VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Algoritmų analizė

Maksimali nepriklausoma viršūnių aibė

(namų darbas nr. 0)

Tomas Giedraitis
VU MIF Informatika
3 kursas 3 grupė

Vilnius
2021

Turinys

1. Uždavinio formuluotė

Duota: Neorientuotas grafas $G = (V, E)$
Rasti: Maksimalią nepriklausomą viršūnių aibę S

Grafų teorijoje nepriklausoma viršūnių aibė $V' \subseteq V$ yra aibė, kurią sudaro grafo viršūnės, iš kurių nė viena nėra sujungta briauna su kuria nors kita iš tos aibės esančia viršūne. Dar vadinama antiklika, nes viršūnių aibė yra nepriklausoma tada ir tik tada, kai ši aibė yra komplementaraus grafo klika (kai bet kokios dvi šios aibės viršūnės yra gretimos)[1].

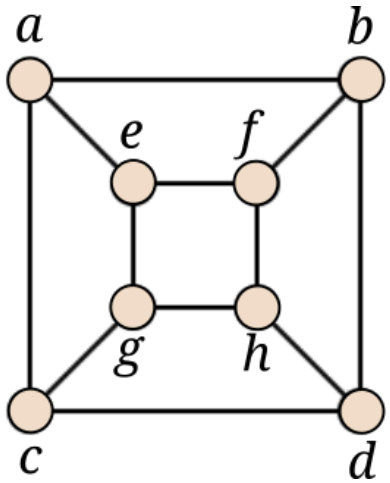
Tuo tarpu maksimali nepriklausoma viršūnių aibė S (angl. *Maximum Independent Set*[1]) yra elementų skaičiumi $|V'|$ didžiausia įmanoma nepriklausoma viršūnių aibė duotam grafui G.

Maksimalios nepriklausomos viršūnių aibės radimas yra NP-sunkus uždavinys[2].

2. Paprastas pavyzdys su sprendimu

Pateikiamas paprastas pavyzdys, kai pradinių duomenų yra mažai (1 pav.).

Duota: Grafas $G = (V, E)$,
$V = \{a, b, c, d, e, f, g, h\}$,
$E = \{ab, ac, ae, bd, bf, cd, cg, dh, ef, eg, fh, gh\}$



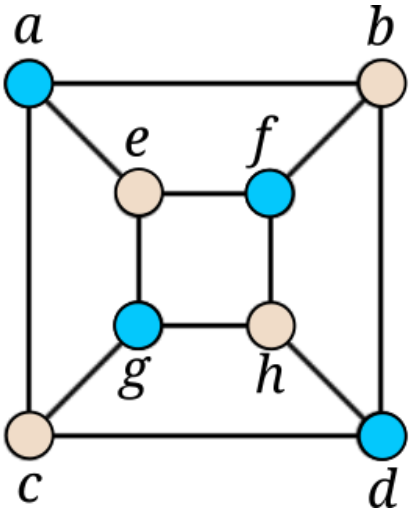
1 pav. Grafas G

Sprendžiame pilno perrinkimo būdu. Kadangi mums reikia maksimalios aibės, pradėsime perrinkimą nuo didžiausio viršūnių poaibio, lygaus visų grafo viršūnių aibei, ir vis mažinsime poaibių dydį, kol rasime pirmąją nepriklausomą viršūnių aibę. Briaunas tarp viršūnių tikrinsime leksikografine tvarka.

Sprendimas:
Imame poaibį $V_1 = V = \{a, b, c, d, e, f, g, h\}$. Tikriname, ar yra briauna *ab.* Taip, yra, taigi V_1 poaibis jau netiks.

Patikrinus su poaibiais, kurių dydis yra 7, 6 ir 5, reikiamos aibės irgi nerasime.

Tikriname su poaibiais dydžio 4. Poaibiai, kuriuose kartu su *a* viršūne yra ir *b* arba *c*, netinka. Tuomet šiuo momentu tikriname poaibius, kuriuose yra *a* ir *d* viršūnės. Kartu su *b, c* į ieškomą poaibį, žinoma, negalės patekti ir *e, h* viršūnės. Taigi prieiname prie poaibio *{a, d, f, g}*, ir matome, kad jo elementai sudaro nepriklausomą viršūnių aibę. Renkamės šį variantą, gautas atsakymas – *{a,d,f,g}*, o aibės dydis yra 4 (2 pav.).



2 pav. Maksimali nepriklausoma viršūnių aibė *S*

3. Kas apie šį uždavinį žinoma pasaulyje

Maksimalios nepriklausomos viršūnių aibės radimo uždavinys yra vienas iš svarbiausių NP-pilnų problemų grafų teorijoje, jis yra naudojamas įrodant daugumos teorinių uždavinių skaičiavimo sudėtingumą, ir turi daug aplikacijų įvairiose srityse, pavyzdžiui, žemėlapių ženklinime (angl. *labelling*), įvairiuose planavimo uždaviniuose (angl. *scheduling*), ar net molekulinėje biologijoje[3] – pvz. atrandant stabilias genetines komponentes genų inžinerijoje.

Per daug neišsiplečiant, verta paminėti, kad šis uždavinys turi sąryšį su kitais grafų teorijos uždaviniais: minimaliu viršūnių denginiu, viršūnių spalvinimu, minimaliu briaunų denginiu.[4] Taip pat, kaip jau aukščiau minėta, grafo nepr. maks. viršūnių aibė ir jam komplementaraus grafo klika yra komplementarūs konceptai. Toliau šių konceptų sąryši nagrinėja ir Ramsey teorija[5].

Kompiuterių moksle yra nagrinėjamos kelios skaičiavimo problemos, susijusios su nepriklausomomis viršūnių aibėmis[6]:

- Viršūnių grupavimo (angl. *vertex packing*) uždavinys.
- Maksimalaus svorio nepriklausomos viršūnių aibės radimas svoriniame grafe.

- Visų nepriklausomų viršūnių aibių grafe radimas, kurios nėra poaibiai kitų nepr. virš. aibių (angliškai ši sąvoka – *maximal independent set*, tuo tarpu mūsų nagrinėjama – *maximum independent set*).
- Nepriklausomos viršūnių aibės egzistavimo problema.

Pirmosios trys problemos yra svarbios praktiniuose taikymuose, o ketvirtoji yra būtina tam, kad galima būtų pritaikyti NP-sunkumo teoriją šių uždavinių (susijusių su nepr. viršūnių aibėmis) grupei.

Algoritmai šio uždavinio sprendimui[7]:

- Tikslieji algoritmai. Paprasčiausias iš jų yra pilnojo perrinkimo. Kiti tikslieji algoritmai atsižvelgia į grafo specifiką, taip siekiant sumažinti algoritmo žingsnių skaičių. 2017 metų žiniomis, ši problema gali būti tiksliai išspręsta su laiko sudėtingumu $O(1.1996^n)$, nadojant polinominį erdvės sudėtingumą. Tikslūs algoritmai, implementuoti kompiuterinėmis programomis: *igraph*, *LightGraphs*.
- Aproksimuojantys algoritmai. Taikomi tik tam tikroms grafų klasėms (planarūs grafai – viena iš šių klasių). Godusis algoritmas taip pat patenka tarp šių algoritmų. Aproksimuojantis algoritmas, implementuotas Python programavimo kalba – *NetworkX*.

Pastaruoju metu dirbtiniai neuroniniai tinklai irgi naudojami nagrinėti šio ir panašių sudėtingų uždavinių sprendimams[8].

4. Pilno perrinkimo sudėtingumas

Jau anksčiau aprašytu pilno perrinkimo būdu blogiausias atvejis bus tada, kai turėsime pilnąjį grafą – tuomet reikės perrinkti visus viršūnių aibės poaibius, kurių dydis yra daugiau nei 1, ir vis tiek nepriklausomos viršūnių aibės nerasime iš bent dviejų elementų nerasime. Tokių poaibių yra $2^n - n$, tačiau paprastumo dėlei galime paimti ir visus įmanomus poaibius, t.y. 2^n, nes pagrindinis sudėtingumo faktorius čia yra eksponentė. Kiekviename poaibyje dydžio k reikės pasirinkti k viršūnių, kurias įkelsime į sąrašą, ir tuomet blogiausiu atveju reikės patikrinti k-1 briauną, kur k kinta nuo 1 iki n. Didesnis sudėtingumas, gautas kiekvieną sykį vietoje skaičiaus k naudojant skaičių n, skirsis tik per kažkokį konstantinį daugiklį, taigi kiekvienam poaibiui, vietoje jam specifinio skaičiaus k, pakelto kvadratu, imsime n^2. Taigi, galime sakyti, kad pilno perrinkimo viršutinis sudėtingumo įvertis yra $O(n^2 2^n)$ nuo viršūnių skaičiaus n.

5. Euristinis algoritmas

Kuomet pasirenkame naują viršūnę į jau pradėtą formuoti poaibį, į šį poaibį nebepateks tos viršūnės, kurios su ja yra sujungtos briauna. Taigi galime taikyti godaus algoritmo strategiją – kiekvienu nauju pasirinkimu siekiame prarasti mažiausią viršūnių–kandidačių skaičių, kitaip tariant, imame tą viršūnę į mūsų nepriklausomos viršūnių aibės poaibį, kurios laipsnis (*deg*) yra mažiausias. Jei laipsniai lygūs, vėlgi imsime leksikografine tvarka (arba, jei viršūnes žymėtumėme sveikaisias skaičias – jų didėjimo tvarka). Svarbu paminėti, kad šalia kaupiame tas

viršūnės, kurios jau nebegalės patekti į poaibį, ir visas jai incidenčias briaunas, skaičiuodami viršūnių laipsnius, ignoruojame.

Išspręskime jau minėtą pavyzdį šiuo būdu.

Sprendimas:
Kadangi visų 8-ių viršūnių laipsniai yra lygūs 3, renkamės *a*.

Turimas sąrašas: *{a}*.
Nebetinkančių viršūnių sąrašas tampa *{b, c, e}*.

Randame likusių galimų viršūnių laipsnius (pagal jau aprašytą taisyklę):

deg(d) = 1 (nes yra briauna *dh*, o briaunas *cd, bd* ignoruojame)
deg(f) = 1
deg(g) = 1
deg(h) = 3

Akivaizdžiai *h* viršūnė netinka į mūsų formuojamą poaibį. Visos kitos turi laipsnį, lygų 1, tai pirmumo tvarka renkamės viršūnę *d*.

Turimas sąrašas: *{a,d}*
Nebetinkančių viršūnių sąrašas: *{b, c, e, h}*.

Toliau skaičiuojame likusių viršūnių laipsnius:

deg(f) = 0
deg(g) = 0

Kadangi abiejų likusių viršūnių laipsniai yra 0, renkamės jas abi iš karto, ir baigiame. Atsakymą gavome tokį patį, kaip ir perrinkimo būdu: *{a,d,f,g}* – maksimalią nepriklausomą viršūnių aibę (2 pav.).

5.1. Algoritmo tikslumas

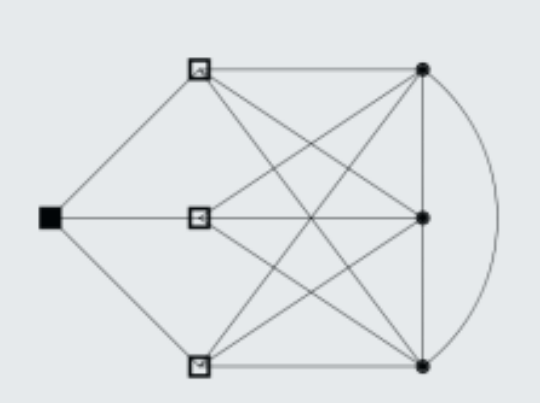
Šiuo atveju mums pasisekė, nes gavome maksimalią nepriklausomą viršūnių aibę. Tačiau šis algoritmas nėra visada korektiškas. Problema yra tame, kad sudarę viršūnių aibę godžiuoju būdu, mes nežinome, ar ši aibė yra maksimali.

Šios problemos netobulas sprendimas galėtų būti algoritmo kartojimas dar sykį, pradedant nuo antros pagal tinkamumą (arba antros iš eilės, jei viršūnės laipsnis toks pat) pradinės viršūnės, ir žiūrint, ar gauname didesnę aibę. Po to vėl galima kartoti. Jei su visomis pradinėmis viršūnėmis jau pakartota, ir dar sykį norėtume įvykdyti algoritmą, tada imame pirmuoju atveju panaudotą pradinę viršūnę ir vietoje tuo metu pasirinktos antros viršūnės imame kitą, mažiau tinkamesnę

viršūnę (arba vienodai tinkamą), ir taip toliau. Tačiau dažniausiai tikimasi gauti maksimalią nepriklausomą viršūnių aibę jau iš pirmųjų kelių kartų.

Taip pat, galima atsižvelgti ir į gautos aibės dydį bei nagrinėjamo grafo specifiką siekiant nuspręsti, ar algoritmo gauta aibė yra maksimali. Pavyzdžiui, jei turime pilnąjį grafą be kilpų, iš karto yra aišku, kad maksimalios nepriklausomos viršūnių aibės dydis bus 1, tad galime rinktis bet kurią viršūnę. Jei grafas neturi briaunų, tai maks. nepr. viršūnių skaičius bus lygus grafo viršūnių skaičiui $|V|$. Jei grafas turi tik vieną briauną, maks. nepr. viršūnių skaičius bus lygus $|V|$-1. Tai, kad grafas neturi kilpų ir dvi viršūnes jame gali jungti tik viena briauna, irgi duoda informacijos apie maks. nepr. viršūnių skaičių.

Pavyzdys, kada godusis algoritmas neduos teisingo atsakymo (3 pav.). Kai pradine viršūne pasirinksime mažiausio laipsnio viršūnę (pavaizduota užtušuotu kvadratėliu), tuomet rezultato aibės dydis bus 2 (pirmoji viršūnė ir viena iš viršūnių, pažymėtų skrituliuku), o optimalus uždavinio sprendimas yra 3 viršūnės (tuščiaviduriai kvadratėliai).



3 pav. Grafas, kuriame godusis algoritmas neduoda gero rezultato

5.2. Algoritmo sudėtingumas

Naudodami gretimumo matricą grafui atvaizduoti, sudedame kiekvienos eilutės elementus, ir taip gauname visų viršūnių laipsnius, kas užims $n \times (n\text{-}1) \cong n^2$ operacijų. Išrinkę vieną viršūnę, perskaičiuojame likusių viršūnių laipsnius (kiekvienai viršūnei – dvi operacijos: tikrinimas, ar jungiasi su jau pasirinkta viršūne, ir vieneto atėmimas iš jos laipsnio, jei atsakymas teigiamas) ir renkamės kitą viršūnę. Visų maks. nepr. aibės viršūnių pasirinkimas – ne daugiau nei n operacijų, laipsnius perskaičiuojame ne daugiau nei n kartų, kaskart ne daugiau nei n viršūnėms. Taigi šio euristinio algoritmo sudėtingumas būtų $n^2 + 2n^2 + n = 3n^2 + n$ žingsnių, taigi tai būtų $O(n^2)$.

Literatūros sąrašas

Trumpiniai:

[WIKI_IS]: https://en.wikipedia.org/wiki/Independent_set_(graph_theory) [žiūrėta 2021-05-15]

Sąrašas:

1. Pierluigi Crescenzi, Viggo Kann. A compendium of NP optimization problems. https://www.csc.kth.se/~viggo/wwwcompendium/node34.html [žiūrėta 2021-05-10]

2. Garey, M. R.; Johnson, D. S. Strong NP-Completeness Results: Motivation, Examples, and Implications. Journal of the ACM. 25 (3): 499–508. doi:10.1145/322077.322090. https://dl.acm.org/doi/10.1145/322077.322090 [žiūrėta 2021-05-15]

3. Mathieu Mari. Study of greedy algorithm for solving maximum independent set problem. Technical report, University of Liverpool, 2017. https://www.di.ens.fr/~mmari/content/papers/rapport.pdf [žiūrėta 2021-05-10]

4. [WIKI_IS]: Sekcija “Properties: Relationship to other graph parameters”.

5. Graham, Ron; Butler, Steve (2015). Rudiments of Ramsey Theory (2nd ed.). American Mathematical Society. p. 1. ISBN 978-0-8218-4156-3. https://www.ams.org/books/cbms/123/cbms123-endmatter.pdf [žiūrėta 2021-05-15]

6. [WIKI_IS]: Sekcija “Finding independent sets”.

7. [WIKI_IS]: Sekcija “Finding maximum independent sets” bei “Software for searching maximum independent set”.

8. Sungsoo Ahn, Younggyo Seo, Jinwoo Shin. Learning What to Defer for Maximum Independent Sets. 2020. https://arxiv.org/abs/2006.09607 [žiūrėta 2021-05-15]