Dinaminio programavimo metodas

Raimondas Čiegis

Matematinio modeliavimo katedra, e-paštas: rc@vgtu.lt

Lapkričio 30 d., 2022

Šioje paskaitoje susipažinsime su vieno populiaraus klasikinio uždavinio sprendimo algoritmais. Parodysime, kaip galime juos sudaryti naudojant dinaminio programavimo metodą.

Šioje paskaitoje susipažinsime su vieno populiaraus klasikinio uždavinio sprendimo algoritmais. Parodysime, kaip galime juos sudaryti naudojant dinaminio programavimo metodą.

Nagrinėsime ne tik metodus, leidžiančius rasti tikslų uždavinio sprendinį, bet ir euristinius algoritmus, kuriais apskaičiuosime tik sprendinio artinius. Tačiau euristikos tinka ir tada, kai duomenų skaičius yra labai didelis.

Keliaujančio pirklio uždavinys

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

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

Keliaujančio pirklio uždavinys

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.

Keliaujančio pirklio uždavinys

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 $w(e_{ij}) = \infty$.

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

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

vadiname maršrutu.

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

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

vadiname maršrutu.

Reikia rasti trumpiausią pirklio maršrutą.

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

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

vadiname maršrutu.

Reikia rasti trumpiausią pirklio maršrutą.

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

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.

Pirmajame algoritme $GS(v_0)$ maršruto paiešką pradedame grafo viršūnėje v_0.

Iš kiekvienos viršūnės einame į artimiausią dar neaplankytą grafo viršūnę.

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.

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.

$N(v)$ žymėsime 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** $\big($ Q $\neq \emptyset$ $\big)$ **do**
(3) **for all** $\big($ $v_j \in$ Q $\cap$ N(v) $\big)$ **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

GS algoritmo sudėtingumas

$$\mathcal{O}(|V| + |E|).$$

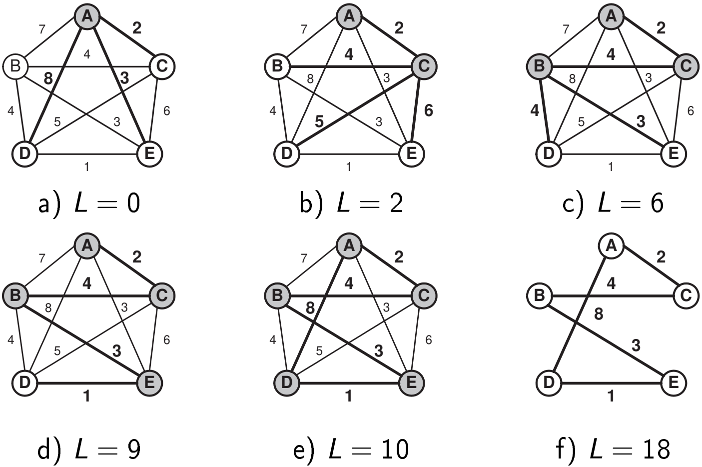

FIGURE: Pirklio maršruto sudarymas GS algoritmu

Pažymėkime maršruto, apakaičiuoto GS algoritmu, ilgį $L_n(GS)$, o optimalaus maršruto ilgį L_n.

Pažymėkime maršruto, apakaičiuoto GS algoritmu, ilgį $L_n(GS)$, o optimalaus maršruto ilgį L_n.

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

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

Pažymėkime maršruto, apakaičiuoto GS algoritmu, ilgį $L_n(GS)$, o optimalaus maršruto ilgį L_n.

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.

Optimalus pirklio maršrutas nepriklauso nuo viršūnės, iš kurios pradedame paiešką.

Tačiau godusis algoritmas negarantuoja, kad rasime optimalų maršrutą, todėl sudarome sudėtingesnę euristiką GS2, kai GS algoritmu $|V|$ kartų ieškome optimalaus maršruto, pradėdami vis iš kitos grafo viršūnės.

Optimalus pirklio maršrutas nepriklauso nuo viršūnės, iš kurios pradedame paiešką.

Tačiau godusis algoritmas negarantuoja, kad rasime optimalų maršrutą, todėl sudarome sudėtingesnę euristiką GS2, kai GS algoritmu $|V|$ kartų ieškome optimalaus maršruto, pradėdami vis iš kitos grafo viršūnės.

GS2 algoritmo sudėtingumas

$$\mathcal{O}(|V|^2 + |V|\,|E|).$$

TABLE: Pirklio maršruto radimas pilnojo variantų perrinkimo metodu ir euristiniais algoritmais

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

TABLE: Pirklio maršruto radimas pilnojo variantų perrinkimo metodu ir euristiniais algoritmais

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 PP skaičiavimo laikas padidėja n kartų, palyginti su mažesnės dimensijos uždaviniu.

Modifikuotas pilnojo variantų perrinkimo algoritmas MPP

Šiame algoritme skaičiuojame jau sudarytos maršruto dalies ilgį.

Modifikuotas pilnojo variantų perrinkimo algoritmas MPP

Šiame algoritme skaičiuojame jau sudarytos maršruto dalies ilgį.

Jei jis tampa didesniu už trumpiausio žinomo maršruto ilgį L_{min}, tai tolesnę paiešką gilyn nutraukiame.

Table: Pirklio maršruto radimas modifikuotu pilnojo 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

TABLE: Pirklio maršruto radimas modifikuotu pilnojo 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 MPP algoritmo skaičiavimo laikas padidėja $0,35n$ kartų, palyginti su mažesnės dimensijos uždaviniu, todėl šis algoritmas yra daug efektyvesnis už pilnojo perrinkimo algoritmą.

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.

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, jo sprendinį išsaugome ir naudojame daug kartų.

Pirklio maršrutas yra uždaroji ciklinė kreivė, todėl galime ją pradėti nagrinėti nuo bet kurios viršūnės.

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 iš v_1 pirmiausia einame ¿ v_k viršūnę, tai gauname naują uždavin¿:

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.

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 nežinome, kuri viršūnė v_k bus aplankyta pirmoji, todėl gauname variacinę lygtį

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

Mes nežinome, kuri viršūnė v_k bus aplankyta pirmoji, todėl gauname variacinę lygtį

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

Jeigu žinotume visų mažesnių uždavinių

$$g\big(k, V \setminus \{v_1, v_k\}\big), \quad k = 2, \dots, n$$

sprendinius, tai, atlikę mažos apimties perrinkimą, nesunkiai galėtume rasti ir viso keliaujančio pirklio uždavinio sprendinį.

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

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

Rekursijos pabaiga yra triviali:

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

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

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

Rekursijos pabaiga yra triviali:

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

Kadangi algoritmą realizuojame "iš apačios ¿ viršų", tai šios sąlygos ir vykdomos algoritmo pradžioje!

Spręsdami keliaujančio pirklio uždavinį dinaminio programavimo metodu pirmiausia apskaičiuojame visas $g(k, \emptyset)$ reikšmes, Spręsdami keliaujančio pirklio uždavinį dinaminio programavimo metodu pirmiausia apskaičiuojame visas $g\big(k, \emptyset\big)$ reikšmes,

paskui skaičiuojame $g\big(k, \{v_j\}\big)$ reikšmes, Spręsdami keliaujančio pirklio uždavinį dinaminio programavimo metodu pirmiausia apskaičiuojame visas $g\big(k, \emptyset\big)$ reikšmes,

paskui skaičiuojame $g\big(k, \{v_j\}\big)$ reikšmes,

šį procesą tęsiame tol, kol randame uždavinio sprendinį

$$g\big(1, V \setminus \{v_1\}\big).$$

Spręsdami keliaujančio pirklio uždavinį dinaminio programavimo metodu pirmiausia apskaičiuojame visas $g(k, \emptyset)$ reikšmes,

paskui skaičiuojame $g(k, \{v_j\})$ reikšmes,

šį procesą tęsiame tol, kol randame uždavinio sprendinį

$$g(1, V \setminus \{v_1\}).$$

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.

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 $g\left(i, S\right)$.

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 $n!$.

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 $n!$.

Pridėjus papildomą grafo viršūnę, sprendimo laikas pailgėja maždaug du kartus.

Priminsime, PP algoritmo atveju skaičiavimo laikas pailgėdavo n kartų, MPP algoritmui šis augimas buvo mažesnis $n/3$. Bet vis tiek, tai daug blogesnis rezultatas, nei sprendžiant uždavinį dinaminio programavimo metodo algoritmu.

Optimalaus maršruto radimas dinaminio programavimo metodu.

Nagrinėkime orientuotąjį pilnąjį grafą G, kuris apibrėžia kelių ilgius tarp penkių miestų. Kelių ilgiai yra tokie:

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

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

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

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

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

Toliau nagrinėjame maršrutus, kurie aplanko du tarpinius miestus

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

Prisiminkime, kad turime saugoti ne tik optimalių maršrutų ilgius, bet ir pačius maršrutus.

Toliau nagrinėjame maršrutus, kurie aplanko du tarpinius miestus

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

Prisiminkime, 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:

$$g\big(2, \{v_3, v_4\}\big) = \min(14 + 31,\, 16 + 31) = 45, \quad \{v_2,\, v_3,\, v_4,\, v_1\}\,,$$
$$g\big(2, \{v_3, v_5\}\big) = \min(14 + 35,\, 13 + 43) = 49, \quad \{v_2,\, v_3,\, v_5,\, v_1\}\,,$$
$$g\big(2, \{v_4, v_5\}\big) = \min(16 + 35,\, 13 + 19) = 32, \quad \{v_2,\, v_5,\, v_4,\, v_1\}\,,$$

$$g\big(3, \{v_2, v_4\}\big) = \min(13 + 23,\, 24 + 29) = 36, \quad \{v_3,\, v_2,\, v_4,\, v_1\}\,,$$
$$g\big(3, \{v_2, v_5\}\big) = \min(13 + 34,\, 14 + 21) = 35, \quad \{v_3,\, v_5,\, v_2,\, v_1\}\,,$$
$$g\big(3, \{v_4, v_5\}\big) = \min(24 + 35,\, 14 + 19) = 33, \quad \{v_3,\, v_5,\, v_4,\, v_1\}\,,$$

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

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

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

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

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

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:

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

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 $\{ v_1,\ v_3,\ v_2,\ v_5,\ v_4,\ v_1 \}$.