4.3. Keliaujančio pirklio uždavinys

Ankstesniuose skyriuose nagrinėjome algoritmus, kurių sudėtingumą įvertinome polinomine grafo viršūnių ar briaunų skaičiaus funkcija. Tokiais atvejais nesunkiai galime rasti tikslų uždavinio sprendinį net ir tada, kai grafas turi kelis tūkstančius viršūnių.

Šiame skyriuje susipažinsime su gerokai sudėtingesnio uždavinio sprendimo algoritmais. Nagrinėsime ne tik metodus, leidžiančius rasti tikslų uždavinio sprendinį, bet ir euristinius algoritmus, kuriais apskaičiuosime tik sprendinio artinius. Tačiau euristikos tinka ir tada, kai grafo viršūnių ir briaunų skaičius yra labai didelis.

4.3.1. Uždavinio formulavimas

Turime n miestų, kurie sudaro grafo $G = (V, E)$ viršūnių aibę

$$V = \{v_i : \ 1 \leqslant i \leqslant n\}.$$

Kai kurie miestai yra sujungti keliais, ir žinome atstumus tarp šių miestų. Keliai sudaro grafo G briaunų aibę:

$$E = \{e_{ij} : \ e_{ij} = (v_i, v_j),\ 1 \leqslant i, j \leqslant n\}.$$

Pažymėkime $w(e_{ij}) = |(v_i, v_j)|$ kelio, jungiančio viršūnes v_i ir v_j, ilgį. Jeigu du miestai v_i ir v_j nėra sujungti keliu, tai $w(e_{ij}) = \infty$. Kadangi iš vieno miesto išeina nedaug kelių, tai dažniausiai grafo briaunų matricą saugome suspaustu formatu.

Pirklys, išėjęs iš pirmo miesto, turi aplankyti visus miestus ir grįžti į pradinį miestą. Į kiekvieną miestą jis gali patekti tik vieną kartą. Tokį kelią

$$p = \{v_1,\ w_2,\ w_3,\ \ldots,\ w_n,\ v_1\},\ \ w_j = v_{i_j}$$

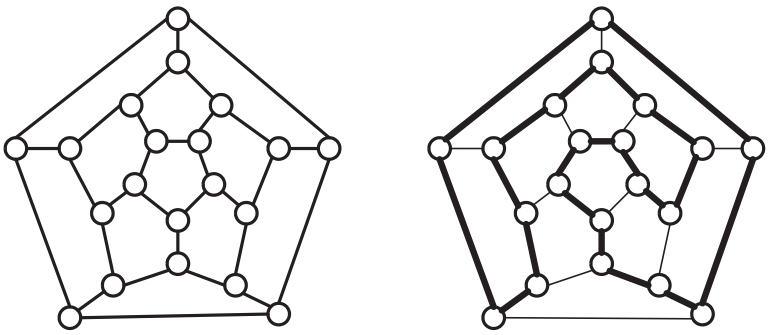
vadiname *maršrutu*. Reikia rasti *trumpiausią* pirklio maršrutą.

Tarsime, kad bent vienas maršrutas visada egzistuoja, nors nežinome nė vienos paprastos pakankamosios sąlygos, kuri tai garantuotų. Aišku, toks maršrutas tikrai egzistuoja, jei grafas yra pilnasis, t. y. visos jo viršūnės yra sujungtos tarpusavyje briaunomis.

Jeigu grafas G yra neorientuotas, tai sprendžiame *simetrinį* keliaujančio pirklio uždavinį, priešingu atveju uždavinys vadinamas *nesimetriniu*.

Hamiltono maršrutai. Keliaujančio pirklio uždavinys yra atskiras atvejis bendresnio uždavinio apie Hamiltono maršrutų radimą. Maršrutas, apeinantis visas grafo viršūnes po vieną kartą, vadinamas Hamiltono maršrutu. Jei pradinė ir galinė maršruto viršūnės sutampa, toks maršrutas vadinamas Hamiltono ciklu. Grafas, turintis bent vieną Hamiltono maršrutą, vadinamas Hamiltono grafu.

1859 m. Hamiltonas (*W. R. Hamilton*) viename laiške aprašė tokį žaidimą: pirmas žaidėjas dodekaedre (žr. 4.6 paveikslą) pažymi penkias gretimas grafo viršūnes, tada antrasis žaidėjas turi baigti ciklą, jungiantį visas dvidešimt viršūnių.



a) b)

4.6 pav. Hamiltono žaidimas: a) pradinis dodekaedras, b) Hamiltono ciklas

Norėdami įrodyti, kad grafas yra Hamiltono, turime rasti bent vieną maršrutą. Dažnai sprendžiame ir sudėtingesnį uždavinį – ieškome visų Hamiltono maršrutų. Tada, suradę visus ciklus, nesunkiai išsprendžiame keliaujančio pirklio uždavinį.

Pažymėsime, kad nors visų maršrutų radimo uždavinys atrodo sudėtingesnis už Hamiltono grafo sąlygos tikrinimą (t. y. vieno maršruto paiešką), tačiau abiejų uždavinių sudėtingumas gali būti toks pat, kai grafe nėra nė vieno maršruto.

4.3.2. Pilnas variantų perrinkimas

Paprasčiausias būdas, kaip rasti visus Hamiltono ciklus, – generuoti visus skirtingus viršūnių išdėstymo būdus

$$p = \{v_1, w_2, w_3, \ldots, w_n, v_1\}, \;\; w_j = v_{i_j}$$

ir patikrinti, kurie iš jų apibrėžia maršrutą, t. y. egzistuoja visos briaunos

$$(v_1, w_2) \in E, \quad (w_i, w_{i+1}) \in E,\ i = 2, \ldots, n-1, \quad (w_n, v_1) \in E\,.$$

Tikrinimo metu naudojame grafo viršūnių gretimumo matricą. Jeigu sprendžiame keliaujančio pirklio uždavinį, tai šiuo etapu apskaičiuojame ir gauto maršruto ilgį.

Tarkime, kad naują viršūnių kombinaciją randame, atlikę $\mathcal{O}(1)$ veiksmų. Iš viso skirtingų variantų yra $(n-1)(n-2)\cdots 1 = (n-1)!$. Tikrindami, ar viršūnių kombinacija sudaro maršrutą, nagrinėjame n briaunų svorius, todėl visus variantus perrenkame atlikę $\mathcal{O}(n!)$ veiksmų. Priminsime Strirlingo formulę

$$\sqrt{2\pi n}\,\left(\frac{n}{e}\right)^n \le n! \le \sqrt{2\pi n}\,\left(\frac{n}{e}\right)^n e^{1/(12n)}\,,$$

taigi pilnojo variantų perrinkimo apimtis yra $\mathcal{O}\Big(\Big(\frac{n}{e}\Big)^{n+1/2}\Big)$, ši funkcija didėja greičiau už bet kokią laipsninę funkciją.

Pateiktajame variantų perrinkimo algoritme neatsižvelgta į tai, kad daugumos taikymų sprendžiamų uždavinių grafo viršūnių gretimumo matrica yra reta. Tada galima smarkiai sumažinti nagrinėjamų kelių skaičių. Tarkime, kad grafo viršūnių kaimynų skaičius neviršyja m. Tada skirtingų ciklų yra ne daugiau nei $\mathcal{O}(m^n)$. Tačiau ir šiuo atveju keliaujančio pirklio uždavinio sprendimo algoritmas yra nepolinominio sudėtingumo.

4.3.3. Euristikos

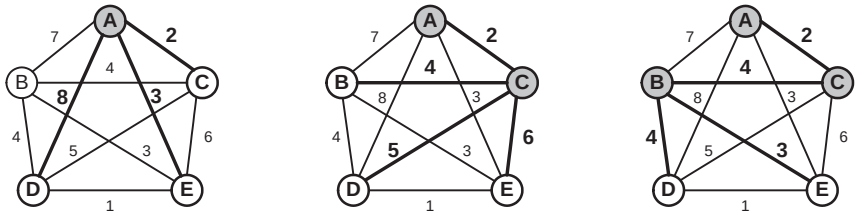
Dažnai užtenka apskaičiuoti pakankamai gerą trumpiausio maršruto artinį, tačiau jį norime rasti labai greitai. Susipažinsime su dviem algoritmais, kuriuos sudarysime naudodami godžiųjų algoritmų strategiją.

Pirmajame algoritme $GS(v_0)$ maršruto paiešką pradedame grafo viršūnėje v_0. Iš kiekvienos viršūnės einame į artimiausią dar neaplankytą grafo viršūnę. Paskutiniu žingsniu vėl grįžtame į pradinę maršruto viršūnę v_0. Priminsime, kad $N(v)$ žymi viršūnės v kaimynų aibę.

Kaliaujančio pirklio uždavinio euristinis algoritmas GS

double **GS** (u)
begin
(1) Q = V / {u}, S = {u}, L= 0, v= u;
(2) **while** (Q $\neq \emptyset$) **do**
(3) **for all** ($v_j \in$ Q $\cap$ N(v)) **do**
(4) Rasti v^*: w(v, v^*) $\leq$ w(v, v_j);
end do
(5) Q := Q / {v^*}, S := S $\cup$ {v^*};
(6) L := L + w(v, v^*), v= v^*;
end do
(7) L := L + w(v, u);
(8) return (L);
end GS

4.7 paveiksle parodyta, kaip GS algoritmu sudarome maršrutą pilnajame grafe, turinčiame penkias viršūnes.



a) $L = 0$ b) $L = 2$ c) $L = 6$

d) $L = 9$ e) $L = 10$ f) $L = 18$

4.7 pav. Pirklio maršruto sudarymas GS algoritmu

GS algoritmo sudėtingumas $\mathcal{O}(|V|+|E|)$, taigi turint net ir pilnąjį grafą, algoritmas yra kvadratinio sudėtingumo $\mathcal{O}(|V|^2)$. Pažymėkime maršruto, apakaičiuoto GS algoritmu, ilgį $L_n(GS)$, o optimalaus maršruto ilgį L_n.

4.4 teorema. *Tegul C yra $n \times n$ dydžio matrica, apibrėžianti atstumus tarp miestų. Tarsime, kad matrica yra simetrinė, t. y. $c_{ij} = c_{ji}$ ir atstumai tenkina trikampio nelygybę*

$$c_{ij} \leqslant c_{ik} + c_{kj}, \quad 1 \leqslant i, j, k \leqslant n.$$

Tada teisingas toks maršruto, apskaičiuoto GS algoritmu, ilgio įvertis:

$$L_n(GS) \leqslant \frac{1}{2}\big(\lceil \log n \rceil + 1\big) L_n.$$

Optimalus pirklio maršrutas nepriklauso nuo viršūnės, iš kurios pradedame paiešką. Tačiau godusis algoritmas negarantuoja, kad rasime optimalų maršrutą, todėl sudarome sudėtingesnę euristiką, kai GS algoritmu $|V|$ kartų ieškome optimalaus maršruto, pradėdami vis iš kitos grafo viršūnės.

Kaliaujančio pirklio uždavinio euristinis algoritmas GS2

```
double GS2 ()
begin
  (1)   L= ∞;
  (2)   for all  ( v ∈ V ) do
  (3)           LN = GS(v);
  (4)           if ( LN < L ) L = LN;
        end do
  (5)   return (L);
end GS2
```

GS2 algoritmo sudėtingumas $\mathcal{O}(|V|^2 + |V|\,|E|)$, taigi turint pilnąjį grafą, algoritmas yra kubinio sudėtingumo $\mathcal{O}(|V|^3)$. Tačiau net ir turėdami 2-3 tūkstančius viršūnių maršrutą apskaičiuojame per kelias sekundes.

Pateiksime dar vieną euristinį algoritmą. Jis panašus į Primo algoritmą, kuriuo skaičiuojame grafo minimalų dengiantįjį medį. Tačiau keliaujančio pirklio uždaviniui tokia strategija leidžia apibrėžti tik euristiką. Kiekvienu algoritmo žingsniu randame trumpiausią briauną $e = (u, v) \in E$, jungiančią jau parinktas maršruto viršūnes S su likusiomis grafo viršūnėmis $Q = V \setminus S$. Tada viršūnę v įtraukiame į maršrutą $S := S \cup \{v\}$. Apibrėžkime aibę briaunų, jungiančių aibės S viršūnes su likusiomis grafo viršūnėmis

$$D = \big\{ e \in E :\ e = (u, v),\ u \in S,\ v \in V \setminus S \big\}.$$

Kaliaujančio pirklio uždavinio euristinis algoritmas GS3

double **GS3** ()
begin
(1) $S = \{v_1 \in V\}$;
(2) **while** ($|S| < |V|$) **do**
(3) Randame briauną $e = (u, v)$: $w(e) = \min_{z \in D} w(z)$;
(4) $S := S \cup \{v\}$;
(5) L := L + w(u, v);
end do
(6) return (L);
end GS3

Realizuodami šį algoritmą kiekvienai dar neparinktai viršūnei $v \in Q$ saugome informaciją apie artimiausią viršūnę $u \in S$. Tada kiekvienu žingsniu atliekame $\mathcal{O}(|V|)$ veiksmų, juo randame naują viršūnę ir atnaujiname informaciją apie artimiausias maršruto viršūnes. Todėl GS3 algoritmo sudėtingumas – $\mathcal{O}(|V|^2)$.

Be įrodymo pateiksime teoremą apie GS3 algoritmu randamo maršruto optimalumą.

4.5 teorema. *Tegul C yra $n \times n$ dydžio matrica, apibrėžianti atstumus tarp miestų. Tarsime, kad matrica yra simetrinė, t. y. $c_{ij} = c_{ji}$ ir atstumai tenkina trikampio nelygybę*

$$c_{ij} \leqslant c_{ik} + c_{kj}, \quad 1 \leqslant i, j, k \leqslant n.$$

Tada teisingas GS3 algoritmu apskaičiuoto maršruto ilgio įvertis

$$L_n(GS3) \leqslant 2L_n.$$

4.4 pavyzdys. Pirklio maršrutas orientuotajame grafe. Sprėsime pirklio optimalaus maršruto radimo uždavinį, kai miestus jungiančių kelių matrica yra nesimetrinė, t. y. grafas yra orientuotasis. Uždavinio grafą parinkome iš TSPLIB uždavinių bibliotekos, sprendėme *ftv33* uždavinį.

4.1 lentelėje pateikti rezultatai, gauti pilnojo perrinkimo metodu, randančiu optimalųjį maršrutą, ir euristiniais algoritmais GS ir GS2. Čia n žymi miestų skaičių, T yra pilnojo perrinkimo algortmo vykdymo laikas sekundėmis. Stulpeliuose PP, GS2 ir GS pateikti maršrutų ilgiai, apskaičiuoti pilnojo perrinkimo ir euristiniais algoritmais.

4.1 lentelė. Pirklio maršruto radimas pilnojo variantų perrinkimo metodu

n	T	PP	GS2	GS
10	0,12	482	482	486
11	1,35	539	574	574
12	15,9	661	737	756
13	202	694	763	816
14	2773	694	775	775

Matome, kad pilnojo perrinkimo algoritmo skaičiavimo laikas padidėja n kartų, palyginti su mažesnės dimensijos uždaviniu, todėl optimalaus maršruto skaičiavimas penkiolikos miestų grafe jau užtruktų net vienuoliką valandų. Tik turėdami dešimt miestų ir algoritmu GS2 radome optimalų maršrutą, visais kitais atvejais euristiniai algoritmai apskaičiavo tiksliojo sprendinio artinius.

Norėdami sumažinti veiksmų skaičių, stengiamės išnaudoti visą turimą informaciją apie optimalų sprendinį. Daugelio metodų pagrindinis tikslas – išsiaiškinti, kurie maršrutų variantai negali būti sprendiniu, ir jų nenagrinėti. Susipažinsime su trimis tokiais metodais.

Modifikuotas perrinkimo algoritmas

Šiame algoritme skaičiuojame jau sudarytos maršruto dalies ilgį. Jei jis tampa didesniu už trumpiausio žinomo maršruto ilgį L_{min}, tai tolesnę paiešką gilyn nutraukiame.

Pateikiame modifikuotą viršūnių aplankymo algoritmą.

VisitM (u)
begin
(1) **if** (t == $|V|$) **then**
(2) **if** ($(u, v_1) \in E$) **then**
(3) $\pi(v_1) = u$;
(4) printCiklas(π);
(5) **if** (L < L_{min}) L_{min} = L;
end if
(6) **else**
(7) **for all** ($v \in N(u)$) **do**

```
(8)          if ( spalva(v) = balta && L + w(u,v) < L_min ) then
(9)            t := t + 1,   spalva(v) = pilka;
(10)           π(v) = u,   L := L+ w(u,v);
(11)           Visit(v);
             end if
           end do
         end if else
(12)   t := t − 1,   spalva(u) = balta;
end VisitM
```

4.5 pavyzdys. Pirklio maršruto radimas modifikuotu perrinkimo algoritmu. Spręsime pirklio optimalaus maršruto radimo uždavinį, suformuluotą 4.4 pavyzdyje. 4.2 lentelėje pateikti rezultatai, gauti modifikuotu perrinkimo metodu ir euristiniais algoritmais GS ir GS2. Čia n žymi miestų skaičių, T yra perrinkimo algoritmo vykdymo laikas sekundėmis. Stulpeliuose MPP, GS2 ir GS pateikti apskaičiuotų maršrutų ilgiai.

4.2 lentelė. Pirklio maršruto radimas modifikuotu variantų perrinkimo metodu

n	T	MPP	GS2	GS
14	8,29	694	775	775
15	52,6	732	830	851
16	244	735	833	851
17	1538	749	832	851

Matome, kad modifikuoto perrinkimo algoritmo skaičiavimo laikas padidėja $0,35n$ kartų, palyginti su mažesnės dimensijos uždaviniu, todėl šis algoritmas yra daug efektyvesnis už pilnojo perrinkimo algoritmą (pastaruoju algoritmu optimalų maršrutą septyniolikos miestų grafe apskaičiuotume tik po keturių mėnesių). Tačiau ir modifikuoto algoritmo sudėtingumo funkcijos asimptotika yra eksponentinė, todėl galime prognozuoti, kad optimalaus maršruto paieška aštuoniolikos miestų grafe jau užtruktų apie 3 valandas. Todėl dažniausiai tenkinamės euristiniu algoritmu apskaičiuotais maršrutais, kurie maždaug 10 procentų ilgesni už optimaliuosius.

4.3.4. Dinaminio programavimo metodas

Pilnojo variantų perrinkimo pagrindinis trūkumas tas, kad daug kartų nagrinėjame tuos pačius maršruto poaibius. Panašios problemos būdingos ir kitiems rekursiją naudojantiems algoritmams. Tai išsamiai aptarėme poskyryje, skirtame rekursinių algoritmų analizei.

2 skyriuje jau parodėme, kad dinaminio programavimo metodas grindžiamas dviem sąlygomis:

1. Visą uždavinį galime nesunkiai išspręsti, jei žinome optimalius mažesnės apimties uždavinių sprendinius. Šią sąlygą vadiname variacine optimalumo sąlyga.

2. Kiekvieną mažesnį uždavinį sprendžiame tik vieną kartą ir jo sprendinį išsaugome ir naudojame daug kartų.

Pirklio maršrutas yra uždaroji ciklinė kreivė, todėl galime ją pradėti nagrinėti nuo bet kurios viršūnės. Tarkime, kad tai yra v_1 viršūnė. Jei optimaliame maršrute pirmiausia einame į v_k viršūnę, tai gauname naują uždavinį:

- Reikia rasti trumpiausią kelią nuo v_k iki v_1, kai kiekvieną aibės V viršūnę aplankome po vieną kartą.

Tegul $S \subset V$ yra aibės V viršūnių poaibis. Pažymėkime $g(k, S)$ funkciją, kuri apibrėžia ilgį trumpiausio kelio, einančio iš v_k iki v_1, kai aplankome po vieną kartą kiekvieną aibės S viršūnę. Tada, spręsdami keliaujančio pirklio uždavinį, ieškome funkcijos $g\big(1, V \setminus \{v_1\}\big)$ reikšmės. Jei pirmoji optimalaus ciklo atkarpa yra (v_1, v_k), tai

$$g\big(1, V \setminus \{v_1\}\big) = w(v_1, v_k) + g\big(k, V \setminus \{v_1, v_k\}\big)\,,$$

čia $w(v_i, v_k)$ pažymėjome atkarpos (v_1, v_k) ilgį. Mes, aišku, nežinome, kuri viršūnė v_k bus aplankyta pirmoji, todėl gauname variacinę lygybę

$$g\big(1, V \setminus \{v_1\}\big) = \min_{2 \leqslant k \leqslant n} \big(w(v_1, v_k) + g\big(k, V \setminus \{v_1, v_k\}\big)\big)\,.$$

Taigi, jei žinotume visų mažesnių uždavinių $g\big(k, V \setminus \{v_1, v_k\}\big), k = 2, \ldots, n$ sprendinius, tai nesunkiai galėtume rasti ir keliaujančio pirklio uždavinio sprendinį.

Šią strategiją taikome rekursyviai mažesnės apimties uždaviniams, gauname pagrindinę *variacinę optimalumo* sąlygą

$$g\big(i, S\big) = \min_{v_k \in S} \big(w(v_i, v_k) + g\big(k, S \setminus \{v_k\}\big)\big)\,.$$

Rekursijos pabaiga yra triviali:

$$g\big(i,\emptyset\big) = w(v_i, v_1)\,.$$

Spręsdami keliaujančio pirklio uždavinį dinaminio programavimo metodu pirmiausia apskaičiuojame visas $g\big(k,\emptyset\big)$ reikšmes, paskui $g\big(k,\{v_j\}\big)$ reikšmes, šį procesą tęsiame tol, kol randame uždavinio sprendinį $g\big(1, V\setminus\{v_1\}\big)$. Kadangi iš anksto nežinome optimalaus maršruto, tenka apskaičiuoti visų pagalbinių mažesnių uždavinių sprendinius. Tačiau skirtingai nuo pilnojo variantų perrinkimo išvengiame pasikartojančių kelio atkarpų analizės.

Mums svarbus ne tik trumpiausio ciklo ilgis, bet ir pats maršrutas, todėl visada įsimename numerį tos viršūnės $v_j \in S$, kuri suteikia minimalią reišmę funkcijai $g\big(i,S\big)$.

Eksperimentai, atlikti su atsitiktinai generuotais grafais, parodė, kad tokio algoritmo sudėtingumas $\mathcal{O}(n^2 2^n)$, t. y. nagrinėjamų variantų skaičius yra daug mažesnis už bendrą variantų skaičių $n!$.

4.6 pavyzdys. Optimalaus maršruto radimas dinaminio programavimo metodu. Nagrinėkime orientuotąjį pilnąjį grafą G, kuris apibrėžia kelių ilgius tarp penkių miestų. Kelių ilgiai yra tokie:

$$\mathbf{S} = \begin{pmatrix} 0 & 12 & 8 & 16 & 23 \\ 9 & 0 & 14 & 16 & 13 \\ 15 & 13 & 0 & 24 & 14 \\ 7 & 20 & 16 & 0 & 14 \\ 21 & 12 & 28 & 12 & 0 \end{pmatrix}.$$

Tarsime, kad pirklys pradeda savo kelionę iš pirmosios viršūnės (optimalus ciklas, aišku, nepriklauso nuo šio pasirinkimo). Atkarpos (v_i, v_k) ilgį žymėsime c_{ik}.

Pirmiausia randame $g(i,\emptyset) = c_{i1}$ reikšmes

$$g(2,\emptyset) = 9, \quad g(3,\emptyset) = 15, \quad g(4,\emptyset) = 7, \quad g(5,\emptyset) = 21\,.$$

Toliau nagrinėjame maršrutus iš viršūnės v_i iki v_1, kai pakeliui aplankome vieną viršūnę v_k. Remdamiesi variacine optimalaus ciklo sąlyga, gauname lygybę

$$g\big(i,\{v_k\}\big) = c_{ik} + g(k,\emptyset) = c_{ik} + c_{k1}, \quad i \neq 1,\ k \neq 1, i\,.$$

Apskaičiuojame funkcijos $g\big(i,\{v_k\}\big)$ reikšmes

$$\begin{aligned}
&g\big(2,\{v_3\}\big)=14+15=29,\ \ g\big(2,\{v_4\}\big)=23,\ \ g\big(2,\{v_5\}\big)=34,\\
&g\big(3,\{v_2\}\big)=13+9=22,\ \ g\big(3,\{v_4\}\big)=31,\ \ g\big(3,\{v_5\}\big)=35,\\
&g\big(4,\{v_2\}\big)=20+9=29,\ \ g\big(4,\{v_3\}\big)=31,\ \ g\big(4,\{v_5\}\big)=35,\\
&g\big(5,\{v_2\}\big)=12+9=21,\ \ g\big(5,\{v_3\}\big)=43,\ \ g\big(5,\{v_4\}\big)=19\,.
\end{aligned}$$

Toliau nagrinėjame maršrutus, kurie aplanko du tarpinius miestus

$$g\big(i,\{v_j,v_k\}\big)=\min\big(c_{ij}+g\big(j,\{v_k\}\big),\ \ c_{ik}+g\big(k,\{v_j\}\big)\big)\,.$$

Priminsime, kad turime saugoti ne tik optimalių maršrutų ilgius, bet ir pačius maršrutus. Gauname tokias $g\big(i,\{v_j,v_k\}\big)$ reikšmes ir optimalius maršrutus:

$$\begin{aligned}
&g\big(2,\{v_3,v_4\}\big)=\min(14+31,\,16+31)=45, && \{v_2,\,v_3,\,v_4,\,v_1\}\,,\\
&g\big(2,\{v_3,v_5\}\big)=\min(14+35,\,13+43)=49, && \{v_2,\,v_3,\,v_5,\,v_1\}\,,\\
&g\big(2,\{v_4,v_5\}\big)=\min(16+35,\,13+19)=32, && \{v_2,\,v_5,\,v_4,\,v_1\}\,,\\
&g\big(3,\{v_2,v_4\}\big)=\min(13+23,\,24+29)=36, && \{v_3,\,v_2,\,v_4,\,v_1\}\,,\\
&g\big(3,\{v_2,v_5\}\big)=\min(13+34,\,14+21)=35, && \{v_3,\,v_5,\,v_2,\,v_1\}\,,\\
&g\big(3,\{v_4,v_5\}\big)=\min(24+35,\,14+19)=33, && \{v_3,\,v_5,\,v_4,\,v_1\}\,,\\
&g\big(4,\{v_2,v_3\}\big)=\min(20+29,\,16+22)=38, && \{v_4,\,v_3,\,v_2,\,v_1\}\,,\\
&g\big(4,\{v_2,v_5\}\big)=\min(20+34,\,14+21)=37, && \{v_4,\,v_5,\,v_2,\,v_1\}\,,\\
&g\big(4,\{v_3,v_5\}\big)=\min(16+35,\,14+43)=51, && \{v_4,\,v_3,\,v_5,\,v_1\}\,,\\
&g\big(5,\{v_2,v_3\}\big)=\min(12+29,\,28+22)=41, && \{v_5,\,v_2,\,v_3,\,v_1\}\,,\\
&g\big(5,\{v_2,v_4\}\big)=\min(12+23,\,12+29)=35, && \{v_5,\,v_2,\,v_4,\,v_1\}\,,\\
&g\big(5,\{v_3,v_4\}\big)=\min(28+24,\,12+31)=43, && \{v_5,\,v_4,\,v_3,\,v_1\}\,.
\end{aligned}$$

Kitu žingsniu nagrinėjame maršrutus iš v_i iki v_1, kurie eina per tris tarpinius miestus:

$$\begin{aligned}
g\big(i,\{v_j,v_k,v_l\}\big)=\min\big\{&c_{ij}+g\big(j,\{v_k,v_l\}\big),\ \ c_{ik}+g\big(k,\{v_j,v_l\}\big),\\
&c_{il}+g\big(l,\{v_j,v_k\}\big)\big\}\,.
\end{aligned}$$

Gauname tokius optimalius maršrutus ir jų ilgius:

$$g\big(2,\{v_3,v_4,v_5\}\big)=\min(\,47,\ 67,\ 56\,)=47,\quad \{v_2,\,v_3,\,v_5,\,v_4,\,v_1\},$$
$$g\big(3,\{v_2,v_4,v_5\}\big)=\min(\,45,\ 61,\ 49\,)=45,\quad \{v_3,\,v_2,\,v_5,\,v_4,\,v_1\},$$
$$g\big(4,\{v_2,v_3,v_5\}\big)=\min(\,69,\ 51,\ 55\,)=51,\quad \{v_4,\,v_3,\,v_5,\,v_2,\,v_1\},$$
$$g\big(5,\{v_2,v_3,v_4\}\big)=\min(\,57,\ 64,\ 50\,)=50,\quad \{v_5,\,v_4,\,v_3,\,v_2,\,v_1\}.$$

Turėdami šią informaciją, apskaičiuojame optimalų ciklą, kuriuo keliaudamas, pirklys greičiausiai apvažiuos visus miestus:

$$g\big(1,\{v_2,v_3,v_4,v_5\}\big)=\min_{2\leqslant j\leqslant 5}\big\{c_{ij}+g\big(j,\{v_2,v_3,v_4,v_5\}\setminus\{v_j\}\big)\big\}.$$

Tokio ciklo ilgis yra

$$g\big(1,\{v_2,v_3,v_4,v_5\}\big)=\min(\,59,\ 53,\ 67,\ 73\,)=53,$$

o pats ciklas apibrėžiamas taip $\{\,v_1,\ v_3,\ v_2,\ v_5,\ v_4,\ v_1\}$.

4.3.5. Šakų ir rėžių metodas

Šis metodas padeda sumažinti nagrinėjamų variantų skaičių. Priminsime pagrindinius jo žingsnius. Sakykime, kad sprendžiame variacinį uždavinį

$$f(X^*)=\min_{X\in D} f(X).$$

1. **Skaidymo žingsnis.** Leistinųjų sprendinių aibę skaidome į baigtinį skaičių mažesnių aibių $D=\bigcup D_i$. Medžio šaknyje užrašome uždavinį $P(f,D)$, šaknis turi m vaikų, kurie apibrėžia uždavinius $P(f,D_i)$, t. y. tikslo funkciją $f(X)$ minimizuojame srityse D_i. Toks skaidymo procesas tęsiamas rekursijos būdu, kol nesunkiai apskaičiuojame gautojo uždavinio sprendinį.

2. **Rėžių skaičiavimas.** Skaičiuojame tikslo funkcijos $f(X)$ reikšmių apatinį $LB(f,D_i)$ ir viršutinį $UB(f,D_i)$ rėžius

$$LB(f,D_i)\leq f(X)\leq UB(f,D_i),\quad X\in D_i.$$

Tada galime apskaičiuoti funkcijos mažiausios reikšmės $f(X^*)$ viršutinį rėžį $UB(f,D)$ visoje aibėje D:

$$UB(f,D)=\min_{1\leq i\leq m} UB(f,D_i).$$

Žinodami šiuos rėžius dažnai gerokai sumažiname nagrinėjamų sričių skaičių. Jeigu $LB(f,D_i)>UB(f,D)$, tai tokios srities toliau nebetiriame, nes joje tikrai nėra optimalaus sprendinio.

3. **Viršūnės išrinkimo taisyklė.** Nurodome, kurią medžio viršūnę nagrinėsime eiliniu žingsniu. Pavyzdžiui, galime imti viršūnę, kurios apatinis rėžis mažiausias.

Šiuos žingsnius realizuosime spręsdami pirklio uždavinį.

Skaidymo žingsnis

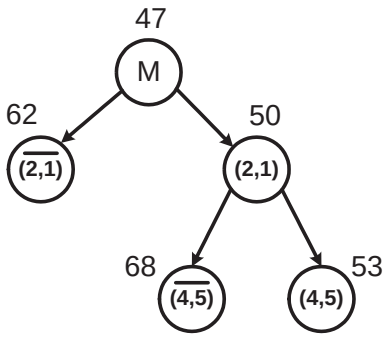
Pažymėkime M visų pirklio maršrutų aibę, joje ieškome trumpiausio maršruto. Tada pasirenkame vieną briauną $e \in E$ ir padalijame M į dvi aibes

$$M = M(e) \cup M(\bar{e}),$$

čia $M(e)$ sudaro visi maršrutai, kuriems priklauso briauna e, o $M(\bar{e})$ priklauso tik tie maršrutai, kuriuose nėra šios briaunos. 4.8 paveiksle pateikta grafo viršūnių gretimumo matrica ir parodyta, kaip sudarome optimalaus kelio paieškos medį. Tada, pavyzdžiui, viršūnė, kurios apatinis rėžis 68, apibrėžia visus maršrutus, kuriems priklauso briauna $(2, 1)$ ir nepriklauso briauna $(4, 5)$.

	1	2	3	4	5
1	B	25	40	31	27
2	5	B	17	30	25
3	19	15	B	6	1
4	9	50	24	B	6
5	22	8	7	10	B

a)



b)

4.8 pav. Trumpiausio maršruto radimas šakų ir rėžių algoritmu: a) grafo viršūnių gretimumo matrica, b) paieškos medis ir viršūnių apatiniai rėžiai

Algoritmo efektyvumas smarkiai priklauso nuo to, kurią grafo briauną pasirenkame, skaidydami medžio viršūnę. Vieną taisyklę pateiksime, kai išnagrinėsime rėžių skaičiavimo algoritmus.

Rėžio skaičiavimas

Jau nagrinėdami modifikuotą perrinkimo algoritmą, įsitikinome, kad net labai paprastas apatinio rėžio įvertis gali smarkiai sumažinti nagrinėjamų

variantų skaičių. Tame algoritme rėžį skaičiavome nesudėtingai, jis buvo lygus jau parinktų briaunų ilgių sumai. Aišku, kad algoritmas bus efektyvesnis, jei tiksliau įvertinsime pasirinktai medžio viršūnei priklausančių maršrutų ilgių apatinį rėžį.

Stengsimės įvertinti svarbiausią informaciją apie optimalų pirklio maršrutą ir išnaudosime visus duomenis apie jau atliktus žingsnius.

1. Bet kuriame pilnajame maršrute iš kiekvienos grafo viršūnės viena briauna išeina ir viena briauna į ją įeina. Taigi iš grafo viršūnių gretimumo matricos kiekvienos eilutės ir stulpelio turime parinkti po vieną elementą. Pastebėsime, kad atvirkščias teiginys nėra teisingas, nes, pavyzdžiui, briaunų rinkinys

   $$(2,1),\ (1,2),\ (3,5),\ (5,4),\ (4,3)$$

   tenkina šią sąlygą, bet neapibrėžia maršruto.

2. Jeigu žinome, kad briauna e_{ij} nepriklauso nė vienam aibės maršrutui, tai tokios briaunos svorį padidiname iki ∞.

3. Jeigu žinome, kad briauna e_{ij} priklauso visiems aibės maršrutams, tai iš grafo viršūnių gretimumo matricos pašaliname i-tąją eilutę ir j-ąjį stulpelį, o briaunos e_{ij} ilgį pridedame prie jau sudaryto kelio ilgio.

4. Išrinkdami perspektyvias briaunas, tikriname, kad jos nesudarytų ciklo, kol dar neaplankytos visos grafo viršūnės.

Pasinaudosime viena svarbia uždavinio savybe. Jei iš grafo viršūnių gretimumo matricos kurios nors eilutės ar stulpelio visų koeficientų atimsime l, tai visų maršrutų ilgiai taip pat sumažės tokiu pačiu dydžiu. Todėl duotosios matricos maršrutų ilgių apatinį rėžį randame taip:

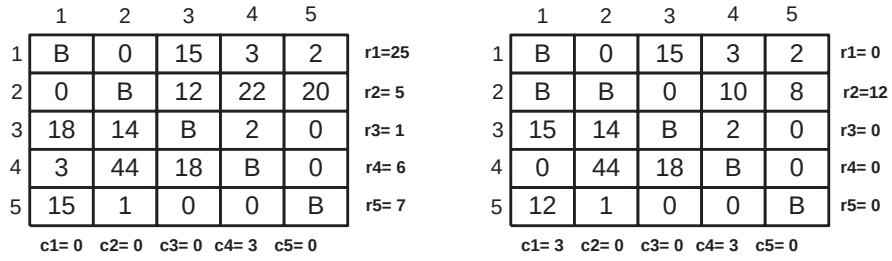
a) kiekvienoje matricos eilutėje randame mažiausią koeficiento reikšmę r_i ir ją atimame iš kiekvieno tos eilutės koeficiento;

b) kiekviename modifikuotos matricos stulpelyje randame mažiausią koeficiento reikšmę c_i ir ją atimame iš kiekvieno to stulpelio koeficiento.

Tada grafo maršrutų M ilgių apatinis rėžis yra

$$LB(M) = \sum_{i=1}^{N}(r_i + c_i).$$

4.9 paveikslo *a* dalyje parodyta, kaip apskaičiuojame 4.8 paveiksle pateikto grafo visų maršrutų ilgių apatinį rėžį.



4.9 pav. Apatinio rėžio skaičiavimas: a) visų grafo maršrutų aibėje, b) maršrutų, kuriems nepriklauso briauna $(2,1)$, aibėje $M(\bar{e}_{21})$

Dabar apskaičiuosime maršrutų, kuriems nepriklauso grafo briauna $(2,1)$, ilgių apatinį rėžį. Modifikuotoje matricoje pakeičiame atitinkamo koeficiento reikšmę $w_{21} = \infty$ ir vėl taikome tą patį algoritmą. Tada gauname, kad $r_2 = 12, c_1 = 3$, todėl (žr. 4.9 paveikslo b dalį):

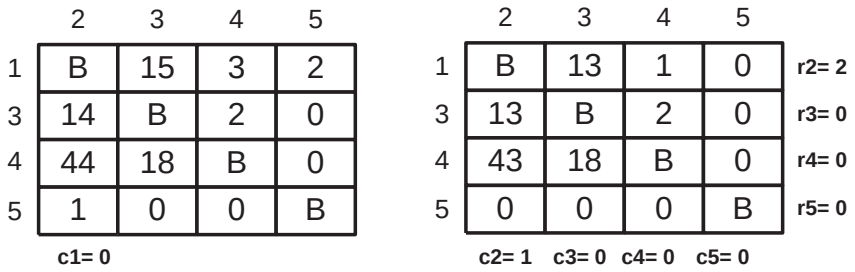
$$LB\big(M(\overline{(2,1)})\big) = 47 + 12 + 3 = 62.$$

Taigi šis įvertis skaičiuojamas labai greitai, nes užtenka nagrinėti tik vienos eilutės ir vieno stulpelio koeficientus.

Įvertinsime maršrutų, kuriems priklauso grafo briauna $(2,1)$, ilgių apatinį rėžį. Iš modifikuotos matricos išbraukiame antrąją eilutę ir pirmąjį stulpelį. Kadangi šioje matricoje $w_{21} = 0$, tai maršruto ilgio prognozės nedidiname. Kol kas turime tik vieną kelio atkarpą, kuri prasideda viršūnėje v_2 ir baigiasi viršūnėje v_1. Norėdami išvengti trumpo ciklo susidarymo, pakeičiame koeficiento w_{12} reikšmę, ją padidiname $w_{12} = \infty$. Gautoji 4×4 dydžio matrica ir jos įvertinimo eiga pavaizduoti 4.10 paveiksle.

Dabar galime paaiškinti taisyklę, kaip parenkame briauną šakos žingsnyje. Siekiame rasti tokią briauną e, kad įvertis $LB\big(M(\overline{e})\big)$ būtų kuo didesnis. Tikimės, kad tai palengvins tokios aibės variantų analizę ir padidins tikimybę, kad jos nereikės detaliai tirti. Taip pat darome prielaidą, kad tos briaunos, kurių svoriai modifikuotoje matricoje yra lygūs nuliui, su didesne tikimybe priklausys optimaliam maršrutui. Todėl įvertiname apatinius rėžius visų aibių $M(\overline{e_{ij}})$, kurioms $w_{ij} = 0$, ir išrenkame tą briauną, kuriai šis rėžis yra didžiausias. Mūsų pavyzdžiu turėjome šešias tokias briaunas, o didžiausią rėžį gavome imdami (2,1) briauną.

Skaidymo žingsnį baigiame, kai gauname 2×2 dydžio matricą, tada suskaičiuojame tikslų maršruto ilgį ir patikriname, ar naujasis maršrutas yra



4.10 pav. Maršrutų, kuriems priklauso briauna (2,1), ilgių apatinio rėžio skaičiavimas: a) pradinė matrica, b) modifikuota matrica

trumpesnis už anksčiau surastus maršrutus. Jei radome geresnį sprendinį, tai jį įsimename ir pakeičiame viršutinį rėžį $UB(D)$.

Eksperimentai, atlikti su atsitiktinai generuotais grafais, parodė, kad tokio algoritmo sudėtingumas $\mathcal{O}(1,26^n)$, t. y. šakų ir rėžių algoritmas yra gerokai efektyvesnis už dinaminio programavimo algoritmą.