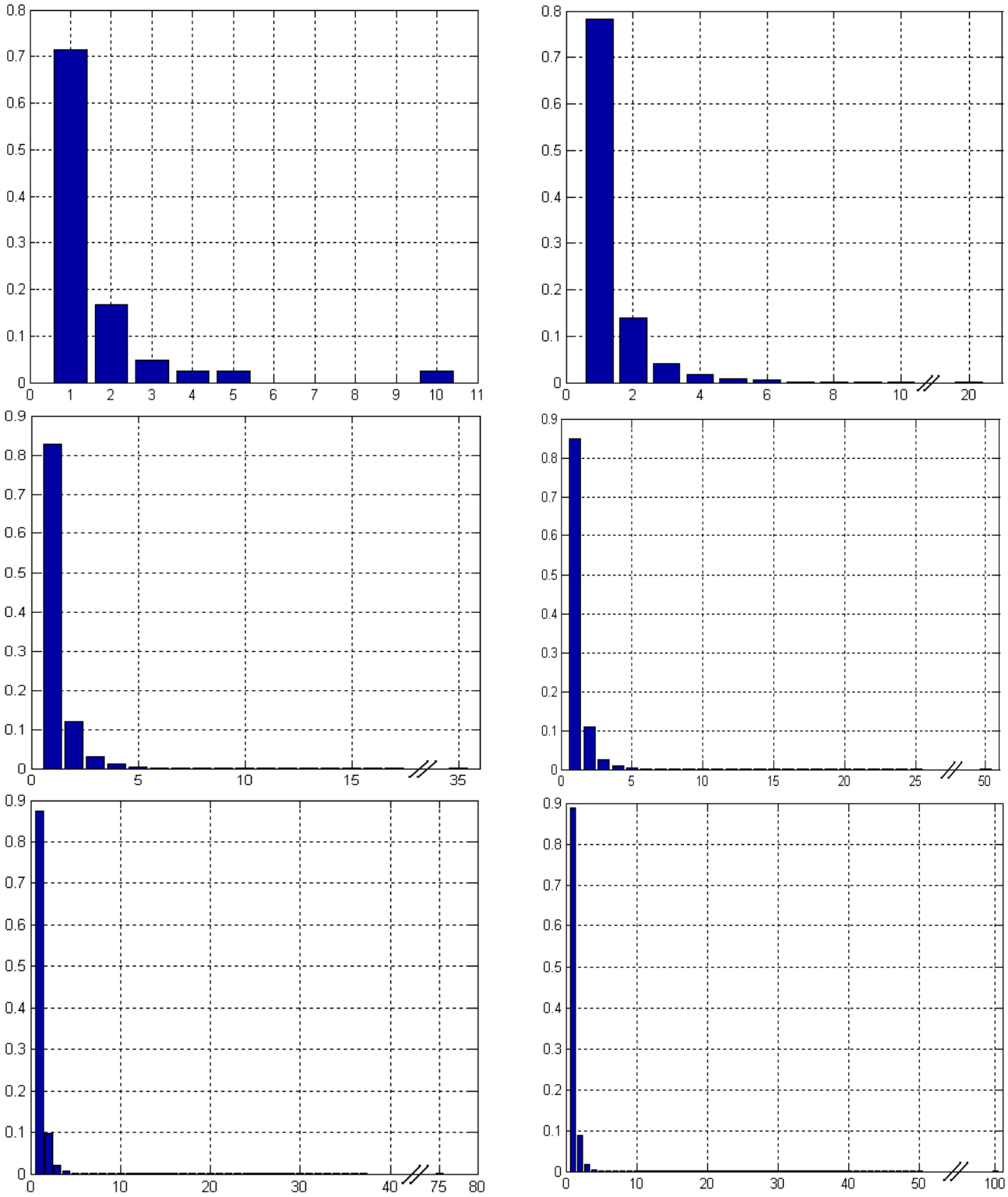
Priedai : `partitions_with_max_part.m` - skirstinių pagal didžiausią dalį sudarymo metodas.
`partitions_with_min_part.m` - skirstinių pagal mažiausią dalį sudarymo metodas.

Gauti skirstiniai neatitinka Binominio ar Puasono skirstinio, neatitinka jokio skirtinio esančio MATLAB R2011a Statistics Toolkit. Vertinti skirstinius artiniu (turbūt) neverta, nes turimi skirstiniai yra tikslūs, o ne jų įverčiai. Vertinti skirstinius artiniu gali būti naudinga norint gauti asimptotines savybes. Skirstinių grafikai pavaizduoti pav.2.5.1 - vaizduojami skaičių skaidinių skirstiniai pagal didžiausią dalį, skaičiams 10, 20, 35, 50, 75, 100.



pav. 2.7 Skaičių skaidinių skirstiniai pagal didžiausią skaidinio dalį

Skaičių skaidinių skirstinių pagal mažiausią dalį grafikai pav.2.5.2 - vaizduojami skirstiniai skaičiams 10, 20, 35, 50, 75, 100.



pav. 2.8 Skaičių skaidinių skirstiniai pagal mažiausią skaidinio dalį

2.5.2 Skaičiaus skaidinio generavimas gautų skirstinių pagalba

pmax(N,*k*) - žymi tikimybę, kad skaičiaus N skaidinyje, didžiausia dalis yra *k*. Reikia sugeneruoti atsitiktinį dydį, pasiskirsčiusį pagal duotą skirstinį, tai ir bus skaidinio didžiausia dalis. Kad sugeneruotume atsitiktinį dydį, pasiskirsčiusį pagal norimą skirstinį, reikia atsitiktinių skaičių, pasiskirsčiusių pagal tolyduįį tolygųjį skirstinį intervale [0, 1], generatoriaus, ir paties pasiskirstymo.

Skaidant skaičių, sudaroma jo didžiausia dalis, po to sekanti dalis, kuri yra mažesne arba lygi prieš tai esančiai ir t.t. Sugeneravus didžiausią dalį *d*, lieka neišskaidytas skaičius N-*d* , ir toliau jau skaidomas jis. Bet sudarinėti šiam (likusiam) skaičiui didžiausią dalį pagal skirstinį *pmax*(N-*d*,*k*) negalima, nes prieš tai sudaryta didžiausia dalis gali būti mažesnė už *pmax*(N-*d*,*k*) skirstinyje esančią dalį. (pavyzdžiui, skaidant 8-tą, pirma dalis gali būti 2, visos kitos dalys turi būti mažesnės, bet likusią dalį - 6 - įmanoma išskaidyti į (3,3)). Todėl skirstinį *pmax*(N-*d*,*k*) reikia "nukirpti", kad didžiausias galimas skaičius skirstinyje būtų ≤ *d*. T.y. *pmax*(N-*d*, *d*+*k*)=0, kai *k*≥1. Taip "apkerpant" skirstinį, jį reikia sunormuoti, kad visų tikimybių suma būtų lygi 1. "Apkirptą" skirstinį žymėsime *pmax*(N,*d*,*k*) (N - skaidomas skaičius, *d* - didžiausia galima reikšmė, *k* - generuojama dalis)).

Tada skaičius N skaidomas atliekant žingsnius :

Įvedama *likusi_dalis* = N, kuri yra neišskaidytoji dalis, t.y. skaidomas skaičius minus sudarytos dalys ir įvedama *d* = N.

1. Sugeneruojamas skaičius *d* pagal "apkirptą" skirstinį *pmax*(*likusi_dalis*, *d, k*), sugeneruotas skaičius laikomas didžiausia galima skaidinio dalimi (pavyzdys: skaidant 8 gali būti sugeneruotas 2-as, ir laikomas didžiausia dalimi).

2. Apskaičiuojama likusi skaidomo skaičiaus dalis *likusi_dalis* - *d* (pavyzdys: likusi dalis yra 6).

3. Jei sudarytos dalys nesudaro skaidomo skaičiaus, grįžtama į žingsnį **1**.

Tokiu būdu sudaromi skaidiniai yra vienodai galimi, t.y. visi skirstiniai turi vienodą tikimybę būti sudarytais, kas patikrinama kitame skyrelyje 2.5.3.

Priedai : `number_partition_random_straight_descending.m` - algoritmo realizacija.

2.5.3 Skaičiaus skaidinių generavimo tikimybių tyrimas

Kaip patikrinti, ar sudarytas skaičiaus skaidinio generavimo metodas generuoja skirtingus skaidinius su vienodomis tikimybėmis ?

Tai padaryta pasinaudojant visų skaidinių generavimo metodu. Sugeneruojami visi skaidiniai pasirinktam skaičiui ir kiekvienam skaidiniui, pradedant nuo didžiausios jo dalies, mažėjimo tvarka, kiekvienai daliai apskaičiuojama tikimybė būti sugeneruotai **2.5.2** skyriuje siūlomo metodo pagalba. Rezultate gaunamos kiekvieno skaidinio sugeneravimo tikimybės.

Išnagrinėtiems skaičiams, kurie yra nuo 1 iki 32, kiekvieno skaičiaus visiems skaidiniams, tikimybės yra vienodos, kas liudija, kad metodas generuoja skaičių skaidinius su vienodomis tikimybėmis. Skaičiaus skaidinių skaičius, atskiro skaidinio sugeneravimo tikimybės ir visų skaičiaus skaidinių sugeneravimo tikimybių suma (kuri turi būti lygi vienetui) pateikiami lentelėje 2.5.3.

Priedas `number_partition_random_analitical_test.m` yra metodo realizacija MATLAB kalba.

Skaidomas skaičius	Skaidinių skaičius	Skaidinio pasirinkimo tikimybe	Tikimybių suma	Skaidomas skaičius	Skaidinių skaičius	Skaidinio pasirinkimo tikimybe	Tikimybių suma
1	1	1	1.0000	17	297	0.00336	1.0000
2	2	0.5000	1.0000	18	385	0.002597	1.0000
3	3	0.3333	1.0000	19	490	0.00204	1.0000
4	5	0.2000	1.0000	20	627	0.00159	1.0000
5	7	0.1429	1.0000	21	792	0.00126	1.0000
6	11	0.0909	1.0000	22	1002	0.000998	1.0000
7	15	0.0667	1.0000	23	1255	0.000796	1.0000
8	22	0.0455	1.0000	24	1575	0.000634	1.0000
9	30	0.0333	1.0000	25	1958	0.000511	1.0000
10	42	0.0238	1.0000	26	2436	0.000411	1.0000
11	56	0.0179	1.0000	27	3010	0.000332	1.0000
12	77	0.0130	1.0000	28	3718	0.000269	1.0000
13	101	0.00990	1.0000	29	4565	0.000219	1.0000
14	135	0.00741	1.0000	30	5604	0.000178	1.0000
15	176	0.00568	1.0000	31	6842	0.000146	1.0000
16	231	0.00433	1.0000	32	8349	0.0001198	1.0000

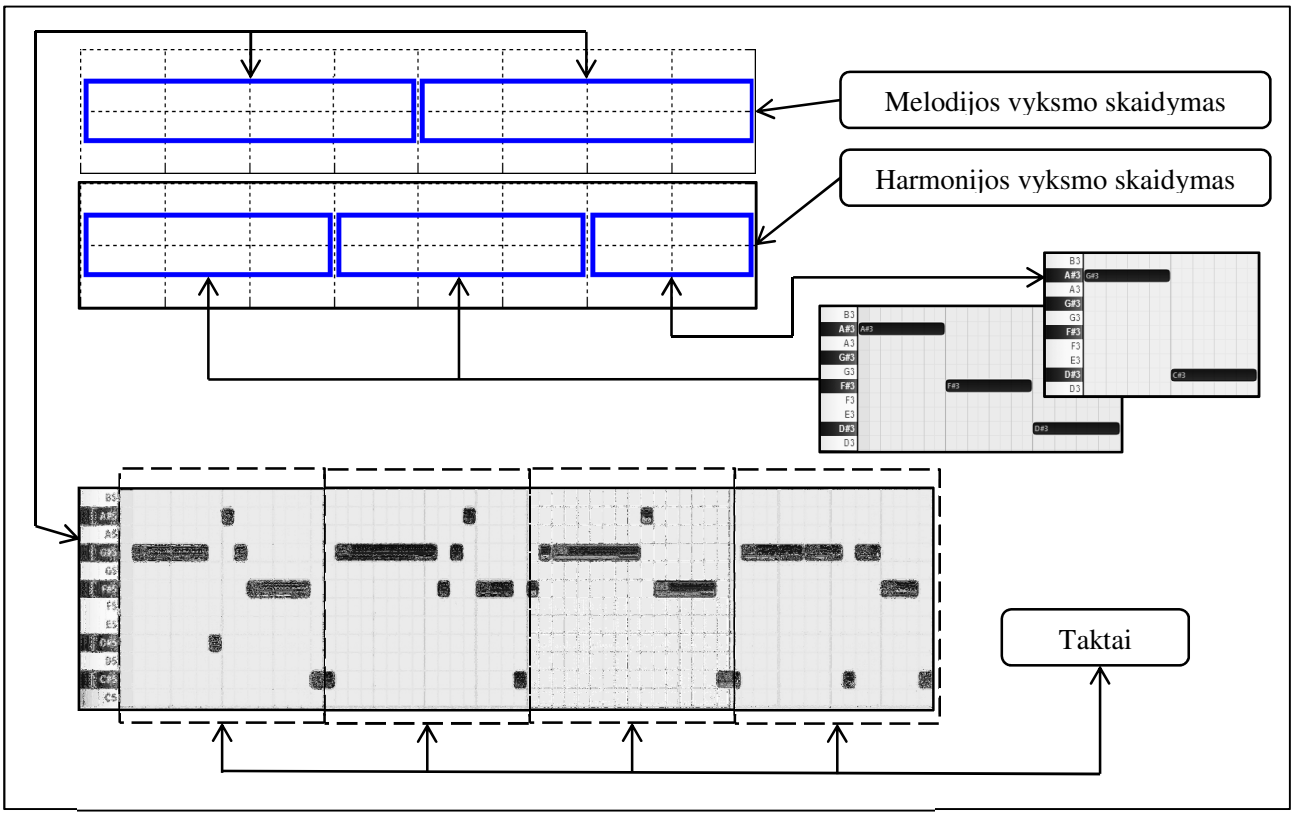
Lentelė 2.5.1 **S**kaičiaus skaidinių generavimo tikimybių tyrimas

Priedai : `number_partition_random_analitical_test.m` - algoritmo realizacija.

3 DISKUSIJA

Vyksmų sudarymui - skaičiaus skaidymui - naudojamas visų skaičiaus skaidinių sudarymo metodas. Įvertinus skaidinių kartotinumą, atitinkamai įvesto parametro (kartotinumo parametro) išrenkamas atitinkamas skaidinys ir užpildomas judėjimais. Metodas gali būti naudojamas vyksmams, kurių dydis neviršija ≈100, nes didesniems skaičiams skaičiavimai užtrunka ilgai. Didesniems skaičiams gali būti naudojami atsitiktinių skaidinių sudarymo metodai, skaidant "didelius" skaičius i kelis mažesnius, ir toliau naudojant visų skaidinių sudarymo metodą. Bet toks skaidymo būdas netirtas.

Metodo pademonstravimui sudarytas nedidelis "kūrinys" - pavyzdys pav.3.1 (taip pat yra prieduose). Pavyzdyje esanti kompozicija (vienas vyksmas) sudaryta iš dviejų instrumentų - melodijos ir boso. Melodijai vyksmas išskaidytas į 4 vienodas dalis - judėjimus, bosui vyksmas išskaidytas į 2 vienodas dalis - judėjimus. Judėjimai sudaromi pradedant nuo vieno takto ir modifikuojant jį. Taktas sudaromas pasirenkant atsitiktines natas iš vienos oktavos 5 natų - C#,D#,F#,G#,A# (juodų klavišų) ir suskaidant intervalą (takto intervalą - 4 dūžiai) į reikiamą dalių kiekį.



pav. 3.1 Nedidelis "kūrinys"

Priedai : Prisegtame dvd diske "M one of randoms 1.mp3" ir "M one of randoms 2.mp3" sudarytos kompozicijos įrašo dvi versijos: skambanti pianino garsu bei skambanti pianino ir smuiko garsu.

IŠVADOS

Sudaryta schema kūrinio struktūros sudarymui.

Išnagrinėtas skaičių skaidymo uždavinys, kuris panaudotas schemos dalių realizavimui.

Išnagrinėti skaidinių skaičiaus apskaičiavimo būdai ir ivertintas jų sudėtingumas laiko prasme.

Gautas įvertis - $O(N^2)$.

Sudarytas visų galimų skaidinių generavimo algoritmas.

Ištirtas asimptotinis algoritmo sudėtingumas laiko prasme - $O\left(exp\left(\pi\sqrt{2N/3}\right)\right)$.

Sudarytas visiškai atsitiktinio skaidymo sudarymo metodas.

Ištirtos gaunamų skaidinių gavimo tikimybės ir jų priklausomybė nuo skaidinio.

Gauta skaidinio sudarymo tikimybės išraiška.

Gauti skaičiaus skaidinių pasiskirstymai pagal didžiausią ir mažiausią dalį.

Sudarytas atsitiktinio skaidinio generavimo su vienodomis skaidinių pasirodymo tikimybėmis metodas.

ŠALTINIAI

[1] Gerhard Nierhaus, Algorithmic Composition, SpringerWienNewYork 2009

[2] Eduardo Reck Miranda, Composing music with computers, Focal Press 2002

[3] Howard Goodall's How Music Works, BBC, 2006

[4] Gareth Loy, Musimatics2, The MIT Press 2007

[5] Coursera, Introduction to digital sound design

[WM] http://mathworld.wolfram.com/PartitionFunctionP.html, 2013.05.25

PRIEDAI

Metodų realizacijos MATLAB kalba

partition_function_seq.m

```
function P_values = partition_function_seq(N)
    P_values = ones(N+1,1);

    for n = 1 : N
        REZ = 0;

        for k = 1 : n
            r1 = n - k*(3*k-1)/2;
            r2 = n - k*(3*k+1)/2;

            if r1 >= 0;
                r1 = P_values(r1+1);
            else
                r1 = 0;
            end

            if r2 >= 0;
                r2 = P_values(r2+1);
            else
                r2 = 0;
            end

            REZ = REZ + ((-1)^(k+1)) * (r1 + r2);
        end

        P_values(n+1) = REZ;
    end

    if N == 0;  P_values = 1;    end
    if N < 0;   P_values = 0;    end
end
```

number_partitioning.m

```
function Result = number_partitioning (N)

    MAX_ARRAY_SIZE = partition_function_seq(N);
    MAX_ARRAY_SIZE = MAX_ARRAY_SIZE (end);
    A = zeros (MAX_ARRAY_SIZE, N);      % matrix of partitions, (in rows)
    A(1,1) = N;

    L_I = 1;                            % L_I - last index
    N_I = 2;                            % N_I - now index
```

```
    for bN = 2 : N                      % bN - biggest number in partition
        for k = 1 : N/bN                % number of times bN is included
            for oP = 1 : L_I            % op  - older patterns to use for calculation

                if A(oP,1) >= k*bN

                    A(N_I,:)  = A(oP,:);
                    A(N_I,1)  = A(N_I,1) - k*bN;
                    A(N_I,bN) = k;

                    N_I = N_I + 1;

                end
            end
        end

        L_I = N_I - 1;
    end

    Result = A;
end
```

partitions_with_max_part.m

```
function D = partitions_with_max_part (N)

seq = partition_function_seq (N);

ALL_PARTITIONS = zeros(N);
ALL_PARTITIONS(1,1) = 1;
D = ALL_PARTITIONS(1,:);

for I = 2 : N
    D = zeros(1,N);

    for i = 0 : I/2
        D(I-i) = seq(i+1);
        k = I-i;
    end

    for i = k-1 : -1 : 1
        D(i) = seq(I-i+1);

        temp = ALL_PARTITIONS(I-i, :);
        temp = sum(temp(i+1 : I-i));

        D(i) = D(i) - temp;
    end

    ALL_PARTITIONS(I, :)= D;
end

D = ALL_PARTITIONS;
```

partitions_with_min_part.m

```
function D = partitions_with_min_part (N)

seq = partition_function_seq (N);

ALL_PARTITIONS = zeros(N);
ALL_PARTITIONS(1,1) = 1;

for I = 2 : N
    D = zeros(1,N);
    D(I) = 1;
    k = floor(I/2);

    for i = k : -1 : 1
        D(i) = seq(I-i+1);

        temp = ALL_PARTITIONS(I-i, :);
        temp = sum(temp(1 : i-1));

        D(i) = D(i) - temp;
    end

    ALL_PARTITIONS(I, :)= D;
end

D = ALL_PARTITIONS;
```

partitions2array.m

```
function Result = partitions2array (partitions)

    N     = size(partitions,2);
    P     = size(partitions,1);
    array = zeros(P,N);

    for p = 1 : P
        k = 1;
        for i = N : -1 : 1
            for j = 1 : partitions(p,i)
                array(p,k) = i;
                k = k + 1;
            end
        end
    end

    Result = array;
end
```

number_partition_random_straight_descending.m

```
function REZ = number_partition_random_straight_descending (N)

    % partitions with biggest number count
    D = partitions_with_max_part (N);
    cpD = D;

    % cumulative probability distribution %%%%%%%%%%%%%%%%%%%%%
    for i = 1 : N
        cpD(i,:) = cpD(i,:) / sum(cpD(i,:));

        for j = 2 : N
            cpD(i,j) = cpD(i,j) + cpD(i,j-1);
        end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    partition    = zeros(N,1);
    part_left    = N;
    biggest_part = N;

    k = 1;

    while part_left > 0

        r    = rand;
        cpDt = cpD(part_left, 1:biggest_part)/cpD(part_left, biggest_part);

        if (r <= cpDt(1))
            biggest_part = 1;
        else
            for j = 1 : biggest_part-1
                if (r > cpDt(j)) && (r <= cpDt(j+1))
                    biggest_part = j+1;
                end
            end
        end

        partition(k) = biggest_part;
        partition(k) = sort(partition(k));
        k = k + 1;
        part_left = part_left - biggest_part;

    end

    temp = length(partition);
    for i = 1 : temp
        if partition(i) == 0;
            partition = partition(1 : (i-1));
            break
        end
    end


REZ = partition;
```

number_partition_random_analitical_test

```
function REZ = number_partition_random_analitical_test (N)

P = partition_function_seq(N);
P = P(end);

% partitions with biggest number count %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
D = partitions_with_max_part (N);
pD  = D;

% cumulative probability distribution %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1 : N
    pD (i,:) = pD(i,:) / sum(pD(i,:));   % probability distribution
    cpD(i,:) = pD(i,:);                  % cumulative probability distribution

    for j = 2 : N
        cpD(i,j) = cpD(i,j) + cpD(i,j-1);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ALL_PARTITIONS = number_partitioning (N);
ALL_PARTITIONS = partitions2array (ALL_PARTITIONS);

part_freq = zeros(P, 1);

for i = 1 : P

    part_left    = N;
    biggest_part = N;
    part_prob    = 1;

    for j = 1 : N

        pDt = pD(part_left, 1:biggest_part)/cpD(part_left, biggest_part);

        biggest_part = ALL_PARTITIONS (i,j);

        part_prob = part_prob * pDt(biggest_part);

        part_left = part_left - biggest_part;

        if part_left == 0; break; end

    end

    part_freq(i) = part_prob;

end

REZ = part_freq;
```