VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

MATEMATINIO MODELIAVIMO KATEDRA

Danas Motiejauskas

PAPRASTO SKYLĖTO DAUGIAKAMPIO SKAIDYMO ALGORITMAI

ALGORITHMS FOR DECOMPOSITION OF SIMPLE POLYGON WITH HOLES

Baigiamasis magistro darbas

Technomatematikos studijų programa, valstybinis kodas 62401P109

Matematinio modeliavimo specializacija

Matematikos studijų kryptis

Vilnius, 2010

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

MATEMATINIO MODELIAVIMO KATEDRA

TVIRTINU
Katedros vedėjas

_____
(Parašas)

_____
(Vardas, pavardė)

_____
(Data)

Danas Motiejauskas

PAPRASTO SKYLĖTO DAUGIAKAMPIO SKAIDYMO ALGORITMAI

ALGORITHMS FOR DECOMPOSITION OF SIMPLE POLYGON WITH HOLES

Baigiamasis magistro darbas

Technomatematikos studijų programa, valstybinis kodas 62401P109

Matematinio modeliavimo specializacija

Matematikos studijų kryptis

Vadovas_____ _____ _____
(Moksl. laipsnis/pedag. vardas, vardas, pavardė) (Parašas) (Data)

Konsultantas_____ _____ _____
(Moksl. laipsnis/pedag. vardas, vardas, pavardė) (Parašas) (Data)

Konsultantas_____ _____ _____
(Moksl. laipsnis/pedag. vardas, vardas, pavardė) (Parašas) (Data)

Vilnius, 2010

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
MATEMATINIO MODELIAVIMO KATEDRA

TVIRTINU
Katedros vedėjas

_____
(Parašas)

_____
(Vardas, pavardė)

_____
(Data)

Matematikos studijų kryptis

Technomatematikos studijų programa, valstybinis kodas 62401P109

Matematinio modeliavimo specializacija

BAIGIAMOJO MAGISTRO DARBO UŽDUOTIS

…….....Nr. .....
Vilnius

Studentui (ei) .....……...…….....….....….....….....
(Vardas, pavardė)

Baigiamojo darbo tema: .....…….....

.....

.....

patvirtinta 200…m. ……..…… d. dekano potvarkiu Nr. …….

Baigiamojo darbo užbaigimo terminas 201…m. ……..…… d.

BAIGIAMOJO DARBO UŽDUOTIS:

.....

.....

.....

.....

.....

.....

.....

Baigiamojo darbo rengimo konsultantai:

……

.....

.....

(Moksl. laipsnis/pedag. vardas, vardas, pavardė)

Vadovas ....
(Parašas) (Moksl. laipsnis/pedag.vardas, vardas, pavardė)

Užduotį gavau

……..
(Parašas)

……..
(Vardas, pavardė)

……..
(Data)

Vilniaus Gedimino technikos universitetas
Fundamentinių mokslų fakultetas
Matematinio modeliavimo katedra

ISBN ISSN
Egz. sk. ……..
Data ….-….-….

Technomatematikos studijų programos baigiamasis magistro darbas
Pavadinimas **Paprasto skylėto daugiakampio skaidymo algoritmai**
Autorius **Danas Motiejauskas** Vadovas doc. dr. **Olga Suboč**

Kalba
[X] lietuvių
[ ] užsienio

Anotacija

Baigiamajame magistro darbe nagrinėjama paprasto skylėto daugiakampio skaidymo į dalis, kurių viršūnių skaičius neviršyja nustatyto skaičiaus problema. Apibrėžiamas uždavinys ir jo svarba. Apžvelgiami egzistuojantys skaidymo algoritmai, padedantys išspręsti uždavinį, bei jų realizacijos. Pateikiamos trianguliacijos ir padalinimo į apytiksliai iškilius daugiakampius algoritmų modifikacijos, jų privalumai ir trūkumai. Įvertinamas šių modifikuotų algoritmų sudėtingumas. Eksperimentinėje dalyje pateikiami skaičiavimo eksperimentų rezultatai, jų analizė ir palyginimas su teoriniais algoritmų sudėtingumo įverčiais. Remiantis skaičiavimo eksperimentų rezultatais pateikiamos išvados ir siūlymai.

Darbą sudaro 6 dalys: įvadas, uždavinio formulavimas, sprendimo algoritmai, skaičiavimo eksperimentų rezultatai, išvados ir pasiūlymai, literatūros sąrašas.

Darbo apimtis – 26 p. teksto be priedų, 10 iliustr., 6 lent., 23 bibliografiniai šaltiniai.

Atskirai pridedami darbo priedai.

Prasminiai žodžiai: paprastas daugiakampis, skylėtas daugiakampis, skaidymas, trianguliacija, iškilus daugiakampis, skaičiuojamoji geometrija.

Vilnius Gediminas Technical University
Faculty of Fundamental Sciences
Department of Mathematical Modelling

ISBN ISSN
Copies No. ……
Date ….-….-….

Technomathematics study programme master thesis.

Title: **Algorithms for decomposition of simple polygon with holes**

Author **Danas Motiejauskas** Academic supervisor **doc.dr. Olga Suboč**

Thesis language

[X] Lithuanian

[ ] Foreign (English)

Annotation

This study deals with decomposition of simple polygon with holes into components so that every piece does not exceed some defined number of vertices. We define the problem and its appliances. Existing studies and algorithms for polygon decomposition are covered. We propose modifications of polygon triangulation and approximate convex decomposition algorithms. Also the complexity analysis of both algorithms is made. In the experimental part of the work results of computing experiments are presented, analyzed and compared to the theoretical complexity bounds.

Structure: introduction, problem definition, algorithms for solving the problem, results of the experiments, conclusions and suggestions, references.

Thesis consist of: 26 p. text without appendixes, 10 pictures, 6 tables, 23 bibliographical entries.

Appendix included.

Keywords: simple polygon with holes, decomposition, parts' number of vertices, triangulation, approximate convex decomposition, computational geometry.

TURINYS

PAVEIKSLŲ SĄRAŠAS 7
LENTELIŲ SĄRAŠAS 7
ĮVADAS 8
1. UŽDAVINIO FORMULAVIMAS 8
2. SPRENDIMO ALGORITMAI 9
2.1. Daugiakampio skaidymo į apytiksliai iškilius daugiakampius algoritmas 10
2.2. Daugiakampio trianguliacijos algoritmo modifikacija 17
3. SKAIČIAVIMO EKSPERIMENTŲ REZULTATAI 21
IŠVADOS IR PASIŪLYMAI 24
LITERATŪRA 25

PAVEIKSLŲ SĄRAŠAS

1 pav. a) paprastas daugiakampis su skylėmis, b) galimas padalinimas 9
2 pav. Daugiakampio kontūrai ir iškilus lukštas (pagal [18]) 11
3 pav. Briaunos (5,7) ir (8,1) yra tiltai, o jų atitinkamos kišenės: (5;6), (6;7) ir (8;9), (9;0), (0;1) (pagal [18]) 11
4 pav. Trumpiausi keliai nuo tilto iki kiekvienos įdubusios viršūnės 12
5 pav. a) ACD algoritmo skaičiavimų rezultatas, b) tikslus padalinimas į iškilius daugiakampius (pagal [18]) 13
6 pav. a) ACD algoritmo skaičiavimų rezultatas, b) tikslus padalinimas į iškilius daugiakampius ..14
7 pav. Iš viršūnės *v* matoma daugiau viršūnių nei iš viršūnės *w* 14
8 pav. Iš viršūnių *v* ir *w* nesimato nei viena išorinio kontūro viršūnė 15
9 pav. Trianguliuotas daugiakampis is trianguliacijos šalinimas 17
10 pav. *a)* apylygis daugiakampio padalinimas, *b)* netolygus daugiakampio padalinimas, visos dalys yra 10 viršūnių dydžio, išskyrus viršutinę – trikampį 18

LENTELIŲ SĄRAŠAS

1 lentelė. Modifikuotos trianguliacijos algoritmo skaičiavimo trukmės priklausomybė nuo didžiausio leidžiamo dalių viršūnių skaičiaus V_{max} 22
2 lentelė. Modifikuotos trianguliacijos algoritmo skaičiavimo trukmės priklausomybė nuo pradinio daugiakampio viršūnių skaičiaus N, kai $V_{max} = 10$ 22
3 lentelė. Modifikuoto ACD algoritmo skaičiavimo trukmės priklausomybė nuo didžiausio leidžiamo dalių viršūnių skaičiaus V_{max} 23
4 lentelė. Modifikuoto ACD algoritmo skaičiavimo trukmės priklausomybė nuo pradinio daugiakampio viršūnių skaičiaus N, kai $V_{max} = 10$ 23
5 lentelė. Modifikuotos trianguliacijos ir modifikuoto ACD algoritmų gautų dalių skaičiai 23
6 lentelė. Modifikuotos trianguliacijos ir modifikuoto ACD algoritmų gautų dalių viršūnių skaičių vidurkiai 23

ĮVADAS

Šiuolaikinės geografijos informacijos sistemos (GIS) leidžia kaupti, redaguoti ir platinti labai didelio tikslumo geografinius duomenis. Didelė dalis objektų yra saugoma daugiakampių pavidalu (pvz., įvairių teritorijų ribos, miškai, ežerai ir pan.). Kadangi duomenys yra didelio tikslumo, daugiakampį sudarančių viršūnių skaičius gali būti labai didelis ir siekti net kelis milijonus. Tačiau kai kurie duomenų saugojimo formatai ir programos, skirtos darbui su geografiniais duomenimis, turi tam tikrų apribojimų – tai nesugebėjimas nuskaityti ir interpretuoti skyles daugiakampyje bei vieno objekto viršūnių skaičiaus apribojimai. Skylė daugiakampyje aprašoma kaip papildomas vienas ar keletas kontūrų. Jei programa ar duomenų saugojimo formatas neturi galimybės nuskaityti arba įrašyti skylių kontūrus – prarandami duomenys arba objektas yra neteisingai atvaizduojamas. Tas pats atsitinka kai ribojamas objekto viršūnių skaičius. Pavyzdžiui, bandome išsaugoti daugiakampį iš 1000 viršūnių formatu, kuris palaiko tik 256 viršūnes, arba tą patį daugiakampį atvaizduojame programa, kuri iš duomenų rinkmenos nuskaito daugiausiai 500 viršūnių. Abiem atvejais prarandame informaciją.

Šių problemų sprendimas – daugiakampio skaidymas į mažesnes dalis. Toks suskaidymas taip pat gali supaprastinti sudėtingas duomenų apdorojimo operacijas. Viena iš dažniausiai pasitaikančių problemų – skaičiuojant visiškai užpildoma kompiuterio operatyvioji atmintinė ir operacinė sistema dalį duomenų iš jos nukelia į standųjį diską, o tai gali ilgai užtrukti. Taip atsitinka atliekant sudėtingus skaičiavimus ir kai duomenis sudaro vienas, bet labai didelis daugiakampis. Suskaidžius jį į mažesnes dalis kiekviena jų analizuojama atskirai, dėl to yra mažesnės atminties sąnaudos, ir operatyvioji atmintinė jau nebeužpildoma iki tos ribos, kai ją reikia atlaisvinti nukėlus duomenis į standųjį diską.

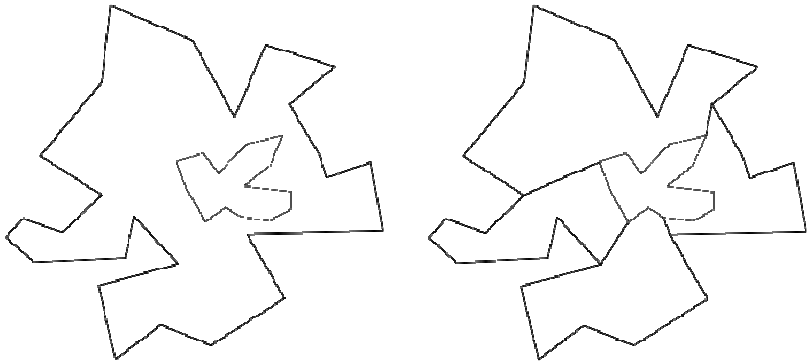
1. UŽDAVINIO FORMULAVIMAS

Turime paprastą daugiakampį (1 pav. a). Jį reikia sudalinti į paprastus daugiakampius taip (1 pav. b), kad:

- juose nebūtų skylių,
- juos sudarančių taškų skaičius neviršytų tam tikro nustatyto skaičiaus $V_{\max}$,
- dalių būtų kuo mažiau.

Paprastas daugiakampis yra kontūrų aibė δP, kur δP_0 yra išorinis kontūras, o likę kontūrai δP_i yra skylių kontūrai. Daugiakampis yra paprastas, jei jokios nesusijungiančios briaunos nesikerta. Daugiakampius, esančius skylėse, traktuosime kaip atskirus daugiakampius, ir juos nagrinėsime atskirai.

Daugiakampis nebūtinai yra iškilus, dėl to iškyla sunkumų. Dalinti bet kaip į dalis, kurias sudarančių taškų skaičius yra lygus nustatytam, negalime, nes galime padalinti į daugiau nei dvi dalis. Be to, skylių kontūras irgi gali būti sudėtingas, kas taip pat sukelia sunkumų.



a *b*

1 pav. a) paprastas daugiakampis su skylėmis, b) galimas padalinimas

2. SPRENDIMO ALGORITMAI

Daugiakampių skaidymo uždavinys yra gana plačiai išnagrinėtas ir yra taikomas daugelyje įvairių uždavinių [15]. Dažniausniai daugiakampis skaidomas į paprastesnius komponentus tam, kad būtų supaprastintas kitas uždavinys. Skirtingiems taikomiesiems uždaviniams efektyviai spręsti yra suformuluota įvairių sąlygų, kurias turi atitikti pradiniai duomenys [5]: 45–61] (pvz., Garey triangliacijos algoritmas efektyviai trianguliuoja tik monotoniškus daugiakampius). Todėl yra įvairių algoritmų, skaidančių daugiakampius taip, kad jie atitiktų tas sąlygas, pvz., monotoniškumas [5]: 45–61]; iškilumas [6], [12], [14], [16]; spiralės [10], [14], žvaigždės [1], [3] formos daugiakampiai; mažiausias galimas dalių perimetras [19] arba jų skaičius [14]. Taip pat yra tyrimų, susijusių su daugiakampio skaidymu, kai yra apbrėžtas viršūnių skaičius. Tai trianguliacijos [5], [7], [13] ir skaidymo į keturkampius [9], [21] algoritmai. Bet kai kurie iš šių algoritmų naudoja Štainerio taškus, kadangi tokie algoritmai yra labiau pritaikyti dalijimo tinklo generavimui, naudojamam inžineriniuose skaičiavimuose. Dėl to jų naudojimas suformuluotam uždaviniui spresti yra ribotas.

Kadangi daugiakampis nebūtinai yra iškilus, galime pasinaudoti algoritmais, kurie suskaldo daugiakampį į iškilius daugiakampius. Iškilų daugiakampį padalinti į mažesnius, atsižvelgant į anksčiau aptartas sąlygas, yra labai paprasta. Tačiau skaidymo į iškilius daugiakampius algoritmai yra ganėtinai sudėtingi – $O(n^3)$ sudetingumo [15]. Be to, gauti iškilieji daugiakampiai yra pakankamai maži ir jų yra labai daug. Dėl to gali iškilti nemalonių situacijų. Tarkime, turime daugiakampį, kurį padalinus į dvi dalis, uždavinys bus išspręstas. Sudalinus jį į iškiliuosius daugiakampius uždavinys taip pat gali būti išspręstas, tačiau daugiakampių bus labai daug ir jie bus labai maži. Dar vienas sunkumas – visi tikslaus dalinimo į iškilius daugiakampius algoritmai, atsiradus skylėms, tampa

nepolinominio sudėtingumo. Akivaizdu, kad spręsti nepolinominio sudėtingumo uždavinį tam, kad būtų galima palengvinti sau darbą, yra neprotinga.

Be tikslaus skaidymo algoritmų yra ir apylikslio padalinimo į iškiliuosius daugiakampius algoritmas (Approximate Convex Decomposition, ACD [18]). Algoritmas gali būti sustabdomas tada, kai įdubų skaičius ir gylis sumažėja iki nustatyto parametro, pvz., kai pašalinamos ypač gilios įdubos. Be to, algoritmas apdoroja ir skyles, dėl to jo sudėtingumas beveik nepadidėja.

Uždavinį galime performuluoti taip – turimą daugiakampį padalinti į *n*-kampius. Jei *n* lygus 3, gauname trianguliacijos uždavinį. Kadangi daugiakampį visada galime trianguliuoti ir, be to, galime tai padaryti labai greitai (paprasti algoritmai yra $O(n \log n)$ sudėtingumo, bet yra ir $O(n)$ sudėtingumo metodų), patogu naudoti šį metodą suformuluotam uždaviniui spręsti.

2.1. Daugiakampio skaidymo į apytiksliai iškilius daugiakampius algoritmas

Šio algoritmo kūrėjai pastebėjo, kad uždaviniai, kuriuos yra paprasčiau spręsti, kai daugiakampiai yra iškilūs, panašiai supaprastėja, kai daugiakampiai yra *beveik* iškilūs. Tai yra, naudojant skaidymo į apytiksliai iškilius daugiakampius algoritmą, kitus uždavinius spręsti yra beveik taip lengva kaip ir prieš tai, tačiau skaidymas žymiai supaprastėja.

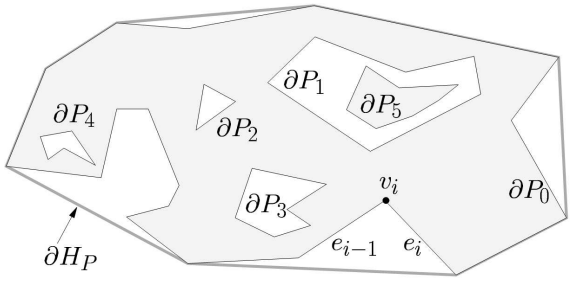
Metodo bendros idėjos:

- apskaičiuoti kiekvienos daugiakampio viršūnės įdubos laipsnį;
- panaikinti skyles, t.y. perkirpti daugiakampį taip, kad skylės kontūras taptų išoriniu kontūru. Skylių kirpimas įvyksta pirmas dėl to, kad skylės įdubos laipsnis yra nustatomas kitaip nei išorinio kontūro viršūnių, dėl to yra visada didesnis nei bet kurios viršūnės ant išorinio kontūro;
- perkirpti daugiakampį nuo viršūnės su didžiausiu įdubos laipsniu iki patogiausios. Patogiausia viršūnė randama indeksuojant viršūnes pagal jos atstumą iki pradinės viršūnės ir pagal tai, kiek sumažės įdubos laipsnis.

Toliau pateiksime platesnį algoritmo aprašymą bei galimybes jį pritaikyti mūsų suformuluotam uždaviniui.

Turime paprastą daugiakampį. Paprastas daugiakampis yra kontūrų aibė, kur δP_0 yra išorinis kontūras, o likę – skylių kontūrai. Daugiakampis yra paprastas, jei jokios nesusijungiančios briaunos nesikerta.

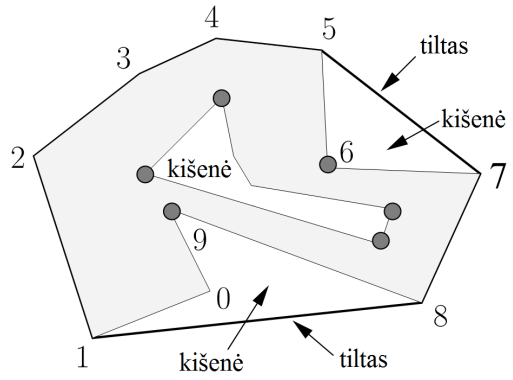
Daugiakampio *P iškilusis lukštas* H_P – tai mažiausias iškilus daugiakampis, į kurį telpa *P* (2 pav.). Jei $P = H_P$, sakome, kad *P* yra iškilus. *P* viršūnės, kurios nėra H_P viršūnės, vadinamos *įdubomis*, vidinis kampas tarp tokios viršūnės briaunų yra didesnis už 180°. Daugiakampio iškilus lukštas greitai apskaičiuojamas Shamos algoritmu [22].



2 pav. Daugiakampio kontūrai ir iškilus lukštas (pagal [18])

Įdubos laipsnio matavimui šiame algoritme vartojamos *tilto* ir *kišenės* sąvokos. *Tiltas* yra iškilaus lukšto briaunos, sujungiančios nesančias greta daugiakampio kontūro δP_0 viršūnes, t. y. TILTAI(P) = $\delta H_P \backslash \delta P$. *Kišenė* yra didžiausia grandinė briaunų, nesančių iškilaus lukšto kontūre, t. y. KIŠENĖS(P) = $\delta P \backslash \delta H_P$. Tokiu būdu įdubos gali būti tik kišenėse. Kiekvienas tiltas turi atitinkamą kišenę, t. y. δP_0 briaunų grandinę, kurios pradžia ir pabaiga sutampa su tilto pradžia ir pabaiga. Skylės irgi yra kišenės, tačiau jos neturi atitinkamų tiltų (3 pav.).

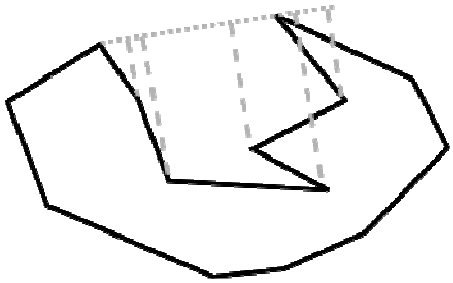
Algoritmas vadovaujasi *skaldyk ir valdyk* principu – daugiakampis dalinamas į 2 dalis, toliau yra dalinama kiekviena iš dalių ir t. t. Algoritmas rekursiškai dalina daugiakampį tol, kol didžiausias įdubos laipsnis yra didesnis už nustatomą parametrą τ. Parametras τ vadinamas neiškilumo paklaida. Kuo labiau mažės τ, tuo smulkiau bus dalinamas daugiakampis, kad jį sudarančių daugiakampių įdubos laipsnis neviršytų τ. Akivaizdu, kad kai $\tau = 0$, daugiakampis bus padalinamas į visiškai iškilius gabalus.



3 pav. Briaunos (5; 7) ir (8; 1) yra tiltai, o jų atitinkamos kišenės: (5; 6), (6; 7) ir (8; 9), (9; 0), (0; 1) (pagal [18])

Pirmas algoritmo žingsnis – apskaičiuoti kiekvienos viršūnės įdubos laipsnį. Jei viršūnė priklauso išoriniam kontūrui – įdubos laipsnis yra atstumas nuo jos iki atitinkamo tilto (4 pav.). Jei viršūnė priklauso skylės kontūrui – elgiamasi kitaip. Randamos 2 tolimiausios viena nuo kitos skylės kontūro viršūnės p_i ir $cw(p_i)$. Apskaičiuojamas atstumas nuo šių dviejų viršūnių iki išorinio kontūro. Iš šių dviejų atstumų pasirenkamas mažesnis, prie jo pridedamas atstumas nuo p_i ir $cw(p_i)$ ir maksimalus išorinio kontūro viršūnių įdubos laipsnis. Gauname visos skylės įdubos laipsnį, kuris

yra didesnis nei bet kurios išorinio kontūro viršūnės įdubos laipsnis. Tokiu būdu pirmiausia atsikratoma skylių.



4 pav. Trumpiausi keliai nuo tilto iki kiekvienos įdubusios viršūnės

Tolesnis algoritmo žingsnis yra surasti viršūnę su maksimaliu įdubos laipsniu. Ji bus kirpimo linijos pradžia. Šios linijos pabaiga randama suteikus viršūnėms indeksus: 0, jei per ją kerpant daugiakampį padalinsime į daugiau nei 2 gabalus, kitais atvejais – skaičių pagal formulę:

$$ind_i = \frac{1+\alpha \cdot \text{concave}(i)}{\beta \cdot \text{dist}(i,r)}, \quad (1)$$

čia concave*(i)* – viršūnės *i* įdubos laipsnis, dist*(i, r)* – atstumas nuo viršūnės *i* iki kirpimo linijos pradžios *r*, α ir β – pasirenkami parametrai. Šis indeksas įvertina kiek sumažės daugiakampio bendras įdubimo laipsnis, jei kirpimo linijos pabaiga bus tam tikra viršūnė. Taip indeksas įvertina ir žmogiškąjį faktorių – žmogus yra linkęs kirpti per siauriausią vietą, t. y. taip, kad kirpimo linija būtų kuo trumpesnė. Todėl indekso skaičiavimui naudojamas ir kirpimo linijos ilgis. Suskaičiavę indeksus kirpimo linijos pabaigos viršūnę pasirenkame taip, kad jos indeksas būtų didžiausias. Atlikę kirpimą gauname 2 daugiakampius. Tada kiekvienam iš jų taikome tokį pat algoritmą.

Nors aprašytas algoritmas sprendžia kitokį uždavinį nei mūsų suformuluotas, bet jį galima pritaikyti. Vienas iš būdų yra toks:

- Atsikratyti skylių pagal ACD algoritmą.
- Skaidyti daugiakampį pagal ACD, kol visi komponentų kontūrai bus ne ilgesni nei nustatyta.
- Jei ACD algoritmo vykdymas sustojo, o komponentų kontūrai vis dar ilgesni nei nustatyta – daliname komponentus taip, kaip mums yra patogu. Jokių problemų neturėtų iškilti, nes komponentai yra iškilūs arba beveik iškilūs.

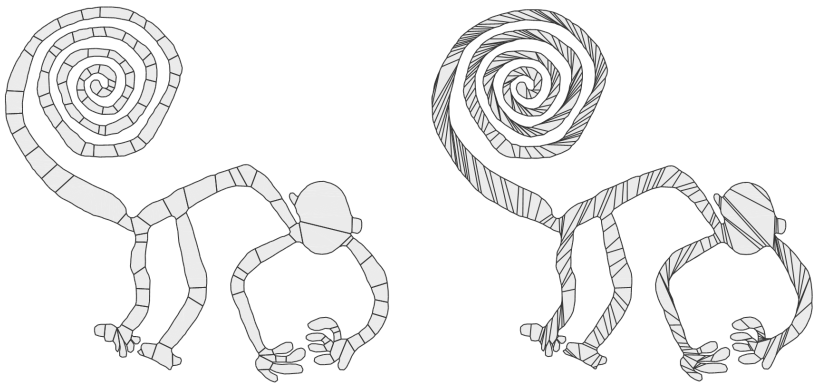
Metodo privalumai:

- Apdorojamos skylės.
- Greitesnis nei skaidymo į tiksliai iškilius daugiakampius algoritmai – $O(nr)$ sudėtingumo, čia n – viršunių kiekis, r – įdubų, gilesnių nei τ**,** skaičius. Mūsų atveju gali neprireikti naikinti visų įdubų, todėl vykdymas bus greitesnis.

- Gaunami komponentai žmogaus supratimui yra pakankamai prasmingi ir gražūs (5 pav. a), t. y. nėra daugybės ilgų ir siaurų daugiakampių kaip tikslaus skaidymo algoritmuose (5 pav. b).
- Pradėti kirpti iš labiausiai įdubusios viršūnės yra protinga, nes iš labiau įdubusios viršūnės galima matyti daugiau kitų viršūnių, todėl galima išrinkti labiausiai atitinkančią paieškos kriterijus viršūnę.
- Skaidymo linijos pasirinkimo kriterijus galima nustatyti pačiam. Tokiu būdu galime gauti vienos ar kitos norimos formos daugiakampius.

Metodo trūkumai:

- Tarkime, turime daugiakampį, kurį padalinus į dvi dalis, uždavinys bus išspręstas. Tuo tarpu ACD algoritmus kirps daugiakampį į didelį ir labai mažą. Po keleto žingsnių turėsime vieną didelį daugiakampį ir daug mažų. Atlikus kažkiek skaičiavimų mūsų uždavinys liko neišspręstas, tik iš dalies supaprastintas.
- Apskritai komponentų skaičius bus didesnis nei įmanomas mažiausias.



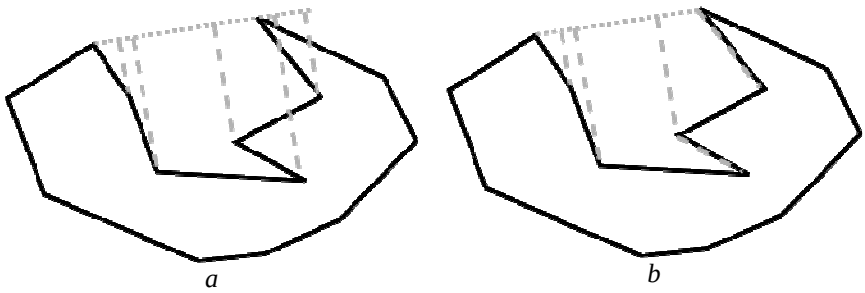
a *b*

5 pav. a) ACD algoritmo skaičiavimų rezultatas, b) tikslus padalinimas į iškilius daugiakampius (pagal [18])

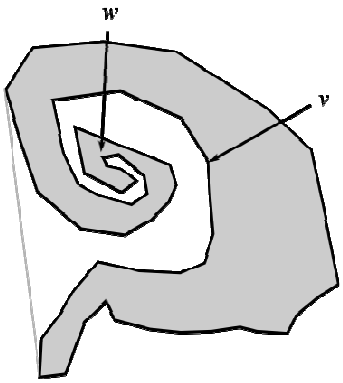
Skaidymo į apytiksliai iškilius daugiakampius metodą galima šiek tiek modifikuoti. Kai ieškoma, į kurią viršūnę iš labiausiai įdubusios reikia kirpti, galime viršūnėms suteikti indeksą ne tik atsižvelgdami į atstumą ir jos įdubos laipsnį, bet ir į mūsų suformuluoto uždavinio specifiką. Visas daugiakampio viršūnes pernumeruojame taip, kad kirpimo pradžios viršūnė būtų pirma. Tada viršūnė *w*, kurios naujas numeris yra lygus V_{max}, bus labiausiai tinkama kirpimui. Kitos viršūnės, esančios atstumu V_{max}–2 nuo *w*, taip pat bus tinkamos kirpimui. Pavyzdžiui, perkirpus į pirmą viršūnę, kuri yra V_{max}-2 atstumu nuo *w*, gauname tokį daugiakampį, kad jį perkirpus pusiau (jei įmanoma) gausime 2 daugiakampius, kurių viršūnių skaičius lygus V_{max}. Pažymėsime visas viršūnės, tinkančias kirpimui didesniu indeksu, nei visas kitas viršūnes. Tada tą skaičių pridėsime prie pradinio in-

dekso (1), ir gausime bendrą indeksą. Šis indeksas įvertina ne tik kiekvienos viršūnės įtaką daugiakampio bendram įdubos laipsniui, bet ir gautų dalių viršūnių skaičių.

ACD metodo privalumas – kirpimo pradžia yra viršūnė su didžiausiu įdubos laipsniu. Tokiu būdu mes panaikiname didžiausią įdubą, dėl to gauti daugiakampiai yra labiau iškilūs nei pradinis. Tokiuose daugiakampiuose iš kurios nors viršūnės bus matoma daugiau viršūnių, todėl bus didesnė tikimybė, kad tiksliau atlikti padalijimą. Įdubos laipsnį algoritmo kūrėjai siūlo skaičiuoti dviem metodais. Pirmas ir paprastesnis metodas (tiesios linijos metodas arba SL metodas, nuo angl. *Straight Line*) – viršūnės įdubos laipsnį skaičiuoti kaip atstumą nuo viršūnės iki atitinkamo tilto (6 pav. a). Antras būdas (trumpiausio kelio metodas arba SP metodas, nuo angl. *Shortest Path*) – surasti trumpiausią kelią nuo viršūnės iki atitinkamo tilto, šio kelio ilgis ir bus viršūnės įdubos laipsnis (6 pav. b). Trumpiausias kelias šiuo atveju negali kirsti daugiakampio briaunų. Tuo tarpu trumpiausias atstumas yra tiesi linija, kuri daugiakampio braunas kirsti gali. SP metodas yra priimtinesnis tuo atveju, jei norima, kad bendras daugiakampio įdubos laipsnis mažėtų monotoniškai. Tačiau mūsų atveju to nėra reikalaujama, todėl geriau pasirinkti paprastesnį SL metodą. Tuo labiau, kad SP metodas turi trūkumų, kurie pablogina modifikuoto ACD algoritmo rezultatus. Pavyzdžiui, turime spiralės formos daugiakampį (7 pav.). Skaičiuojant įdubos laipsnį pirmuoju metodu kirpimo linijos pradžia bus viršūnė v. Skaičiuojant įdubos laipsnį SP metodu kirpimo linijos pradžia bus viršūnė w. Iš viršūnės *v* matoma daugiau viršūnių (9 viršūnės), nei iš *w* (matomos 2 viršūnės). Todėl yra didesnė tikimybė kad rasime reikiamą kirpimo linijos pabaigą.

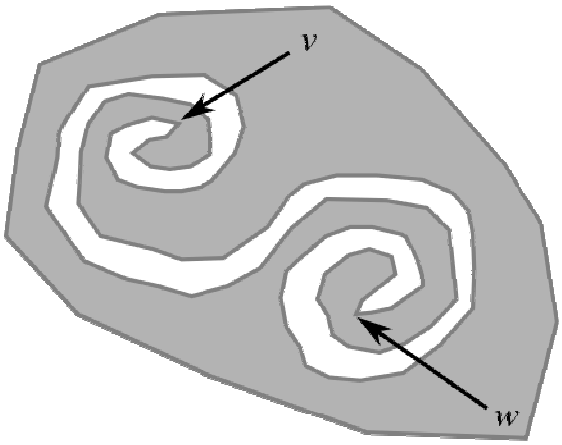


6 pav. a) ACD algoritmo skaičiavimų rezultatas, b) tikslus padalinimas į iškilius daugiakampius



7 pav. Iš viršūnės *v* matoma daugiau viršūnių nei iš viršūnės *w*

Kadangi ACD algoritme daugiakampio skylė yra traktuojama taip pat kaip įduba, tik be atitinkamo tilto, reikia paskaičiuoti ir skylės įdubos laipsnį bei rasti kirpimo linijos pradžią. Tai irgi siūloma atlikti dviems būdais, panašiai į išorinio kontūro viršūnės įdubos laipsnio skaičiavimą. Pirmasis būdas – rasti dvi labiausiai nutolusias skylės kontūro viršūnes. Šis metodas atitinka SL metodą. Tai atliekama Melkmano algoritmu [20], kuris randa daugiakampio skersmenį. Antrasis būdas – surasti dvi viršūnes, kelias tarp kurių yra ilgiausias, šis metodas atitinka SP metodą išorinio kontūro viršūnės įdubos laipsnio skaičiavimui. Siūloma tai atlikti suradus skylės viduriniosios ašies transformaciją (angl. Medial Axis Transform). Dėl to kad skylė yra paprastas daugiakampis, viduriniosios ašies transformacija bus medis, kuris sujungia visas skylės viršūnes. Tada tiesiog surandame dvi viršūnes, kurios yra labiausiai nutolusios viena nuo kitos šiame medyje. Šiame darbe pasirinkta naudoti pirmą metodą nes jis yra paprasčiau realizuojamas ir atitinka pasirinktą išorinio kontūro viršūnės įdubos laipsnio skaičiavimo metodą SL. Taip pat antras metodas turi esminį trūkumą – gali būti parinktos tokios kirpimo linijos pradinės viršūnės, iš kurių nesimatys nei viena išorinio kontūro viršūnė (8 pav.). Tada negalėsime atsikratyti skylės ir reikės ieškoti kirpimo linijos pradžios kokiu nors kitu būdu.



8 pav. Iš viršūnių *v* ir *w* nesimato nei viena išorinio kontūro viršūnė

Apžvelgėme pagrindines ACD algoritmo idėjas, jo privalumus bei trūkumus ir modifikavimo būdą. Toliau pateiksime modifikuoto ACD algoritmo pseudokodą.

Algoritmas ModACD(P, V_{max})
Pradiniai duomenys: paprastas daugiakampis P, maksimalus vienos dalies viršūnių skaičius V_{max}.
Rezultatai: daugiakampio P dalių aibė $\{C\}$.

1. if $|V_P| <= V_{max}$ then
2. return P.
3. else
4. ApskaičiuokNeiškilumoLaipsnius(P).

5. n := viršūnė, kurios neiškilumo laipsnis yra didžiausias.
6. $\{M\}$:= MatomosViršūnės(P, n).
7. foreach v iš $\{M\}$ do
8. ApskaičiuokTinkamumoIndeksą(v)
9. end do
10. Iš $\{M\}$ pasirink viršūnę t tokią, kurios tinkamumo indeksas didžiausias.
11. $\{C\}$:= Kirpk(P, n, t).
12. for C_i iš $\{C\}$ do
13. $\{G\}$:= ModACD(C_i).
14. Print {G}.

Dabar įvertinsime ModACD algoritmo sudėtingumą. Įdubos laipsnio skaičiavimas visoms viršūnėms ir viršūnės su didžiausiu įdubos laipsniu radimas (4 ir 5 eilutės) yra tiesinio sudėtingumo operacijos. Matomų viršūnių paieška yra $O(n \log n)$ sudėtingumo (Lee algoritmas, [5]: 323–333, [17]), bet yra ir $O(n)$ sudėtingumo algoritmų [3], [11], [23]. Vienos viršūnės tinkamumo indeksą apskaičiuojame per $O(1)$ laiką, todėl net jei matomos yra visos viršūnės, tinkamumo indekso apskaičiavimas ir didžiausios reikšmės radimas užtrunka $O(n)$ laiko. Daugiakampio skaidymas į dvi dalis taip pat yra tiesinio sudėtingumo uždavinys. Taigi, vienos iteracijos sudėtingumas yra $O(n \log n)$, jei matomumui naudojamas Lee algoritmas, ir $O(n)$, jei naudojami tiesinio sudėtingumo matomumo algoritmai.

Algoritmo metu vykdoma tiek iteracijų, kiek yra reikiamo dydžio dalių. Blogiausiu atveju dalių skaičius yra n–2, tokiu atveju daugiakampis suskaidomas trikampiais. Skaidyti trikampiais šio algoritmo pagalba nėra prasminga, nes yra greitų trianguliacijos algoritmų. Tačiau toks suskaidymas, nors ir mažai tikėtinas, yra galimas net ir nustačius $V_{\max}$ didesnį už 3. Tokiu (blogiausiu) atveju algoritmo bendras sudėtingumas yra $O(n^2 \log n)$ ($O(n^2)$ naudojant greitą matomumo algoritmą). Geriausiu atveju (suskaidžius į dalis, kurių dydžiai labai artimi arba lygūs $V_{\max}$) dalių skaičius yra $\frac{n-2}{V_{\max}-2}$. Taip gauname, kad geriausiu atveju ModACD algoritmo sudėtingumas yra:

$$O\left(n \log n \left(\frac{n-2}{V_{\max}-2}\right)\right).$$

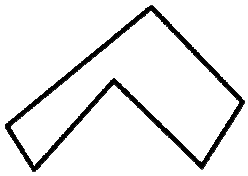
Kai naudojamas greitas matomumo algoritmas gauname tokį sudėtingumo įvertį:

$$O\left(n \left(\frac{n-2}{V_{\max}-2}\right)\right).$$

2.2. Daugiakampio trianguliacijos algoritmo modifikacija

Kaip jau buvo minėta, uždavinį galime performuluoti kaip daugiakamio dalinimą i *n*-kampius. Tai yra panašu į daugiakampio trianguliacijos uždavinį. Galima įrodyti, kad daugiakampio trianguliacija visada egzistuoja ir gaunamų trikampių skaičius yra *N*–2, čia *N* yra daugiakampio viršūnių skaičius [5]: 45–61]. Taip pat paprasta pastebėti, kad jei turime daugiakampio trianguliacijos linijas, pašalinus vieną liniją pridedama viena viršūnė. Pavyzdžiui, turime daugiakampį, pavaizduotą 6 pav. Apatiniame kairiajame kampe pradedame nuo trikampio. Pašaliname vieną liniją, gauname keturkampį. Dešiniajame kampe pradėję nuo trikampio ir pašalinę vieną liniją taip pat gauname keturkampį. Taip pašalinę 2 trianguliacijos apskaičiuotas linijas šešiakampį suskaidome į 2 keturkampius Taigi galime trianguliuoti daugiakampį vienu iš žinomų metodų, ir tada naikinti kirpimo linijas taip, kad gautume reikiamo dydžio daugiakampius.

Trianguliacijos algoritmai yra greiti, palyginti su daugiakampio tikslaus padalinimo į iškilius daugiakampius. Vieno iš paprastesnių algoritmų (Garey algoritmas), naudojančio daugiakampio monotoniškumo sąlygą, sudėtingumo įvertis yra $O(n \log n)$ [5]: 45–61]. 1991 m. B. Chazelle pasiūlė tiesinio sudėtingumo atgoritmą, tačiau jis yra labai sudėtingas [7]. 1998 m. M. Held taip pat pasiūlė tiesinio sudėtingumo algoritmą, kuris apdoroja skyles ir gali dirbti su sudėtingais daugiakampiais [13].



9 pav. Trianguliuotas daugiakampis is trianguliacijos šalinimas

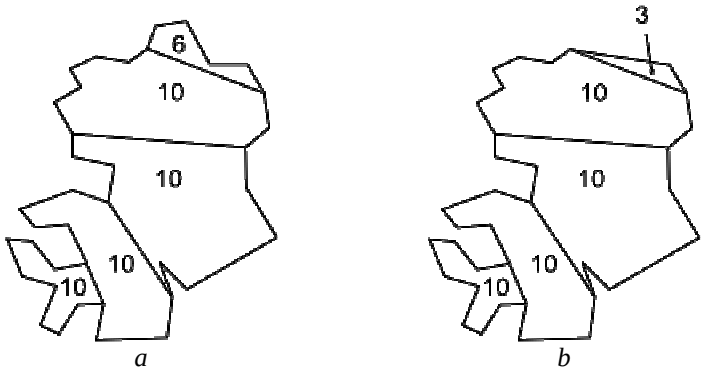
Metodo privalumai:

- Algoritmas yra paprastesnis nei daugiakampio skaidymo į apytiksliai iškilias dalis algoritmas.
- Daugumos gautų dalių viršūnių skaičius yra lygus V_{max}. Tik vienos dalies viršūnių skaičius yra mažesnis. Pavyzdžiui, jei dalinamas daugiakampis iš 38 viršūnių į dalis iki 10 viršūnių, gauname 4 dalis ir 10 viršūnių ir vieną dalį su 6 viršūnėmis (7 pav. *a*).

Metodo trūkumai:

- Kadangi daugumos gautų dalių viršūnių skaičius yra lygus V_{max}, esant tam tikram viršūnių skaičiui, paskutinės dalies viršūnių skaičius gali būti labai mažas, palyginti su kitomis dalimis. Pavyzdžiui, jei dalinamas daugiakampis iš 35 viršūnių į dalis iki 10 viršūnių, gauname 4 dalis iš 10 viršūnių ir vieną dalį tik su 3 viršūnėmis (7 pav. *b*).

- Algoritmas neveikia, jei daugiakampyje yra skylių, t.y. skylių neturi būti, jis jų neapdoroja. Dėl to skyles reikia kirpti kitu algoritmu, pavyzdžiui, ACD algoritmo modifikacija.



10 pav. *a)* apylygis daugiakampio padalinimas, *b)* netolygus daugiakampio padalinimas, visos dalys yra 10 viršūnių dydžio, išskyrus viršutinę – trikampį

Toliau pateiksime algoritmo pseudokodą ir paaiškinsime kai kuriuos jo veikimo ypatumus.

Algoritmas ModifikuotaTrianguliacija(P, V_{max})

Pradiniai duomenys: paprastas daugiakampis P be skylių, maksimalus vienos dalies viršūnių skaičius V_{max}.

Rezultatai: daugiakampio P dalys tokios, kad kiekvienos viršūnių skaičius neviršyja $V_{\max}$.

1. $\{D\}$:= Monotonizuok(P).
2. $\{C\}$:= SkaidykPagalĮstrižaines(P, $\{D\}$).
3. $\{D\}$:= tuščias įstrižainių sąrašas.
4. foreach C_i iš $\{C\}$ do
5. D_i := *Trianguliuok*(C_i).
6. Pridėk D_i prie įstrižainių sąrašo $\{D\}$.
7. end do
8. PašalinkĮstrižaines(P, D, $V_{\max}$).
9. $\{C\}$:= SkaidykPagalĮstrižaines(P, D).
10. return $\{C\}$.

Visų pirma procedūra Monotonizuok randa įstrižainės, kuriomis suskaidžius daugiakampį gaunamos monotoniškos dalys. Kiekvieną dalį trianguliuojame ir gauname trianguliacijos įstrižaines. Po to iš visų įstrižainių aibės pašaliname įstrižaines taip, kad suskaidžius daugiakampį pagal jas, gautume dalis, ne didesnes nei $V_{\max}$. Kirpimo įstrižaines radome visas iš karto, o ne rekursiškai, kaip ACD algoritme. Todėl skaidyti rekursiniu algoritmu tokias struktūras, kaip daugiakampis (kuris gali būti pakankamai didelis) nėra optimalu. Dėl to buvo sugalvotas iteracinis algoritmas, kuris

suskaido daugiakampį vadovaudamasis gautomis įstrižainėmis. Šiame algoritme funkcija Print(*s*) reiškia daugiakampio *s* pridėjimą prie rezultatų. Tai gali būti ir spausdinimas, ir įdėjimas į daugiakampių masyvą, kuris algoritmo pabaigoje gali būti grąžinamas kaip rezultatas.

Algoritmas SkaidykPagalĮstrižaines(*P*, *D*)
Pradiniai duomenys: daugiakampis *P*, orientuotas prieš laikrodžio rodyklę, ir įstrižainės *D*.
Rezultatai: daugiakampio *P* dalys.

1. foreach *v* iš *P* viršūnių do
2. while *v* gali priklausyti neaplankytoms *P* dalims do
3. *s* := naujas daugiakampis
4. *s*.pridekViršūnę(*v*)
5. Rask tokią neaplankytą įstrižainę *d*, kurios kampas pagal laikrodžio rodykle su briauna, ateinančia į viršūnę *v*, yra mažiausias. Jei tokios nėra – imk briauną, išeinančią iš viršūnės *v*.
6. v_n := įstrižainės *d* kita viršūnė
7. while v_n != *v* do
8. *s*.pridekViršūnę(v_n)
9. Rask tokią neaplankytą įstrižainę *d*, kurios kampas pagal laikrodžio rodykle su briauna arba įstrižaine, kuria atėjome į viršūnę v_n, yra mažiausias. Jei tokios nėra – imk briauną, išeinančią iš viršūnės v_n.
10. v_n:=įstrižainės *d* kita viršūnė
11. end do
12. Print(*s*)
13. end do
14. end do

Ar viršūnė *v* gali priklausyti neaplankytoms *P* dalims nustatome pagal dalių, kurioms gali priklausyti *v*, skaičių bei pažymėdami, kiek kartų viršūnė jau buvo pridėta į kurią nors dalį. Skaičių dalių, kurioms gali priklausyti *v*, nustatome pagal įstrižainių, ateinančių į *v* arba išeinančių iš *v*, skaičių pagal šią formulę $n_d = n_i + 1$. Čia n_d yra dalių, kurioms gali priklausyti *v*, skaičius; n_i – įstrižainių, kurioms priklauso *v*, skaičius.

Apskaičiavę trianguliaciją gauname įstrižaines, kurios dalina daugiakampį į trikampius. Bet mums nereikia visų įstrižainių, nes $V_{\max}$ gali būti didesnis už 3. Todėl dalį įstrižainių šalinsime.

Algoritmas PašalinkĮstrižaines(P, D, V_{max})

Pradiniai duomenys: paprastas daugiakampis P be skylių, jo trianguliacija D ir maksimalus vienos dalies viršūnių skaičius V_{max}.

Rezultatai: įstrižainių sąrašas $\{D_k\}$ toks, kad suskaidžius daugiakampį P pagal įstrižainės iš D, dalių viršūnių skaičius neviršys V_{max}.

1. $\{D_k\}$:= tuščias sąrašas įstrižainių, kurios turi būti paliktos
2. $\{V_m\}$:= tuščia aplankytų viršūnių aibė
3. v_{nd} := pirma pasitaikiusi viršūnė, kuri nepriklauso nei vienai įstrižainei
4. v_{start} := v_{nd} + 1 // pradžios viršūnė
5. count := 3
6. foreach v iš P viršūnių, pradedant v_{start} do
7. if v priklauso bent vienai įstrižainei then
8. Surūsiuok įstrižaines pagal kampą (pagal laikrodžio rodyklę), kurį įstrižainė sudaro su briauna, iš kurios atėjome į v.
9. d_{prev} := NULL
10. foreach d iš įstrižainių, kurioms priklauso v do
11. if d yra sąraše $\{D_k\}$ then
12. count := 3
13. else
14. if (d_{prev} != NULL) && (d ir d_{prev} kiti galai nėra ant tos pačios briaunos) then
15. pridėk įstrižainę (d; d_{prev}) prie $\{D_k\}$
16. if count < V_{max} then
17. pridek kitą d galą prie $\{V_m\}$
18. d_{prev} := d
19. pašalink d iš D
20. count++
21. else
22. pridėk d prie $\{D_k\}$
23. $\{D_k\}$:= NULL
24. count := 3
25. end do
26. end if
27. pridėk v prie $\{V_m\}$
28. end do
29. return $\{D_k\}$

Modifikuotos trianguliacijos algoritmo sudėtingumą sudaro monotonizavimo, trianguliacijos, įstrižainių pašalinimo bei daugiakampio skaidymo operacijų sudėtingumai. Monotonizavimo procedūros sudėtingumas yra lygus $O(n \log n)$ (įrodymas – [5]: 45–61]). Didžiąją monotonizavimo algoritmo sudėtingumo dalį sudaro viršūnių rūšiavimas pagal y koordinatę.

Monotonizavus daugiakampį jį reikia suskaidyti, kad galėtume apskaičiuoti trianguliaciją. Taip pat skaidoma pašalinus nereikalingas įstrižaines. Procedūros *SkaidykPagalĮstrižaines* vykdymo metu aplankome kiekvieną viršūnę ir tada aplankome kiekvieną viršūnės įstrižainę. Taip mes aplankome kiekvieną įstrižainę du kartus. Didžiausias įstrižainių skaičius gali būti n–3. Todėl viso įstrižainių aplankymų daugiausiai gali būti $O(n)$. Kiekvieną kitą įstrižainę randame pagal mažiausią kampą su briauna e_{i-1}. Minimumo paieškos sąraše veiksmo sudėtingumas yra $O(\log n)$. Visos likusios opreacijos yra $O(1)$ sudėtingumo. Taigi, procedūros *SkaidykPagalĮstrižaines* bendras sudėtingumas yra $O(n \log n)$. Taip pat galima šios procedūros pradžioje išrūšiuoti visas įstrižaines kiekvienoje viršūnėje, o vykdymo metu tiesiog imti viršūnės įstrižaines iš eilės. Tada kiekvienos kitos įstrižainės suradimo sudėtingumas yra lygus $O(1)$, o visų įstrižainių rūšiavimo sudėtingumas $O(n \log n)$.

Monotoniško daugiakampio trianguliacijos sudėtingumas yra $O(n)$ (įrodymas – [5]: 45–61]). Toliau vykdomos *PašalinkĮstrižaines* procedūros veikimo principas panašus į *SkaidykPagalĮstrižaines*. Viso aplankoma $O(n)$ įstrižainių. Kiekvieną įstrižainę randame ieškodami mažiausio kampo, šis veiksmas yra $O(\log n)$ sudėtingumo (arba procedūros vykdymo pradžioje galime išrūšiuoti įstrižaines, šis veiksmas užtrunka $O(n \log n)$). Likę veiksmai yra $O(1)$ sudėtingumo, taigi gauname, kad bendras procedūros *PašalinkĮstrižaines* sudėtingumo įvertis yra $O(n \log n)$.

Kadangi visų modifikuotos trianguliacijos algoritmo veiksmų sudėtingumo įverčiai neviršija $O(n \log n)$, bendras modifikuotos trianguliacijos algoritmo sudėtingumo įvertis yra lygus $O(n \log n)$.

3. SKAIČIAVIMO EKSPERIMENTŲ REZULTATAI

Algoritmai realizuoti Python programavimo kalba. Testavimas buvo atliekamas programinės įrangos (toliau – PĮ) paketu Safe FME (nuo angl. Feature Manipulation Engine – geografinių objektų valdymo programinė įranga). Realizuoti algoritmai naudoti kaip šios PĮ plėtiniai. Testuota su realiais geografiniais duomenimis.

Atliekant skaičiavimo eksperimentus buvo siekiama patikrinti algoritmų sudėtingumo įverčių teisingumą. Modifikuotos trianguliacijos algoritmo sudėtingumo įvertis nepriklauso nuo didžiausio leidžiamo dalių viršūnių skaičiaus. Tam patikrinti buvo skaidomas daugiakampis iš 5586 viršūnių į dalis iš daugiausiai 10, 20, 40 ir 80 viršūnių. Skaičiavimo trukmės nurodytos 1 lentelėje. Iš rezultatų matyti, kad modifikuotos trianguliacijos vykdymo trukmė iš tikrųjų nepriklauso nuo didžiausio

leidžiamo dalių viršūnių skaičiaus. Tačiau šis algoritmas priklauso nuo pradinio daugiakampio viršūnių skaičiaus. Kad patikrinti šią priklausomybę atlikti eksperimentai su daugiakampiais iš 40, 80, 160, 320 ir 640 viršūnių, kurie buvo skaidomi į dalis iš daugiausiai 10 viršūnių. Skaičiavimo trukmės ir jų santykiai nurodyti 2 lentelėje. Eksperimentai rodo, kad skaičiavimo trukmės atitinka sudėtingumo įvertį.

1 lentelė. Modifikuotos trianguliacijos algoritmo skaičiavimo trukmės priklausomybė nuo didžiausio leidžiamo dalių viršūnių skaičiaus V_{max}

Daugiakampio viršūnių skaičius N	Didžiausias leidžiamas dalių viršūnių skaičius V_{max}	Skaičiavimo trukmė t, s	$t(V_{max})/t(2V_{max})$
5586	80	1,79	
5586	40	1,80	1,01
5586	20	1,92	1,07
5586	10	1,82	0,95

2 lentelė. Modifikuotos trianguliacijos algoritmo skaičiavimo trukmės priklausomybė nuo pradinio daugiakampio viršūnių skaičiaus N, kai $V_{max} = 10$

Daugiakampio viršūnių skaičius N	Skaičiavimo trukmė t, s	$t(2N)/t(N)$
40	0,013	2,38
80	0,031	2,32
160	0,072	2,27
320	0,163	2,24
640	0,364	

Daugiakampio skaidymo į apytiksliai iškilius daugiakampius algoritmo sudėtingumo įvertis priklauso ir nuo viršūnių skaičiaus, ir nuo didžiausio leidžiamo dalių viršūnių skaičiaus. Atlikti eksperimentai su tokias pačiais pradiniais duomenimis, kaip ir tikrinant modifikuotos trianguliacijos algoritmo sudėtingumo įvertį. Skaičiavimo trukmės priklausomybė nuo didžiausio leidžiamo dalių viršūnių skaičiaus yra pateikta 3 lentelėje, o nuo viršūnių skaičiaus – 4 lentelėje. Abi priklausomybės atitinka teorinį sudėtingumo įvertį.

3 lentelė. Modifikuoto ACD algoritmo skaičiavimo trukmės priklausomybė nuo didžiausio leidžiamo dalių viršūnių skaičiaus V_{max}

Daugiakampio viršūnių skaičius N	Didžiausias leidžiamas dalių viršūnių skaičius V_{max}	Skaičiavimo trukmė t, s	$t(V_{max})/t(2V_{max})$
160	80	0,094	
160	40	0,129	1,37
160	20	0,162	1,26
160	10	0,212	1,31

4 lentelė. Modifikuoto ACD algoritmo skaičiavimo trukmės priklausomybė nuo pradinio daugiakampio viršūnių skaičiaus N, kai $V_{max} = 10$

Daugiakampio viršūnių skaičius N	Skaičiavimo trukmė t, s	$t(2N)/t(N)$
40	0,019	4,42
80	0,084	2,55
160	0,214	3,74
320	0,800	3,41
640	2,723	

5 ir 6 lentelėse palyginami abejų algoritmų gautų dalių skaičiai ir dalių dydžių vidurkiai. Matome, kad rezultatai yra panašūs.

5 lentelė. Modifikuotos trianguliacijos ir modifikuoto ACD algoritmų gautų dalių skaičiai

Daugiakampio viršūnių skaičius N	Dalių skaičius	
	Modifikuota trainguliacija	Modifikuotas ACD algoritmas
40	5	5
80	12	12
160	25	26
320	60	60
640	119	100

6 lentelė. Modifikuotos trianguliacijos ir modifikuoto ACD algoritmų gautų dalių viršūnių skaičių vidurkiai

Daugiakampio viršūnių skaičius N	Dalių viršūnių skaičių vidurkiai	
	Modifikuota trainguliacija	Modifikuotas ACD algoritmas
40	9,4	9,4
80	8,4	8,4
160	8,3	8,1
320	7,5	7,5
640	7,4	8,4

1 priede pateikti paveikslai, kuriuose parodyti daugiakampių suskaidymai. Kaip matome, dauguma modifikuotos trianguliacijos algoritmo vykdymo metu gautų kirpimo įstrižainių yra artimos horizontalioms linijoms, todėl dauguma dalių yra plačios ir žemos. Tuo tarpu modifikuoto ACD algoritmo gautos dalys yra įvairių formų, jos yra gražesnės ir toks skaidymas yra labiau panašus į žmogau darytą skaidymą. Nors tam tikrais atvejais ir šis algoritmas atkerpa dalis, kurios yra ilgos ir siauros.

IŠVADOS IR PASIŪLYMAI

- Pateiktos dvi daugiakampio skaidymo strategijos, kurios užtikrina, kad daugiakampio dalių viršūnių skaičiai bus nedidesni, nei nustatytas parametras $V_{\max}$. Viena iš šių strategijų paremta daugiakampio trianguliacija, kita – daugiakampio skaidymo į apytiksliai iškilius daugiakampius algoritmu (ACD).
- Atlikta modifikuotos trianguliacijos ir modifikuoto ACD algoritmo sudėtingumo analizė. Modifikuotos trianguliacijos algoritmo sudėtingumo įvertis yra $O(n \log n)$. Modifikuoto ACD algoritmo sudėtingumo įvertis geriausiu atveju yra $O\left(n\left(\frac{n-2}{V_{\max}-2}\right)\right)$, blogiausiu – $O(n^2)$.
- Skaitiniais eksperimentais patvirtinti algoritmų sudėtingumo įverčiai.
- Iš skaitinių eksperimentų rezultatų nustatyta, kad abu algoritmai suskaido daugiakampį į panašų skaičių dalių. Gautų dalių viršūnių skaičių vidurkiai taip pat yra panašūs.
- Modifikuotas ACD algoritmas suskaido į gražesnės formos dalis. Tačiau modifikuotos trianguliacijos algoritmo rezultatas nėra toks gražus.
- Kad algoritmų sudėtingumo įverčiai būtų geriausi, rekomenduojama naudoti tiesinio sudėtingumo trianguliacijos [7], [13] ir daugiakampio matomumo [3], [11], [23] algoritmus.

LITERATŪRA

[1] Aggarwal, A.; Chazelle, B. 1984. *Effiecient algorithm for partitioning a polygon into star-shaped polygons*. Report. Yorktown Heights: IBM T. J. Watson Res. Center.

[2] Al-Essa, B. 1997. Polygon Partitioning: Monotone Triangulation [interaktyvus]. [Žiūrėta 2010 m. birželio 13 d.]. Prieiga per internetą: http://www.me.cmu.edu/faculty1/shimada/cg97/bader/index.html

[3] Asano, T.; Asano, T.; Guibas, L. J.; Hershberger, J.; Imai, H. 1986. Visibility of disjoint polygons. *Algorithmica* 1: 49–63.

[4] Avis D., Toussaint G. T. 1981. An efficient algorithm for decomposing a polygon into star-shaped polygons, *Pattern Recognition* 13: 395–398.

[5] de Berg, M.; Cheong, O.; van Kreveld, M.; Overmars, M. 2008. *Computational geometry: Algorithms and applications*. Berlin Heidelberg: Springer-Verlag.

[6] Chazelle, B. 1982. A theorem on polygon cutting with applications, in *Proc. 23rd Annual IEEE Symposium Foundations of Computer Science*. Chicago, 339–349.

[7] Chazelle, B. 1991. Triangulating a simple polygon in linear time, in *Discrete & Computational Geometry*: vol. 6, issue 5. New York: Springer-Verlag, 485–524.

[8] Chazelle, B.; Dobkin D. P. 1985. Optimal Convex Decomposition, in *G.T. Toussaint (Ed.), Computational Geometry*. Amsterdam, 63–133.

[9] Everett, H.; Lenhart, W.; Overmars, M.; Shermer T.; Urrutia, J. 1992. Strictly convex quadrilateralizations of polygons, in *Proc. 4th Canadian Conference on Computational Geometry*. St. John‘s, 77–83.

[10] Feng, H. Y. F.; Pavlidis, T. 1975. Decompositions of polygons into simpler components: Feature generation for syntactic pattern recognition, *IEEE Transactions on Computers* C-24: 636–650.

[11] Guibas, L. J.; Hershberger, J.; Leven, D.; Sharir, M.; Tarjan, R. E. 1987. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica* 2: 209–233.

[12] Greene, D. H. 1983. The decomposition of polygons into convex parts, in *F. P. Preparata (Ed.), Computational Geometry, volume 1 of Adv. Comput. Res*. Greenwich: JAI Press, 235–259.

[13] Held, M. 1998. *FIST: Fast industrial-strength triangulation of polygons*. Technical report, University at Stony Brook.

[14] Keil, J. M. 1985. Decomposing a polygon into simpler components, *SIAM Journal on Computing*, 14(4): 799–817.

[15] Keil, J. M. 1998. Polygon decomposition, in *J.-R. Sack and J. Urrutia(Ed.), Handbook of Computational Geometry*, Amsterdam: Elsevier Science Publishers B.V., 491–518.

[16] Keil, J. M.; Snoeyink, J. 2002. On the time bound for convex decomposition of simple polygons, *International Journal of Computational Geometry and Applications* 12(3): 181–192.

[17] Lee, D. T. 1978. Proximity and reachability in the plane. Report R-831, Dept. Elect. Engrg., Univ. Illinois, Urbana, IL.

[18] Lien, J.-M.; Amato, N. M. 2004. Approximate convex decomposition of polygons, in *Proc. 20th Annual ACM Symposium on Computational Geometry (SoCG)*. New York, 17–26.

[19] Lingas, A.; Levcopoulos, C.; Sack, J.-R. 1987. Algorithms for minimum length partitions of polygons, *BIT* 27: 474-479.

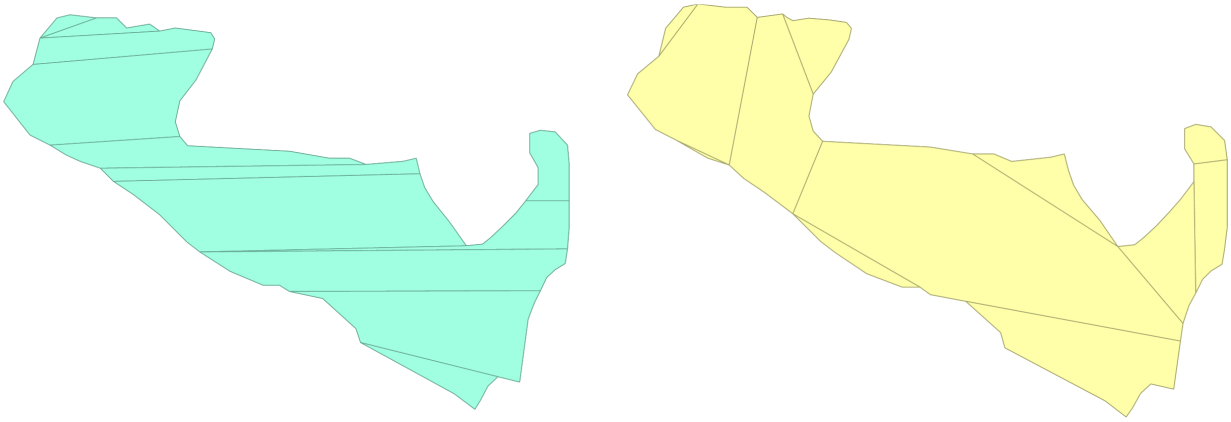
[20] Melkman A. A. 1985. Online Construction of the convex hull of a simple polyline, *Information Processing Letters* 25(1): 11–12.

[21] Ramaswami, S.; Ramos, P.; Toussaint, G. T. 1998. Converting triangulations to quadrangulations, *Computational Geometry: Theory and Applications* 9: 257–276.

[22] Shamos, M. I. 1978. *Computational geometry*. Ph.D. thesis, Yale University. Yale.

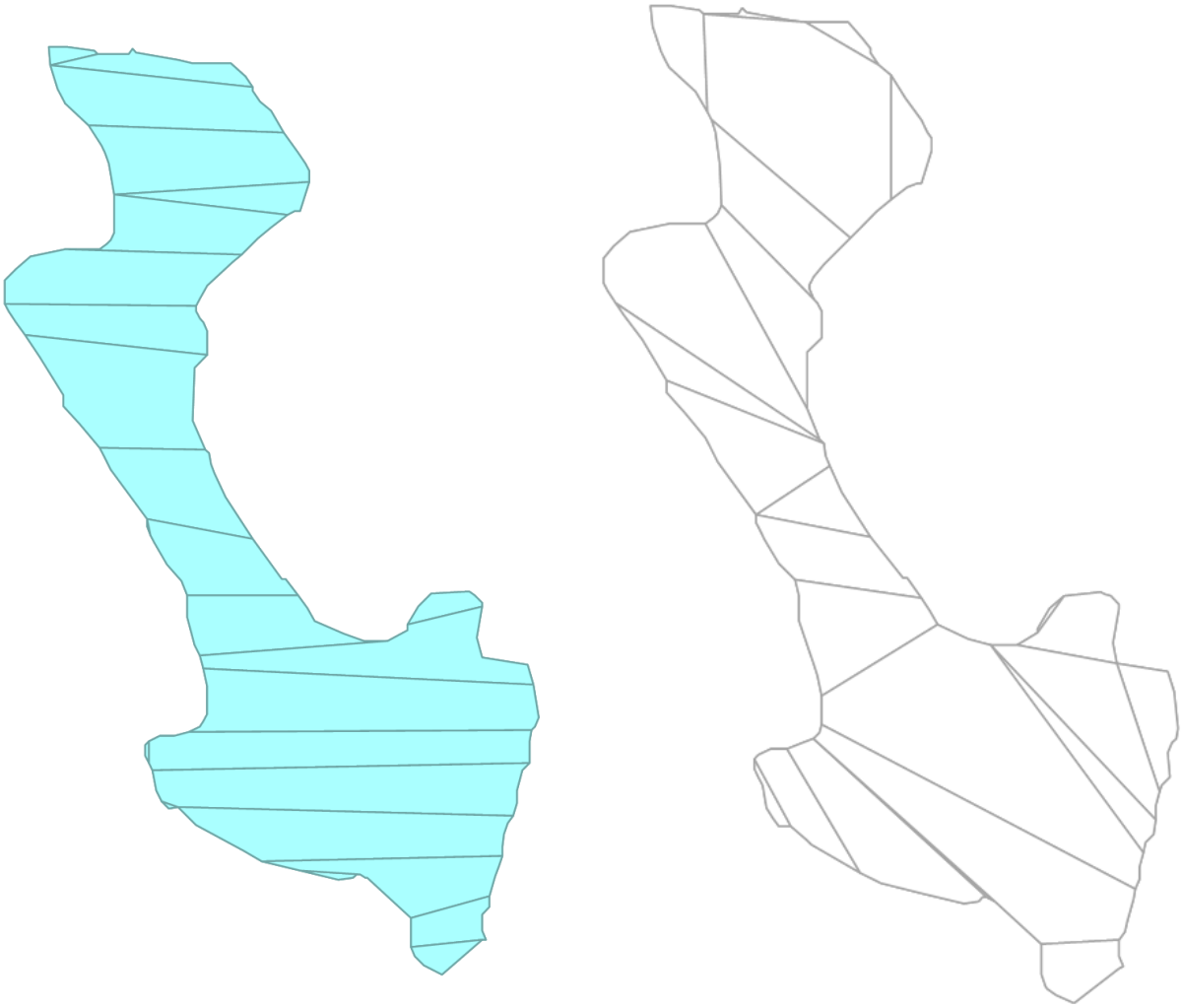
[23] Welzl, E. 1985. Constructing the visibility graph for n line segments in $O(n^2)$ time. Information Processing Letters 20: 167–171.

1 PRIEDAS



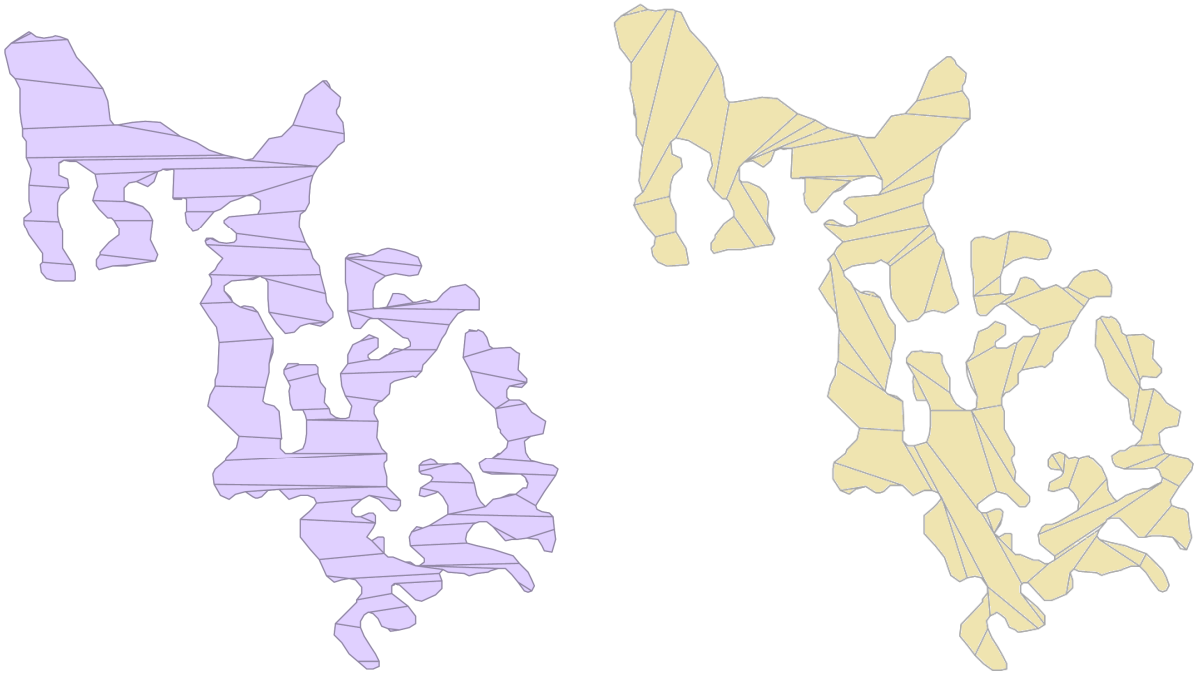
Modifikuota trianguliacija Modifikuotas ACD

1 pav. Daugiakampio iš 80 viršūnių suskaidymas į dalis iš daugiausiai 10 viršūnių



Modifikuota trianguliacija Modifikuotas ACD

2 pav. Daugiakampio iš 160 viršūnių suskaidymas į dalis iš daugiausiai 10 viršūnių

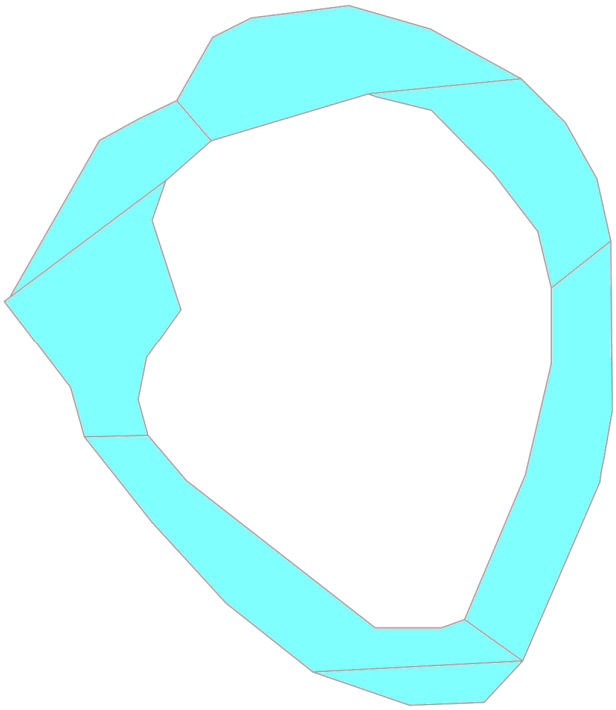


Modifikuota trianguliacija

Modifikuotas ACD

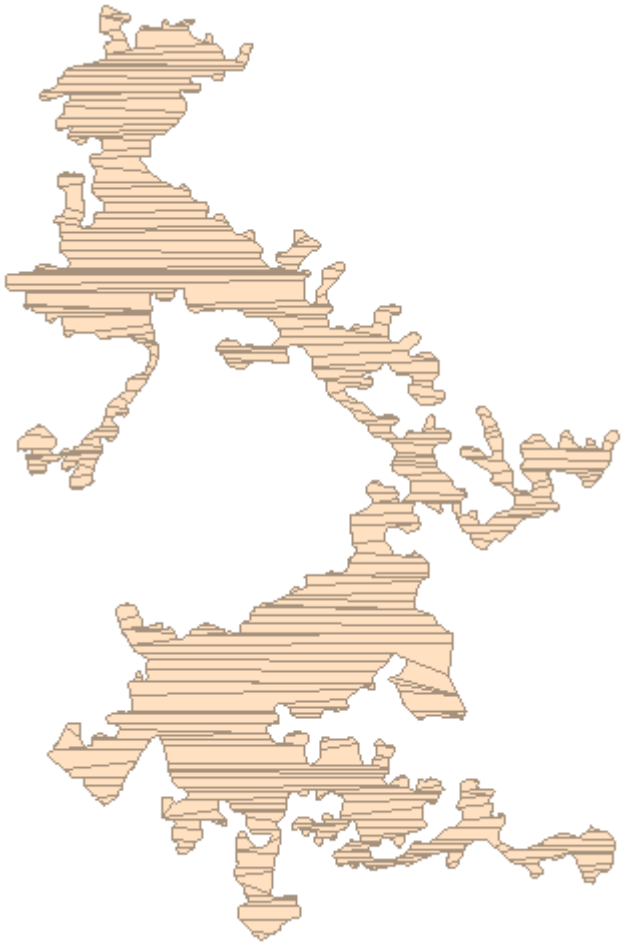
3 pav. Daugiakampio iš 640 viršūnių suskaidymas į dalis iš daugiausiai 10 viršūnių

4 pav. Skylių kirpimas



5 pav. Daugiakampio su skyle suskaidymas. Daugiakampio viršūnių skaičius yra 47, dalių viršūnių skaičius neviršyja 10

6 pav. Daugiakampio iš 1300 viršūnių padalijimas modifikuotu ACD algoritmu



7 pav. Daugiakampio iš 3200 viršūnių padalijimas modifikuotos trianguliacijos algoritmu