DAUGAVPILS UNIVERSITĀTE
Dabaszinātņu un matemātikas fakultāte
Matemātikas katedra
Maģistra studiju programma "Matemātika"

Studiju kurss

Diskrētā matemātika

14.lekcija

Docētājs: Dr. P. Daugulis

2012./2013.studiju gads

Saturs

Lekcijas mērķis:

- apgūt svarīgākos grafu algoritmus.

Lekcijas kopsavilkums:

- daži grafu algoritmi balstās uz visu variantu pārlasi.

Svarīgākie jēdzieni: plūsmas maksimizācijas uzdevums, maksimālas neatkarīgas kopas atrašanas problēma, krāsošanas problēma, topoloģiskās šķirošanas problēma, ceļojošā tirgoņa problēma.

Svarīgākie fakti un metodes: pilnas un maksimālas plūsmas atrašanas algoritmi ar uzlabošanas metodi, Forda-Falkersona teorēma, plūsmas maksimizācijas pielietojumi, maksimālu neatkarīgu kopu meklēšanas un krāsošanas algoritmi, topoloģiskās šķirošanas algoritms.

1. Grafu algoritmi II

1.1. Plūsmas maksimizācija

1.1.1. Problēma

Inženierzinātnēs ir izplatīti uzdevumi, kas ir saistīti ar fizikālu vai cita veida plūsmu vadīšanu to transportēšanas tīklos.

1.1. piemērs. Maksimālās caurlaides spējas noteikšana autoceļu tīklā, naftas vadu tīklā, datortīklā, loģistikas vai ražošanas tīklā, asinsvadu vai nervu tīklā. Viens no svarīgākajiem jautājumiem, kas var būt uzdots šādā situācijā, ir jautājums par maksimālo pieļaujamo plūsmu tīklā.

$\Gamma = (V, E, c)$ orientēts nosvērts grafs ar vienu avotu s un vienu noteku t (*tīkls*). $c : E \to \mathbb{R}^+$ - *šķautņu caurlaides funkciju*.

Funkcijas f vērtību ar argumentu $e = (u, v)$ apzīmēsim ar $f(e)$ vai $f(u, v)$.

Tīkla griezums - šķautņu kopa, kas atdala avotu no notekas.

Plūsmu tīklā asociēsim ar funkciju $f : E \to \mathbb{R}^+ \cup \{0\}$.

Par virsotnes v *f-diverģenci (plūsmu caur virsotni)* sauc lielumu

$$div(v) = \sum_{u:(u,v)\in E} f(u,v) - \sum_{w:(v,w)\in E} f(v,w).$$

Funkciju f sauksim par *plūsmu (pieļaujamu plūsmu)* tīklā Γ, ja

1) $\forall\ (u,v) \in E$ izpildās nosacījums $0 \leq f(u,v) \leq c(u,v)$ (plūsma pa jebkuru šķautni nepārsniedz šķautnes caurlaides spēju);
2) $\forall\ v \in V$, izņemot avotu un noteku, izpildās nosacījums $div(v) = 0$ (plūsma nekur neuzkrājas).

$div(t)$ - *plūsmas lielums*, $|f|$.

f - pieļaujamā plūsma tīklā. Šķautne e attiecībā uz f ir

- *tukša*, ja plūsma pa to ir vienāda ar nulli: $f(e) = 0$,
- *piesātināta*, ja plūsma pa to ir vienāda ar caurlaides spēju: $f(e) = c(e)$.

Plūsmu sauksim par

- *pilnu*, ja katrs maršruts no avota uz noteku satur vismaz vienu piesātinātu šķautni.
- *maksimālu*, ja tās lielums ir maksimāls starp visām pieļaujamām plūsmām.

Plūsmas maksimizācijas uzdevums: dotajam tīklam $\Gamma = (V, E, c)$ atrast maksimālu plūsmu.

1.1.2. Algoritmi

Pilnas plūsmas atrašana

Pirmajā tuvinājumā mēs varētu konstruēt pieļaujamo plūsmu, kas ir pilna.

Pilnas plūsmas konstruēšanas algoritmu.

1) Sākuma brīdī definēsim nulles plūsmu $f(e) = 0$ katrai šķautnei e, konstruējam palīggrafu $\Gamma' = \Gamma$.

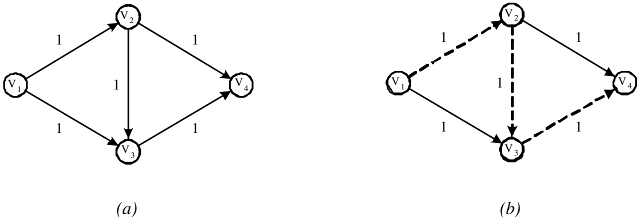
2) Meklējam grafā Γ' virzītu ķēdi p no avota uz tīklu. Ja tāda ķēde neeksistē, tad ir iegūta pilna plūsma, apstājamies. Ja tāda ķēde eksistē, tad ejam uz soli (3).

3) Palielinām plūsmu katrā ķēdes p posmā par lielumu f_p, kas ir vienāds ar minimālo caurlaides spēju ķēdes p šķautnēs (tā, lai vismaz viena šķautne būtu piesātināta). Kopējā plūsma palielinās par f_p, plūsma joprojām ir pieļaujama.

4) Modificējam grafu Γ': samazinam caurlaides spēju ķēdes p šķautnēs par f_p, izdzēšam grafā Γ' visas piesātinātās šķautnes. Konstruējam šī soļa rezultātā jaunu grafu Γ'. Pārejam uz soli 2.

Šāda tipa metodi sauksim par *uzlabojošo ķēžu metodi*.

Diemžēl pilna plūsma var arī nebūt maksimāla, tāpēc uzdevums par maksimālo plūsmu vēl nav atrisināts.



3.70. attēls. (a) tīkla piemērs, (b) pilnas un nemaksimālas plūsmas piemērs

Forda-Falkersona algoritms

Pilnas plūsmas konstruēšanas algoritmā mēs apskatījām tikai virzītas ķēdes, kas ir virzītas no avota uz noteku.

Izrādās, ka ir lietderīgi apskatīt visas iespējamās, ne obligāti virzītās ķēdes no avota uz noteku.

Var redzēt, ka plūsmu var palielināt, ja eksistē ķēde no avota uz noteku ar šādu īpašību:

- katra šķautne, kas ir vērsta no avota uz noteku, nav piesātināta,
- katra šķautne, kas ir vērsta no notekas uz avotu, nav tukša

(*"no full forward or empty backward edges"*).

Var pierādīt, ka algoritms, kurā tiek pakāpeniski palielināta plūsma tik ilgi, līdzko nav ķēžu ar šo īpašību, tiešām atrod maksimālo plūsmu.

Šāda tipa algoritmi balstās uz *Forda-Falkersona* jeb *max-flow mincut* teorēmu: maksimālas plūsmas lielums ir vienāds ar minimālu griezuma kapacitāti.

Gan pilnas, gan arī maksimālas plūsmas meklēšana ar uzlabojošo ķēžu metodi tiek realizēta, konstruējot un uzturot mainīgu palīgtīklu, kura šķautņu caurlaides spējas tiek modificētas algoritma darba laikā, lai atspoguļotu konstruētās plūsmas ietekmi uz tīkla atlikušo kapacitāti. Virzītie maršruti šajā palīgtīklā tiek noteikti, izmantojot īsākā ceļa meklēšanas algoritmus.

Operāciju skaits

Var pierādīt, ka plūsmas maksimizāciju var realizēt ar algoritmu, kas patērē $O(|V||E|^2)$ operācijas.

1.1.3. Pielietojumi

Papildus tiešajiem pielietojumiem transporta tīklos šim uzdevumam ir arī daži neacīmredzami pielietojumi.

Stabilo laulību uzdevums

Ir dota zēnu kopa Z un meiteņu kopa M. Starp dažiem zēniem un meitenēm eksistē savstarpējas simpātijas, kuras mēs vienkāršoti modelēsim ar šķautni, ja starp divām virsotnēm eksistē šķautne, tad uzskatīsim, ka atbilstošie cilvēki var veidot stabilu laulību.

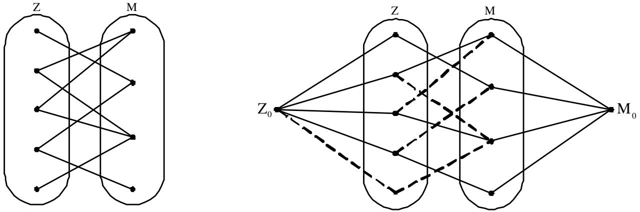
Uzdevums ir šāds: noteikt, kāds ir maksimālais stabilo laulību skaits, kas ir iespējams dotajam kopu pārim (M, Z) ar uzdotu savstarpējo simpātiju sarakstu.

Šo uzdevumu var atrisināt, izmantojot plūsmas maksimizācijas pieeju.

Grafi modificēsim šādā veidā:

- pievienosim vēl divas virsotnes M_0 un Z_0,
- savienosim virsotni M_0 ar visām kopas M virsotnēm,
- savienosim Z_0 ar visām kopas Z virsotnēm,
- uzskatīsim M_0 par avotu un Z_0 - par noteku,
- katru šķautni orientēsim virzienā no avota uz noteku un piešķirsim katrai orientētajai šķautnei caurlaides spēju 1.

Var redzēt, ka maksimālā plūsma šādā tīklā definē maksimālu pāru skaitu.



3.71. attēls. Stabilo laulību uzdevuma risināšana ar plūsmas maksimizācijas metodi

Apakškopu sistēmas pārstāvju izvēle

Ir dota kopa A un tās apakškopu kopa $\{A_i\}_{i\in I}$. Uzdevums ir atrast apakškopu sistēmas *dažādo pārstāvju kopu* - kopu $R = \{a_i\}_{i\in I}$, kas apmierina īpašības $a_i \in A_i$ un $|R| = |I|$.

Definēsim tīklu šādā veidā:

- virsotņu kopa ir $\{A_i\}_{i\in I} \bigcup A \bigcup \{x, y\}$, kur x un y ir divas jaunas virsotnes;

- savienosim katru kopas A virsotni ar x;
- savienosim katru kopas $\{A_i\}_{i\in I}$ virsotni ar y;
- savienosim katru elementu a ar visām virsotnēm A_i, kurām izpildās nosacījums $a \in A_i$.
- uzskatīsim x par avotu un y par noteku,
- orientēsim šķautnes virzienā no avota uz noteku un piekārtosim katrai šķautnei caurlaides spēju 1.

Var redzēt, ka apakškopu sistēmai eksistē pārstāvju kopa tad un tikai tad, ja maksimāla plūsma šādā tīklā ir vienāda ar apakškopu skaitu (elementu skaitu kopā $\{A_i\}_{i\in I}$ jeb $|I|$).

Sakarīguma skaitļa noteikšana

Izmantojot Mengera teorēmu, var piedāvāt algoritmu, kas atrod grafa sakarīguma skaitli, pielietojot plūsmas maksimizāciju: maksimālais virsotņu šķirtu ķēžu skaits, kas savieno divas virsotnes, ir vienāds ar maksimālas plūsmas lielumu grafam atbilstošajā tīklā, ja katrai neorientētai šķautnei atbilst divas orientētas šķautnes ar caurlaides spēju 1.

1.2. Maksimālu neatkarīgu kopu meklēšana

1.2.1. Problēma

Dots nosvērts (neorientēts) grafs. Meklēsim neatkarīgu virsotņu vai šķautņu kopu ar maksimālu svaru.

1.2.2. Algoritmi

Šo uzdevumu ir iespējams atrisināt, tikai apskatot visas iespējamās virsotņu vai šķautņu apakškopas.

Ja grafam ir n virsotnes, tad ir jāapskata visas 2^n apakškopas, un algoritma darba laiks ir vismaz $O(2^n)$. To ir iespējams izdarīt, piemēram, ar bektrekinga algoritma palīdzību - tiek pēctecīgi ģenerētas visas apakškopas un uzglabāta apakškopa ar maksimālo svaru starp visām jau ğenerētajām apakškopām.

Heiristisku maksimālas neatkarīgu virsotņu kopas meklēšanas algoritms:

1) izvēlēties grafā Γ virsotni t ar minimālu pakāpi un ievietot to kopā U (sākumā kopa U ir tukša),

2) ja $\Gamma - (t \cup N_\Gamma(t))$ ir tukšs grafs (ja tā virsotņu kopa ir tukša), tad apstāties, ja $\Gamma - (t \cup N_\Gamma(t))$ nav tukšs, tad pārveidot grafu Γ par $\Gamma - (t \cup N_\Gamma(t))$ (iznīcināt kopas $t \cup N_\Gamma(t)$ virsotnes) un iet uz soli 1).

1.3. Grafu krāsošana

1.3.1. Problēma

Ir dots grafs Γ. Uzdevums: nokrāsot grafa virsotnes, ja ir dots krāsu skaits χ. Citos terminos: sadalīt Γ virsotņu kopu χ neatkarīgās apakškopās.

1.3.2. Algoritms

Tā kā virsotņu krāsošana ir tas pats, kas virsotņu kopas sadalīšana neatkarīgu apakškopu apvienojumā, var piedāvāt šādu "rijīgu" krāsošanas algoritmu:

1) izvēlēties grafā Γ maksimālu neatkarīgu virsotņu kopu S, fiksēt kārtējo krāsu K;

2) nokrāsot kopas S virsotnes krāsā K;

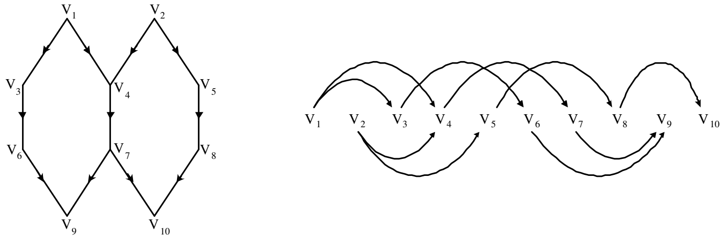
3) ja $\Gamma - S$ ir tukšs grafs, tad apstāties, ja $\Gamma - S$ nav tukšs, tad pārveidot grafu Γ par $\Gamma - S$ (izdzēst kopas S virsotnes) un iet uz soli 1).

1.4. "Topoloģiskā šķirošana"

1.4.1. Problēma

Ir dots aciklisks orientēts grafs (AOG).

Uzdevums ir sakārtot grafa virsotnes virknē tā, lai katras orientētas šķautnes beigu virsotne virknē būtu pa labi no sākuma virsotnes.



3.72. attēls. Topoloģiskās šķirošanas piemērs

Šo uzdevumu sauc par *AOG topoloģiskās šķirošanas* uzdevumu.

1.2. piemērs. Viens no tā pielietojumiem ir šāds: pieņemsim, ka ir liels projekts, kas sastāv no vairākiem darbiem, darbi var būt atkarīgi viens no otra to izpildes nozīmē (piemēram, darbs B nevar būt iesākts, pirms nav pabeigts darbs A), modelēsim projektu ar orientētu grafu, kura virsotnes ir darbi un šķautnes savieno atkarīgus darbus to izpildes kārtībā. Topoloģiskās šķirošanas uzdevuma atrisinājumu šajā gadījumā var interpretēt kā darbu virkni, kas apmierina atkarības nosacījumus.

1.4.2. Algoritms

Topoloģiskās šķirošanas problēma var tikt atrisināts, izmantojot šādu algoritmu:

1) dotajā grafā atradīsim visu avotu kopu (pārskatot grafa matricu vai izmantojot dziļummeklēšanu);
2) pievienosim kopu S konstruējamās virknes labajā galā jebkurā kārtībā;
3) pārveidosim Γ par $\Gamma - S$;

4) ja $V(\Gamma) = \varnothing$, tad apstājamies, ja nē, tad ejam uz soli 1).

1.5. Eilera cikla problēma

1.5.1. Problēma

Dots neorientēts Eilera grafs - katras virsotnes pakāpe ir pāra skaitlis. Uzdevums: atrast šajā grafā Eilera ciklu.

1.5.2. Algoritms

Fleury algoritms

Īss apraksts:

1. sākam ar patvaļīgu šķautni,
2. izvēlamies jebkuru šķautni, kas ir incidenta iepriekšējai šķautnei un nav tilts modificētajā grafā, ja ir izvēle, ja nav izvēles, tad ņemam tiltu,
3. modificējam grafu - izdzēšam izvēlēto šķautni,

4. ja modificētais grafs ir tukšs, tad apstājamies, ja nav tukšs, tad ejam uz soli 2).

Dots neorientēts Eilera grafs Γ.

1. Sākam ar patvaļīgu virsotni v_0. Definējam mainīgo grafu $\Gamma_t = \Gamma$, mainīgo virsotni $v_t = v_0$.
2. Izvēlamies šķautni $e = (v_t, v)$ grafā Γ_t, kuras viens gals ir v_t, tādu, ka
   (a) e nav tilts grafā Γ_t,
   (b) e ir tilts grafā Γ_t, un visas pārējās šķautnes, kas ir incidentas ar v_t, ir tilti.

   Modificējam mainīgo grafu un virsotni: $\Gamma_t := \Gamma_t - e$, $v_t := v$.
3. Ja Γ_t ir bezšķautņu grafs, tad apstājamies, ja nē, ejam uz soli 2).

1.6. Hamiltona cikla un "ceļojošā tirgoņa problēma"

1.6.1. Problēma

Ir dots nosvērts grafs. Uzdevums ir atrast šajā grafā Hamiltona ciklu, kura šķautņu svaru summa ir minimāla.

Speciālgadījumā, ja visu šķautņu svari ir vienādi, iegūsim uzdevumu par Hamiltona cikla atrašanu.

1.1. piezīme. Šādu uzdevumu sauksim par "ceļojošā tirgoņa problēmu", jo tas ir cēlies no uzdevuma par tirgoni, kam ir jāapmeklē visas pilsētas valstī un jāatgriežas sākotnējā pilsētā, ceļojot pēc iespējas īsāku maršrutu.

1.6.2. Algoritmi

Vispārīgā gadījumā šim uzdevumam nav atrasts algoritms, kas patērē būtiski mazāk laika nekā *izsmeļošās pārlases* algoritms - visu virsotņu permutāciju apskatīšana.

Tātad, ja grafam ir n virsotnes, tad algoritma darbības laiks ir vismaz $O(n!)$.

Speciālu grafu klasēm ˇšī uzdevuma risināšanai ir izstrādātas efektīvas heiristiskās metodes.

2. 14.mājasdarbs

2.1. Obligātie uzdevumi

14.1 Ģenerēt nosvērtu orientētu avotu ar 20 virsotnēm un realizēt maksimālās plūsmas meklēšanas algoritmu.

14.2 Ģenerēt neorientētu grafu ar 20 virsotnēm un realizēt šādus algoritmus izmantojot pilno pārlasi:

(a) maksimālas neatkarīgas virsotņu kopas meklēšanas algoritmu,
(b) virsotņu krāsošanas algoritmu ar minimālu krāsu skaitu,
(c) Hamiltona cikla meklēšanas algoritmu.