ALGORITMICA GRAFURILOR

C. Croitoru

2006-2007

Motivaţie: Grafurile sunt considerate cele mai utilizate "structuri" abstracte din Informaticā .

Scop: Familiarizarea studenţilor cu principalele noţiuni şi rezultate din Teoria Grafurilor şi aplicarea acestora în proiectarea unor algoritmi eficienţi pentru diversele probleme de optimizare pe grafuri.

Conţinut: Probleme, Algoritmi, Complexitate; Vocabular al Teoriei Grafurilor; Probleme de drum (parcurgeri, drumuri minime, conexiune); Arbori parţiali de cost minim (union-find, complexitate amortizatā); Cuplaje; Fluxuri; Reduceri polinomiale pentru probleme de decizie pe grafuri; Abordāri ale problemelor NP-hard pe grafuri; Grafuri planare.

Bibliografie:

- Manualul cursului :
CROITORU C., *Tehnici de bază în optimizarea combinatorie*, Editura Universităţii "Al. I. Cuza" Iaşi, Iaşi,1992.(Slide-urile sunt făcute utilizând acest text).

- Se poate consulta şi capitolul 3 din
CROITORU C., *Introducere in proiectarea algoritmilor paraleli,* Editura Matrix Rom, Bucureşti, 2002.

- TOMESCU I., *Probleme de combinatorica şi teoria grafurilor*, Editura did şi ped, Bucureşti, 1981.

- DIESTEL R., Graph Theory, Electronic Edition

-T.H. CORMEN, C.E. Leiserson, R.L. Rivest, C. Stein: *Introduction to Algorithms* , The MIT Press 2001, 2nd edition

0. Probleme,Algoritmi,Complexitate

Vom considera o **problemă** (computațională) ca fiind o aplicație

$$\mathcal{P} : I \to O,$$

(I mulțimea intrărilor problemei, mulțimea instanțelor problemei; O mulțimea ieșirilor, răspunsurilor, soluțiilor) care pentru fiecare intrare $i \in I$ oferă o ieșire $\mathcal{P}(i) \in O$.

Dacă $O = \{da, nu\}$ atunci $\mathcal{P}$ se va numi problemă de decizie, $\mathcal{P}(i) \in \{da, nu\}$ va fi răspunsul (la întrebarea pusă de $\mathcal{P}$), iar forma uzuală de prezentare a problemei va fi:

$\mathcal{P}$ **Intrare:** $i \in I$.
Intrebare: descriere ... ?

Exemplu:

Compus **Intrare:** $n \in \mathbf{N}$ $n \geq 2$.
Intrebare: Există $p, q \in \mathbf{N}$ $p, q \geq 2$ și $n = pq$?

Un alt tip de probleme care apar frecvent sunt cele de căutare :

mulţimea O conţine pentru fiecare $i \in I$ măcar o soluţie acceptabilă în raport cu un anumit criteriu precizat, iar problema cere găsirea unei astfel de soluţii.

O clasă specială de astfel de probleme este cea a problemelor de optimizare, care sunt de maximizare sau minimizare. Exemple:

Drum **Intrare:** G un graf, a, b două vârfuri ale lui G.
Ieşire: P un drum în G de la a la b (dacă $\exists$).

DrMin **Intrare:** G graf, a, b vârfuri în G,
o funcţie de lungime a muchiilor lui G.
Ieşire: P^* un drum în G de la a la b cu suma lungimilor muchiilor minimă, printre toate drumurile de la a la b în G.

MaxCut **Intrare:** G graf,
o funcţie de lungime a muchiilor lui G.
Ieşire: O bipartiţie (S, T) a mulţimii vârfurilor grafului G cu suma lungimilor muchiilor dintre cele două clase maximă.

Oricărei probleme de optimizare i se poate asocia o problemă de decizie (care ne va da informaţii asupra dificultăţii ei computaţionale). Pentru cele două probleme de optimizare de mai sus, avem:

DrMin-D

Intrare: G graf, a, b vârfuri în G, $k \in \mathbf{N}$,
o funcţie de lungime a muchiilor lui G.
Întrebare: Există P drum în G de la a la b
cu suma lungimilor muchiilor $\leq k$?

MaxCut-D

Intrare: G graf, $k \in \mathbf{N}$,
o funcţie de lungime a muchiilor lui G.
Întrebare: Există (S, T) bipartiţie a mulţimii
vârfurilor lui G cu suma lungimilor
muchiilor dintre cele două clase $\geq k$?

Vom considera în continuare doar probleme de decizie.

Pentru a fi rezolvate cu ajutorul calculatorului problemele sunt codificate.

Vom considera Σ o mulţime finită fixată numită alfabet, Σ^* mulţimea tuturor cuvintelor peste Σ.

Obiectele care apar în descrierea unei probleme (numere raționale, grafuri, funcții, matrici etc.) vor fi reprezentate cu ajutorul unor cuvinte $w \in \Sigma^*$. Lungimea cuvântului w se va nota cu $|w|$. Orice mulțime de cuvinte peste Σ, deci o submulțime a lui Σ^* se numește limbaj (peste Σ).

Problemele de decizie au forma *Dat un anumit obiect, are el o anumită proprietate?* Cum obiectele le reprezentăm cu ajutorul cuvintelor înseamnă că problema se reduce la întrebarea *Dat un cuvânt, are el o anumită proprietate?*

Vom considera problemă de decizie o funcție

$$P : \Sigma^* \to \{da, nu\}.$$

Limbajul asociat problemei P este

$$P^{-1}(da) = \{w | w \in \Sigma^* \text{ și } P(w) = da\}.$$

Vom considera că un algoritm este o funcție *

$$A : \Sigma^* \to \{da, nu, nedecidabil\}.$$

Limbajul $L \subseteq \Sigma^*$ este acceptat de algoritmul A dacă $L = \{w | w \in \Sigma^* \text{ și } A(w) = da\}$.

Limbajul $L \subseteq \Sigma^*$ este decis de algoritmul A dacă $\forall w \in L : A(w) = da$ și $\forall w \notin L : A(w) = nu$.

Limbajul $L \subseteq \Sigma^*$ este acceptat de algoritmul A în timp polinomial dacă L este acceptat de A și $\exists k \in \mathbf{N}$ astfel încât pentru orice $w \in L$ algoritmul A evaluează $A(w) = da$ în timpul $\mathcal{O}(|w|^k)$.

Limbajul $L \subseteq \Sigma^*$ este decis de algoritmul A în timp polinomial dacă $\exists k \in \mathbf{N}$ astfel încât pentru orice $w \in \Sigma^*$ algoritmul A evaluează $A(w) = da$, dacă $w \in L$ și $A(w) = nu$, dacă $w \notin L$, în timpul $\mathcal{O}(|w|^k)$.

*Preferăm un mod informal de prezentare a noțiunilor de algoritm și timp de execuție

Clasa de complexitate $\mathbf{P}$:

$$\mathbf{P} = \{L \subset \Sigma^* | \exists A \text{ alg. a.î. } L \text{ e decis de } A \text{ în timp polinomial}\}.$$

Dacă P e o problemă de decizie, atunci ea este rezolvabilă în timp polinomial dacă limbajul $L = P^{-1}(da)$ satisface $L \in \mathbf{P}$. Se notează aceasta (cam abuziv) $P \in \mathbf{P}$.

De exemplu, problema $DrMIN\text{-}D$ este rezolvabilă în timp polinomial dacă funcţia de lungime (specificată în intrarea ei) este cu valori nenegative. Dacă se permit şi valori negative, atunci nu se cunoaşte nici o demonstraţie a apartenenţei $DrMin\text{-}D \in \mathbf{P}$ (şi nici nu se crede c-ar exista una). Apartenenţa
$Compus \in \mathbf{P}$ s-a demonstrat abia în anul 2002.

Observaţii. 1. Dacă notăm

$$\mathbf{P}' = \{L \subset \Sigma^* | \exists A \text{ alg. a.î. } L \text{ e acceptat de } A \text{ în timp polinomial}\},$$

se observă imediat că $\mathbf{P} \subseteq \mathbf{P}'$ şi nu e dificil să se arate şi incluziunea inversă, deci $\mathbf{P} = \mathbf{P}'$.
2. Se verifică uşor că dacă $P \in \mathbf{P}$ atunci şi $\Sigma^* \setminus P \in \mathbf{P}$.

Verificare în timp polinomial.

Un algoritm de verificare este o funcție

$$A : \Sigma^* \times \Sigma^* \to \{da, nu, nedecidabil\}.$$

Pentru $A(x, y)$, x este intrarea iar y este certificatul.

Limbajul $L \subseteq \Sigma^*$ este verificat de algoritmul de verificare A dacă
$L = \{w | w \in \Sigma^* \text{ și } \exists y \in \Sigma^* \text{ a. î. } A(w, y) = da\}$.

Clasa de complexitate $\mathbf{NP}$:

$$\mathbf{NP} = \{L \subseteq \Sigma^* \mid \exists A \text{ algoritm de verificare}$$
$$\text{cu timp de lucru polinomial a.î.}$$
$$L = \{x \in \Sigma^* \mid \exists y \in \Sigma^*, \ \exists k \in \mathbf{N} \text{ a. î.}$$
$$|y| = \mathcal{O}(|x|^k) \text{ și } A(x, y) = da\}$$
$$\}.$$

Dacă P e o problemă de decizie, atunci ea este problemă (din) $\mathbf{NP}$ dacă limbajul $L = P^{-1}(da)$ satisface $L \in \mathbf{NP}$.

Observaţie. $\mathbf{NP}$ este mnemonic pentru Nedeterminist Polinomial şi nu pentru NePolinomial.

Un argument că nu e bine să asimilăm $\mathbf{NP}$ cu nepolinomial este şi faptul că $\mathbf{P} \subseteq \mathbf{NP}$.
Justificarea este imediată: Dacă $L \in \mathbf{P}$, atunci există $A : \Sigma^* \to \{da, nu, nedecidabil\}$ algoritm care decide L în timp polinomial. Considerăm $A' : \Sigma^* \times \Sigma^* \to \{da, nu, nedecidabil\}$, satisfăcând $A'(x, x) = A(x)$ pentru orice $x \in \Sigma^*$. Se vede uşor că L este verificat de A' în timp polinomial.

Din definiţiile de mai sus ar fi fost mai normal să notăm $\mathbf{VP}$ (verificabil polinomial). Sintagma Nedeterminist Polinomial se justifică dacă am considera algoritmi nedeterminişti în care controlul execuţiei este astfel încât după fiecare pas este posibil să se execute unul oarecare dintre paşii specificaţi într-o mulţime finită de paşi succesori. Un astfel de algoritm acceptă un cuvânt de intrare dacă este posibilă o execuţie care să conducă la rezultatul da. Se poate arăta că aceasta definiţie a acceptării nedeterministe este echivalentă cu cea de verificare dată mai sus, în care *certificatul* este utilizat pentru efectuarea alegerilor corecte ale paşilor următori ai unei execuţii a algoritmului nedeterminist.

NP notează clasa problemelor de decizie pentru care răspunsurile afirmative au *certificate* care pot fi folosite pentru a demonstra succint (în timp polinomial) corectitudinea lor.

Intuitiv, **NP** este clasa tuturor problemelor de decizie pentru care se poate verifica un răspuns pozitiv (da) rapid dacă ni se dă o soluție.

De exemplu, pentru problema $MaxCut\text{-}D$ un răspuns afirmativ are drept certificat o partiție (S^*, T^*) a mulțimii vârfurilor grafului (iar proprietatea că suma lungimilor muchiilor cu o extremitate în S^* și cealaltă în T^* nu depășește pragul k se face în timp polinomial). Deci, $MaxCut\text{-}D \in \mathbf{NP}$.

Dacă am considera următoarea problemă de decizie:

UnDrum **Intrare:** G un graf, a, b două vârfuri ale lui G.
Întrebare: $\exists$ un unic drum de la a la b în G?

Cum s-ar putea justifica un răspuns da la o problemă $UnDrum$? Dacă prezentăm un drum anume drept certificat, el nu ne asigură că nu mai există și altele. O demonstrație succintă (graful e mare) pare a nu exista. Deci nu știm dacă $UnDrum \in \mathbf{NP}$.

Clasa de complexitate $co\text{-}\mathbf{NP}$:

$$co\text{-}\mathbf{NP} = \{L \subseteq \Sigma^* \mid \Sigma^* \setminus L \in \mathbf{NP}\}.$$

O problemă de decizie $P \in co\text{-}\mathbf{NP}$ dacă
$L = P^{-1}(da) \in co\text{-}\mathbf{NP}$ (echivalent, $L = P^{-1}(nu) \in \mathbf{NP}$).

Exemplu de problemă din $co\text{-}\mathbf{NP}$:

NeHam **Intrare:** G un graf.
Întrebare: Este adevărat că în G nu există un circuit care să treacă exact o dată prin fiecare vârf al său?

Observație. $\mathbf{P} \subseteq \mathbf{NP} \cap co\text{-}\mathbf{NP}$.

Reduceri polinomiale.

Fie $L_1, L_2 \subseteq \Sigma^*$. Spunem că L_1 se reduce polinomial la L_2, și notăm aceasta prin $L_1 \propto L_2$, dacă există $f : \Sigma^* \to \Sigma^*$ o funcție polinomial calculabilă astfel încât $\forall w \in \Sigma^*$: $w \in L_1$ dacă și numai dacă $f(w) \in L_2$.

f se numește funcție de reducere și algoritmul polinomial F care calculează f, algoritm de reducere polinomială.

Observații. 1. Dacă $L \in \mathbf{P}$ și $L' \propto L$, atunci $L' \in \mathbf{P}$.
Fie A un algoritm polinomial care decide L și F un algoritm de reducere polinomială a lui L' la L. Atunci, $A' = A \circ F$ este un algoritm polinomial care decide L'. ($\forall x \in \Sigma*$, $A'(x) = da \Leftrightarrow A(F(x)) = da \Leftrightarrow F(x) \in L \Leftrightarrow x \in L'$; A' e polinomial deoarece mulțimea polinoamelor e închisă la operația de compunere).
2.Relația $\propto$ este tranzitivă: $L_1 \propto L_2, L_2 \propto L_3 \Rightarrow L_1 \propto L_3$.

Limbajul $L \subseteq \Sigma^*$ este $\mathbf{NP}$-dificil (engl. $\mathbf{NP}$-hard) dacă $\forall L' \in \mathbf{NP}$ are loc $L' \propto L$.

Limbajul $L \subseteq \Sigma^*$ este $\mathbf{NP}$-complet dacă
$L \in \mathbf{NP}$ și L este $\mathbf{NP}$-dificil.

Terminologia se transferă pentru probleme de decizie:

Spunem că problema de decizie P_1 se reduce polinomial la problema de decizie P_2, și notăm $P_1 \propto P_2$, dacă $L_1 = P_1^{-1}(da) \propto L_2 = P_2^{-1}(da)$.

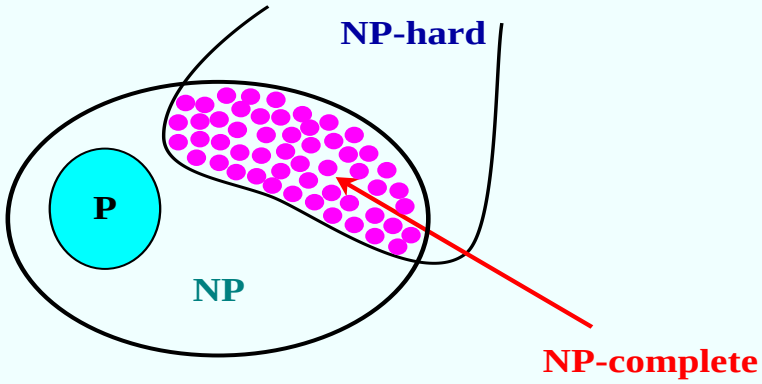
Problema de decizie P este **NP**-dificilă dacă
$\forall P' \in \mathbf{NP}$ are loc $P' \propto P$.

Problema de decizie P este **NP**-completă dacă
$P \in \mathbf{NP}$ și P este **NP**-dificilă.

Folosind observația 1 și faptul că $\mathbf{P} \subseteq \mathbf{NP}$ rezultă următoarea

Teoremă *Dacă P o problemă oarecare **NP**-completă satisface $P \in \mathbf{P}$ atunci $\mathbf{P} = \mathbf{NP}$.*

Rezultă că problemele **NP**-complete formează submulțimea lui **NP** care conține cele mai dificile probleme. Dacă găsim un algoritm polinomial pentru una dintre ele, putem construi un algoritm polinomial pentru oricare altă problemă din **NP**. Din păcate, deși nu există o demonstrație, oamenii cred că, de fapt, aceste probleme nu admit algoritmi polinomiali de rezolvare. Diagrama următoare sugerează relațiile care se crede că există între aceste clase de complexitate:



Se obișnuiește să se spună că o problemă de optimizare este $\mathbf{NP}$-dificilă, dacă problema de decizie asociată este așa. Pentru ca să ne aliniem la definițiile precedente, vom spune că o problemă oarecare P (nu neaparat de decizie) este $\mathbf{NP}$-dificilă dacă existența unui algoritm polinomial pentru P implică $\mathbf{P} = \mathbf{NP}$.

Comentariu metaforic: Saying that a problem is NP-hard is like saying 'If I own a dog, then it can speak fluent English.' You probably don't know whether or not I own a dog, but you're probably pretty sure that I don't own a talking dog. Nobody has a mathematical proof that dogs can't speak English. Nevertheless, no sane person would believe me if I said I owned a dog that spoke fluent English. So the statement 'If I own a dog, then it can speak fluent English' has a natural corollary: No one in their right mind should believe that I own a dog! Likewise, if a problem is NP-hard, no one in their right mind should believe it can be solved in polynomial time. [Jeff Erickson, Univ. of. Illinois]

Singura clasă de complexitate pentru care nu am sugerat un exemplu este cea a problemelor $\mathbf{NP} - complete$. Primul om care a demonstrat existența unei astfel de probleme este **Cook**, care în 1971 a arătat că $SAT \in \mathbf{NP}$, unde

SAT

Instanță: $U = \{u_1, \ldots, u_n\}$ o mulțime finită de variabile booleene.
$C = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ o formulă în formă conjunctivă peste U:
$C_i = v_{i_1} \vee v_{i_2} \vee \ldots \vee v_{i_{k_i}} \quad \forall i = \overline{1,m}$, unde
$\forall i_j \quad \exists \alpha \in \{1, \ldots, n\}$ astfel încât
$v_{i_j} = u_\alpha$ sau $v_{i_j} = \overline{u}_\alpha$.

Intrebare: Există o atribuire $t : U \to \{A, F\}$ a. î. $t(C) = A$?

Evaluarea lui $t(C)$ se bazează pe formulele uzuale din calculul boolean:

$A \wedge A = A$, $F \wedge A = F \wedge F = A \wedge F = F$,

$F \vee F = F$, $F \vee A = A \vee F = A \vee A = A$,

$\overline{F} = A$, $\overline{A} = F$, $(F) = F$,$(A) = A$,

și e clar că se poate face în timp polinomial.

Apartenența lui SAT la $\mathbf{NP}$ e clară, un certificat pentru răspunsul da este o atribuire t_0 care poate fi verificată în timp polinomial.

O demonstrație că orice limbaj din $\mathbf{NP}$ se reduce polinomial la SAT necesită o abordare formală noțiunilor de algoritm și timp de execuție (pe care le-am considerat aici la nivel intuitiv) și o omitem.

$3SAT$ este restricția lui SAT în care fiecare clauză C_i are exact trei literali ($k_i = 3$), un literal v_{i_j} fiind, așa cum este descris mai sus, o variabilă sau negația ei.

Se poate arăta ușor că $SAT \propto 3SAT$ și deci se obține că $3SAT$ este $\mathbf{NP}$-completă (apartenența la $\mathbf{NP}$ e clară fiind o restricție a lui SAT care e din $\mathbf{NP}$, iar tranzitivitatea relației $\propto$ împreună cu teorema lui Cook termină demonstrația).

Schița de demonstrație de mai sus este uzuală în demonstrațiile de $\mathbf{NP}$-completitudine și o explicităm:

Pentru a arăta că o problemă de decizie P este $\mathbf{NP}$-completă se procedează astfel:

1. Se arată că $L = P^{-}1(da)$ satisface $L \in \mathbf{NP}$.
2. Se selectează un limbaj L' despre care știm că este $\mathbf{NP}$-complet.
3. Se încearcă construirea unui algoritm de reducere F de la L' la L.
4. Se demonstrează că F e algoritm de reducere.
5. Se arată că F este algoritm polinomial.

Pentru pasul 2 se poate consulta

http://www.nada.kth.se/~viggo/problemlist/

I. Vocabular al Teoriei grafurilor

1. Definiţia unui graf

Un **graf** este o pereche $G = (V(G), E(G))$, unde

- $V(G)$ este o mulţime finită nevidă, iar

- $E(G)$ este o submulţime a mulţimii $\mathcal{P}_2(V(G))$ a părţilor cu două elemente ale lui $V(G)$.

$V(G)$ se numeşte **mulţimea vîrfurilor** grafului G şi numărul elementelor sale

$\left|V(G)\right|$, este **ordinul** grafului G;

$E(G)$ este **mulţimea muchiilor** grafului G şi numărul său de elemente, $\left|E(G)\right|$, este **dimensiunea** grafului G.

Atunci cînd nu există posibilitatea confuziilor, vom nota simplu, $G = (V, E)$.

Dacă $e = \{u, v\} \in E(G)$ este o muchie a graf-ului G vom nota $e = uv$ (pentru simplificarea scrierii) și vom spune că:

- muchia e **unește** vîrfurile u și v;
- vîrfurile u și v sunt **adiacente** în G;
- muchia e este **incidentă** cu vîrfurile u și v;
- vîrfurile u și v sunt **vecine** în G;
- vîrf. u și v sunt **extremitățile** muchiei e.

Dacă $v \in V(G)$, atunci mulțimea

$$N_G(v) = \{w | w \in V(G), vw \in E(G)\}$$

se numește **vecinătatea** vîrfului v în G.
Se mai notează $N_G(v) = \Gamma_G(v)$.

Remarcăm faptul că graful G poate fi definit (în mod echivalent) ca o pereche $\Big(V(G), \Gamma_G\Big)$ unde

$$\Gamma_G \colon V(G) \to \mathcal{P}\Big(V(G)\Big)$$

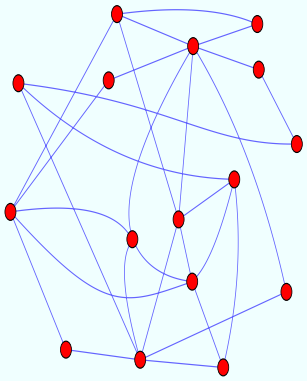
asociază fiecărui vîrf vecinătatea sa.

Două muchii e și e' care au o extremitate comună se numesc **adiacente**.

Intuitiv, un graf $G = (V(G), E(G))$ poate fi reprezentat (după cum sugerează și numele său) cu ajutorul unei figuri plane formată dintr-o mulțime de **mici forme geometrice** aflată în corespondență cu mulțimea de vîrfuri $V(G)$, două forme fiind unite printr-o **curbă simplă** dacă și numai dacă, perechea de vîrfuri corespunzătoare lor este o muchie a grafului G.

Corespondența dintre vârfurile grafului și figurile geometrice considerate este vizualizată

uneori cu etichete atașate vârfurilor. De asemenea, muchiile pot fi etichetate. În plus, sunt utilizate diferite atribute grafice pentru expresivitatea desenului, diagramei.
De exemplu,



și

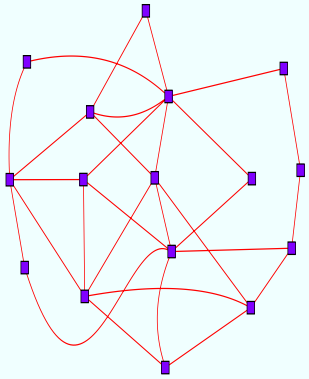
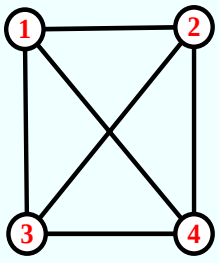
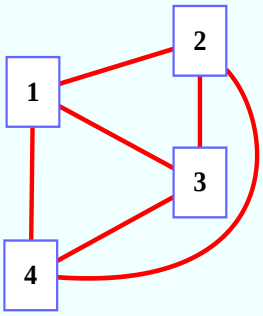
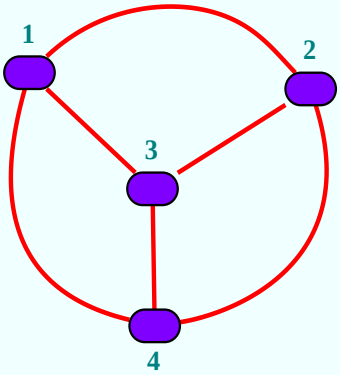


figura reprezintă același graf, deși lipsa etichetelor face dificilă realizarea acestui fapt.
Următoarele trei reprezentări ale grafului $G = (\{1, 2, 3, 4\}, \{12, 13, 14, 23, 24, 34\})$ sunt mult mai clare:



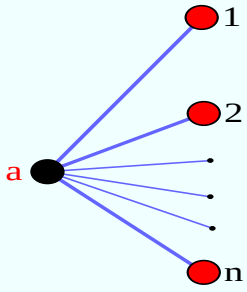




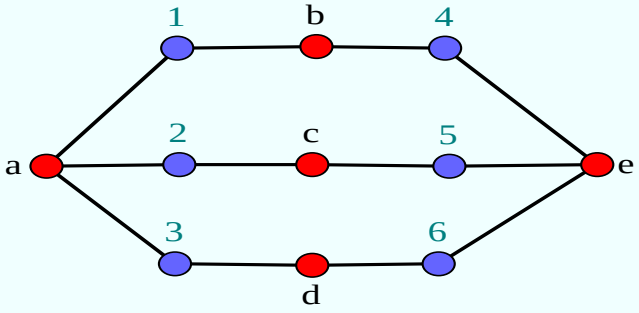
O **mulțime independentă de vîrfuri** (sau **mulțime stabilă**) în G este o mulțime $S \subseteq V(G)$ de vîrfuri cu proprietatea că

$$\mathcal{P}_2(S) \cap E(G) = \emptyset$$

(adică o mulțime de vîrfuri neadiacente două cîte două).
Cardinalul maxim al unei mulțimi stabile se numește **numărul de stabilitate** sau **numărul de independență** al grafului G și se notează cu $\alpha(G)$.
De exemplu, în graful G de mai jos mulțimea $\{a\}$ este mulțime stabilă (maximală în raport cu incluziunea), dar numărul de stabilitate este n, mulțimea $\{1, \ldots, n\}$ fiind stabilă de cardinal maxim ($n \geq 1$). În graful H s-au evidențiat două mulțimi stabile care partiționează mulțimea vârfurilor, iar $\alpha(H) = 6$.



G



H

Problema următoare este naturală, ușor de formulat și apare deseori în diferite aplicații:

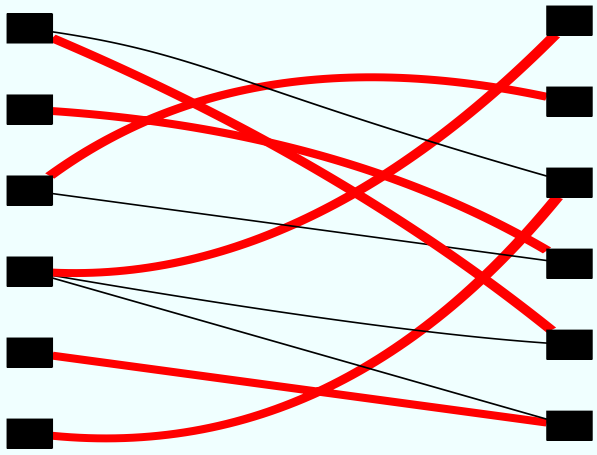
P1 Intrare: G un graf.
Ieșire: $\alpha(G)$ și un "martor":
S m.stabilă în G, cu $|S| = \alpha(G)$.

Deși foarte simplă (de formulat, la o primă vedere), problema este NP-dificilă. Problema de decizie corespunzătoare,

SM Intrare: G un graf, $k \in \mathbf{N}$.
Intrebare: Există S m.stabilă în G, cu $|S| \geq k$?

este NP-completă (Karp, 1972). Probabil că o cauză a dificultății acestei probleme este faptul că două mulțimi stabile maximale (în raport cu incluziunea) pot avea raportul cardinalelor oricât de mare (vezi graful G din figura de la pagina precedentă).

O **mulţime independentă de muchii** sau **cuplaj** în graful G este o mulţime de muchii neadiacente două cîte două . Cardinalul maxim al unei mulţimi independente de muchii în G se numeşte **numărul de muchie-independenţă** al grafului G şi se notează $\nu(G)$.
Exemplu, pentru graful G:



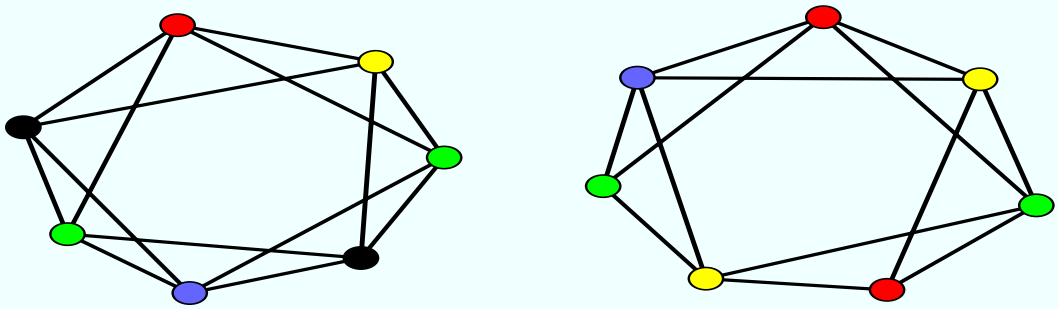
numărul de muchie independenţă, $\nu(G)$ este 6, un cuplaj cu acest număr de muchii fiind pus în evidenţă. Problema

P2 Intrare: G un graf.
Ieşire: $\nu(G)$ şi un "martor":
M cuplaj în G, cu $|M| = \nu(G)$.

este foarte asemănătoare cu P1 (este de fapt, o restricţie a lui P1 pentru intrări care sunt line-grafuri) şi totuşi s-a arătat că este în **P**(Edmons, 1965).
Diferenţa de complexitate provine, probabil, din faptul că raportul dintre cardinalele a două cuplaje maximale în raport cu incluziunea nu poate depăşi 2.

Dacă $G = (V(G), E(G))$este un graf şi $p \in \mathbf{N}^*$, se numeşte $p-$**colorare a (vîrfurilor)** lui G o aplicaţie $c : V(G) \to \{1, \ldots, p\}$ cu proprietatea că $c^{-1}(i)$ este o mulţime stabilă în G, $\forall i \in \{1, \ldots, p\}$ (remarcăm că, din definiţia mulţimilor stabile, $\emptyset$ este o mulţime stabilă).
Numărul cromatic al grafului G, notat $\chi(G)$, este cea mai mică valoare a lui $p \in \mathbf{N}^*$ pentru care G admite o p-colorare.

Exemplu: În graful G desenat mai jos, sunt evidenţiate 2 colorări una cu 5 culori şi una cu 4 culori. Se poate argumenta că $\chi(G) = 4$.



5-colorare

4-colorare

rosu= culoarea 1
galben= culoarea 2
verde=culoarea 3
albastru=culoarea 4
negru=culoarea 5

Problema următoare apare în diverse situaţii practice (de exemplu în problemele de orar, sau în problemele de acoperire din wireless networks):

P3 Intrare: G un graf.
Ieşire: $\chi(G)$ şi un "martor":
o $\chi(G)$-colorare a lui G.

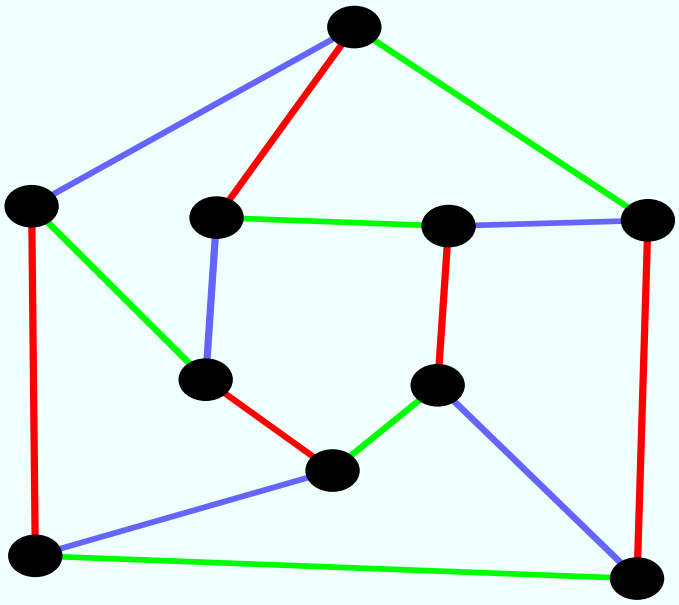
Din problema P1 știm că mulțimile stabile ale grafului sunt greu de "stăpânit", și cum problema P3 cere de fapt să partiționăm mulțimea de vârfuri a grafului într-un număr cât mai mic de mulțimi stabile, este normal ca și această problemă să fie NP-dificilă. Problema de decizie

COL	Intrare:	G un graf, $k \in \mathbf{N}$.
	Intrebare:	Admite G o k-colorare?

este NP-completă chiar dacă o restricționăm la cazul particular $k = 3$. Există însă restricții ale problemei pentru care avem apartenență la **P** (de exemplu, dacă G este un graf perfect).

O $p-$**colorare a muchiilor** lui G este o aplicație $c : E(G) \to \{1, \ldots, p\}$ cu proprietatea că $c^{-1}(i)$ este un cuplaj al lui G, $\forall i \in \{1, \ldots, p\}$.
Indicele cromatic al grafului G, notat $\chi'(G)$, este cea mai mică valoare a lui $p \in \mathbf{N}^*$ pentru care G admite o p-colorare a muchiilor.

Exemplu: In graful de mai jos



este evidenţiată o 3-colorare a muchiilor; este clar că este şi optimă, întrucât muchiile incidente în acelaşi vârf trebuie să aibă culori distincte. Problema

P4 Intrare: G un graf.
Ieşire: $\chi'(G)$ şi un "martor":
o $\chi'(G)$-colorare a muchiilor lui G.

s-a dovedit a fi NP-dificilă, deși era de așteptat ca (lucrând cu cuplaje, iar problema P2 fiind din **P**) ca ea să fie rezolvabilă polinomial. Diferența față de P3 (P4 este de fapt o restricție a lui P3 pe clasa line-grafurilor) este că în timp ce P3 s-a demonstrat că nu poate fi ușor aproximabilă (în timp polinomial), problema P4 poate fi rezolvată "aproximativ" cu o eroare de o unitate (deci dacă se greșește, atunci colorarea obținută are cel mult o culoare în plus față de cea optimă).

Două grafuri, $G = (V(G), E(G))$ și $H = (V(H), E(H))$ se numesc **izomorfe**, și notăm aceasta prin $G \cong H$, dacă există o bijecție

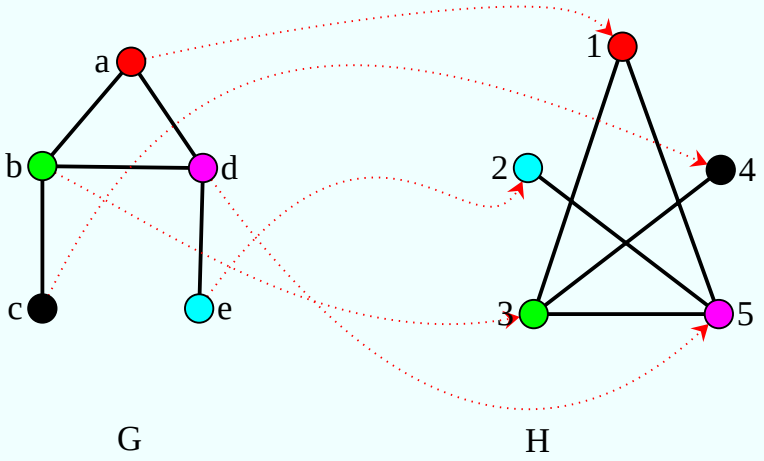
$$\varphi \colon V(G) \to V(H)$$

cu proprietatea că aplicația

$$\psi \colon E(G) \to E(H),$$

definită pentru orice $uv \in E(G)$ prin $\psi(uv) = \varphi(u)\varphi(v)$ este o bijecție.

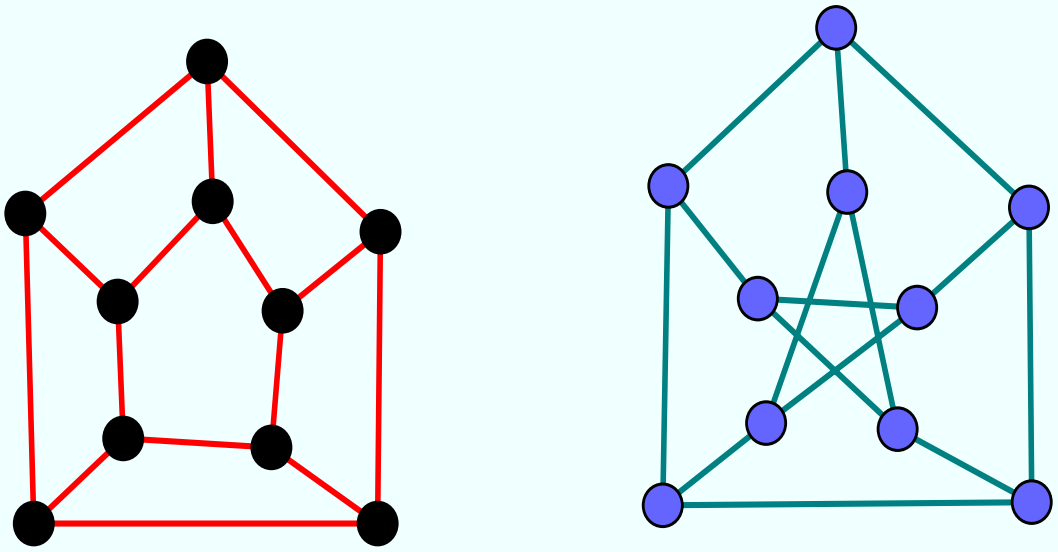
(deci, două grafuri sînt izomorfe dacă există o bijecţie între mulţimile lor de vîrfuri care induce o bijecţie între mulţimile lor de muchii).
Grafurile următoare sunt izomorfe, aşa cum se sugerează în figură



Problema următoare este utilă în multe probleme de modelare discretă (de exemplu în chimie)

ISO Intrare: G, H grafuri.
Intrebare: $G \cong H$?

Nu s-a demonstrat dacă această problemă este sau nu NP-completă (aparteneța la NP este clară). Se pare că face parte dintr-o clasă de probleme aflată între **P** și **NP**). Exemplul următor aratā dificultatea problemei (cele două grafuri au același ordin, aceeași dimensiune, vârfurile au același număr de muchii incidente, și totuși ele nu-s izomorfe: in primul apar circuite de lungime 4, iar în al doilea nu !)



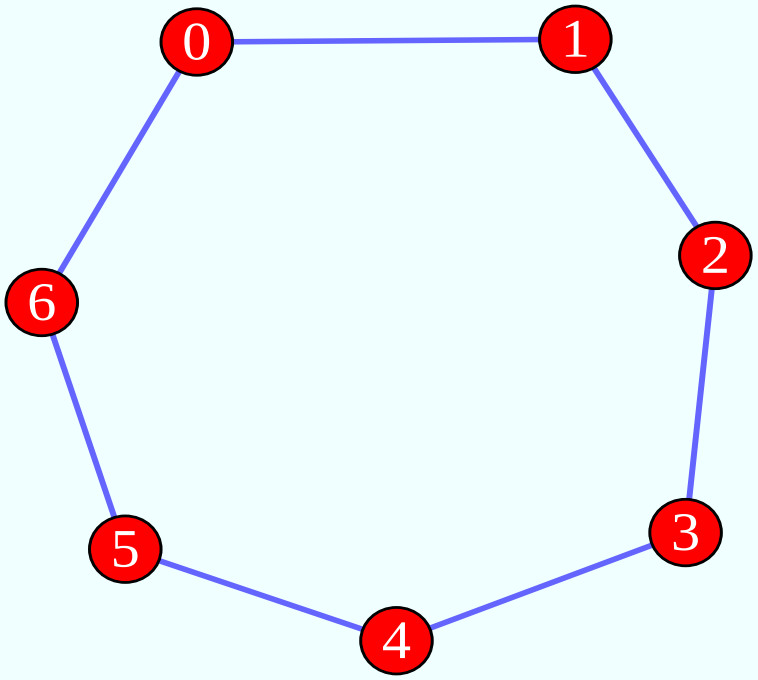
G H

Dacă $G = (V(G), E(G))$este un graf, un **automorfism** al lui G este o permutare a lui $V(G)$

(φ: $V(G) \to V(G)$, φ bijectivă) cu proprietatea că induce o permutare a lui $E(G)$
(ψ: $E(G) \to E(G)$, $\psi(uv) = \varphi(u)\varphi(v)$, $\forall uv \in E(G)$, este bijectivă).
Mulţimea automorfismelor grafului G formează, în raport cu operaţia de compunere a aplicaţiilor, un grup numit **grupul automorfismelor grafului** G, notat $Aut(G)$.
$Aut(G)$ este **tranzitiv** dacă

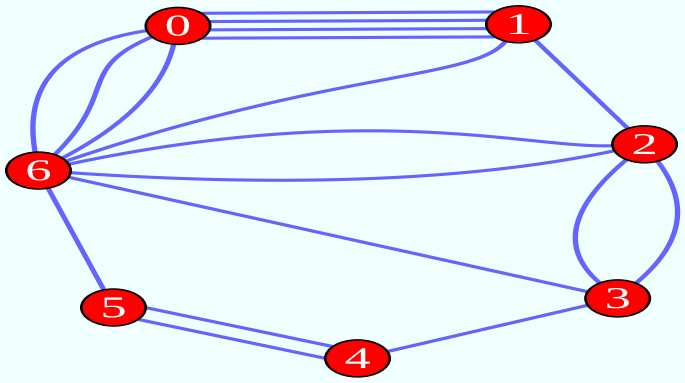
$$\forall v \in V(G),\ \{w \mid \exists \varphi \in Aut(G) : \varphi(v) = w\} = V(G)$$

Exemplu:

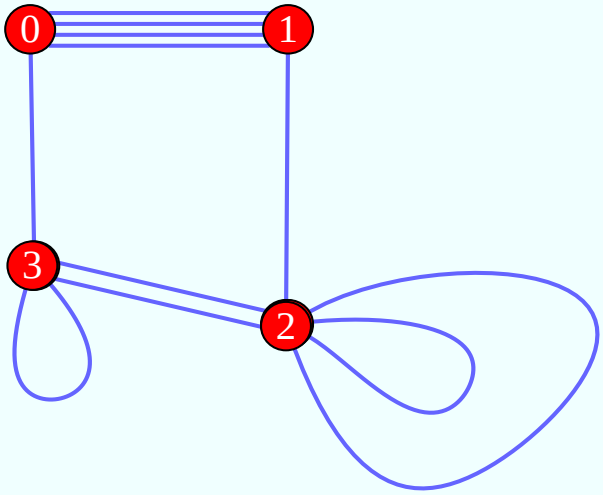


2. Variaţii în definiţia unui graf

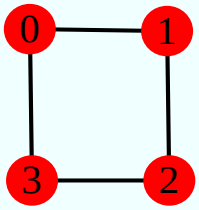
a) Dacă în definiţia unui graf, se consideră $E(G)$ o multimulţime pe $\mathcal{P}_2\Big(V(G)\Big)$, adică este dată o funcţie $m\colon \mathcal{P}_2\Big(V(G)\Big) \to \mathbf{N}$, se obţine noţiunea de **multigraf**.
Un element $e \in \mathcal{P}_2\Big(V(G)\Big)$ cu $m(e) > 0$ este muchie a multigrafului, **simplă** dacă $m(e) = 1$, **multiplă** dacă $m(e) > 1$.
Oricărui multigraf M i se poate asocia un graf $G(M)$, numit **graful suport** al lui M, obţinut prin înlocuirea fiecărei muchii multiple cu o singură muchie cu aceleaşi extremităţi.
Pictural, modificările de reprezentare sunt evidente; graful suport al multigrafului desenat mai jos, este graful desenat pe pagina precedentă.



b) Dacă în definiția unui graf se consideră $E(G)$ ca o multimulțime pe mulțimea părților nevide cu cel mult două elemente ale lui $V(G)$, atunci G se numește **graf general** sau **pseudograf**.
O muchie $e \in E(G)$, $e = \{v\}$ se numește **buclă** în vîrful v.
Exemplul următor arată un graf general M și graful său suport.



M



G(M)

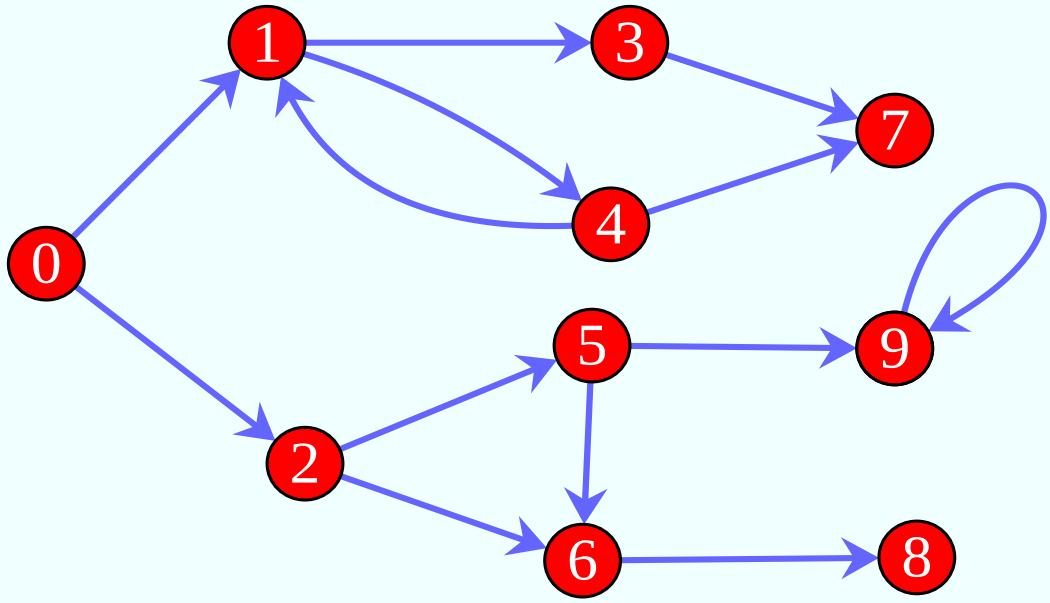
Pentru evitarea confuziilor, uneori grafurile – așa cum le-am definit– se mai numesc și *grafuri simple* .

c) Un **digraf** este o pereche $D = (V(D), A(D))$ unde $V(D)$ este o mulțime finită nevidă (mulțimea vîrfurilor digrafului D), iar
$A(D) \subseteq V(D) \times V(D)$ este mulțimea **arcelor** digrafului D.

Dacă $a = (u, v)$ este arc în D, notăm $a = uv$ și spunem că
u și v sînt **adiacente**;
a este **incident din** u ;
a este **incident spre** v;
u **dominã** pe v;
a este **incident** cu u **spre exterior**;
a este **incident** cu v **spre interior**;
u este **extremitatea inițială** a lui a și v este **extremitatea finală** a lui a.

Pictural, digrafurile se reprezintă la fel ca și grafurile, adăugînd curbei ce unește două figuri asociate vârfurilor o săgeată pentru a preciza **perechea** de vîrfuri corespunzătoare arcului desenat.

Exemplu:

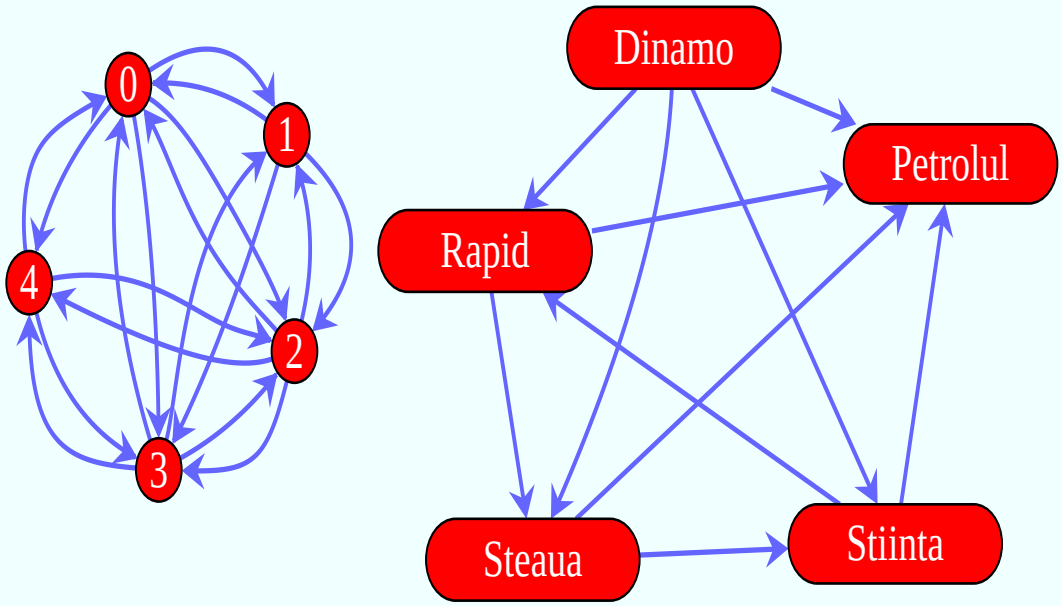


O pereche de arce de forma vw și wv se numește **pereche simetrică** de arce.
Dacă D este un digraf, **inversul** său D' este digraful obținut din D prin înlocuirea fiecărui arc vw cu opusul său wv.

Dacă D este un digraf, atunci înlocuind fiecare arc cu mulțimea de vîrfuri care îl formează, obținem, în general, un graf general $M(D)$. Graful suport al acestuia se numește **graful suport** al digrafului D.

Dacă $M(D)$ este graf atunci D se numește **graf orientat** (poate fi gîndit ca obţinut prin "orientarea" fiecărei muchii a grafului $M(D)$).

Un **digraf complet simetric** este un digraf în care fiecare pereche de vîrfuri este unită prin exact o pereche de arce simetrice.
Un **turneu** este un digraf în care orice două vîrfuri sînt unite prin exact un arc.



d) Grafurile **infinite** se obţin prin înlăturarea condiţiei de finitudine a mulţimii de vîrfuri şi (sau) muchii. Acestea se consideră a fi numărabile, iar tratarea lor utilizează instrumente care nu sunt neaparat combinatorii (de exemplu, mecanisme generative). Un graf G **local finit** este un graf infinit în care $N_G(v)$ este finită pentru orice vîrf v.

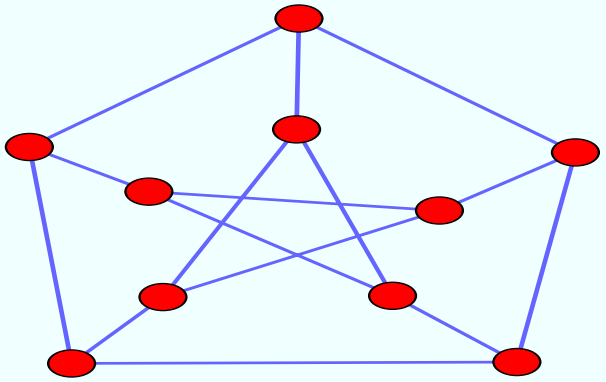
e) **Hipergrafurile** se obţin renunţând la condiţia că muchiile pot avea cel mult două vârfuri, (se obţin astfel **hipermuchiile**). Ele se mai numesc **sisteme finite de mulţimi** şi vom arăta că pot fi studiate via grafurile bipartite, deşi există rezultate combinatorii importante şi cu aplicaţii directe (de exemplu în bazele de date) în formalismul care urmează extinderea tratării grafurilor (din punct de vedere combinatoriu sau algebric).

3. Grade

Dacă $G = (V, E)$este un graf și $v \in V$ un vîrf al său, atunci **valența** sau **gradul** lui v în G, notat $d_G(v)$ sau $\rho_G(v)$ este

$$|\{e \mid e \in E, e \text{ incidentă cu } v\}|.$$

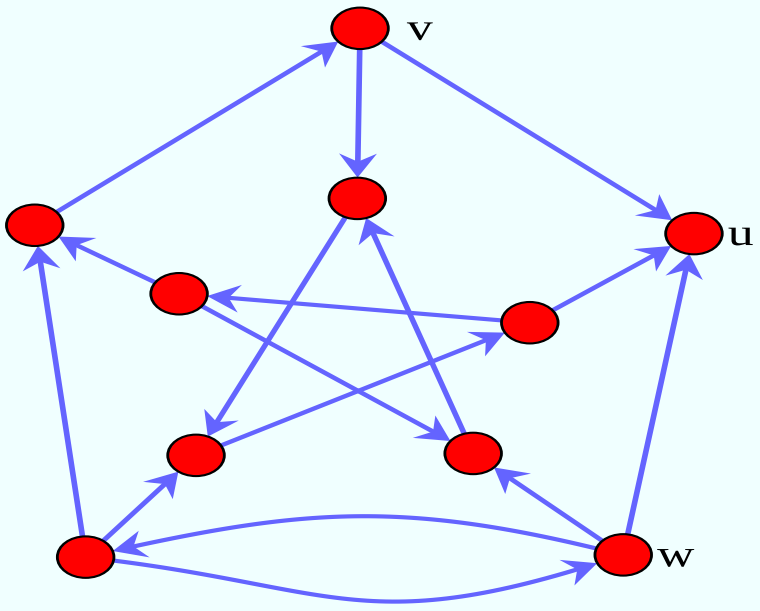
Un vîrf de grad 0 se numește **izolat**; un vîrf de grad 1 se numește **pendant**. Dacă toate vîrfurile lui G au aceeași valență ρ atunci G se numește **graf** $\rho-$**valent** sau $\rho-$**regulat**. Un graf 0−valent se numește **graf nul**. Un graf 3−valent se numeste **graf trivalent** sau **cubic**. Un exemplu de graf trivalent este graful lui *Petersen*:



Gradul maxim al unui vârf al grafului G se notează cu $\mathbf{\Delta(G)}$, iar gradul minim $\delta(\mathbf{G})$.

Concepte analoage se pot defini și pentru digrafuri. Dacă v este un vîrf al digrafului D atunci **valența interioară** sau **gradul interior** notat $\rho_{in}(v)$ sau $\rho_D^-(v)$ sau $d_D^-(v)$, este numărul arcelor incidente cu v spre interior; **valența exterioară** sau **gradul exterior** notat $\rho_{out}(v)$ sau $\rho_D^+(v)$ sau $d_D^+(v)$, este numărul arcelor incidente cu v spre exterior.

De exemplu, în digraful D desenat mai jos avem
$d_D^-(v) = 1,\ d_D^+(v) = 2;$
$d_D^-(u) = 3,\ d_D^+(u) = 0;$
$d_D^-(w) = 1,\ d_D^+(w) = 3;$

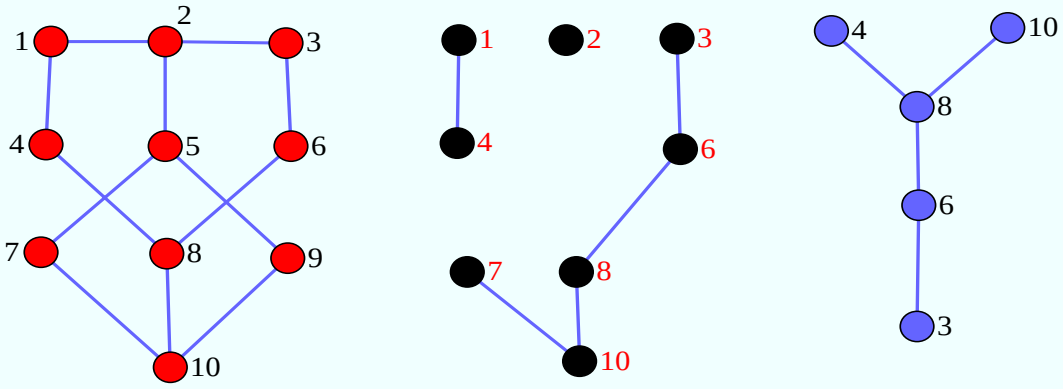


4. Subgrafuri

Un **subgraf** al grafului $G = (V(G), E(G))$este un graf $H = (V(H), E(H))$ care satisface: $V(H) \subseteq V(G)$ și $E(H) \subseteq E(G)$.

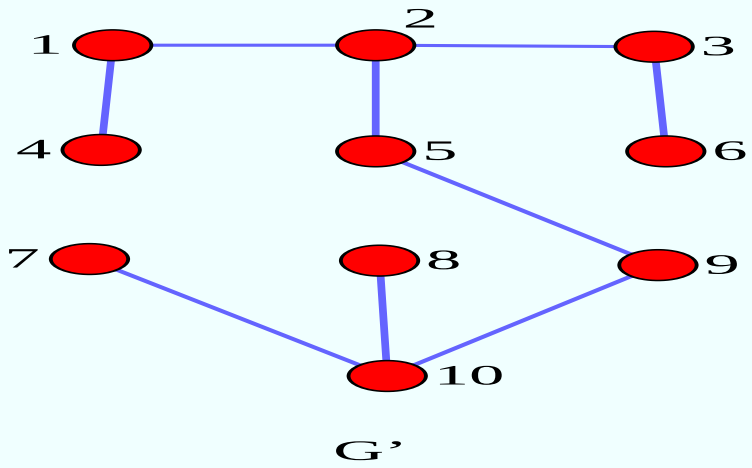
Dacă în plus, $V(H) = V(G)$ atunci H se numește **graf parțial** al lui G (în limba engleză, spanning subgraph).

Dacă $A \subseteq V(G)$ atunci $[A]_G = (A, \mathcal{P}_2(A) \cap E(G))$ se numește **subgraf indus** în G de mulțimea de vîrfuri A (se mai notează și $G[A]$).
În figura următoare, H este subgraf al lui G iar subgraful indus de mulțimea de vârfuri $\{3, 4, 6, 8, 10\}$ este $G[\{3, 4, 6, 8, 10\}]$:



Subgraful $[V(G) \setminus A]_G$ se notează $G - A$ și este **subgraful** lui G **obținut prin îndepărtarea vîrfurilor din** A; în particular, dacă $A = \{v\}$, atunci $G - \{v\}$ se notează $G - v$.

Dacă $E' \subseteq E(G)$ atunci $\langle E' \rangle_G = (V(G), E')$ este **graful parțial secționat de** E' **în** G. $G - E'$ este prin definiție $\langle E(G) \setminus E' \rangle_G$, iar $G - e = G - \{e\}$ $(e \in E(G))$. Pentru G graful din figura precedentă și $E' = \{12, 14, 23, 25, 36, 59, 710, 810, 910\}$, obținem că $\langle E' \rangle_G$ este graful G' :

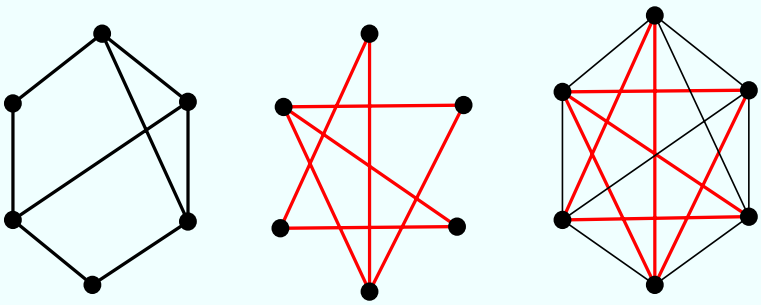


Concepte similare se pot defini în mod analog pentru multigrafuri, grafuri generale sau digrafuri.

5. Operații cu grafuri

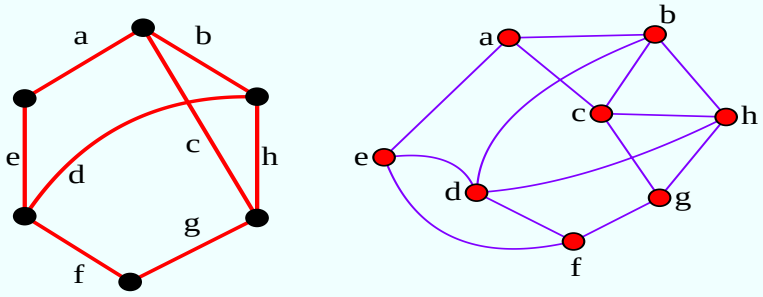
Dacă $G = (V(G), E(G))$este un graf, atunci :

-**complementarul** său este graful $\overline{\mathbf{G}}$ cu $V(\overline{G}) = V(G)$ și $E(\overline{G}) = \mathcal{P}_2\Big(V(G)\Big) \setminus E(G)$.



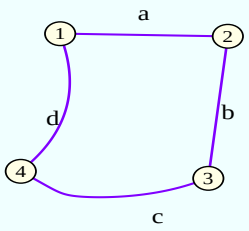
Graful initial Complementarul Graful complet

-**graful reprezentativ al muchiilor lui** G este graful $\mathbf{L(G)}$ cu $V(L(G)) = E(G)$ și $E(L(G)) = \{ee' \mid e, e' \in E(G), e$ și e' adiacente în $G\}$.

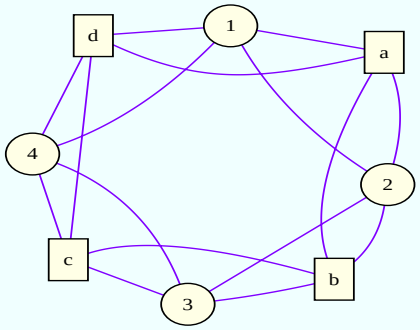


Graful initial Line-graful sau

-**graful total** al grafului G este graful $\mathbf{T(G)}$ cu $V(T(G)) = V(G) \cup E(G)$ și $E(T(G)) = \{xy | x, y \in V(G) \cup E(G), x$ și y adiacente sau incidente în $G\}$.

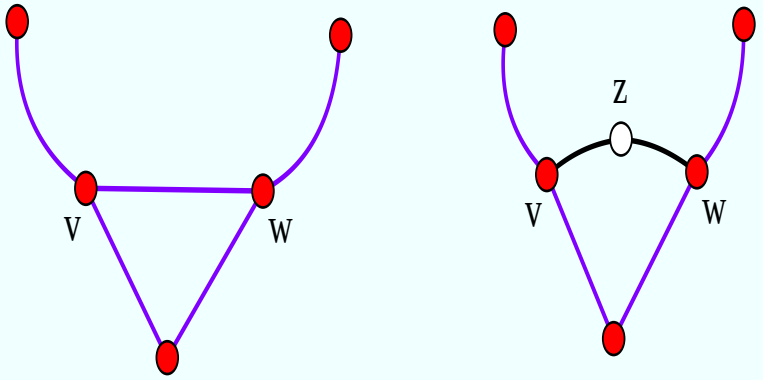


Graful initial



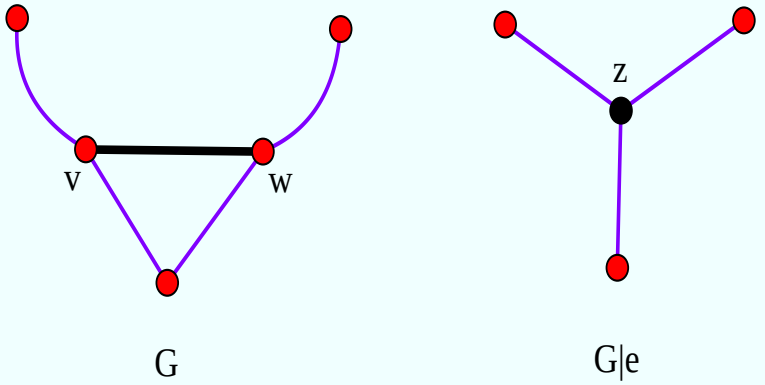
Graful total

-**graful obţinut din G prin inserţia unui vîrf** (z) *pe o muchie* $(e = vw)$ este graful $G' = (V(G) \cup \{z\}, E(G) \setminus \{vw\} \cup \{vz, zw\})$ $(z \notin V(G), e \in E(G))$.



Două grafuri obţinute prin inserţii succesive de vîrfuri pe muchiile aceluiași graf se numesc **homeomorfe**.

-graful obținut din G prin contracția muchiei $e = vw \in E(G)$ este graful $\mathbf{G|e} = (V(G) \setminus \{v, w\} \cup \{z\}, E([V(G) \setminus \{v, w\}]_G) \cup \{yz \mid yv \text{ sau } yw \in E(G)\})$.



G G|e

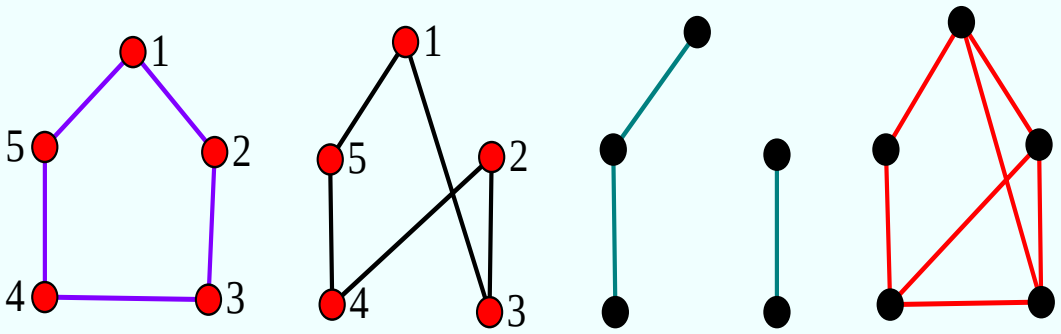
Dacă H se poate obține prin contracții succesive de muchii din graful G, se spune că G **este contractibil la** H.

Fie $G = (V(G), E(G))$**și** $G' = (V(G'), E(G'))$ **două grafuri.**

- Dacă $V(G) = V(G')$ atunci **reuniunea** celor două grafuri și **intersecția** lor se definesc

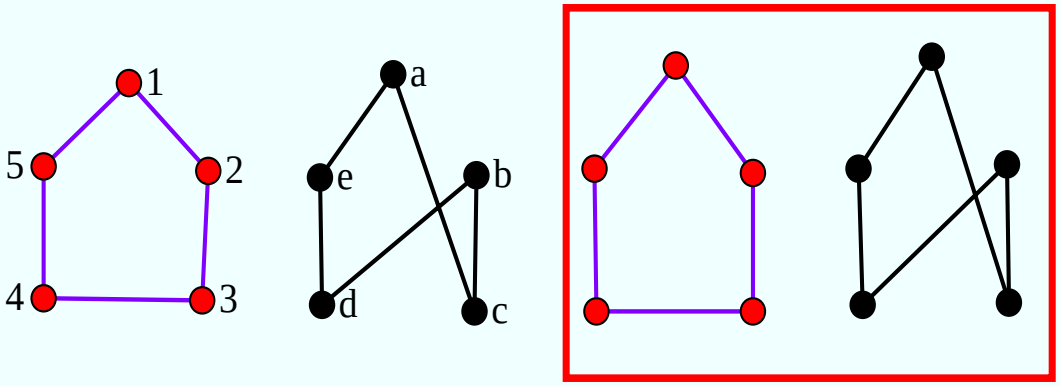
$\mathbf{G \cup G'} = (V(G), E(G) \cup E(G'))$,

$\mathbf{G \cap G'} = (V(G), E(G) \cap E(G'))$.



G G' Intersectia Reuniunea

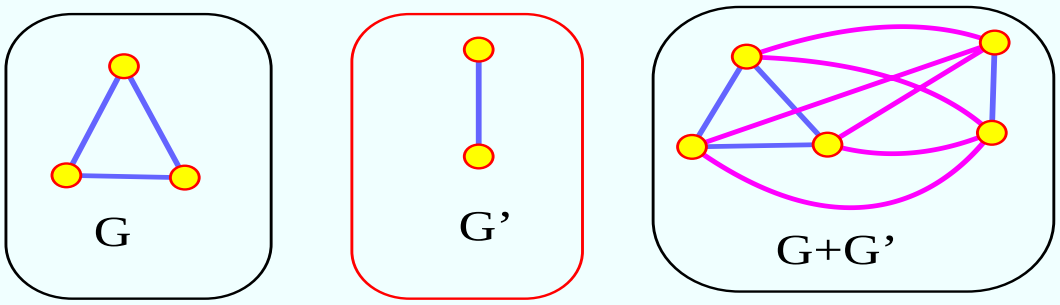
-Dacă $V(G) \cap V(G') = \emptyset$ atunci $\mathbf{G \cup G'} = (V(G) \cup V(G'), E(G) \cup E(G'))$ se numește **reuniunea disjunctă** a grafurilor G și G'. Reuniunea disjunctă a k grafuri izomorfe cu G se notează kG.



G G' Reuniunea disjuncta

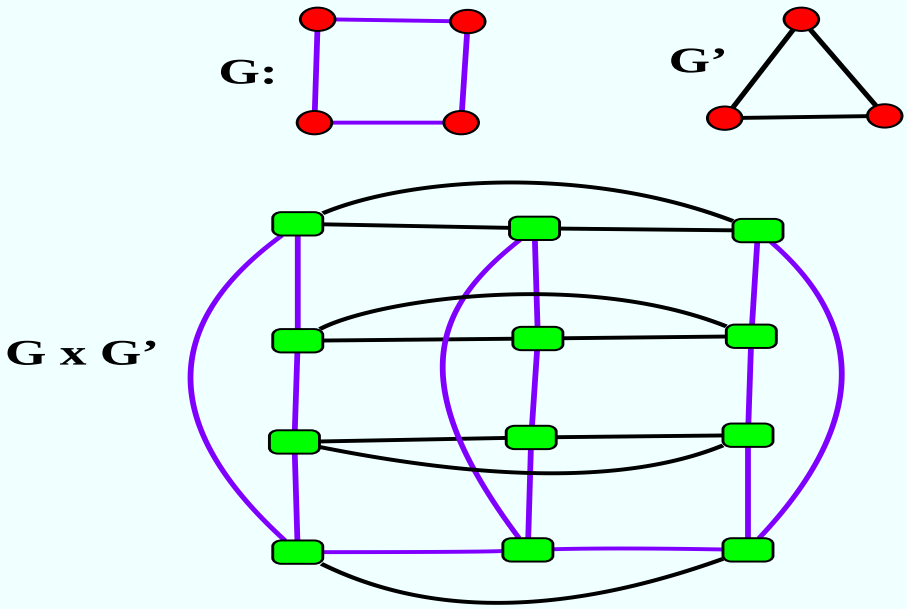
-**Suma** a două grafuri G și G' este graful

$$\mathbf{G + G'} = \overline{\overline{G} \cup \overline{G'}}.$$



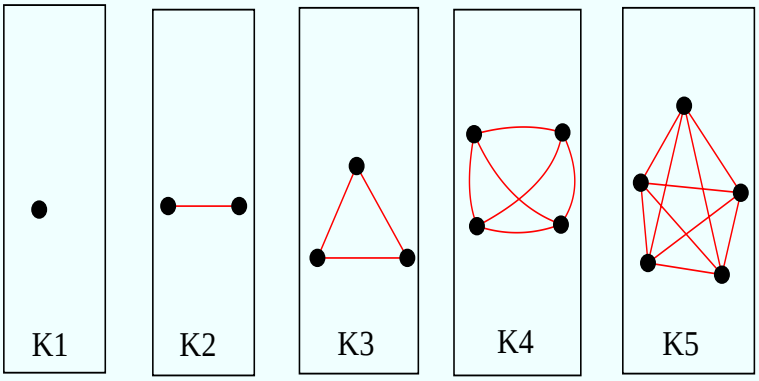
-**Produsul cartezian** al grafurilor G și G' este graful $\mathbf{G \times G'}$ cu $V(G \times G') = V(G) \times V(G')$ și

$$E(G \times G') = \{(v,w)(v',w') | v, v' \in V(G), w, w' \in V(G')$$
$$v = v' \text{și } ww' \in E(G') \text{sau}$$
$$w = w' \text{ și } vv' \in E(G)\}$$

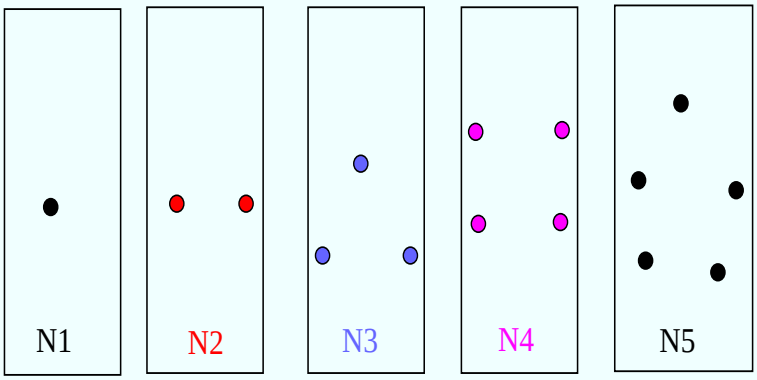


6. Clase de grafuri

Graful complet de ordin n : K_n
cu $\left|V(K_n)\right| = n$ și $E(K_n) = \mathcal{P}_2\Big(V(K_n)\Big)$.



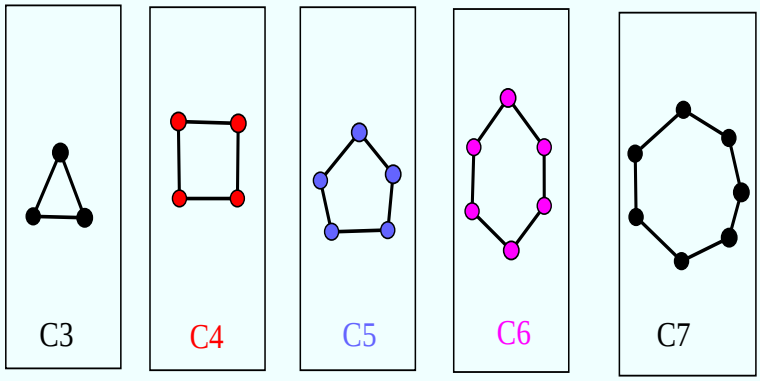
Graful nul de ordin n : $N_n = \overline{K}_n$.



Circuitul de ordin n $(n \geq 3)$: C_n

cu $V(C_n) = \{1, \ldots, n\}$ și

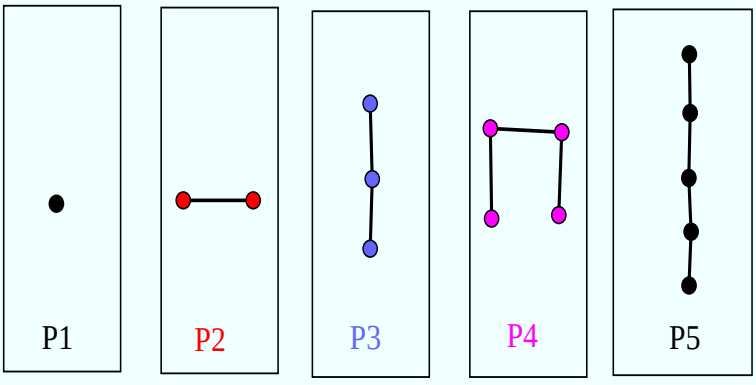
$E(C_n) = \{12, 23, \ldots, n-1n, n1\}$.



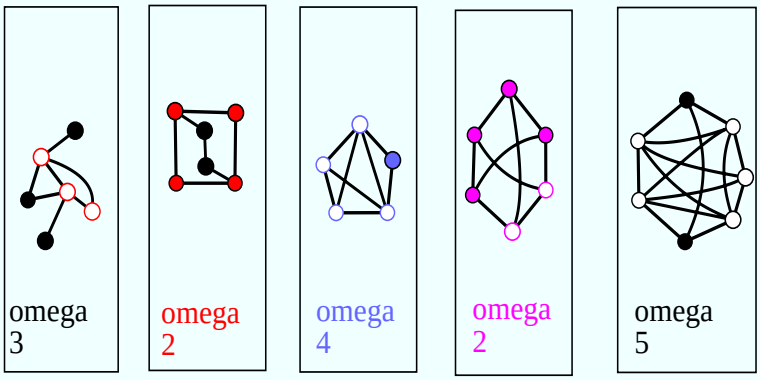
Drumul de ordin n : P_n

$P_1 = K_1$, $P_2 = K_2$;

$n \geq 3$: $P_n = C_n - e$ $(e \in E(C_n))$.

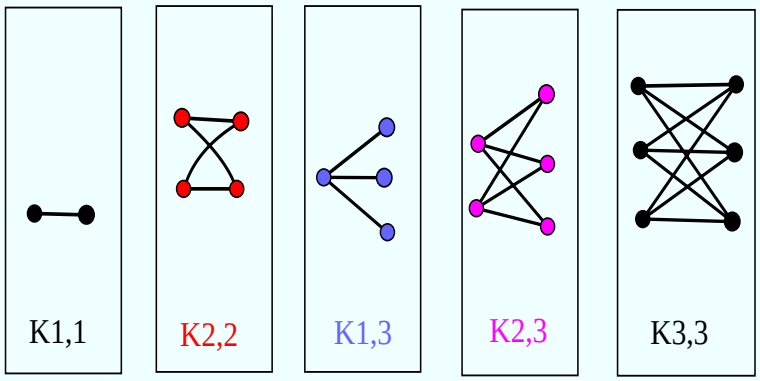


Un subgraf complet (de ordin q) al unui graf G se numește **clicã** (q-**clicã)** a lui G.
Cardinalul maxim al unei clici a lui G se numește **numãrul de clicã** sau **numãrul de densitate** al lui G și se notează $\omega(G)$. Cum, evident $\omega(G) = \alpha(\overline{G})$, rezultã cã determinarea numãrului de clicã al unui graf și a unei clici de cardinal maxim este problema **P1** pcu intrarea $\overline{G}$.
Exemple:

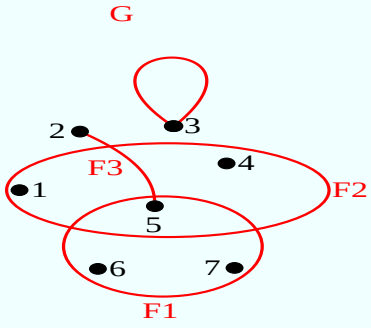


Un **graf bipartit** este un graf G cu proprietatea cã $V(G)$ se poate partiționa în douã mulțimi independente în G.
Dacã S și T satisfac $S \cup T = V(G)$, $S \cup T = \emptyset$ și S, T sînt independente și nevide în G, atunci graful bipartit G se notează $G = (S, T; E(G))$.

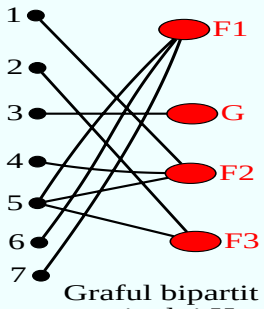
Deci, dacă $G = (S, T; E(G))$ este un graf bipartit, atunci $\forall e \in E(G)$ are o extremitate în S și cealaltă în T.
Dacă $\forall v \in S$ și $\forall w \in T \qquad vw \in E(G)$, atunci graful bipartit $G = (S, T; E(G))$ se numește **graf bipartit complet** și se notează $K_{s,t}$ unde $s = |S|$ și $t = |T|$.



Pentru orice hipergraf $H = (V, \mathcal{E})$, se poate asocia un graf bipartit $G_H = (V, \mathcal{E}; E(G_H))$, unde $\forall v \in V$, $\forall F \in \mathcal{E}$ $vF \in E(G_H) \Leftrightarrow v \in F$.



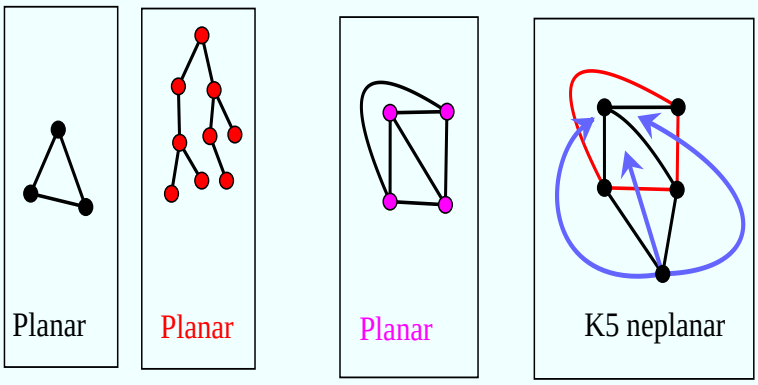
Hipergraful H



Graful bipartit asociat lui H

O construcție inversă evidentă, ne arată că și pentru orice graf bipartit se poate asocia un hipergraf.

Un graf $G = (V(G), E(G))$se numește **planar** dacă poate fi reprezentat în plan astfel încît fiecărui vîrf să-i corespundă un punct al planului, iar muchiilor le corespund curbe simple ce unesc punctele corespunzătoare extremităţilor lor și în plus **aceste curbe se interesectează (eventual) numai în vîrfuri.** Un graf care nu-i planar se numște **neplanar**. Exemple minimale de grafuri neplanare sînt grafurile K_5 și $K_{3,3}$.



Deși problema

PLAN Intrare: G un graf.
Intrebare: Este G planar ?

pare mult mai dificilă decât problema stabilei maxime (P1 din cursul trecut), ea subsumând noțiuni de topologie, s-a dovedit că este din **P** (Hopcroft & Tarjan , 1972, $O(n+m)$).

O modalitate uzuală de a defini clase de grafuri este de a **interzice** apariția unor subgrafuri induse, pentru grafurile acelei clase.
Dacă $\mathcal{F}$ este o mulțime de grafuri, atunci un graf G se numește $\mathcal{F}$-**liber** (sau $\mathcal{F}$-**free**) dacă G nu are ca subgraf indus pe niciunul dintre grafurile lui $\mathcal{F}$. De exemplu, clasa grafurilor nule poate fi definită ca fiind clasa grafurilor K_2-free; clasa grafurilor ale căror componente conexe sunt subgrafuri complete este clasa grafurilor P_3-free;
clasa grafurilor **triangulate** (sau **cordale**) este clasa grafurilor $(C_k)_{k\geq 4}$-free.

7. Drumuri și circuite

Fie $G = (V(G), E(G))$un graf.

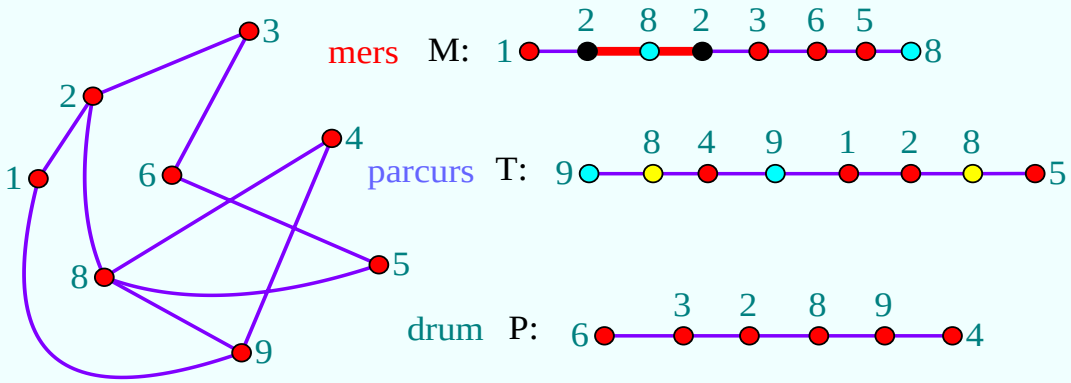
Se numește **mers** (walk) **de lungime** r **de la** v **la** w în G un șir de vîrfuri și muchii

$$(v =)v_0, v_0v_1, v_1, \ldots, v_{r-1}, v_{r-1}v_r, v_r(= w);$$

v și w se numesc **extremitățile mersului**.

Dacă muchiile mersului sînt distincte atunci mersul se numește **parcurs** (trail) în G de la v la w.

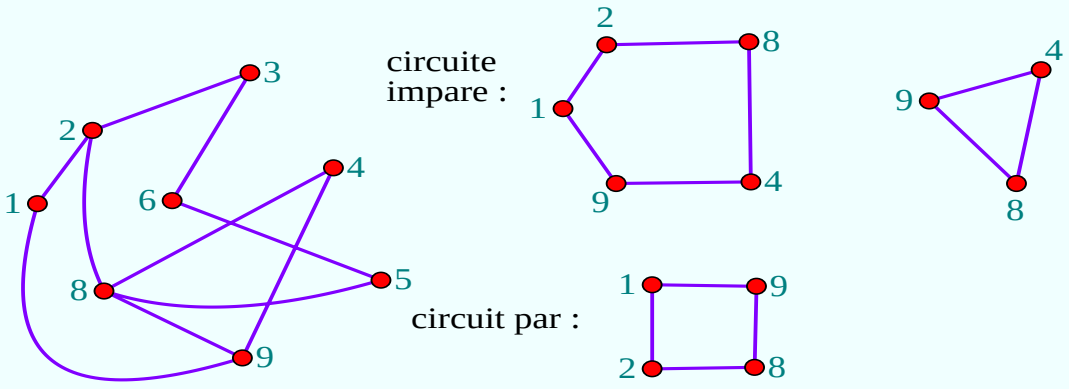
Dacă vîrfurile sînt distincte atunci mersul se numește **drum** (path) de la v la w.



Dacă $v = w$ atunci mersul (parcursul) se numește **închis**.

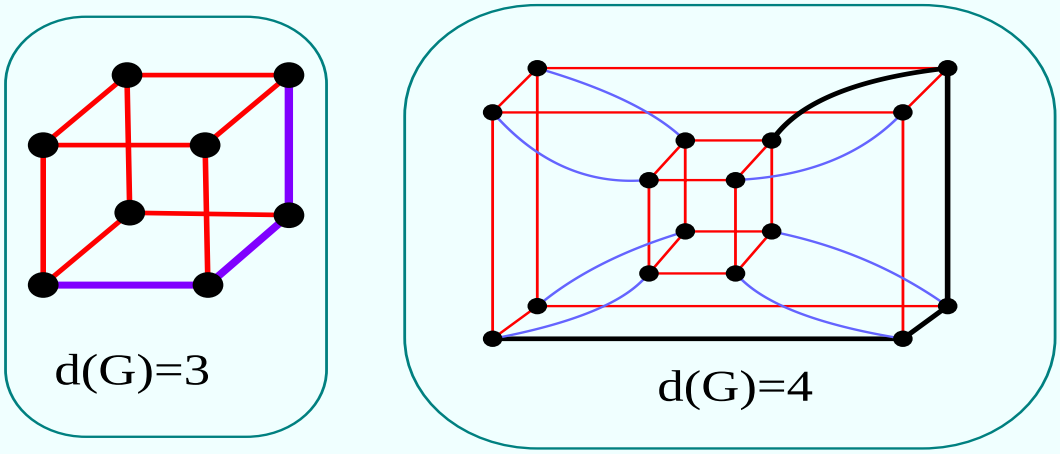
Dacă într-un mers toate vîrfurile sînt distincte, cu excepția extremităților, atunci mersul se numește **circuit** (sau **drum închis**).

Un circuit este **par** sau **impar** după cum lungimea sa (numărul muchiilor) este parā sau imparā.



Lungimea celui mai scurt circuit al grafului G (dacă G are circuite) se numește **grația** (girth) grafului G și se notează cu $g(G)$; lungimea celui mai lung circuit al lui G se numește **circumferința** lui G și se notează $c(G)$.

Dacă v și w sînt vîrfuri ale lui G, lungimea celui mai scurt drum de la v la w în G se numește **distanța** în G de la v la w și se notează $d_G(v, w)$.
Diametrul grafului G, notat $d(G)$ este $d(G) = \max\{d_G(v, w)|v, w \in V(G)\}$.

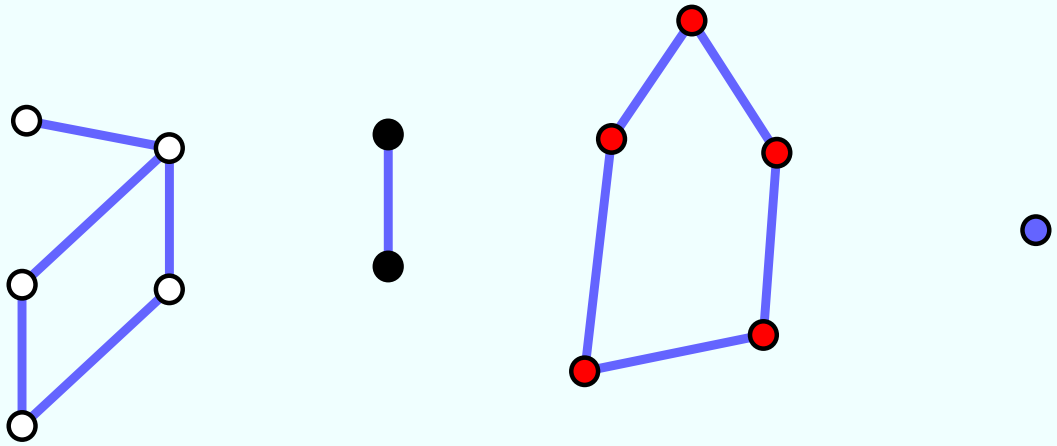


În proiectarea rețelelor de interconectare a procesoarelor este important ca graful G reprezentând rețeaua să aibă gradul maxim $\Delta(G)$ mic (restricție tehnologică) și diametrul $d(G)$ mic în raport cu numărul vârfurilor (restricție de calitate a interconectării).

Definițiile de mai sus se extind, în mod evident,**pentru digrafuri singura modificare** fiind aceea că **se înlocuiesc muchiile cu arce.**

Un graf este **conex** dacă există (măcar) un drum între orice două vîrfuri ale sale; un graf care nu este conex se numește **neconex**.

Orice graf G poate fi unic exprimat ca o reuniune disjunctă de *subgrafuri induse, conexe și maximale cu această proprietate;* aceste subgrafuri se numesc **componentele conexe** ale grafului G (mai precis, se pot defini componentele conexe ca subgrafurile induse de clasele de echivalență determinate pe $V(G)$ de relația de echivalență $\rho \subseteq V(G) \times V(G)$ definită prin : $v \ \rho \ w \Leftrightarrow$ există în G un drum de la v la w).



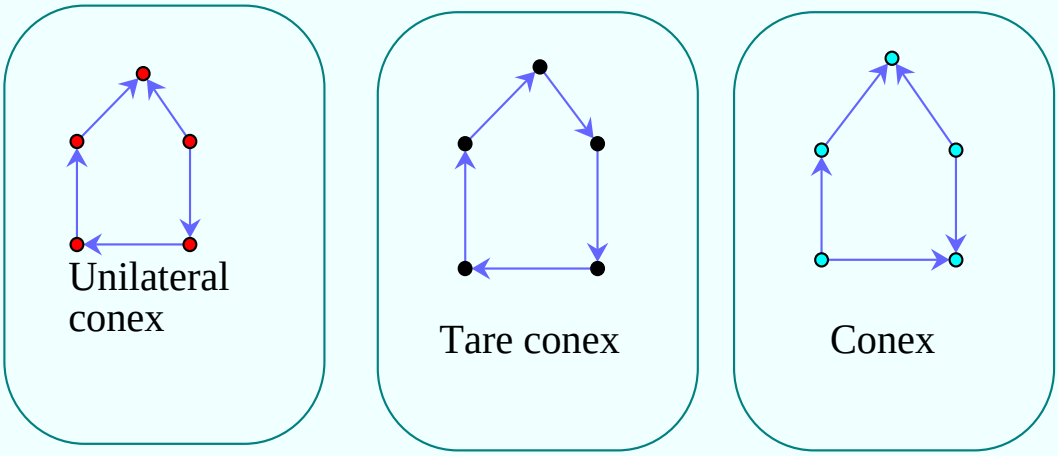
Graful din figura de mai sus are 4 componente conexe: una cu 1 vârf, una cu 2 vârfuri și două cu 5 vârfuri.

Concepte analoage se pot defini și pentru digrafuri; dacă D este un digraf atunci :

- D este **tare conex** dacă $\forall (v, w) \in V(D) \times V(D)$ există un drum în D de la v la w;

- D este **unilateral conex** dacă $\forall (v, w) \in V(D) \times V(D)$ există în D un drum de la v la w **sau** un drum de la w la v;

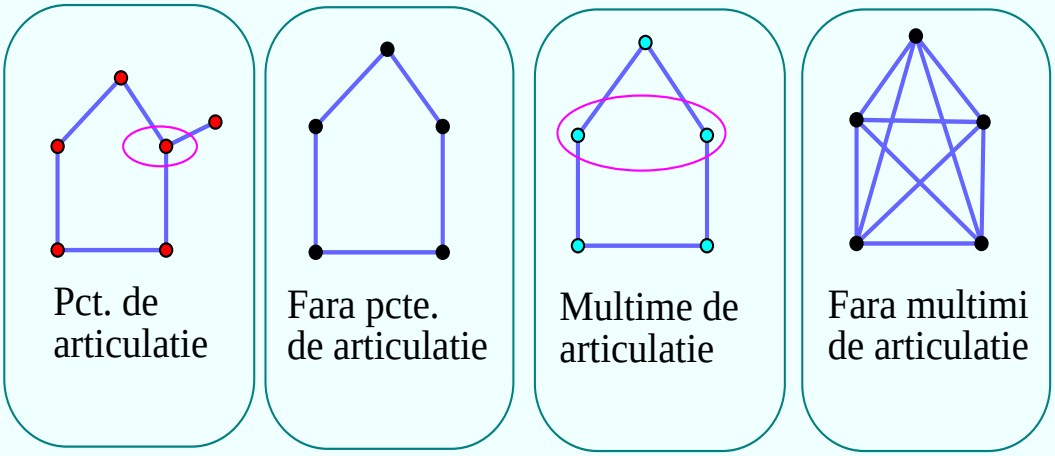
- D este **conex** dacă $G(D)$, graful suport al lui D, este conex.



Un graf conex care nu are circuite se numește **arbore**. Un graf ale cărui componente conexe sînt arbori se numește **pãdure**.

Time to leave the trees !!!

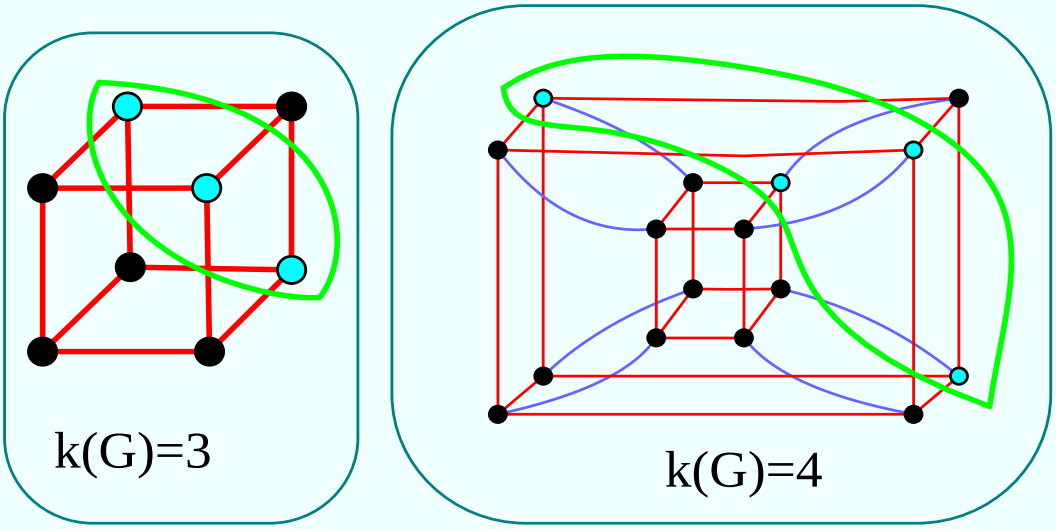
Dacă G este un graf conex, un vîrf $v \in V(G)$ cu proprietatea că $G - v$ este neconex se numește **vîrf (punct) de articulație**; mai general, o mulțime A de vîrfuri ale unui graf G se numește **mulțime separatoare de vîrfuri (mulțime de articulație)** dacă $G - A$ este neconex.



Fie p un număr întreg pozitiv;
un graf G cu măcar p vîrfuri este $\mathbf{p-conex}$ dacă $G = K_p$ sau are cel puţin $p+1$ vîrfuri şi nu are mulţimi separatoare de vîrfuri de cardinal mai mic decît p.

Evident, G este 1-conex dacă şi numai dacă este conex. Un graf 2-conex se mai numeşte şi *bloc*.

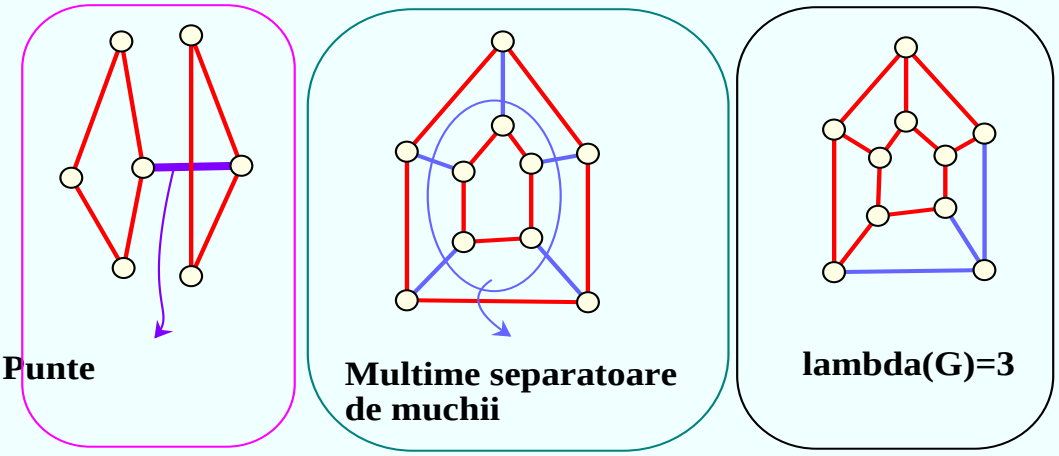
Numărul de conexiune al lui G, notat $\mathbf{k(G)}$, este cel mai mare număr natural p pentru care G este $p-$conex.



Dacă G este un graf conex, o muchie $e \in E(G)$ cu proprietatea că $G - e$ este neconex se numește **punte** în graful G; mai general, o mulțime A de muchii ale unui graf G se numește **mulțime separatoare de muchii** dacă $G - A$ este neconex.

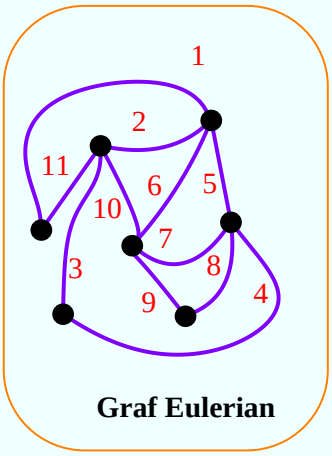
Un graf G cu cel puțin p vîrfuri este **p−muchie-conex** dacă nu admite mulțimi separatoare de muchii de cardinal mai mic decît p.

Numărul de muchie-conexiune al lui G, notat $\lambda(G)$, este cel mai mare număr natural p pentru care G este $p-$muchie-conex .

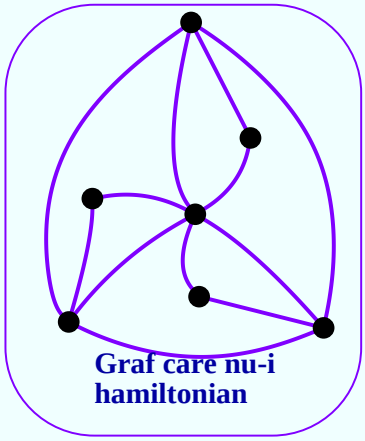


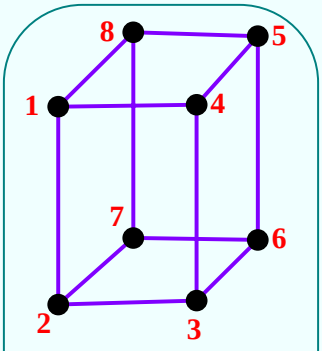
Un graf (sau digraf) se numește **eulerian** dacă admite un parcurs închis care folosește fiecare muchie a grafului (respectiv, fiecare arc al digrafului).

Un (di)graf G se numește **hamiltonian** dacă are un circuit care trece prin fiecare vîrf.



Graf Eulerian


Graf care nu-i hamiltonian



Graf hamiltonian

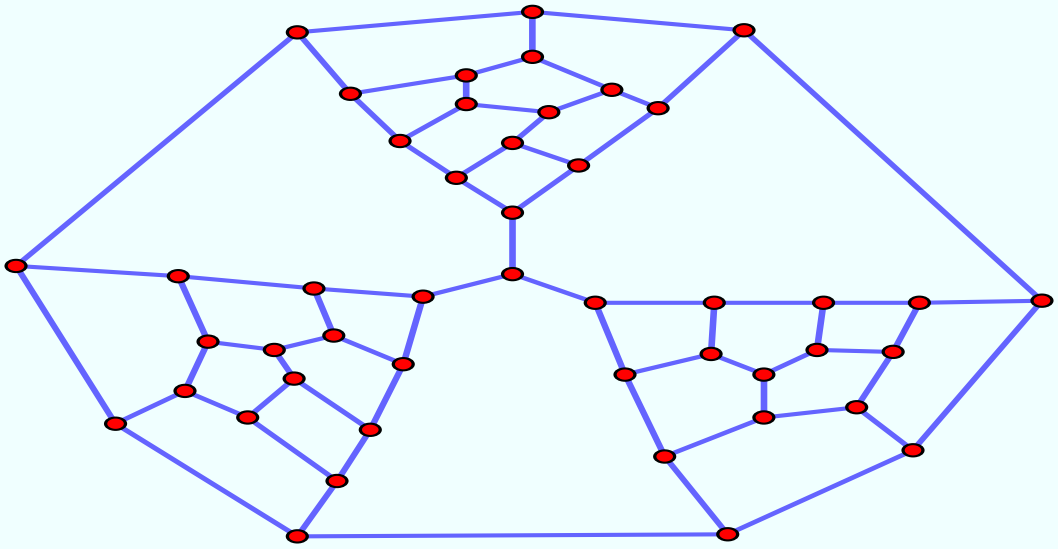
În timp ce problema testării dacă un graf este elerian este foarte simplă (Euler 1736 : conex și cu toate vârfurile de grad par), problema

HAM Intrare: G un graf.
Intrebare: Este G hamiltonian ?

este **NP**-completă. Apartenența la **NP** este evidentă: un circuit hamiltonian, se poate indica printr-o permutare a vârfurilor care poate fi testată în timp liniar. În schimb pentru problema

NH Intrare: G un graf.
Intrebare: Este adevărat că G nu-i hamiltonian?

nu se cunoaște o demonstrație a apartenenței la **NP** (se observă că este din **co-NP**). Nu se poate da o demonstrație succintă că graful de mai jos nu e hamiltonian:



3-conex, planar, si nehamiltonian
(Tutte)

8. Matrici asociate.

Dacă $G = (\{v_1, \ldots, v_n\}, \{e_1, \ldots, e_m\})$ este un graf, atunci

Matricea de adiacenţă a grafului G este matricea $A = (a_{ij})_{n \times n}$, unde

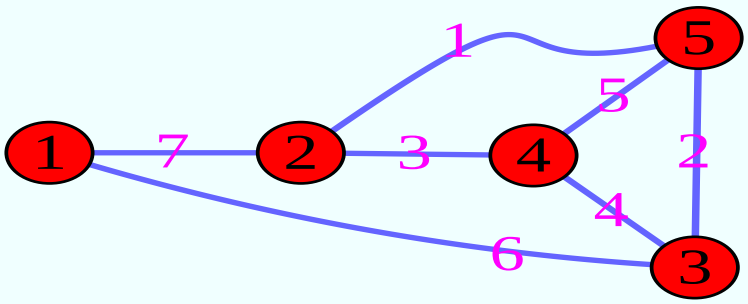
$$a_{ij} = \begin{cases} 1 & \text{dacă } v_i \text{ şi } v_j \text{ sînt adiacente} \\ 0 & \text{altminteri.} \end{cases}$$

Matricea de incidenţă a grafului G este matricea $B = (b_{ij})_{n \times m}$, unde

$$b_{ij} = \begin{cases} 1 & \text{dacă } v_i \text{ şi } e_j \text{ sînt incidente} \\ 0 & \text{altminteri.} \end{cases}$$

În cazul digrafurilor, se pot asocia similar astfel de matrici, în care, evident se poate indica şi orientarea arcelor, folosind elementele 0, 1 şi -1.

Pentru graful din figura de mai jos,



matricea de adiacenţă este:

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix},$$

iar matricea de incidenţă:

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}.$$

Valorile proprii şi polinomul caracteristic ale matricii de adiacenţă se numesc **valorile proprii ale grafului**, respectiv, **polinomul caracteristic al grafului**.

9 Structuri de date utilizate în reprezentarea (di)grafurilor.

Fie $G = (V, E)$ un (di)graf cu $V = \{1, 2, \ldots, n\}$ și $|E| = e$.

Cele mai uzuale structuri de date utilizate pentru reprezentarea (di)grafului G sunt:

a) *matricea de adiacență*

Dacă $A = (a_{ij})_{n \times n}$ este matricea de adiacență a lui G atunci, reprezentarea acesteia cu ajutorul unui tablou bidimensional va necesita $O(n^2)$ operații pentru orice inițializare, deci orice algoritm, care folosește o astfel de reprezentare, are complexitatea $\Omega(n^2)$.

Cu această structură de date testarea dacă două vârfuri sunt sau nu adiacente se face în $O(1)$, dar parcurgerea lui $N_G(v)$ (sau $N_G^+(v)$), pentru un vârf oarecare $v \in V$, necesită $\Omega(n)$ operații.

b) listele de adiacenţă

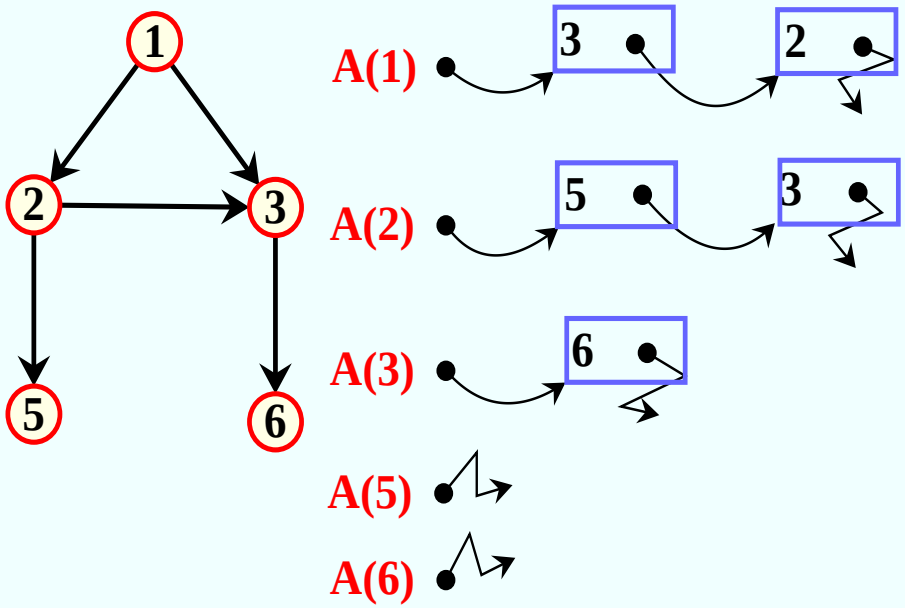
Pentru fiecare vîrf $v \in V$ se consideră o listă $A(v)$ a vecinilor săi în G.
Dacă G este graf, atunci $A(v)$ conţine $N_G(v) = \{v|w \in V$ şi $vw \in E\}$ iar dacă G este digraf atunci $A(v)$ conţine $N_G^+(v) = \{v|w \in V$ şi $vw \in E\}$.

Pentru cazul în care G este graf, fiecare muchie $vw \in E$ va genera două elemente în listele de adiacenţă, unul în $A(v)$ şi celălalt în $A(w)$. Spaţiul total de memorie utilizat va fi de $O(n+2e)$. Pentru cazul în care G este digraf, spaţiul de memorie utilizat este de $O(n+e)$.

Listele de adiacenţă pot fi reprezentate cu ajutorul tablourilor sau ca structuri dinamice de date (liste înlănţuite).

Testarea dacă un vârf fixat v este adiacent cu un vârf oarecare w în G se face în $\Omega(d_G(v))$, dar se poate parcurge $N_G(v)$ în timpul $O(d_G(v))$ şi nu $O(n)$ ca în cazul matricii de adiacenţă.

Pentru digraful desenat mai jos, sunt reprezentate listele de adiacenţă:



Cu noţiunile si terminologia minimală descrisă în secţiunile 1-9 ale acestui prim capitol, se poate trece la probleme algoritmice specifice.

II Probleme de drum în (di)grafuri

1. Parcurgeri sistematice ale (di)grafurilor.

Graph search- paradigmă algoritmică utilizată pentru a desemna o metodă sistematică de parcurgere a mulțimii vârfurilor la care se poate ajunge prin drumuri într-un (di)graf de la un vârf fixat.

Dat $G = (V, E)$ *un (di)graf cu mulțimea vîrfurilor* $V = \{1, \ldots, n\}$ *și* $s \in V$, *se cere să se genereze "eficient" mulțimea*
$S = \{v \in V | \exists D$ *drum în* G *de la* s *la* $v\}$.

(Di)graful G e reprezentat cu **listele de adiacenţă** (e nevoie de aflarea eficientă a mulţimii vecinilor nodului curent, în procesul sistematic de vizitare).

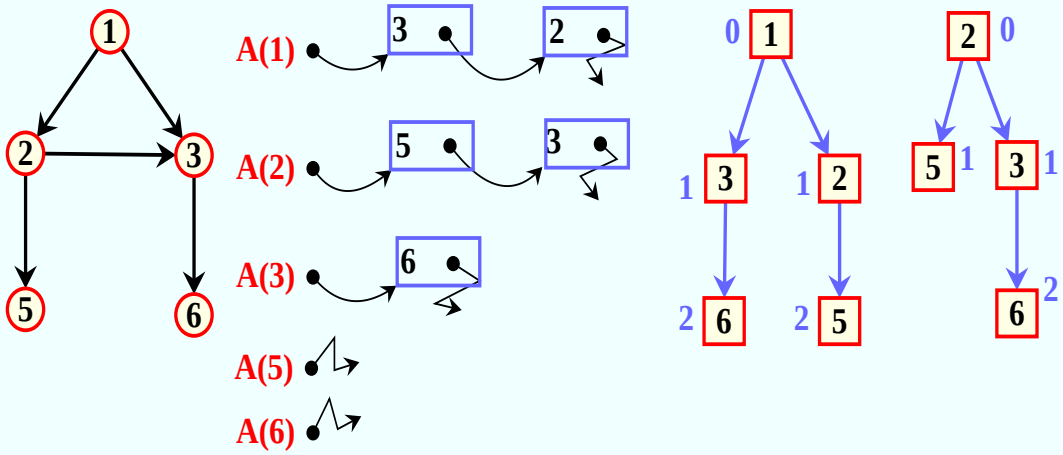
Prezentăm succint cele două tehnici principale de parcurgere.

bfs - breadth first search

Iniţial, $\forall v \in V$ are eticheta $label(v) < 0$.

$label(s) \leftarrow 0 \quad parent(s) \leftarrow 0$;
creează coada Q conţinând s;
while $Q \neq \emptyset$ **do**
{ $v \leftarrow$vârful din capul cozii Q;
şterge vârful din capul cozii Q;
for $w \in A(v)$ **do**
{ **if** $label(w) < 0$ **then**
{ $label(w) \leftarrow label(v) + 1$;
$parent(w) \leftarrow v$;
introdu w în coada Q
}
}
}

Evident:
- $S = \{v \in V | label(v) \geq 0\}$;
- $\forall v \in V \ label(v) = d_G(s, v)$;
- variabila $parent$ definește **arborele bfs** asociat căutării din s: dacă G e graf atunci acesta este arbore parțial al componentei conexe a lui G la care aparține s; dacă G este digraf atunci acesta este o arborescență (arbore orientat cu toate vârfurile accesibile prin drumuri din rădăcină);
- deoarece fiecare listă de adiacență a unui vârf din mulțimea S este traversată exact o dată, complexitatea timp a lui $bfs(s)$ este $O(n_S + m_S)$, unde $n_S = |S|$ iar $m_S = |E([S]_G)|$;
În figura următoare sunt desenați arborii corespunzători lui bfs(1) și bfs(2).



dfs - depth first search

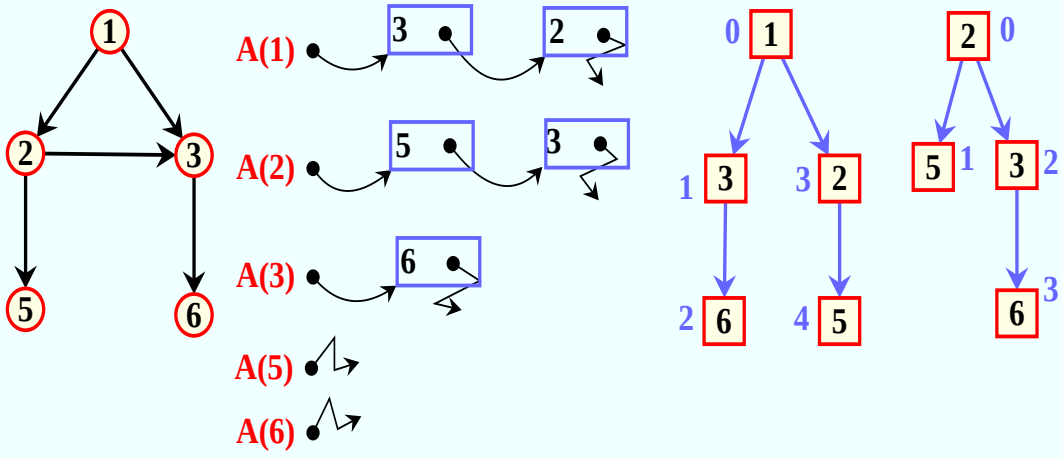
Inițial, $\forall v \in V$ are eticheta $label(v) < 0$

și toți pointerii de parcurgere a listelor de adiacență sunt la început.

$label(s) \leftarrow 0 \quad parent(s) \leftarrow 0$;
creează stiva S conținând s;
$n_S \leftarrow 1$
while $S \neq \emptyset$ **do**
{ $v \leftarrow$vârful din capul stivei S;
$w \leftarrow next[A(v)]$
if $\exists w$ **then**
if $label(w) < 0$ **then**
{ $label(w) \leftarrow n_S$; n_S++;
$parent(w) \leftarrow v$;
introdu w în stiva S
}
else *NOP* (aici; dar se poate utiliza !!)
else șterge vârful din capul stivei S
// s-a terminat căutarea din v;
}

Iarăși, rezultă imediat că
$S = \{v \in V | label(v) \geq 0\}$ și complexitatea timp a lui $dfs(s)$ este $O(n_S + m_S)$.

Pentru exemplul nostru de lucru, arborii dfs(1) și dfs(2) sunt ilustrați mai jos.



Parcurgerile sistematice sunt importante pentru obținerea unor algoritmi eficienți pentru determinarea componentelor conexe în grafuri, pentru determinarea componentelor tari conexe în digrafuri, componentelor 2-conexe în grafuri etc., (care sunt preprocesări uzuale pentru probleme mai complicate).

Ele sunt esențiale în problemele din Inteligența Artificială, unde spațiul stărilor de căutare poate fi văzut ca un graf . De data aceasta graful este dat implicit; în fiecare nod (stare) se dispune de un predicat *precondiție* care este utilizat de o funcție *neighbours* care intoarce lista nodurilor accesibile în contextul curent din acel nod. Graful explicit care se poate construi în principiu, este folosit pentru descrierea algoritmului și analizele de complexitate.

2. Probleme de drum minim.

Fie $G = (V, E)$ un digraf cu mulţimea vîrfurilor $V = \{1, \ldots, n\}$. Considerăm dată o funcţie $a : E \to \mathbf{R}$ cu interpretarea:
$\forall ij \in E \quad a(ij)$ =costul arcului ij (ponderea, lungimea, etc).

Dacă digraful G este reprezentat cu ajutorul listelor de adiacenţă, atunci costul arcului ij este un cîmp în nodul din lista de adiacenţă a lui i ce reprezintă acest arc.

Pentru uşurinţa notaţiilor vom folosi reprezentarea digrafului G cu ajutorul matricii de cost-adiacenţă $A = (a_{ij})_{n \times n}$ cu

$$a_{ij} = \begin{cases} a(ij) & \text{dacă } ij \in E \\ \infty & \text{altfel} \end{cases}$$

Aici ∞ desemnează un număr real mare în raport cu celelalte costuri (de exemplu $\infty > n \times \max_{ij \in E} a(ij)$) şi vom presupune în plus că

$\infty \mp a = \infty,\ \infty + \infty = \infty.$
(Este posibil, de asemenea, ca ∞ să semnifice acces terminat cu insucces în structura de date în care se reprezintă matricea A).

Dacă $i, j \in V$, vom nota cu

$$\mathcal{D}_{ij} = \{D_{ij} \mid D_{ij} \text{ drum în } G \text{ de la } i \text{ la } j\}.$$

Pentru $D_{ij} \in \mathcal{D}_{ij}$

$$D_{ij} : (i =)v_0, v_0v_1, v_1, \ldots, v_{r-1}, v_{r-1}v_r, v_r(= j)$$

mulțimea vîrfurilor este $V(D_{ij}) = \{v_0, v_1, \ldots, v_r\}$ și mulțimea arcelor
$E(D_{ij}) = \{v_0v_1, v_1v_2, \ldots, v_{r-1}v_r\}$.

Orice vîrf $k \neq i, j$ al lui D_{ij}, determină pe D_{ij} două drumuri $D_{ik} \in \mathcal{D}_{ik}$ și $D_{kj} \in \mathcal{D}_{kj}$. Vom nota $D_{ij} = D_{ik} \circ D_{kj}$.

Costul unui drum $D_{ij} \in \mathcal{D}_{ij}$ se definește

$$a(D_{ij}) = 0 + \sum_{ij \in E(D_{ij})} a_{ij}.$$

În particular, $a(D_{ii}) = 0$.

Principalele probleme de drum (de cost) minim care apar în aplicaţii practice (sau sînt utile în rezolvarea altor probleme de optimizare combinatorie) sînt:

P1 Date G digraf;$a : E(G) \to \mathbf{R}$; $s, t \in V(G)$,$s \neq t$.
Să se determine $D^*_{st} \in \mathcal{D}_{st}$, astfel încît
$a(D^*_{st}) = \min\{a(D_{st}) \mid D_{st} \in \mathcal{D}_{st}\}$.

P2 Date G digraf; $a : E(G) \to \mathbf{R}$; $s \in V(G)$.
Să se determine $D^*_{si} \in \mathcal{D}_{si}$ $\forall i \in V(G)$, a.î.
$a(D^*_{si}) = \min\{a(D_{si}) \mid D_{si} \in \mathcal{D}_{si}\}$.

P3 Date G digraf; $a : E(G) \to \mathbf{R}$.
Să se determine $D^*_{ij} \in \mathcal{D}_{ij}$ $\forall i, j \in V(G)$, a.î.
$a(D^*_{ij}) = \min\{a(D_{ij}) \mid D_{ij} \in \mathcal{D}_{ij}\}$.

Observaţii:
1. Cu convenţia folosită în reprezentarea matricilor de cost adiacenţă, se poate considera că $\mathcal{D}_{ij} \neq \emptyset \quad \forall i, j \in V$.

Dacă $a(D_{ij}) < \infty$ atunci D_{ij} este drum (adevărat) în G de la i la j, iar dacă $a(D_{ij}) = \infty$, atunci D_{ij} este drum în digraful complet simetric obţinut din G prin adăugarea arcelor lipsă, cu costul ∞.
Rezultă că toate **mulţimile pe care se consideră minimele** în problemele precedente, sînt **nevide** şi, cum digrafurile considerate sînt finite, rezultă că aceste mulţimi sînt **finite** (în fiecare drum vîrfurile sînt distincte), **deci minimele considerate există**.

2. Algoritmii de rezolvare a problemei (P1) se obţin din algoritmii de rezolvare a problemei (P2) adăugîndu-li-se un test suplimentar (evident) de oprire.

3. Problema (P3) se poate rezolva iterînd un algoritm de rezolvare a problemei (P2). Sînt posibile însă soluţii mai eficiente.

Aplicaţii. Vom schiţa în continuare trei aplicaţii practice posibile ale acestor probleme.

a) $G = (V, E)$ reprezintă o reţea de comunicaţie cu nodurile V şi rutele directe între noduri formînd mulţimea E.

Dacă $a(e)$ reprezintă lungimea arcului e, atunci cele trei probleme de mai sus reprezintă probleme naturale, care se pun în astfel de reţele: *"determinarea drumurilor celor mai scurte"*.

Dacă $p_{ij} \in (0, 1]$ este probabilitatea de funcţionare a arcului $ij \in E$ atunci, presupunînd că arcele funcţionează independent unele de altele, probabilitatea de funcţionare a drumului D este

$$p(D) = \prod_{ij \in E(D)} p_{ij}.$$

Considerînd $a_{ij} = -\log p_{ij}$, problema drumului de cost minim de la s la t semnifică determinarea *drumului cel mai sigur* de la s la t.

b) Reţele *PERT* (Project Evaluation and Review Technique).

Fie $P = \{A_1, \ldots, A_n\}$ mulţimea activităţilor atomice ale unui proiect de anvergură (n este mare). P este o mulţime parţial ordonată cu relaţia de ordine
$A_i < A_j \quad \Leftrightarrow$ activitatea A_j nu poate începe decît după terminarea activităţii A_i.
Se cunoaşte, pentru fiecare activitate A_i timpul de execuţie t_i.

Se cere **să se determine un plan de organizare a proiectului astfel încît timpul total de execuţie să fie minim.** (Notăm că problemele practice sînt mai complexe datorită restricţiilor de utilizare concurentă a resurselor - oameni, utilaje, etc. - de către diversele activităţi).

Ideea generală pe care se bazează pachetele soft care rezolvă astfel de probleme este de a asocia proiectului un digraf aciclic (reţeaua PERT) astfel:

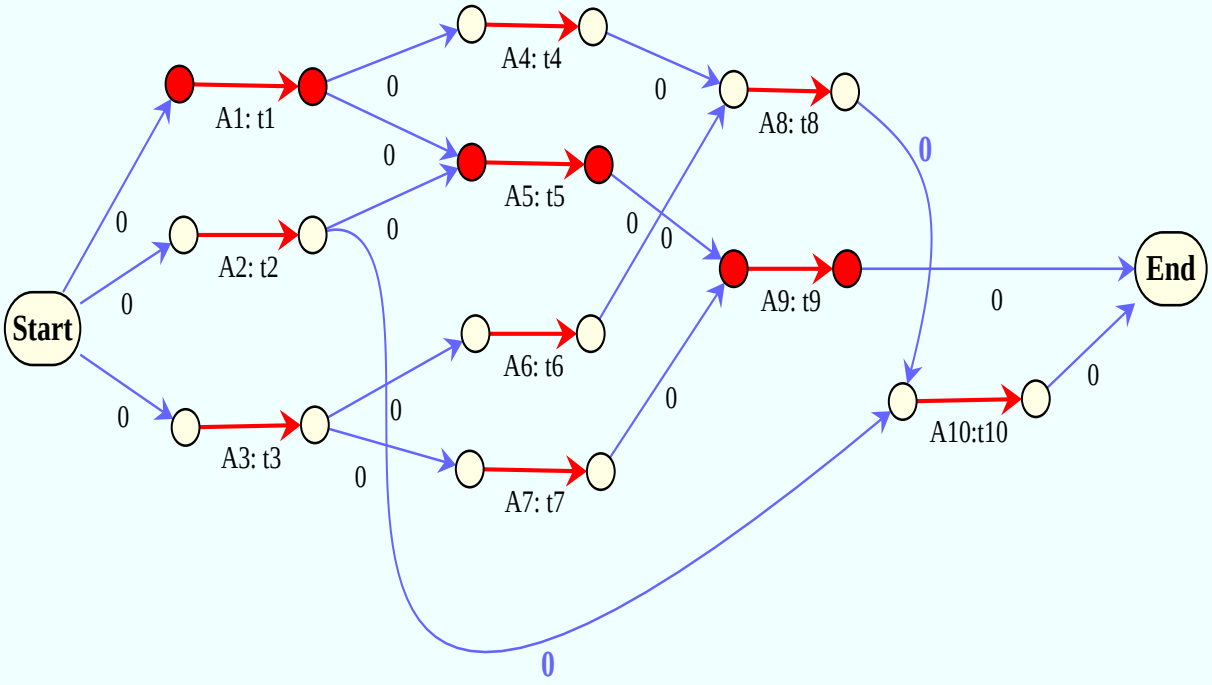
Fiecărei activități A_l i se asociază arcul $i_l j_l$ de cost $a(i_l j_l) = t_l$.
Nodul i_l reprezintă evenimentul de început al activității A_l, iar nodul j_l reprezintă evenimentul de sfîrșit al activității A_l.

Dacă activitatea A_k poate începe imediat după terminarea activității A_l, se introduce în digraf arcul $j_l i_k$ (activitate fictivă) de cost 0.

Se asociază un eveniment s (START) unit prin arce de cost 0 cu elementele minimale ale lui $(P, <)$ și un eveniment t (END) de care vor fi unite prin cîte un arc fiecare element maximal al lui P.

În digraful obținut (care este evident aciclic) costul maxim al unui drum de la s la t reprezintă *cel mai scurt timp posibil de execuție a proiectului.*
Un drum de cost maxim se numește **drum critic**, întrucît întîrzierea oricărei activități corespunzătoare unui arc de pe drumul critic conduce la întîrzierea întregului proiect.

Figura de mai jos ilustrează tipul de digraf (aciclic) care se formează și se evidențiază un posibil drum critic. Dificultatea majoră este în construcția digrafului (modelarea problemei reale) și problema devine extrem de interesantă dacă (dar și NP-dificilă) dacă se introduc și alte tipuri de restricții între activități (nu numai temporale).



c) *Problema rucsacului*. Dispunem de n obiecte de volume $a_1, \ldots, a_n$ și de un rucsac de volum b ($a_i \in \mathbf{Z}_+, i = \overline{1,n}, b \in \mathbf{Z}_+, a_i \leq b \ \forall i = \overline{1,n}$).

Cunoscînd "profitul" $p_i \in \mathbf{R}_+$ adus de introducerea obiectului i în rucsac, se cere **să se determine o încărcare a rucsacului de profit total maxim:**

$$\max\Big\{\sum_{i=1}^{n} p_i x_i \mid \sum_{i=1}^{n} a_i x_i \leq b, x_i \in \{0,1\} \forall i = \overline{1,n}\Big\}.$$

Problema, deşi interesantă în unele aplicaţii (de exemplu la încărcarea vapoarelor într-un port) a fost aleasă pentru a pune în evidenţă legătura dintre metoda programării dinamice (discrete) şi problemele de drum minim într-un digraf.

Considerăm $G = (V, E)$ un digraf cu
$V = \{s\} \cup V_1 \cup V_2 \cup \ldots \cup V_n \cup \{t\}$, unde
$V_i = \{i^0, i^1, \ldots, i^b\}$ este asociat obiectului i, $i = \overline{1,n}$.

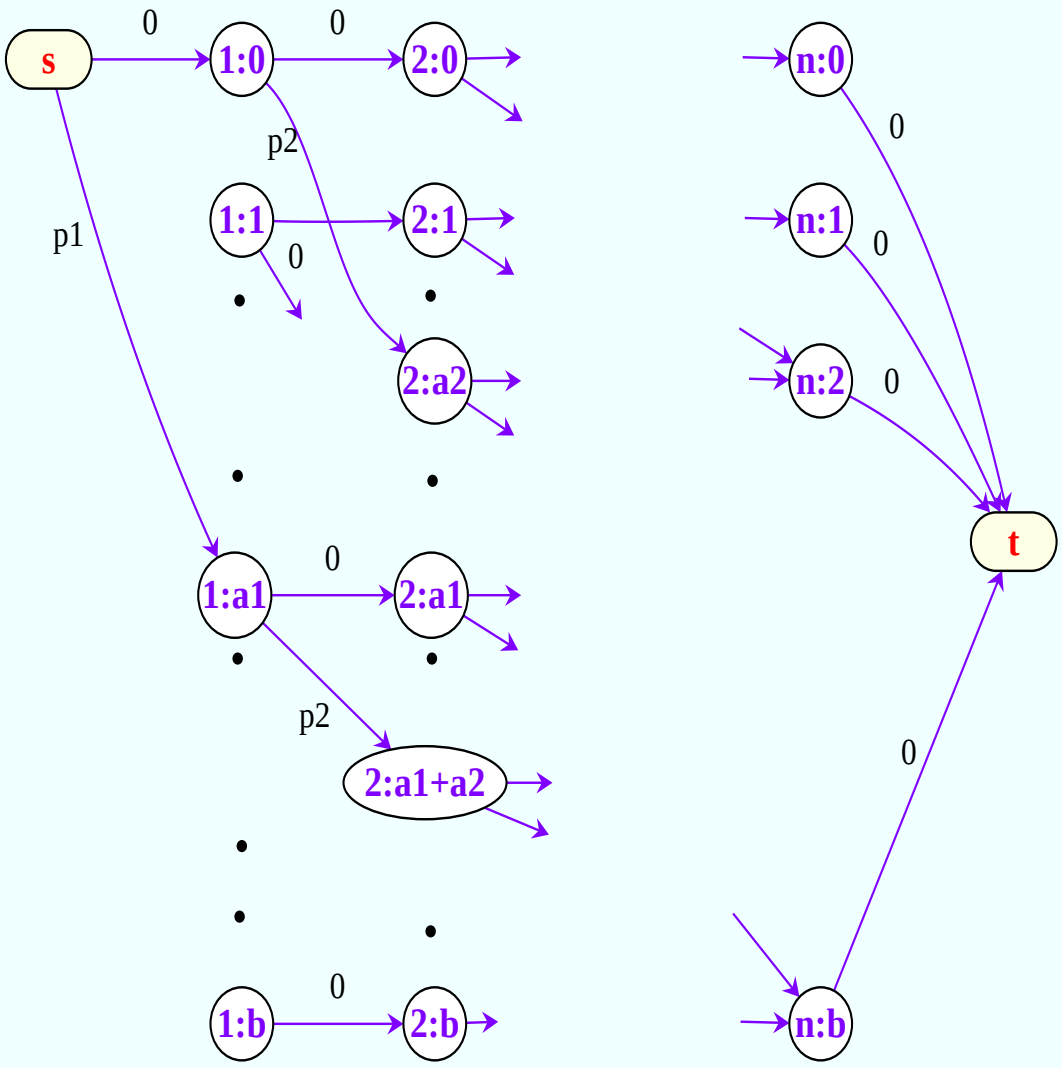
Arcele lui G sînt:

- $s1^0$ și $s1^{a_1}$ cu $a(s1^0) = 0$ și $a(s1^{a_1}) = p_1$.
  (se pune obiectul 1 în rucsac și se obține profitul p_1, ajungându-se la nivelul a_1 de umplere, sau nu se pune obiectul 1 în rucsac, profitul fiind 0 și nivelul de umplere rămânând 0).

- $\forall i = \overline{2, n}$ $\forall j = \overline{0, b}$:
  $(i-1)^j i^j$ cu $a((i-1)^j i^j) = 0$;
  (dacă decidem să nu introducem obiectul i în rucsac, atunci de la încărcarea rucsacului cu primele $i-1$ obiecte se trece la o încărcare cu primele i obiecte în care nu este selectat obiectul i, deci se rămâne pe același nivel de încărcare j, iar profitul ce se va adăuga este 0).
  Dacă $j - a_i \geq 0$ atunci avem și arcul
  $(i-1)^{j-a_i}\ i^j$ cu $a\Big((i-1)^{j-a_i}\ i^j\Big) = p_i$.
  (se poate ajunge la o încărcare j prin introducerea obiectului i de volum a_i la o încărcare a primelor $i-1$ obiecte de nivel $j - a_i$).

- $\forall j = \overline{0, b}$: $n^j t$ cu $a(n^j t) = 0$.

Figura următoare ilustrează construcţia acestui digraf.



Se observă din construcţie, că orice drum de la s la t în G corespunde unei submulţimi de obiecte cu suma volumelor mai mică sau egală cu b şi de profit egal cu costul acestui drum. Reciproc, oricărei mulţimi de obiecte cu suma volumelor nedepăşind b îi corespunde un drum de la s la t în G.

Rezultă că dacă în digraful G se determină un drum de cost maxim de la s la t se rezolvă problema rucsacului.

Notăm că descrierea (statică) a digrafului G poate fi uşor transformată în una procedurală astfel încît digraful să reprezinte doar ilustrarea unei metode de programare dinamincă (prospectivă) pentru rezolvarea problemei rucsacului.

Atenţie ! Problema rucsacului este referită uzual ca una din problemele NP-dificile. Soluţia polinomială descrisă mai sus conduce la un digraf aciclic cu $O(nb)$ vârfuri, care nu-i dimensiunea intrării !!!

Rezolvarea problemei P2

Teoremă. 1. *Fie* $G = (V, E)$ *digraf,* $V = \{1, \ldots, n\}$, $s \in V$ *și* $a : E \to \mathbf{R}$, *astfel încît*

$$(I) \qquad \forall C \text{ circuit în } G, a(C) > 0.$$

Atunci $(u_1, \ldots, u_n)$ **este o soluție a sistemului**

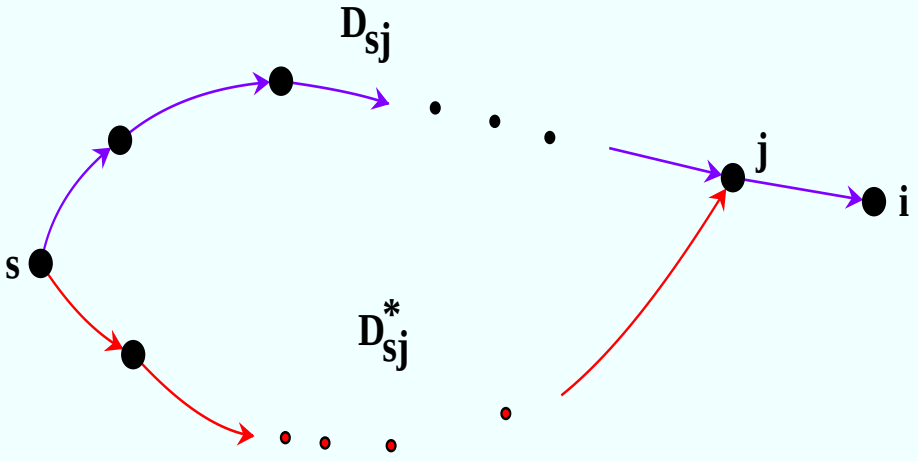
$$(*) \qquad \begin{cases} u_s = 0 \\ u_i = \min\limits_{j \neq i}(u_j + a_{ji}) \quad \forall i \neq s. \end{cases}$$

dacă și numai dacă $\forall i \in V$, $\exists D^*_{si} \in \mathcal{D}_{si}$ *astfel încît* $a(D^*_{si}) = u_i$ *și* $a(D^*_{si}) = \min\{a(D) \mid D \in \mathcal{D}_s i\}$.

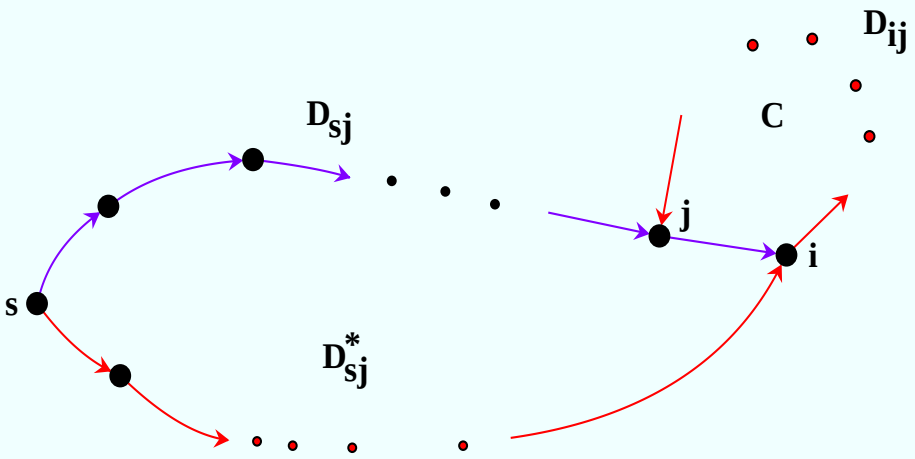
Demonstrație: "$\Leftarrow$" Fie D^*_{si} ($i \in V$) soluții ale problemei (P2) cu $a(D^*_{si}) = \min\{a(D) \mid D \in \mathcal{D}_{si}\}$. Notăm cu $u_i = a(D^*_{si})$ $(i \in V)$.

Ipoteza (I) asigură faptul că $u_s = 0$, adică prima ecuație a sistemului (*) este verificată. Pentru $i \neq s$ drumul D^*_{si} are un penultim vîrf j. Dacă D_{sj} este drumul de la s la j determinat pe D^*_{si} de vîrful j, avem: $u_i = a(D^*_{si}) = a(D_{sj}) + a_{ji} \geq a(D^*_{sj}) + a_{ji} = u_j + a_{ji}$.

Arătăm că $u_i = u_j + a_{ji}$.
Presupunem că $u_i > u_j + a_{ji}$, adică $a(D_{sj}) > a(D^*_{sj})$. Avem două cazuri posibile:
1. $i \notin V(D^*_{sj})$. Atunci $D^1 = D^*_{sj} \circ (j, ji, i) \in \mathcal{D}_{si}$ și $a(D^1) = a(D^*_{sj}) + a_{ji} < a(D_{sj}) + a_{ji} = a(D^*_{si})$, ceea ce contrazice alegerea drumului D^*_{si} (vezi figura următoare).



2. $i \in V(D^*_{sj})$. Fie $D^*_{sj} = D_{si} \circ D_{ij}$ cele două drumuri determinate pe D^*_{sj} de vîrful i. Atunci circuitul $C = D_{ij} \circ (j, ji, i)$ are costul $a(C) = a(D_{ij}) + a_{ji} = a(D^*_{sj}) - a(D_{si}) + a_{ji} = u_j + a_{ji} - a(D_{si}) \leq u_j + a_{ji} - a(D^*_{si}) = u_j + a_{ji} - u_i < 0$, contrazicînd ipoteza (I) (vezi figura următoare).



Deci am demonstrat că $\forall i \neq s \Rightarrow u_i = u_j + a_{ji}$.

Dacă u_i nu satisface (*), atunci ar exista j_1 astfel încât $u_i > u_{j_1} + a_{j_1 i}$. Atunci, ca mai sus, se poate construi un drum de cost mai mic decât u_i de la s la i.

Rezultă că suficiența teoremei este demonstrată.

Notăm că de fapt am dovedit mai sus că $a(D_{sj}) = a(D^*_{sj})$ adică, dacă j este vîrful dinaintea lui i pe un drum de cost minim de la s la i atunci și porțiunea de drum de la s la j este drum de cost minim de la s la j. Inductiv, rezultă :

(Principiul optimalității al lui Bellman) *dacă D^*_{si} este drum de cost minim de la s la i atunci $\forall j \in V(D^*_{si})$, dacă $D^*_{si} = D_{sj} \circ D_{ji}$ atunci D_{sj} (respectiv D_{ji}) sînt drumuri de cost minim de la s la j (respectiv de la j la i).*

"$\Rightarrow$". Dovedim că dacă $(u_1, \ldots, u_n)$ este o soluție a lui (*) atunci

(a) $\exists D_{si} \in \mathcal{D}_{si} : u_i = a(D_{si})$, $\forall i \in V$.
(b) $\forall i \in V\ u_i = \min\{a(D) \mid D \in \mathcal{D}_{si}\} (= a(D^*_{si}))$.

(a) Dacă $i = s$, atunci $u_s = 0$ și drumul D_{ss} satisface $a(D_{ss}) = 0 = u_s$.
Dacă $i \neq s$, considerăm următorul algoritm:

$v \leftarrow i; k \leftarrow 0;$
while $v \neq s$ **do**
{ determină w astfel încît $u_v = u_w + a_{wv}$;
// $\exists w$ pentru că u_v satisface (*)
$i_k \leftarrow v; k++; v \leftarrow w$
}
$i_{k+1} \leftarrow s$

Să observăm că algoritmul determină drumul

$$D : (s =) i_{k+1}, i_{k+1}i_k, \ldots, i_1, i_1i_0, i_0 (= i)$$

cu $D \in \mathcal{D}_{si}$ satisfăcînd $a(D) = a(i_{k+1}i_k) + \cdots + a(i_1i_0) = (u_{i_k} - u_{i_{k+1}}) + (u_{i_{k-1}} - u_{i_k}) + \cdots + (u_{i_0} - u_{i_1}) = u_{i_0} - u_{i_{k+1}} = u_i - u_s = u_i$.

Nu este posibil ca într-o iterație oarecare $w \in \{i_0, \ldots, i_{k-1}\}$, căci atunci s-ar obține un circuit C de cost total 0, contrazicînd ipoteza (I).

Din construcţie, se observă că $u_i = u_{i_1} + a_{i_1 i}$.

(b) Fie $\overline{u}_i = a(D^*_{si})\ \forall i \in V$. Conform primei părţi a demonstraţiei $\overline{u}_i$, $i = \overline{1,n}$, satisfac sistemul $(*)$. Presupunem că $u = (u_1, \ldots, u_n) \neq \overline{u} = (\overline{u}_1, \ldots, \overline{u}_n)$. Cum $u_s = \overline{u}_s = 0$, rezultă că există $i \neq s$ astfel încît $u_i \neq \overline{u}_i$ şi $\forall j \in V(D_{si})$, $j \neq i$, $u_j = \overline{u}_j$, unde D_{si} este drumul construit la (a) pentru $\overline{u}_i$. Atunci avem:
$u_i > \overline{u}_i = \overline{u}_{i_1} + a_{i_1 i} = u_{i_1} + a_{i_1 i}$
(din alegerea lui i)
$\geq u_i$ pentru că u_i satisface (*).

Contradicţia găsită arată că $u = \overline{u}$, deci că u_i reprezintă costuri de drumuri minime.

Observaţii 1. Din demonstraţie rezultă că pentru rezolvarea problemei P2 este suficient să obţinem o soluţie a sistemului $(*)$. Drumurile corespunzătoare se obţin ca la (a).

Algoritmii pe care îi vom prezenta se vor ocupa de rezolvarea sistemului $(*)$. Totuşi, dacă avem $u_i = u_k + a_{ki}$ atunci aşa cum am văzut, k este vîrful dinaintea lui i de pe drumul minim de la s la i de cost u_i.

Rezultă că dacă în algoritmul de rezolvare a lui $(*)$ construim un tablou *înainte*[1..n] cu componente din $V \cup \{0,\}$ cu interpretarea finală "*înainte*[i]=vîrful dinaintea lui i de pe drumul minim de la s la i", atunci vîrfurile acestui drum pot fi determinate în $O(n)$ construind şirul i, *înainte*[i], *înainte[înainte[i]]*,... pînă se depistează vîrful s.

2. Dacă algoritmii de rezolvare a lui $(*)$ vor evita (prin modul de actualizare a vectorului *înainte*) apariţia circuitelor de cost total 0, atunci se observă că,

deşi nu mai are loc unicitatea soluţiei sistemului $(*)$, problema (P2) este rezolvată. Rezultă că aceşti algoritmi vor rezolva problema (P2) în condiţia

(I') $\qquad \forall C$ circuit în $G, a(C) \geq 0$.

3. În cazul grafurilor, rezolvarea problemelor (P1)-(P3) corespunzătoare se poate face utilizînd algoritmii pentru digrafuri, prin înlocuirea fiecărei muchii cu o pereche de arce simetrice de acelaşi cost ca şi muchia pe care o înlocuiesc.
Dificultatea unei astfel de abordări rezultă din introducerea pentru muchii de cost negativ a unor circuite de lungime 2 de cost negativ.

Deci, în cazul grafurilor, algoritmii pentru digrafuri sînt valabili doar dacă toate costurile sînt nenegative.

4. Avînd în vedere că mulţimile $\mathcal{D}_{ij}$ sînt finite, se pot considera probleme analoge problemelor (P1)-(P3) înlocuind min cu max.

Utilizarea ideii uzuale,

$$\max_{x \in A} x = -(\min_{x \in A}(-x))$$

prin înlocuirea costurilor a_{ij} cu $-a_{ij}$ este posibilă doar în cazul digrafurilor în care pentru orice circuit C avem $a(C) \leq 0$.

În particular, această abordare este posibilă în cazul digrafurilor fără circuite (ca în aplicaţiile b) şi c) prezentate).

Dacă digraful iniţial are circuite, problemele de drum de cost maxim se pot dovedi uşor (prin reducerea polinomială la probleme hamiltoniene) a fi NP-dificile.

Rezolvarea problemei (P2) în cazul digrafurilor fără circuite

O **numerotare aciclică** a (vîrfurilor) digrafului $G = (V, E)$ este un vector $ord[v]$ $v \in V$, (cu interpretarea $ord[v]$ = numărul de ordine al vîrfului v) astfel încît

$$\forall vw \in E \Rightarrow ord[v] < ord[w].$$

Are loc următoarea

Lemă. *G este un digraf fără circuite dacă şi numai dacă admite o numerotare aciclică .*

Demonstraţie. Este evident că dacă G admite o numerotare aciclică atunci G nu are circuite (dacă $v_1, v_2, \ldots, v_k, v_1$ sînt vîrfurile unui circuit atunci, cum G are o numerotare aciclică, obţinem $ord[v_1] < ord[v_2] < \ldots < ord[v_k] < ord[v_1]$, contradicţie).

Reciproc, dacă G nu are circuite atunci există un vîrf $v_0 \in V$ astfel încît $d_G^-(v_0) = 0$ (altfel, datoritā finitudinii digrafului, se poate construi un circuit); punem $ord[v_0] \leftarrow 1$, considerām $G \leftarrow G$-v_0 și repetām raționamentul (proprietatea de a nu avea circuite se transmite la subdigrafuri induse).

Aflarea unei numerotări aciclice a unui digraf se numește și **sortare topologică** întrucât se sortează mulțimea V într-un mod compatibil cu "topologia" digrafului.

Vom presupune că digraful este reprezentat cu ajutorul listelor de adiacență. Dimensiunea problemei este $O(n + e)$.
Vom construi un algoritm care să rezolve problema în timp $O(n + e)$.
Acest lucru este posibil datoritā unei utilizări judicioase a structurilor de date.
Linia algoritmului:

- determinăm gradele interioare ale vîrfurilor, parcurgând toate listele de adiacenţă (la întâlnirea lui w în lista de adiacenţă a unui vârf oarecare v se execută $d_G^-(w)$++ ;
- Parcurgem vectorul d_G^- şi vîrfurile de grad interior 0 le memorăm într-o stivă S_0;

(a)- scoatem vîrful din topul stivei S_0 şi-l numerotăm;
(b)- scădem 1 din gradele interioare ale vîrfurilor din lista de adiacenţă a vîrfului tocmai numerotat(''îl scoatem din digraf'') ;
(c)- în modificarea anterioară, crearea unui vîrf de grad interior 0 va implica memorarea lui în stiva S_0;
(d)- reluăm secvenţa $(a) - (c)$ pînă cînd stiva devine vidă.

Dacă nu s-au numerotat toate vîrfurile rezultă că digraful conţine circuite; în cazul epuizării vîrfurilor, s-a obţinut numerotarea aciclică dorită (sortarea topologică).

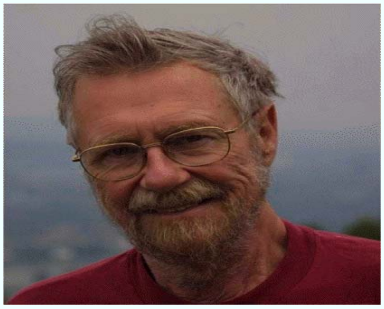
Revenim la rezolvarea problemei P2.

În acest caz, după sortarea topologică a digrafului, vom că considera vârfurile sunt ordonate conform numerotării aciclice(vârful 1 numerotat cu 1, vârful 2 numerotat cu 2 etc.) și în plus vârful s din enunțul problemei **P2** este vârful 1(pentru că nu vor exista drumuri de la vârful s la vârfuri cu număr de ordine mai mic decât el). În aceste ipoteze sistemul $(*)$ se poate rezolva prin substituție:

1. Sortează topologic G; $\{O(n+e)$ operații $\}$
2. $u_1 \leftarrow 0$; *înainte*[1] $\leftarrow 0$;
   for $i := 2$ **to** n **do**
   { $u_i \leftarrow \infty$; *înainte*[i] $\leftarrow 0$;
   for $j := 1$ **to** $i-1$ **do**
   if $u_i > u_j + a_{ji}$ **then**
   { $u_i \leftarrow u_j + a_{ji}$;
   înainte[i] $\leftarrow j$
   }
   }

Complexitatea pasului 2 este, evident $O(1+2+\cdots+n-1) = O(n^2)$.

Rezolvarea problemei (P2) în cazul costurilor nenegative.



Algoritmul lui Dijkstra

Dacă $a_{ij} \geq 0 \ \forall ij \in E$, atunci condiția (I') este îndeplinită și o soluție a sistemului $(*)$ se poate obține cu ajutorul următorului algoritm (Dijkstra, 1961).

Se consideră $S \subseteq V$ astfel încît pe parcursul algoritmului are loc
$(\mathbf{D})$:

$$\begin{cases} \forall i \in S & u_i = \min\{a(D_{si}) \mid D_{si} \in \mathcal{D}_{si}\} \\ \forall i \in V \setminus S & u_i = \min\{a(D_{si}) \mid D_{si} \in \mathcal{D}_{si}, V(D_{si}) \setminus S = \{i\}\} \end{cases}$$

Dacă se reușește construirea lui S astfel încît $S = V$, atunci problema e rezolvată.
Inițial, se va considera $S = \{s\}$ și în $n-1$ pași se adaugă la S cîte un nou vîrf din V.

Algoritmul lui Dijkstra

1. $S \leftarrow \{s\}$; $u_s \leftarrow 0$; *înainte*$[s] \leftarrow 0$;
   for $i \in V \setminus \{s\}$ **do**
   { $u_i \leftarrow a_{si}$; *înainte*$[i] \leftarrow s$ }
   // după aceste inițializări (D) are loc
2. **while** $S \neq V$ **do**
   {
   determină $j^* \in V \setminus S : u_{j^*} = \min\{u_j \mid j \in V \setminus S\}$;
   $S :\leftarrow S \cup \{j^*\}$;
   for $j \in V \setminus S$ **do**
   if $u_j > u_{j^*} + a_{j^*j}$ **then**
   { $u_j \leftarrow u_{j^*} + a_{j^*j}$; *înainte*$[j] \leftarrow j^*$ }
   }

Corectitudinea algoritmului va rezulta dacă vom arăta că, dacă înaintea unei iterații din pasul 2 are loc (D), atunci, după execuția acelei iterații, (D) are loc de asemenea.
Arătăm mai întîi că în ipoteza că (D) are loc, atunci adăugarea lui j^* la S nu contrazice (D). Deci trebuie dovedit că dacă $u_{j^*} = \min\{u_j \mid j \in V \setminus S\}$ atunci $u_{j^*} = \min\{a(D_{sj^*}) \mid D_{sj^*} \in \mathcal{D}_{sj^*}\}$.

Presupunem că există $D^1_{sj^*} \in \mathcal{D}_{sj^*}$ astfel încît $a(D^1_{sj^*}) < u_{j^*}$.
Cum S satisface (D), avem
$u_{j^*} = \min\{a(D_{sj^*}) \mid D_{sj^*} \in \mathcal{D}_{sj^*}, V(D_{sj^*}) \setminus S = \{j^*\}\}$. Rezultă că $V(D^1_{sj^*}) \setminus S \neq \{j^*\}$.
Fie k primul vîrf al drumului $D^1_{sj^*}$ (în parcurgerea sa din s) astfel încît $k \notin S$.
Atunci $a(D^1_{sj^*}) = a(D^1_{sk}) + a(D^1_{kj^*})$.
Din alegerea lui k, avem $V(D^1_{sk}) \setminus S = \{k\}$ și cum (D) are loc, avem $a(D^1_{sk}) = u_k$. Obținem $u_{j^*} > a(D^1_{sj^*}) = u_k + a(D^*_{kj}) \geq u_k$ (costurile sînt nenegative), ceea ce contrazice alegerea lui j^*.
Contradicția obținută arată că, după atribuirea $S := S \cup \{j^*\}$, prima parte a condiției (D) are loc.

Pentru ca și cea de-a doua parte a condiției (D) să aibă loc după această atribuire, să observăm că $\forall j \in V \setminus (S \cup \{j^*\})$ avem
$\min\{a(D_{sj}) \mid D_{sj} \in \mathcal{D}_{sj}, V(D_{sj}) \setminus (S \cup \{j^*\}) = \{j\}\} = \min\Big(\min\{a(D_{sj}) \mid D_{sj} \in \mathcal{D}_{sj}, V(D_{sj}) \setminus S = \{j\}\}, \min\{a(D_{sj}) \mid D_{sj} \in \mathcal{D}_{sj}, V(D_{sj}) \setminus S = \{j, j^*\}\Big)$.

Cum (D) are loc, primul din cele două minime de mai sus este u_j. Fie α_j valoarea celui de-al doilea minim și fie D^1_{sj} drumul pentru care se realizează. Cum $j^* \in V(D^1_{sj})$, avem $\alpha_j = a(D^1_{sj^*}) + a(D^1_{j^*j})$.

Întrucît $S \cup \{j^*\}$ satisfce prima parte a lui (D), avem $a(D^1_{sj*}) = u_{j^*}$ (altfel s-ar contrazice alegerea lui D^1_{sj} înlocuind în D^1_{sj}, porțiunea D^1_{sj*} cu un drum de cost mai mic). Deci $\alpha_j = u_{j^*} + a(D^1_{j^*j})$.

Dacă drumul $D^1_{j^*j}$ este de lungime 1 atunci avem $\alpha_j = u_j + a_{j^*j}$.
Altfel, considerînd k vîrful dinaintea lui j de pe drumul D^1_{sj} avem $k \neq j^*, k \in S$ și $\alpha_j = a(D^1_{sk}) + a_{kj}$. Cum $S \cup \{j^*\}$ satisface prima parte a lui (D), obținem $\alpha_j = u_k + a_{kj}$.

Întrucît S satisface (D), u_k este costul unui drum minim de la s la k cu vîrfurile conținute în S deci α_j este costul unui drum de la s la j cu vîrfurile conținute în S. Rezultă că $\alpha_j \geq u_j$, căci S satisface (D).

Am obținut că singurul caz în care $\alpha_j < u_j$ este atunci cînd $\alpha_j = u_{j^*} + a_{j^*j} < u_j$, situație testată în ciclul **for** al pasului 2.

Rezultă că (D) are loc pe tot parcursul algoritmului și deci valorile finale ale variabilelor u_i reprezintă soluția sistemului $(*)$. Evident, tabloul *înainte* este actualizat pentru memorarea implicită a drumurilor de cost minim.

Complexitatea timp a algoritmului, în descrierea dată este $O(n^2)$ datorită selectării minimelor din pasul 2.

Este posibilă organizarea unor cozi cu prioritate (de exemplu heap-urile) pentru a memora valorile u_i, $i \in U = V \setminus S$, astfel încît extragerea minimului să se facă în $O(1)$, iar actualizările necesare în pasul 2 să se facă în timpul total de $O(m \log n)$ unde $m = |E|$ (executându-se $O(m)$ descreșteri de valori u_i, fiecare necesitând $O(\log n)$ operații; *Johnson ,1977).*

Cea mai bună implementare se obţine utilâzând **heap-uri Fibonacci**, ceea ce conduce la o complexitate timp de $O(m + n \log n)$ (*Fredman şi Tarjan, 1984)*.

O *pădure cu rădăcini* (rooted forest) este un **digraf aciclic** $D = (V, A)$ cu proprietatea că **fiecare vârf are gradul interior cel mult 1.**
Vârfurile de grad interior 0 sunt *rădăcinile* lui D, iar cele cu grad exterior 0 sunt *frunzele* lui D.
Dacă $uv \in A$ atunci u este *părintele* lui v iar v este *copilul* lui u.
Dacă pădurea are o singură rădăcină, atunci ea este un *arbore cu rădăcină*.

O **pădure Fibonacci** este o pădure cu rădăcini $F = (V, A)$ în care copiii fiecărui vârf v pot fi ordonaţi astfel încât **copilul numărul** i **are** la rândul său **cel puţin** $i - 2$ **copii.**

Teoremă. *Într-o pădure Fibonacci $F = (V, A)$ fiecare vârf are cel mult $1 + 2\log|V|$ copii.*

Dem. Notăm cu $\sigma(v)$ numărul vârfurilor accesibile din v în F (ordinul subarborelui cu rădăcina v).
Arătăm că $\sigma(v) \geq 2^{(d^+(v)-1)/2}$, care va implica prin logaritmare afirmaţia din enunţul teoremei.

Se observă că inegalitatea precedentă are loc pentru v frunză, aşa că utilizăm un raţionament inductiv.
Fie $k = d^+(v)$ şi fie v_i copilul numărul i al lui v $(i = 1, \ldots, k)$.

Avem, $\sigma(v_i) \geq 2^{(d^+(v_i)-1)/2} \geq 2^{(i-1)/2}$, întrucât $d^+(v_i) \geq i - 2$.
Deci $\sigma(v) = 1 + \sum_{i=1}^{k} \sigma(v_i) \geq 1 + \sum_{i=1}^{k} 2^{(i-3)/2} \geq \cdots \geq 2^{(k-1)/2}$, şi teorema e demonstrată.

Un **heap Fibonacci** conținând valorile reale $(u_j; j \in U)$ este o pădure Fibonacci $F = (U, A)$ (fiecare vârf j are ordonați copii astfel încât copilul numărul i are cel puțin $i-2$ copii) în care este precizată o mulțime $T \subset U$ astfel încât:

(i) dacă $jk \in A$ atunci $u_j \leq u_k$;
(ii) dacă h este copilul numărul i al lui j și $h \notin T$ atunci h are cel puțin $i-1$ copii;
(iii) dacă j_1 și j_2 sunt două rădăcini distincte atunci $d^+(j_1) \neq d^+(j_2)$.

Teorema anterioară ne asigură că numărul rădăcinilor nu va depăși $2 + 2\log|U|$.

Heapul Fibonacci va fi reprezentat cu ajutorul următoarei structuri de date:
- câte o listă dublu înlănțuită C_j a copiilor fiecărui $j \in U$;
-funcția $p : U \to U$, unde $p(j) =$ părintele lui j (dacă j e rădăcină $p(j) = j$);

-funcția $d^+ : U \to \mathbf{N}$;
-funcția $b : \{0, \ldots, t\} \to U$ (cu $t = 1 + \lceil 2 \log |U| \rceil$) cu proprietatea că $b(d^+(j)) = j$ pentru fiecare rădăcină j;
-funcția $l : U \to \{0, 1\}$ cu $l(j) = 1$ dacă și numai dacă $j \in T$.

Teoremă. *Pentru găsirea și ștergerea de n ori a unui j care minimizează u_j și descreșterea de m ori a unei valori u_j, structura de date poate fi actualizată în timpul $O(m + p + n \log p)$, unde p este numărul de vârfuri din pădurea inițială.*

Dem. Pentru găsirea unui j care minimizează u_j este suficient să parcurgem $u_{b(i)}$ pentru $i = 0, \ldots, t$, deci în $O(\log p)$. Un astfel de element j (cu u_j minim) se poate șterge astfel:
-fie $v_1, \ldots, v_k$ copii lui j;
-ștergem j și arcele ce ies din j din pădure;
-acum $v_1, \ldots, v_k$ au devenit rădăcini, iar condițiile (i) și (ii) nu-s afectate;

-pentru repararea condiţiilor (iii) se execută pentru fiecare $r = v_1, \ldots, v_k$:
repară(r): dacă $\exists s$ rădăcină cu $d^+(r) = d^+(s)$ atunci: dacă $u_r \leq u_s$, adaugă s ca ultim copil al lui r şi *repară(r)*, altfel ($u_r > u_s$), adauă r ca ultim copil al lui s şi *repară(s)*.

În acest fel condiţiile (i) şi (iii) sunt menţinute, iar existenţa rădăcinii s de mai sus, se face cu ajutorul funţiilor b, d şi p (în timpul procesului, se actualizează structura de date).

Descreşterea unei valori u_j pentru un $j \in U$ se face astfel:
declară rădăcină(j):
dacă j are un părinte, fie acesta v, atunci
se şterge arcul vj şi se aplică *repară(r)*;
dacă $v \notin T$ se adaugă v la T, altfel se scoate v din T şi se aplică *declară rădăcină(v):*

Notăm cu $incr(..)$ și $decr(..)$ numărul creșterilor, respectiv descreșterilor lui .. în timpul operațiilor din enunțul teoremei. Avem:

numărul de apeluri ale lui *declară rădăcină*=
$decr(u_j)+decr(T) \leq$
$\leq decr(u_j)+incr(T)+p \leq 2decr(u_j)+p = 2m+p$,
deoarece creștem T cel mult o dată după ce a descrescut un u_j.

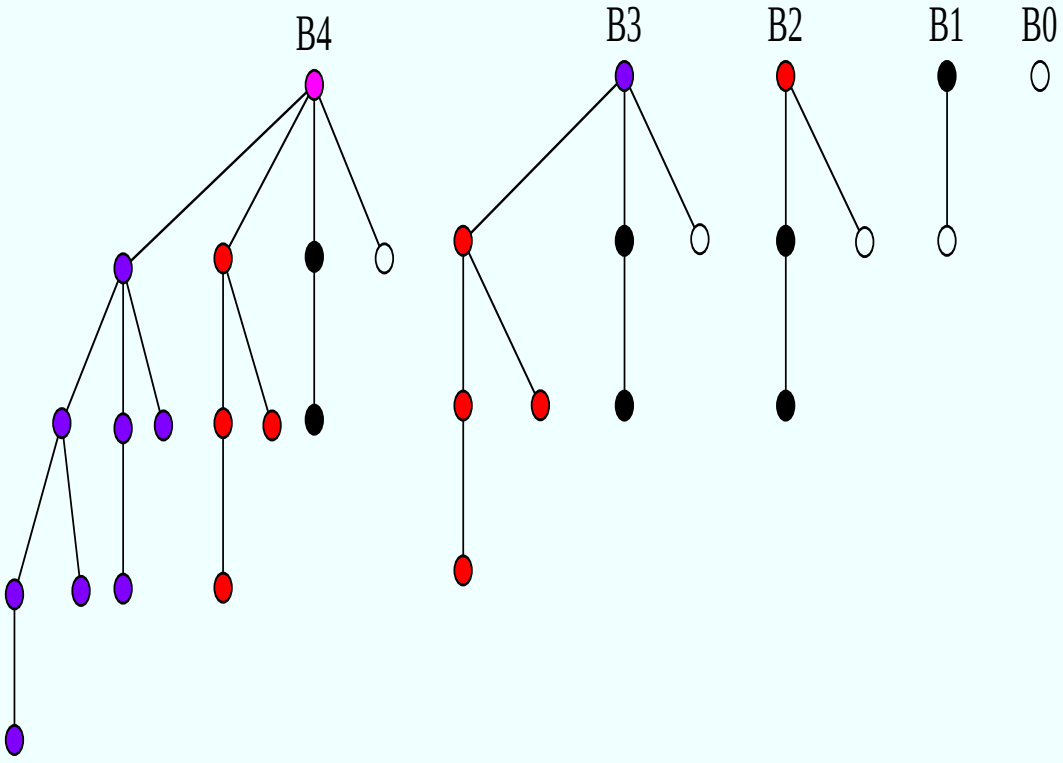
Dacă R este mulțimea rădăcinilor, avem:

numărul de apeluri ale lui *repară*=
$decr(A)+decr(T) \leq$
$\leq decr(A)+incr(R)+p = 2decr(A)+p$
$\leq 2(n \log p +$ numărul de apeluri ale lui *declară rădăcină*$)+p$
$\leq 2(n \log p + 2m + p) + p$.

Cum decizia dacă să se apeleze una sau alta dintre cele două funcții se face în $O(1)$, rezultă că algoritmul are complexitatea $O(m+p+n \log p)$ și teorema e demonstrată.

Corolar. *Algoritmul lui Dijkstra pentru rezolva rea problemei* **P2** *se poate imlementa cu ajutorul heap-urilor Fibonacci în complexitatea timp* $O(m + n \log n)$.

Demonstraţia rezultă din teorema precedentă şi din următoarea figură care indică un mod de construcţie a heap-ului iniţial (binomial):



Dacă se dorește rezolvarea problemei (P1) cu ajutorul algoritmului lui Dijkstra, atunci, la introducerea lui t în S, se poate opri algoritmul. Complexitatea, în cazul cel mai nefavorabil, rămîne aceeași. Totuși, în situaţii practice concrete există posibilitatea de a grăbi introducerea lui t în S utilizînd o funcţie de dirijare a procesului de construcţie a lui S.

O funcţie $g : V \to \mathbf{R}_+$ se numește **estimator consistent** dacă
(i) $\forall i \in V \;\; u_i + g(i) \leq \min\{a(D_{st}) \mid D_{st} \in \mathcal{D}_{st}$ și $i \in V(D_{st})\}$;
(ii) $\forall ij \in E \;\; g(i) \leq a_{ij} + g(j)$.

Să observăm că $g(i) = 0 \;\forall i$ este un estimator consistent (trivial).
Dacă însă $V(G)$ este o mulţime de puncte din plan, atunci $g(i)$=distanţa (euclidiană) de la i la t este un estimator consistent, dacă sînt satisfăcute condiţiile (ii).

Dacă g este un estimator consistent atunci se poate modifica alegerea lui j^* în algoritm astfel: $u_{j^*} + g(j^*) = \min\{u_j + g(j) \mid j \in V \setminus S\}$.
Algoritmul rămîne valabil (demonstrația este identică situației $g(i) = 0\ \forall i$ și se folosește (ii) repetat).
Avantajul este acela că se vor introduce în S vîrfuri care să ne apropie de t.

În implementarea care rezultă din descrierea algoritmului lui Dijkstra, s-a presupus că se dispune de matricea de cost-adiacență a digrafului. În cazul digrafurilor cu multe vîrfuri (în care, de exemplu, listele de adiacență sînt memorate în memoria secundară), sau în cazul digrafurilor date funcțional (se dispune de o procedură care construiește pentru un vîrf dat, lista sa de adiacență) această implementare este neeficientă, respectiv neaplicabilă. O implementare care nu are aceste deficiențe este următoarea datorată lui *Glover, Klingman și Philips (1985)*

Partition Shortest Path (PSP) algorithm:

1. $u_s \leftarrow 0$; *înainte*$(s) \leftarrow 0$;
   $S \leftarrow \emptyset$; $NOW \leftarrow \{s\}$; $NEXT \leftarrow \emptyset$;
2. **while** $NOW \cup NEXT \neq \emptyset$ **do**
   { **while** $NOW \neq \emptyset$ **do**
   { Extrage i din NOW;
   $S \leftarrow S \cup \{i\}$;
   $L \leftarrow N_G^+(i)$;// se generează în L, lista de adiacență și costurile corespunzătoare
   for $j \in L$ **do**
   if $j \notin NOW \cup NEXT$ **then**
   { $u_j \leftarrow u_i + a_{ij}$; *înainte*$(j) \leftarrow i$;
   introdu j în NEXT
   }
   else if $u_j > u_i + a_{ij}$ **then**
   { $u_j \leftarrow u_i + a_{ij}$
   înainte$(j) \leftarrow i$
   }
   }
   if $NEXT \neq \emptyset$ **then**
   { determină $d = \min\{u_i \mid i \in NEXT\}$;
   transferă $\forall i \in NEXT$ cu $u_i = d$ în NOW
   }
   }

Rezolvarea problemei (P2) în cazul general.

Dacă există $ij \in E$ astfel încît $a_{ij} < 0$, algoritmul lui Dijkstra nu mai este valabil în general (introducerea lui j^* în S poate conduce la violarea condiţiei (D)).
Considerînd îndeplinită condiţia (I') vom rezolva sistemul $(*)$ prin aproximaţii succesive.

Considerăm $\forall i \in V$ şi $\forall m = \overline{1, n-1}$
(BM)
$u_i^m = \min\{a(D) \mid D \in \mathcal{D}_{si}, \text{nr arcelor lui D este} \leq m\}$.

Cum orice drum în G are cel mult $n-1$ arce rezultă că dacă reuşim construcţia lui

$\mathbf{u^1} = (u_1^1, \ldots, u_n^1)$, $\mathbf{u^2} = (u_1^2, \ldots, u_n^2)$, ...,
$\mathbf{u^{n-1}} = (u_1^{n-1}, u_2^{n-1}, \ldots, u_n^{n-1})$, atunci $\mathbf{u^{n-1}}$ este soluţia sistemului $(*)$.

Algoritmul care rezultă este următorul:

Algoritmul lui *Bellman, Ford, Moore* ($\sim$ 1960)

1. $u_s^1 \leftarrow 0$; **for** $i \in V \setminus \{s\}$ **do** $u_i^1 \leftarrow a_{si}$;
   // evident (BM) are loc
2. **for** $m := 1$ **to** $n-2$ **do**
   for $i := 1$ **to** n **do**
   $u_i^{m+1} \leftarrow \min\left(u_i^m, \min_{j \neq i}(u_j^m + a_{ji})\right)$

Pentru a demonstra corectitudinea algoritmului, arătăm că dacă $u^m (m \geq 1)$ satisface (BM) atunci și u^{m+1} o satisface. Fie $i \in V$ și considerăm mulțimile de drumuri:

$A = \{D \mid D \in \mathcal{D}_{si}, \text{numărul arcelor lui } D \leq m+1\}$.

$B = \{D \mid D \in \mathcal{D}_{si}, \text{numărul arcelor lui } D \leq m\}$.

$C = \{D \mid D \in \mathcal{D}_{si}, \text{numărul arcelor lui } D = m+1\}$.

Atunci $A = B \cup C$ și

$\min\{a(D) \mid D \in A\} = \min(\min\{a(D) \mid D \in B\}, \min\{a(D) \mid D \in C\})$

Cum u^m satisfac (BM), rezultă că
$\min\{a(D) \mid D \in A\} = \min(u_i^m, \min\{a(D) \mid D \in C\})$.
Fie $\min\{a(D) \mid D \in C\} = a(D^0)$, $D^0 \in C$.
Dacă j este vîrful ce-l precede pe i în D^0 (există, întrucît D^0 are măcar 2 arce) atunci
$a(D^0) = a(D^0_{sj}) + a_{ji} \geq u_j^m + a_{ji}$,
întrucît D^0_{sj} are m arce și u^m satisface (BM).

Rezultă că
$\min\{a(D) \mid D \in A\} = \min\{u_i^m, \min_{j \neq i}(u_j^m + a_{ji})\}$ valoare care în algoritm se atribuie lui u_i^{m+1}.

Observăm că algoritmul are complexitatea de $O(n^3)$ dacă determinarea minimului din pasul 2 necesită $O(n)$ operații.

Determinarea drumurilor minime se face menținînd vectorul *înainte*, inițializat în mod evident în pasul 1 și actualizat corespunzător, la stabilirea minimului din pasul 2.

Observaţii:

1. Dacă la algoritm se adaugă şi pasul 3:

3. **if** ($\exists i \in V$ a.î. $u_i^{n-1} > \min_{j \neq i}(u_j^{n-1} + a_{ji})$)
then"există circuit de cost negativ ".

se obţine posibilitatea testării în $O(n^3)$ a existenţei unui circuit C de cost negativ în digraful G (altfel, din demonstraţia corectitudinii algoritmului ar trebui să nu se poată micşora u_i^{n-1}).

Depistarea circuitului C se face simplu ($O(n)$) utilizînd vectorul *înainte*.

2. Dacă există $k < n-1$ astfel încît $\mathbf{u^k = u^{k+1}}$ atunci algoritmul se poate opri. Mai mult, se poate obţine o implementare a acestui algoritm, care să aibă complexitatea $O(nm)$, folosind o coadă UQ în care se vor păstra vîrfurile i cărora li se modifică u_i curent (se va renunţa, evident, la memorarea tuturor aproximaţiilor succesive).

Rezolvarea problemei (P3).

Considerām
$u_{ij} = \min\{a(D_{ij}) \mid Dij \in \mathcal{D}_{ij}\}\ \forall i, j \in V.$

Problema se reduce la determinarea matricii $U = (u_{ij})_{n\times n}$, atunci cînd se cunoaşte A matricea de cost-adiacentă.

Drumurile de cost minim vor fi obţinute în $O(n)$ dacă odată cu determinarea matricii U se va construi matricea
Înainte=(înainte(i,j))$_{n\times n}$ cu elementele avînd semnificaţia
înainte(i,j)=vîrful dinaintea lui j de pe drumul de cost minim de la i la j în G.

Să observăm că dacă $a_{ij} \geq 0\ \forall ij$, atunci, iterînd algoritmul lui Dijkstra pentru $s \in \{1, \ldots, n\}$, se obţine un algoritm de complexitate $O(n^3)$.

Dacă G nu conține circuite de cost negativ, dar există și arce de cost negativ, iterînd algoritmul lui Bellman Ford pentru $s = \overline{1,n}$ se obține un algoritm de complexitate $O(n^4)$.

Arătăm în continuare că se poate proceda și mai eficient.

Soluţia I^a.
Fie $\alpha : V \to \mathbf{R}$ a. î. $\forall ij \in E\ \alpha(i) + a_{ij} \geq \alpha(j)$.
Considerăm $\overline{a} : E \to \mathbf{R}_+$ dată de
$\overline{a}_{ij} = a_{ij} + \alpha(i) - \alpha(j)$, $\forall ij \in E$.
Avem $\overline{a}_{ij} \geq 0$ și, în plus, oricare ar fi $D_{ij} \in \mathcal{D}_{ij}$,

$$\overline{a}(D_{ij}) = a(D) + \alpha(i) - \alpha(j). \tag{2}$$

Rezultă că se poate itera algoritmul lui Dijkstra pentru obținerea drumurilor de cost $\overline{a}$ minim și din relația (2) se observă că un drum este de cost $\overline{a}$ minim dacă și numai dacă este drum de cost a minim. Rezultă următorul algoritm:

1. Determină α și construiește $\overline{A}$.
2. Rezolvă (P3) pt. $\overline{A}$ construind $\overline{U}$ și *Înainte.*
3. Determină U ($u_{ij} := \overline{u}_{ij} - \alpha(i) + \alpha(j)\ \forall ij$).

Pasul 2 al algoritmului necesită $O(n)^3$) operaţii prin iterarea algoritmului lui Dijkstra.

Pasul 1 se poate realiza în timpul $O(n^3)$, fixînd $s \in V$ și rezolvînd (P2) cu alg. Bellman-Ford.

În adevăr, dacă (u_i, $i \in V$) este soluţie a lui (P2), atunci (u_j, $j \in V$) este soluţie a sistemului $(*) \Rightarrow u_j = \min_{i \neq j}\{u_i + a_{ij}\}$, adică $\forall ij \in E$ $u_j \geq u_i + a_{ij}$, sau, $a_{ij} + u_i - u_j \geq 0$.
Deci, se poate considera $\alpha(i) = u_i\ \forall i \in V$.

Soluţia a II^a.
Fie
$u_{ij}^m = \min\{a(D_{ij}) \mid D_{ij} \in \mathcal{D}_{ij}, V(D_{ij}) \setminus \{i,j\} \subseteq \{1,2,\ldots,m-1\}\}\ \forall i,j \in \{1,2,\ldots,n\}, m = \overline{1,n+1}$.

Atunci, evident $u_{ij}^1 = a_{ij}\ \forall i, j \in V$ (presupunem matricea A având elementele diagonale egale cu 0). În plus,

$$u_{ij}^{m+1} = \min\{u_{ij}^m, u_{im}^m + u_{mj}^m\}\ \forall i, j \in V,\ m = 1, \ldots, n.$$

Această ultimă relație se poate justifica inductiv: un drum de cost minim de la i la j care nu are vîrfuri interioare $\geq m$ poate să nu conțină vîrful m, și atunci are costul u_{ij}^m, sau poate conține vîrful m, și atunci, din principiul optimalității al lui Bellman și ipoteza inductivă, este $u_{im}^m + u_{mj}^m$.

Evident, dacă se obține $u_{ii}^m < 0$ atunci digraful conține un circuit de cost negativ C care trece prin vîrful i, cu $V(C) \setminus \{i\} \subseteq \{1, \ldots, m-1\}$.

Această soluție a problemei (P3) este cunoscută ca algoritmul lui **Floyd-Warshal** și poate fi descris astfel:

```
1:  for i := 1 to n do
    for j := 1 to n do
    {   înainte(i, j) ← i;
        if i = j then { a_ii ← 0;înainte(i, i) ← 0 }
    }

2:  for m := 1 to n do
    for i := 1 to n do
    for j := 1 to n do
        if a_ij > a_im + a_mj then
        {   a_ij ← a_im + a_mj;
            înainte(i, j) ←înainte(m, j)
            if (i = j ∧ a_ij < 0) then
                return "circuit negativ"
        }
```

Evident, complexitatea algoritmului este de $O(n^3)$.

Observaţie. Dacă digraful nu conţine circuite de cost negativ, atunci iniţializînd $a_{ii} \leftarrow \infty$, valorile finale ale elementelor diagonale dau costul minim al unui circuit ce trece prin vîrful corespunzător.

Soluţia a III^a. Considerām $a_{ii} = 0\ \forall i \in V$ (G nu conţine circuite de cost < 0).
Iterarea algoritmului lui Bellman Ford corespunde următoarei abordări. Fie
$u_{ij}^m = \min\{a(D_{ij}) \mid D_{ij} \in \mathcal{D}_{ij}, D_{ij}$ are cel mult m arce$\}$
$\forall i, j \in V$, $\forall m = 1, 2, \ldots, n-1$.

Dacă notām $U^m = (u_{ij}^m)$ cu $m \in \{0, 1, 2 \ldots, n-1\}$, unde U^0 are toate elementele ∞ cu excepţia celor de pe diagonală care-s egale cu 0 atunci, iterarea algoritmului lui Bellman Ford revine la:

1. **for** $i, j \in V$ **do** $u_{ij}^0 \leftarrow \infty$ **if** $i \neq j$ **else** 0;
2. **for** $m := 0$ **to** $n-2$ **do**
   for $i, j \in V$ **do** $u_{ij}^{m+1} = \min_k(u_{ik}^m + a_{kj})$;

(în minimul anterior, comparaţia cu u_{ij}^m din algoritmul Bellman Ford, se realizează pentru $k = j$, şi utilizînd ipoteza că $a_{jj} = 0$). Întregul proces de calcul se poate rescrie matricial dacă se consideră următorul produs pe mulţimea matricilor pătrate cu elemente reale:

$$\forall B, C \in \mathcal{M}_{n\times n} \quad B \otimes C = P = (p_{ij})$$

unde, $p_{ij} = \min_{k=\overline{1,n}}(a_{ik} + b_{kj})$.

Se observă că, dacă se folosește determinarea uzuală a minimului, atunci calculul matricii P este similar înmulțirii uzuale a matricilor. În plus operația $\otimes$ este asociativă.

Cu aceste notații avem
$U^{m+1} = U^m \otimes A$ și inductiv rezultă că

$U^1 = A, U^2 = A^{(2)}, \ldots, U^{n-1} = A^{(n-1)}$
unde $A^{(k)} = A^{(k-1)} \otimes A$ și $A^{(1)} = A$.

În ipoteza că graful nu are circuite de cost negativ, atunci $A^{(2^k)} = A^{(n-1)} \quad \forall k : 2^k \geq n - 1$.

Rezultă că determinarea succesivă a matricilor $A, A^{(2)}, A^{(4)} = A^{(2)} \otimes A^{(2)}, \ldots$ conduce la un algoritm de complexitate $O(n^3 \log n)$ pentru rezolvarea problemei (P3)

(desigur, matricea *Înainte* se va obține în $O(n^3)$ operații ca în demonstrația teoremei 1, după determinarea lui U^{n-1}.)

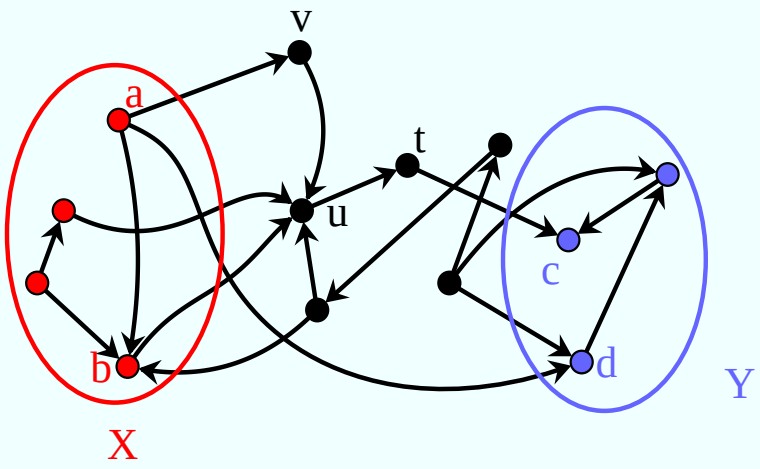
Dacă "produsul" matricial considerat se face cu algoritmi mai performanți atunci se obține o rezolvare eficientă a problemei (n^3 din evaluarea precedentă se poate înlocui cu $n^{\log_2 7} = n^{2,81}$(*Strassen* 1969); sau chiar cu $n^{2,38}$ *(Cooppersmith, Winograd 1987)).*

. . . . . .

3. Probleme de conexiune.
Teorema lui Menger și aplicații.

Definiție. Fie $G = (V, E)$ (di)graf și $X, Y \subseteq V$. Numim **XY-drum** în G orice drum D în G de la un vîrf $x \in X$ la un vîrf $y \in Y$, astfel încît $V(D) \cap X = \{x\}$ și $V(D) \cap Y = \{y\}$.

În figura alăturată, $D_1 : a, v, u, t, c$ și $D_2 : a, d$ sunt singurele XY-drumuri ce pornesc din a :



Vom nota cu $\mathcal{D}(X, Y; G)$ mulțimea tuturor XY-drumurilor în G.

Să observăm că dacă $x \in X \cap Y$ atunci drumul de lungime 0 $D = \{x\}$ este XY-drum.

Vom spune că drumurile D_1 și D_2 sînt disjuncte dacă $V(D_1) \cap V(D_2) = \emptyset$.

Probleme practice evidente, din rețelele de comunicație, dar și unele probleme legate de conexiunea grafurilor și digrafurilor, necesită determinarea unor mulțimi de XY-drumuri disjuncte și cu număr maxim de elemente.

Vom nota cu $p(X, Y; G)$ **numărul maxim de XY-drumuri disjuncte în (di)graful G.**
Teorema care precizează acest număr a fost stabilită de Menger în 1927 și constituie unul din rezultatele fundamentale din teoria grafurilor.

Definiție. Fie $G = (V, E)$ un digraf și $X, Y \subseteq V$. Numim **mulțime XY-separatoare** în G o mulțime $Z \subseteq V$ astfel încît $\forall D \in \mathcal{D}(X, Y; G) \Rightarrow V(D) \cap Z \neq \emptyset$.

Notăm cu

$$\mathcal{S}(\mathbf{X}, \mathbf{Y}; \mathbf{G}) = \{Z \mid Z \quad XY\text{-separatoare în } G\}$$

și cu

$$\mathbf{k}(\mathbf{X}, \mathbf{Y}; \mathbf{G}) = \min\{|Z|; Z \in \mathcal{S}(X, Y; G)\}.$$

Din definiţie, rezultă următoarele proprietăţi imediate ale mulţimilor XY-separatoare:

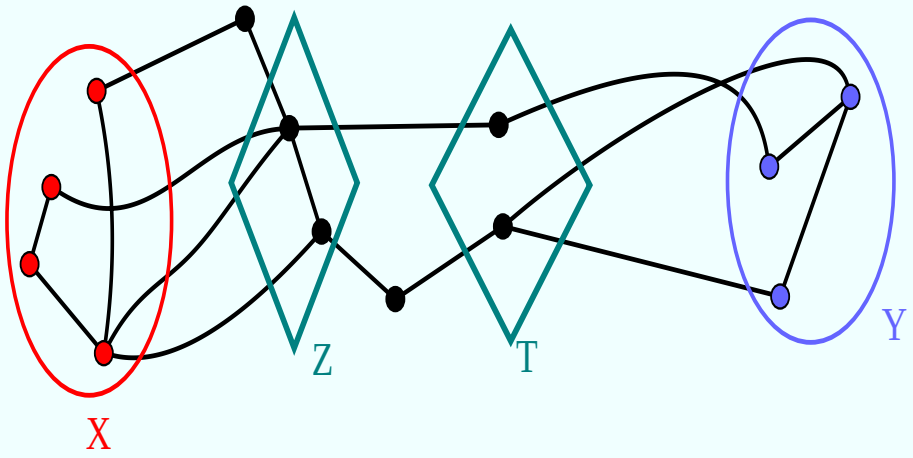
(a) Dacă $Z \in \mathcal{S}(X, Y; G)$ atunci $\forall D \in \mathcal{D}(X, Y; G)$ D nu este drum în $G - Z$.

(b) $X, Y \in \mathcal{S}(X, Y; G)$.

(c) Dacă $Z \in \mathcal{S}(X, Y; G)$ atunci $\forall A$ astfel încît $Z \subseteq A \subseteq V$ avem $A \in \mathcal{S}(X, Y; G)$.

(d) Dacă $Z \in \mathcal{S}(X, Y; G)$ și $T \in \mathcal{S}(X, Z; G)$ sau $T \in \mathcal{S}(Z, Y; G)$ atunci $T \in \mathcal{D}(X, Y; G)$.

Proprietatea (d) este esenţială pentru obţinerea teoremei următoare şi este evidenţiată mai jos



Teoremă. 1. *Fie $G = (V, E)$ (di)graf şi $X, Y \subseteq V$. Atunci*

$$p(X, Y; G) = k(X, Y; G).$$

Demonstraţie: 1^0. Dacă $p = p(X, Y; G)$ şi $D_1, D_2, \ldots, D_p$ sînt XY-drumuri disjuncte în G, atunci $\forall Z \in \mathcal{S}(X, Y; G)$ avem $Z \cap V(D_i) \neq \emptyset$ şi cum D_i sînt disjuncte $(i = 1, p)$:
$|Z| \geq |Z \cap \cup_{i=1}^{p} V(D_i)| = \sum_{i=1,p} |Z \cap V(D_i)| \geq \sum_{i=1,p} 1 = p$.

Deci $\forall Z \in \mathcal{S}(X, Y; G)$ $|Z| \geq p$; în particular $k(X, Y; G) \geq p(X, Y, G)$.

2^0. Arătăm prin inducţie după $a(G) = |V| + |E|$ că $\forall G = (V, E)$ $\forall X, Y \subseteq V$

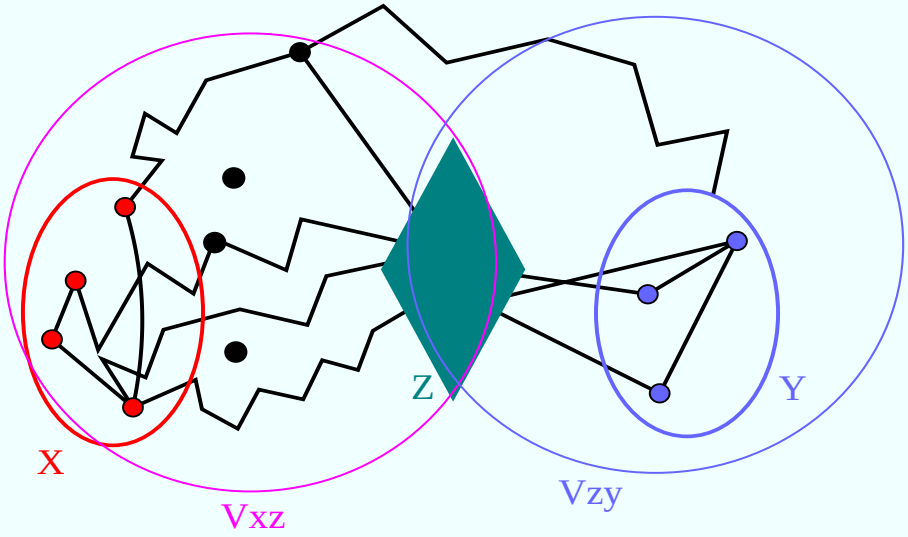
$(*)$ $\exists\ k(X, Y; G)$ XY-drumuri disjuncte în G.

(Evident, din $(*)$ rezultă că $p(X, Y; G) \geq k(X, Y; G)$ şi deci, împreună cu 1^0, teorema e demonstrată).

Cum $(*)$ se verifică pentru (di)grafuri G cu $a(G) = 1, 2$, considerăm în pasul inductiv că $(*)$ are loc pentru orice (di)graf G' şi orice $X', Y' \subseteq V(G')$, cu $a(G') < a(G)$. Pentru a exclude cazurile banale, vom presupune că $X \not\subseteq Y$, $Y \not\subseteq X$ şi $k = k(X, Y; G) > 0$.

Cazul 1. Există $Z \in \mathcal{S}(X, Y; G)$ astfel încît $|Z| = k$, $Z \neq X, Y$.
Considerăm $V_{XZ} = \{v \mid \exists D \in \mathcal{D}(X, Z; G) : \ v \in V(D)\}$ şi $V_{ZY} = \{v \mid \exists D \in \mathcal{D}(Z, Y; G) : \ v \in V(D)\}$.

Să observăm că $V_{XZ} \cap V_{ZY} = Z$
(dacă există $v \in V_{XZ} \cap V_{ZY} - Z$, atunci se obține că Z nu este XY-separatoare; dacă există $z \in Z$ astfel încît $z \notin V_{XZ} \cap V_{ZY}$ atunci $Z - \{z\}$ este XY-separatoare, contrazicînd $|Z| = k(X, Y; G)$).

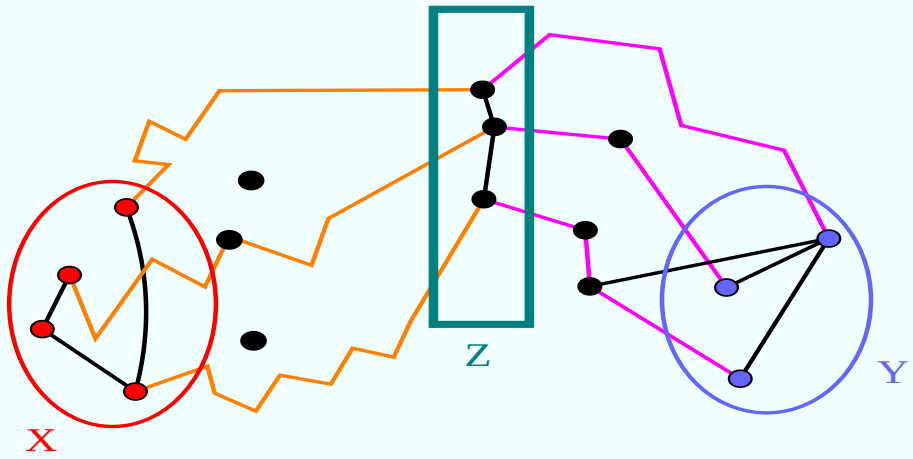


Pe de altă parte, există $x \in X - Z$ (dacă $X \subseteq Z$, atunci cum $X \in \mathcal{S}(X, Y; G)$ și $|Z| = k(X, Y; G)$ rezultă $X = Z$, contrazicînd ipoteza cazului 1) și evident $x \notin V_{ZY}$ (altfel, Z nu ar fi XY-separatoare). Rezultă $|V_{ZY}| < |V|$. În mod similar $|V_{XZ}| < |V|$.

Fie $G_{XZ} = [V_{XZ}]_G$ și $G_{ZY} = [V_{ZY}]_G$. Din observațiile precedente: $a(G_{XZ}), a(G_{ZY}) < a(G)$.

Avem $k(X, Z; G_{XZ}) = k$ și $k(Z, Y; G_{ZY}) = k$ (Z este XZ- separatoare în G_{XZ}, respectiv ZY-separatoare în G_{ZY} și are cardinalul k; dacă în unul din cele două grafuri, ar exista o mulțime T separatoare de cardinal $< k$, atunci, utilizînd observația (d), se contrazice definiția lui k pentru G, X și Y).

Din ipoteza inductivă, rezultă că există k XZ-drumuri disjuncte în G_{XZ} și k ZY-drumuri disjuncte în G_{ZY}. Cum $V_{XZ} \cap V_{ZY} = Z$ și $|Z| = k$, rezultă că aceste $2k$ drumuri se pot concatena două cîte două în G (vezi figura de mai jos) și deci $(*)$ are loc.



Cazul 2. Oricare ar fi Z XY-separatoare astfel încît $|Z| = k$ avem $Z = X$ sau $Z = Y$.

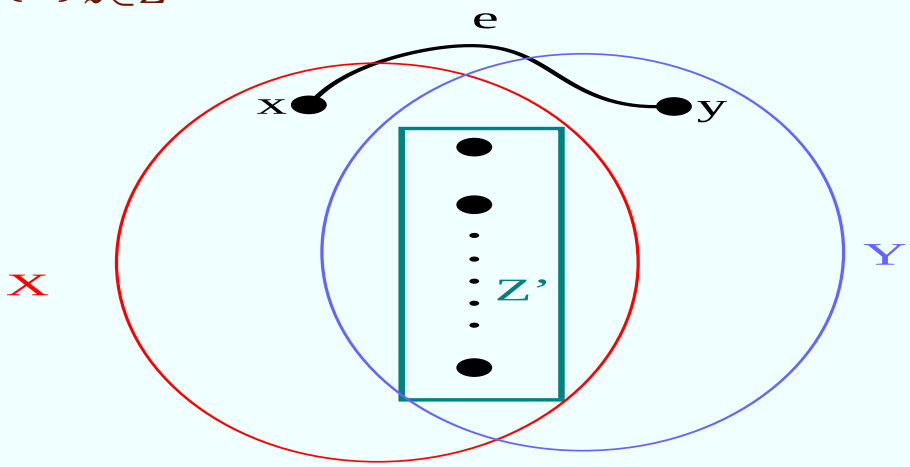
Presupunem, pentru precizarea notațiilor, $Z = X$. Cum $X \not\subseteq Y$, există $x \in X - Y$. $X - \{x\}$ nu este XY-separatoare (are mai puțin de k elemente). Există deci un XY-drum în G. Fie $e = xy$ prima muchie (arc) a acestui drum (există!). Să observăm că $y \notin X$. Considerăm $G' = G - e$. Avem $a(G') < a(G)$, deci $(*)$ are loc pentru G', X și Y.

Dacă $k(X, Y; G') = k$, atunci cele k XY-drumuri disjuncte din G' sînt XY-drumuri disjuncte și în G deci $(*)$ are loc pentru G, X și Y.

Dacă $k(X, Y; G') < k$, atunci în G' există Z' XY-separatoare cu $|Z'| = k-1$ (se aplică, eventual, proprietatea (c)).

Deci Z' nu este XY-separatoate în G ($|Z'| < k$).
Singurele XY-drumuri pe care Z nu le intersectează sînt cele care au drept primă muchie (arc) pe e.
Din definiţia lui k, rezultă că $x \notin Z'$, $y \notin Z'$ şi $|Z' \cup \{x\}| = |Z' \cup \{y\}| = k$.
Din alegerea lui x şi y avem $Z' \cup \{x\} \neq Y$ şi $Z' \cup \{y\} \neq X$.
Din ipoteza cazului 2, rezultă atunci că $Z' \cup \{x\} = X$ şi $Z' \cup \{y\} = Y$.

Cele k drumuri din $(*)$ sînt în acest caz $\{z\}_{z \in Z'}$ şi (x, xy, y).



Cu acestea, teorema este demonstrată.

Observaţii: 1^0. Egalitatea min-max din enunţul teoremei este interesantă şi conduce, aşa cum vom vedea, la rezultate importante, în cazuri particulare.

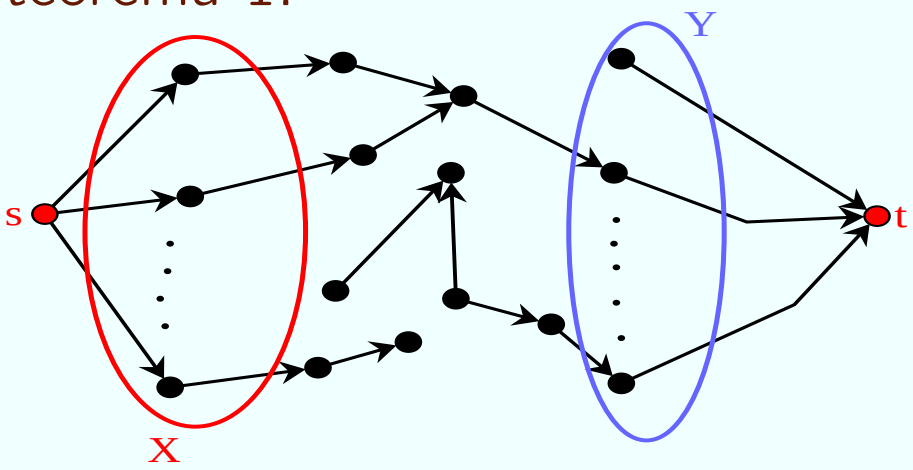
2^0 Teorema se poate demonstra şi algoritmic ca o consecinţă a teoremei fluxului maxim-secţiunii minime, aşa cum vom arăta în capitolul relativ la probleme de flux.

Forma echivalentă în care a fost enunţată şi demonstrată iniţial de Menger (1927) teorema 1 este:

Teoremă. 1'. *Fie $G = (V, E)$ un (di)graf şi $s, t \in V$, astfel încît $s \neq t$, $st \notin E$. Există k drumuri intern disjuncte de la s la t în graful G dacă şi numai dacă îndepărtînd mai puţin de k vîrfuri diferite de s şi t, în graful rămas există un drum de la s la t.*

Notām că două drumuri sînt intern disjuncte dacă nu au vîrfuri comune cu excepţia extremităţilor.

Se observă că dacă se consideră $X = N_G(s)$ şi $Y = N_G(t)$ (respectiv, $N_G^+(s)$ şi $N_G^-(t)$ în cazul digrafurilor) teorema 1' se obţine imediat din teorema 1.



Reciproc, o construcţie inversă celei de mai sus asupra tripletului G, X, Y din teorema 1, aratā că teorema 1 se obţine din teorema 1'.

Am definit un graf G p-conex ($p \in N^*$) dacă $G = K_p$ sau dacă $|G| > p$ și G nu poate fi deconectat prin îndepărtarea a mai puțin de p vîrfuri. Utilizînd teorema 2' obținem

Corolar. *Un graf G este p-conex dacă $G = K_p$ sau $\forall st \in E(\overline{G})$ există p drumuri intern disjuncte de la s la t în G.*

Determinarea numărului $k(G)$ de conexiune a grafului G (cea mai mare valoare a lui p pentru care G este p-conex) se reduce deci la determinarea lui

$$\min_{st \in E(\overline{G})} p(\{s\}, \{t\}; G)$$

problemă care vom dovedi că se poate rezolva în timp polinomial.

Un caz particular interesant al teoremei 1, se obține atunci cînd G este un graf bipartit iar X și Y sînt cele două clase ale bipartiției:

Teoremă. 2. (Konig, 1931) *Dacă* $G = (S, R; E)$ *este un graf bipartit, atunci cardinalul maxim al unui cuplaj este egal cu cardinalul minim al unei mulţimi de vîrfuri incidente cu toate muchiile grafului.*

Demonstraţie: Evident, cardinalul maxim al unui cuplaj în G este
$p(S, R; G)$, care este egal, conform teoremei 1, cu $k(S, R; G)$.
Teorema rezultă imediat dacă observăm că o mulţime de vîrfuri este SR-separatoare dacă şi numai dacă este incidentă cu orice muchie a grafului.

O aplicaţie, fundamentală în numeroase raţionamente combinatorii, a acestei teoreme este teorema lui **Hall** (1935).

Definiţie: Fie I şi S mulţimi finite nevide. Numim *familie de submulţimi ale lui* S *(indexată după* I*)* orice aplicaţie $\mathcal{A} : I \to 2^S$. Vom nota familia $\mathcal{A} = (A_i; i \in I)$ şi vom folosi notaţia funcţională uzuală
$\mathcal{A}(J) = \cup_{j \in J} A_j$ (pentru $J \subseteq I$).

Dacă $\mathcal{A} = (A_i; i \in I)$ este o familie de submulțimi ale lui S, o funcție $r_{\mathcal{A}} : I \to S$ cu proprietatea că $r_{\mathcal{A}}(i) \in A_i$, $\forall i \in I$ se numește *funcție de reprezentare pentru familia $\mathcal{A}$.*

În acest caz, $(r_{\mathcal{A}}(i); i \in I)$ formează un sistem de reprezentanți ai familiei A.

Dacă funcția de reprezentare $r_{\mathcal{A}}$ este injectivă atunci $r_{\mathcal{A}}(I) \subseteq S$ se numește *sistem de reprezentanți distincți ai familiei $\mathcal{A}$, sau transversală.*

Problema centrală în teoria transversalelor este aceea de a caracteriza familiile $\mathcal{A}$ care admit transversale (eventual cu anumite proprietăți). Prima teoremă de acest tip a fost stabilită de *Hall* în 1935:

Teoremă. *3. Familia $\mathcal{A} = (A_i; i \in I)$ de submulțimi ale lui S admite o transversală dacă și numai dacă*

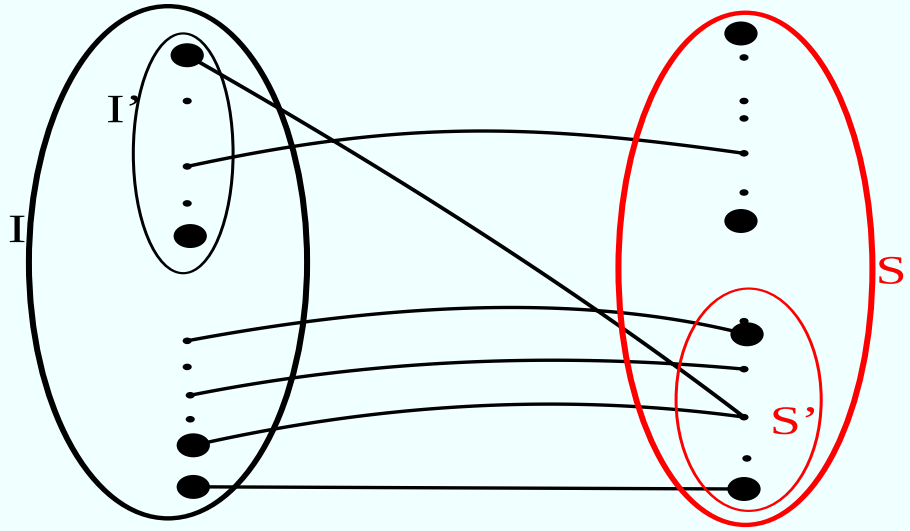
$$(H) \qquad |\mathcal{A}(J)| \geq |J| \quad \forall J \subseteq I.$$

Demonstrație: Necesitatea este evidentă: dacă $\mathcal{A}$ admite o funcție $r_{\mathcal{A}}$ de reprezentare injectivă atunci $\forall J \subseteq I \; r_{\mathcal{A}}(J) \subseteq \mathcal{A}(J)$ și deci $|\mathcal{A}(J)| \geq |r_{\mathcal{A}}(J)| \geq |J|$ (întrucît $r_{\mathcal{A}}$ este injectivă).

Suficiența. Considerăm graful bipartit $G_{\mathcal{A}} = (I, S; E)$ unde am presupus $I \cap S = \emptyset$ (altfel, se consideră copii izomorfe disjuncte) iar $E = \{is \mid i \in I, \; s \in S \; \wedge \; s \in A_i\}$. Se observă că $N_{G_{\mathcal{A}}}(i) = A_i$ și că $\mathcal{A}$ are o transversală dacă și numai dacă $G_{\mathcal{A}}$ are un cuplaj de cardinal $|I|$. În ipoteza că (H) are loc, arătăm că orice mulțime de vîrfuri incidentă cu toate muchiile lui $G_{\mathcal{A}}$ are măcar $|I|$ elemente,

ceea ce dovedește existența cuplajului de cardinal $|I|$ (utilizînd teorema 2).

Fie $X = I' \cup S' \subset I \cup S$ o mulțime de vîrfuri incidentă cu toate muchiile. Rezultă că $N_{G_{\mathcal{A}}}(I - I') \subseteq S'$, adică $\mathcal{A}(I - I') \subseteq S'$. Atunci, $|X| = |I'| + |S'| \geq |I'| + |\mathcal{A}(I - I')|$. Folosind condiția (H) obținem în continuare: $|X| \geq |I'| + |\mathcal{A}(I - I')| \geq |I'| + |I - I'| = |I|$.



O altă teoremă celebră care poate fi obţinută ca o consecinţă imediată a teoremei 2 este teorema lui *Dilworth, 1950.*
Preferăm totuşi, o demonstraţie directă, pentru a evidenţia asemănarea cu cea a teoremei 1. Fie $(P, \leq)$ o mulţime finită parţial ordonată ($\leq$ este o relaţie de ordine pe P).
Dacă $x, y \in P$, spunem că x şi y *sînt comparabile* dacă $x \leq y$ sau $y \leq x$.
Un *lanţ* în $(P, \leq)$ este o submulţime L a lui P cu proprietatea că orice două elemente ale sale sînt comparabile.
Un *antilanţ* în $(P, \leq)$ este o submulţime A a lui P cu proprietatea că $\forall x, y \in A \; x \leq y \Rightarrow x = y$.

Teoremă. 4. (Dilworth, 1950) *Dacă $(P, \leq)$ este o mulţime parţial ordonată finită, atunci numărul minim de lanţuri a căror reuniune (disjunctă) este P este egal cu cardinalul maxim al unui antilanţ.*

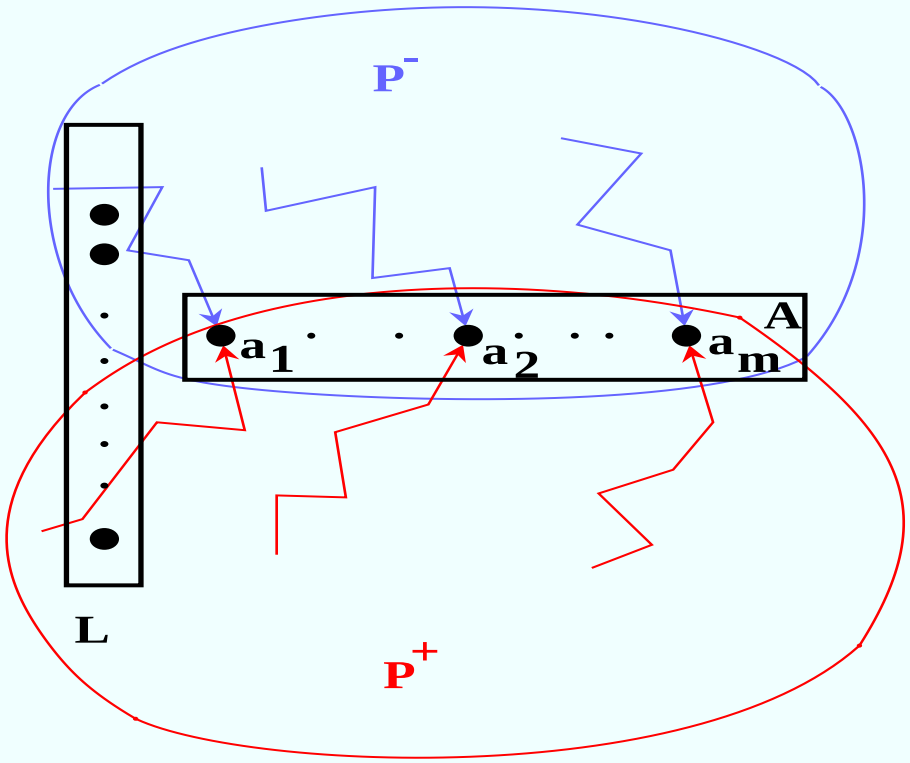
Demonstraţie. Fie $a(P, \leq)$ cardinalul maxim al unui antilanţ al lui $(P, \leq)$.
Arătăm prin inducţie după $|P|$, că există $a(P, \leq)$ lanţuri a căror reuniune este P (inegalitatea inversă este imediată).

Dcaă $|P| = 1$, afirmaţia este trivială, deci presupunem, în pasul inductiv, că teorema are loc pentru orice mulţime parţial ordonată cu mai puţin de $|P| \geq 2$ elemente.
Fie L un lanţ maximal (în raport cu incluziunea) al lui P.
Dacă $a(P - L, \leq) = a(P, \leq) - 1$, atunci teorema are loc din ipoteza inductivă.
Deci, presupunem că în $P - L$ există un antilanţ $A = \{a_1, a_2, \ldots, a_m\}$ cu $m = a(P, \leq)$. Fie

$$P^- = \{x \mid x \in P, \exists a_i \in A : x \leq a_i\}$$

$$P^+ = \{x \mid x \in P, \exists a_j \in A : a_j \leq x\}.$$

Se observă că $P = P^- \cup P^+$, $P^- \cap P^+ = A$ şi că $|P^-|, |P^+| < |P|$ [elementul maximal (minimal) al lui L nu aparţine lui P^- (respectiv, P^+)].



Se poate, deci, aplica ipoteza inductivă pentru a scrie

$P^- = \cup_{i=1,m} L_i^-$ $\quad L_i^-$ lanț $\wedge\ a_i \in L_i^-\ \forall i = 1, m$

$P^+ = \cup_{i=1,m} L_i^+$ $\quad L_i^+$ lanț $\wedge a_i \in L_i^+\ \forall i = 1, m.$

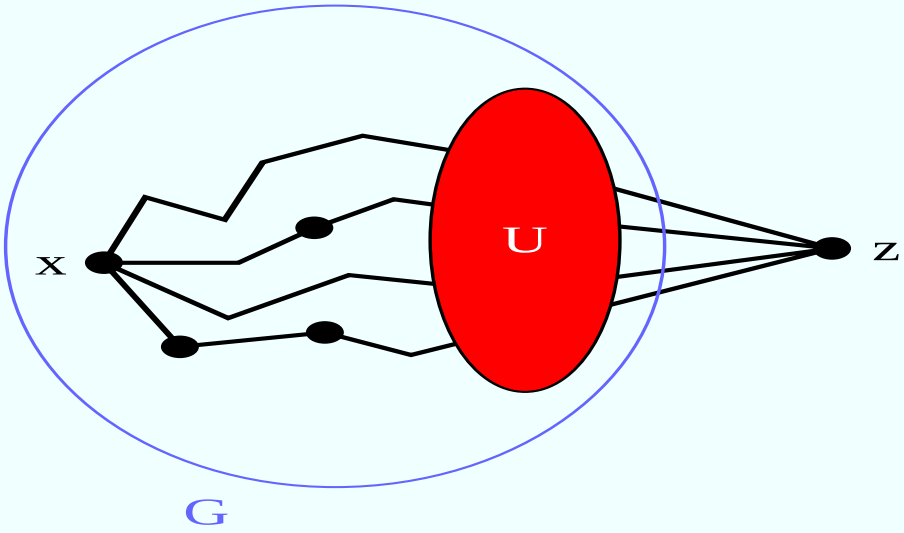
În plus, a_i este element maximal în L_i^- și element minimal în L_i^+.

Rezultă că $(L_i^- \cup L_i^+)_{i=1,m}$ sînt cele $a(P, \leq)$ lanțuri a căror reuniune este P.

Structura grafurilor p-conexe.

Lemã. 1. *Fie $G = (V, E)$ p-conex, $|V| \geq p+1$, $U \subseteq V$ $|U| = p$ şi $x \in V - U$. Există în G p xU-drumuri cu singurul vîrf comun x.*

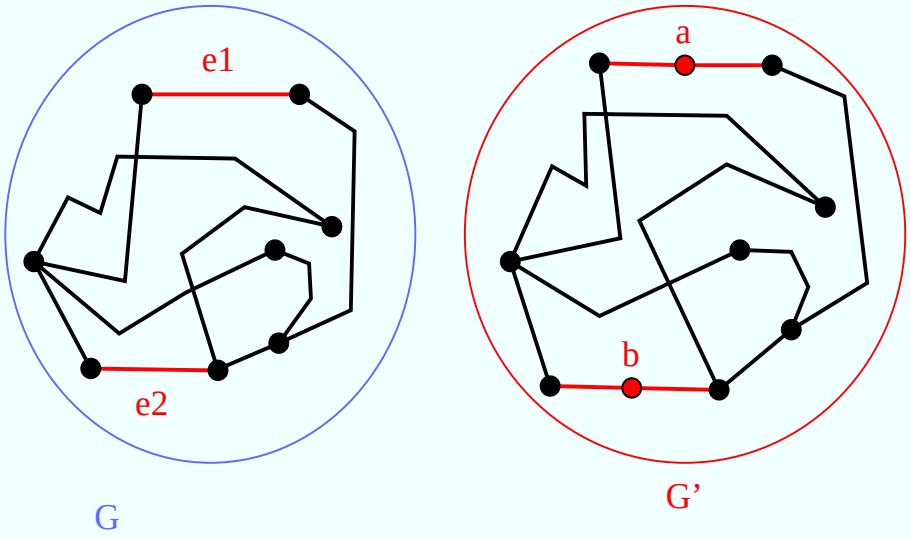
Demonstraţie: Considerām graful $G' = (V \cup \{z\}, E')$, unde $E' = E \cup \{zy \mid y \in U\}$.



G' este p-conex. În adevār, $\forall A$ cu $|A| \leq p-1$ $G' - A$ este conex (dacā $z \in A$, acest lucru este evident din p-conexiunea lui G; dacā $A \subseteq V$, atunci $G' - A$ este conex întrucît $G - A$ este conex şi $\exists y \in U$ cu $zy \in E(G' - A)$). Lema rezultā, aplicînd teorema 1' grafului G' şi perechii x, z.

Lemă. 2. *Dacă $G = (V, E)$ este un graf p-conex $p \geq 2$, atunci oricare ar fi două muchii e_1 și e_2 și $p-2$ vîrfuri $x_1, x_2, \ldots, x_{p-2}$ există un circuit în G care le conține.*

Demonstrție: Inducție după p.
Dacă $p = 2$, trebuie să dovedim că în orice graf 2-conex, prin orice două muchii trece un circuit. Considerăm G' obținut din G prin inserția cîte unui vîrf pe muchiile e_1 (a) și e_2 (b).



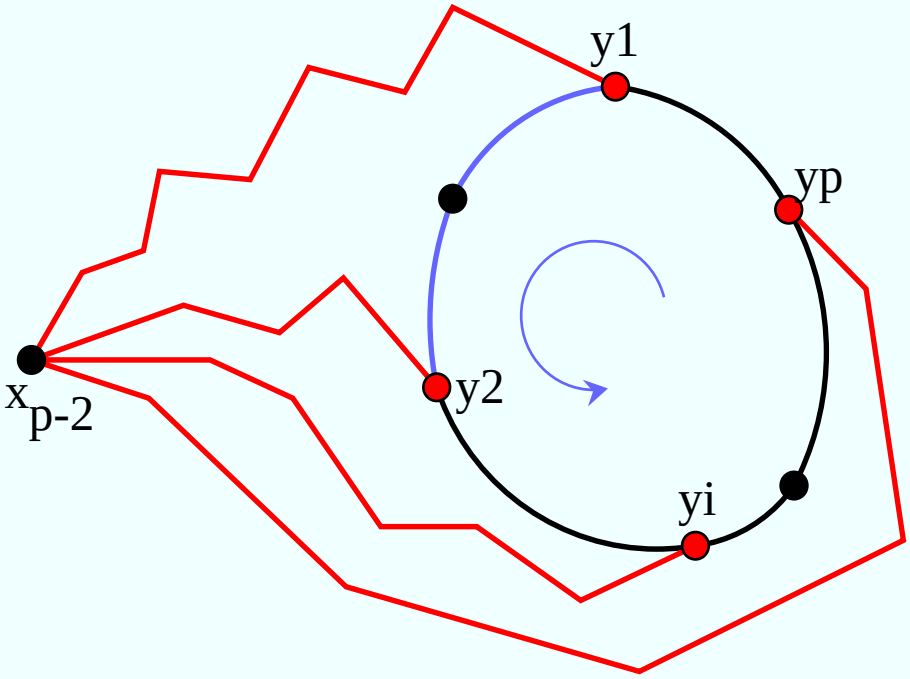
Noul graf este tot 2-conex, deoarece orice vîrf am scoate, nu se pierde conexiunea . Există deci în G' două ab-drumuri disjuncte, care în G oferă un circuit ce conține e_1 și e_2.

Fie $p > 2$ și presupunem afirmaţia adevărată pentru orice graf k-conex $k < p$. Fie G p-conex.

Putem presupune că extremitătile muchiilor e_1 și e_2 nu sînt printre $x_1, x_2, \ldots, x_{p-2}$, deoarece altfel, afirmaţia ar rezulta prin inducţie.

Graful $G - x_{p-2}$ este $(p-1)$-conex. Conform ipotezei inductive exista un circuit μ ce conţine $x_1, x_2, \ldots, x_{p-3}$ și e_1, e_2. Fie Y muţimea vîrfurilor lui μ, $|Y| \geq p$.

Folosind lema 1, există în G p $x_{p-2}y$ drumuri cu $y \in Y$, disjuncte. Putem presupune că pentru orice $x_{p-2}y$ astfel de drum, y este primul vîrf din Y întîlnit, așa că aceste drumuri au cîte un singur vîrf comun cu Y. Dăm o orientare circuitului μ și numerotăm vîrfurile sale conform acestei orientări.Avem deci drumurile $D_{x_{p-2}y_1}, D_{x_{p-2}y_2}, \ldots, D_{x_{p-2}y_p}$.



Vîrfurile $y_1, y_2, \ldots, y_p$ descompun circuitul în drumurile $D_{y_1y_2}, D_{y_2y_3}, \ldots, D_{y_{p-1}y_p}, D_{y_py_1}$.

Există un drum dintre acestea, în care nu e conţinut nici unul din elementele
$x_1, \ldots, x_{p-3}, e_1, e_2$.

Fie acest drum $D_{y_1y_2}$; atunci
$D_{x_{p-2}y_2}, D_{y_2y_3}, \ldots, D_{y_py_1}, D_{y_1x_{p-2}}$ este un circuit ce conţine $x_1, x_2 \ldots, x_{p-2}$, e_1 şi e_2 ,
şi lema este demonstrată.

Teoremă. 5. (Dirac 1953) *Dacă $G = (V, E)$ este un graf p-conex $p \geq 2$, atunci prin orice p vîrfuri ale sale trece un circuit.*

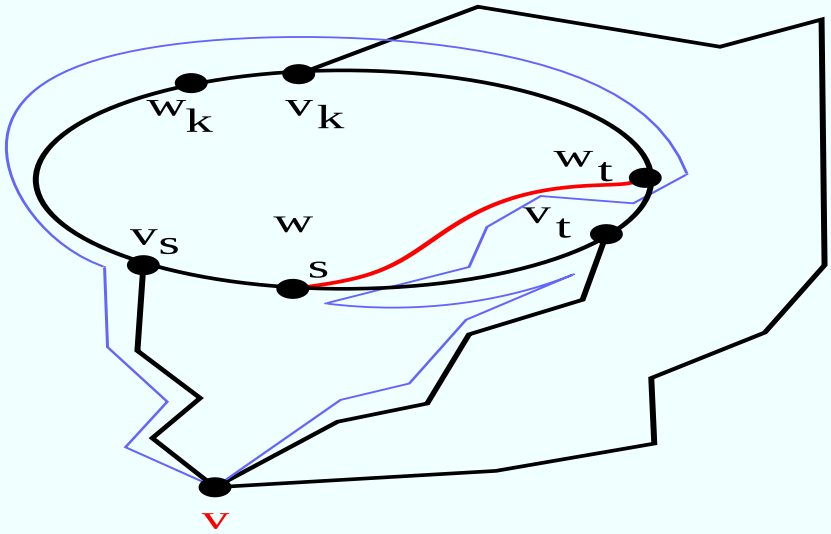
Demonstraţie. Fie $x_1, x_2, \ldots, x_{p-2}, x_{p-1}, x_p$ p vîrfuri oarecare ale lui G. Deoarece graful G este conex, există $e_1 = xx_{p-1}$ și $e_2 = yx_p$ și aplicăm lema 2.

Aplicăm această teoremă, precum și ideea utilizată în demonstraţia lemei 2, pentru a demonstra o condiţie suficientă de hamiltonietate interesantă, datorată lui *Erdös și Chvatal (1972).*

Teoremă. 6. *Fie G p-conex. Dacă $\alpha(G) \leq p$ atunci G este hamiltonian.*

Demonstraţie: Presupunem că G nu e hamiltonian. Vom obţine o contradicţie. Cum G este p-conex , există un circuit de lungime cel puţin p (conform teoremei lui Dirac de mai sus).

Fie C un circuit de lungime maximă în G. Dacă G nu e hamiltonian, există $v \notin C$. Întrucît $|C| \geq p$, conform lemei 2, există p vC-drumuri disjuncte (cu excepţia lui v) fie ele $D_{vv_1}, D_{vv_2}, \ldots, D_{vv_p}$ (numerotarea vîrfurilor este în ordinea întîlnirii lor întro parcurgere fixată a circuitului). Notăm, pentru fiecare v_i, cu w_i vîrful succesor al lui v_i în parcurgerea lui C.



Atunci, $vw_i \notin E$ (altfel am avea circuitul $vw_i, w_i, C - w_i v_i, D_{v_i v}$ de lungime mai mare decît C). Cum $\alpha(G) \leq p$, mulţimea $\{v, w_1, w_2, \ldots, w_p\}$ nu este stabilă. Deci, există $w_s w_t \in E$.

Dar atunci:
D_{vv_s},drumul (invers) pe C de la v_s la w_t, muchia $w_t w_s$,drumul (invers) pe C de la w_s la v_t,şi $D_{v_t v}$ este un circuit de lungime mai mare decît C, contrazicînd ipoteza că C este de lungime maximă.

III. ARBORI

1. Proprietăţi elementare ale arborilor

Definiţie: Un **arbore** este un graf conex şi fără circuite.

Teoremă. 1. *Fie $G = (V, E)$ un graf. Următoarele afirmaţii sînt echivalente:*
(i) *G este arbore.*
(ii) *G este conex şi este minimal cu această proprietate.*
(iii) *G este fără circuite şi este maximal cu această proprietate.*
Observaţie: Maximalitatea şi minimalitatea din condiţiile (ii) şi (iii) se referă la mulţimea muchiilor grafului G şi se consideră în raport cu relaţia de ordine dată de incluziune. Mai precis, cele două afirmaţii se pot formula echivalent astfel:
(ii') *G este conex şi $\forall e \in E(G),\ G - e$ este neconex.*
(iii') *G este fără circuite şi $\forall e \in E(\overline{G}),\ G + e$ are un circuit.*

Definiţie: Fie $G = (V, E)$ un (multi)graf. Se numeşte **arbore parţial** al lui G, un graf parţial $T = (V, E')$ $(E' \subseteq E)$ care este arbore.

Vom nota cu $\mathcal{T}_G$ **mulţimea arborilor parţiali** ai lui G.

Obs. $\mathcal{T}_G \neq \emptyset$ dacă şi numai dacă G este conex.

În adevăr, dacă $\mathcal{T}_G \neq \emptyset$, atunci există un arbore parţial $T = (V, E')$ al lui G. T este conex, deci între orice două vîrfuri ale lui G există un drum cu muchii din $E' \subseteq E$. Prin urmare G este conex.

Reciproc, dacă G este conex, atunci considerăm următorul algoritm:

1. $T \leftarrow G$
2. **while** ($\exists e \in E(T)$ astfel încît $T \setminus \{e\}$ este conex) **do**
   $T \leftarrow T \setminus \{e\}$

Graful T obținut este graf parțial al lui G, este conex (din ipotezã, dupã atribuirea din 1, așa este și din condiția lui **while**, T este conex dupã fiecare iterație) și în plus la oprirea algoritmului, T satisface condiția ii) din teorema 1, deci este arbore.

O altă demonstrație a reciprocei anterioare se bazează pe observația că $G = (V, E)$ este conex dacă și numai dacă oricare ar fi o partiție (V_1, V_2) a lui V există $e = v_1v_2 \in E$ cu $v_i \in V_i$ $i = \overline{1,2}$.

Dacă $|V| = n > 0$ atunci următorul algoritm construiește un arbore parțial al lui G:

1. $T_1 \leftarrow (\{v\}, \emptyset)$ ($v \in V$, oarecare); $k \leftarrow 1$;
2. **while** $k < n$ **do**
   { Fie $v_1v_2 \in E$ cu $v_1 \in V(T_k), v_2 \in V \setminus V(T_k)$;
   //$\exists$ o astfel de muchie din conexiunea lui G
   $V(T_{k+1}) \leftarrow V(T_k) \cup \{v_2\}$;
   $E(T_{k+1}) \leftarrow E(T_k) \cup \{v_1v_2\}$;
   $k := k + 1$
   }

Se observă că T_k este arbore $\forall k = \overline{1,n}$
(inductiv, dacă T_k este arbore, atunci din construcție T_{k+1} este conex și nu are circuite)
și, în plus, se verifică imediat că:
$|V(T_k)| = k$ iar $|E(T_k)| = k-1 \ \ \forall k = 1, 2, \ldots, n$.

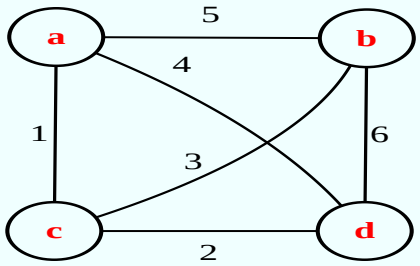
Această demonstrație aplicată unui arbore G cu n vîrfuri dovedește ca G are $n-1$ muchii.

Proprietatea obținută poate fi folosită pentru completarea teoremei 1 cu alte caracterizări ale arborilor:

Teoremă. *1'. Următoarele afirmații sînt echivalente pentru un graf $G = (V, E)$ cu n vîrfuri:*
(i) *G este arbore.*
(ii) *G este conex și are $n-1$ muchii.*
(iii) *G este fară circuite și are $n-1$ muchii.*
(iv) *$G = K_n$ pentru $n = 1, 2$ și $G \neq K_n$ pentru $n \geq 3$ și adăugarea unei muchii la G produce exact un circuit.*

2. Numărarea și enumerarea arborilor parțiali

Familia $\mathcal{T}_G$ a arborilor parțiali ai unui (multi)graf are proprietăți interesante. Vom prezenta o metodă (tip backtrack) de **generare a elementelor lui $\mathcal{T}_G$,** problemă de interes practic în multe aplicații (de exemplu, în chimie).
Fie $G = (V, E), V = \{1, 2, \ldots, n\}, |E| = m$.
Reprezentăm E printr-un tablou $E[1..m, 1..2]$ cu componente din V cu semnificația că dacă $v = E[i, 1]$ și $w = E[i, 2]$, atunci vw este muchia i a grafului G ($i = \overline{1, m}$).
Vom presupune în plus, că primele $d_G(v_0)$ muchii din tabloul E satisfac $E[i, 1] = v_0$ unde $v_0 \in V$ este un vîrf oarecare. Exemplu:



1	c	a
2	c	d
3	c	b
4	a	d
5	a	b
6	b	d

E

Un arbore parțial $T \in \mathcal{T}_G$ va fi identificat cu mulțimea indicilor ce reprezintă muchiile sale în tabloul E (submulțime a lui $\{1, \ldots, m\}$ de cardinal $n-1$).
Pe tot parcursul generării dispunem de un vector global $T[1..n-1]$ cu componente din mulțimea $1..m$ și de un indicator i având semnificația: *în arborele curent care se construiește, primele* $i-1$ *muchii sînt*
$T[1] < T[2] < \ldots < T[i-1] \quad (i \in \{1, \ldots, n\})$.

generare-arbori-parțiali$(int\ i)$;
// se generează toți arborii parțiali ai lui G
având drept prime $i-1$ muchii,elementele
$T(1), \ldots, T(i-1)$
ale tabloului E (ordonate crescător).
variabile locale:
$j \in \{1, \ldots m\}$; S, listă de vîrfuri; $x \in V$;

if $i = n$ **then**
//$\{T(1), \ldots, T(n-1)\}$ formează un
arbore parțial ;
prelucrează T *(listează, memorează etc.)*

else
  if $i = 1$ **then**
    for $j := 1$ **to** $d_G(v_0)$ **do**
    { $T[i] \leftarrow j$;
      A:
      generare-arbori-parţiali$(i + 1)$;
      B:
    }
  else
    for $j := T[i-1] + 1$ **to** $m - (n-1) + i$ **do**
      if $\langle \{T[1], \ldots, T[i-1]\} \cup \{j\} \rangle_G$ nu are circuite
        then
        { $T[i] \leftarrow j$;
          A:
          generare-arbori-parţiali$(i + 1)$;
          B:
        }

Apelul *generare-arbori-parţiali(1)* rezolvă problema enumerării elementelor lui $\mathcal{T}_G$.

Pentru exemplul considerat mai sus ($G = K_4$ și numerotarea precizată pentru muchii) arborii generați sunt următorii 16 :

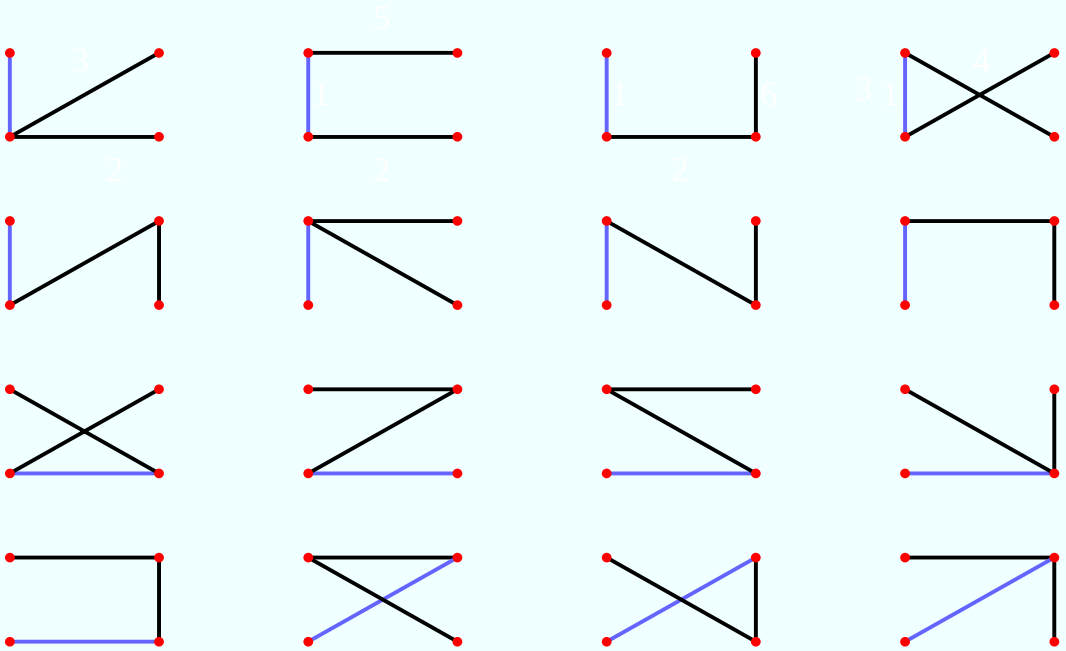


Figura poate fi interpretată și ca lista tuturor arborilor **etichetați** distincți care se pot forma cu cele 4 noduri (nu s-au mai trecut etichetele nodurilor și muchiilor, pentru decongestionarea figurii; totuși s-a marcat muchia $T(1)$ pentru a se evidenția modul de funcționare a algoritmului).

Pentru implementarea eficientă a testului dacă graful parţial

$\left\langle \{T[1], \ldots, T[i-1]\} \cup \{j\} \right\rangle_G$ nu are circuite să observăm că din construcţie,

$$\left\langle \{T[1], \ldots, T[i-1]\} \right\rangle_G$$

nu are circuite , deci componentele sale conexe sînt arbori.

Vom considera un vector global $rad[1..n]$ cu componente din V cu semnificaţia

$rad[v]$ = rădăcina arborelui la care aparţine vîrful v (unul din vîrfurile acestui arbore).

Înaintea apelului *generare-arbori-parţiali(1)* se iniţializează $rad[v] \leftarrow v (\forall v \in V)$, ceea ce corespunde faptului că $\{T[1], \ldots, T[i-1]\} = \emptyset$.

Pe parcursul apelurilor (recursive) se încearcă plasarea muchiei j în mulţimea curentă $T[1], \ldots, T[i-1]$.

Fie $v = E[j, 1]$ și $w = E[j, 2]$.

Atunci $\left\langle \{T[1], \ldots, T[i-1]\} \cup \{j\} \right\rangle_G$ nu are circuite **dacă și numai dacă**
muchia vw nu are extremitățile în aceeași componentă a lui

$$\left\langle \{T[1], \ldots, T[i-1]\} \right\rangle_G,$$

adică **dacă și numai dacă** $rad[v] \neq rad[w]$.

Vectorul rad trebuie întreținut pentru a avea semnificația dorită.
Acest lucru se obține înlocuind în algoritmul descris, instrucțiunile (vide) etichetate A și B. Astfel, A: se va înlocui cu secvența :

$S \leftarrow \emptyset; x \leftarrow rad[v];$
for $u \in V$ **do**
  if $rad[u] = x$ **then**
  { $S \leftarrow S \cup \{u\}$;
    $rad[u] \leftarrow rad[w]$
  }

(în cuvinte, arborele cu rădăcina x se "unește" cu arborele cu rădăcina $rad[w]$; se salvează în S vîrfurile arborelui cu rădăcina x).

După apelul lui *generare-arbori-parţiali*$(i+1)$ trebuie refăcut vectorul rad la valoarea dinainte de apel, deci se va înlocui B: cu

$$\textbf{for } u \in S \textbf{ do } rad[u] := x;$$

Numărul elementelor lui $\mathcal{T}_G$, problemă interesantă chiar și numai pentru analiza algoritmului precedent, se poate determina eficient. Prezentăm în continuare una din soluţiile posibile.

Fie $G = (V, E)$ un multigraf cu $V = \{1, 2, \ldots, n\}$. Cosiderăm $A = (a_{ij})_{n\times n}$ matricea de adiacență a lui G (a_{ij} = multiplicitatea muchiei ij dacă $ij \in E$, altfel 0). Fie

$$D = diag(d_G(1), d_G(2), ..., d_G(n)).$$

Matricea $L[G] = D - A$ se numește **matricea de admitanță a multigrafului** G sau **matricea Laplace a lui** G.

Să observăm că în $L[G]$ suma elemetelor de pe fiecare linie și fiecare coloană este 0.

Teoremă. 2. (Kirchoff-Trent) *(Matrix Tree Theorem) Dacă G este un multigraf cu mulțimea de vîrfuri $\{1, \ldots, n\}$ și $L[G]$ matricea Laplace, atunci*

$$|\mathcal{T}_G| = det(L[G]_{ii}) \quad \forall i \in \{1, \ldots, n\}.$$

$L[G]_{ij}$ notează minorul lui $L[G]$ obținut prin îndepărtarea liniei i și coloanei j.

Demonstrația (pe care o omitem) se bazează pe regula clasică de dezvoltare a unui determinat după o linie, după descompunerea lui $\mathcal{T}_G$ în arborii care conțin o muchie fixată și cei care nu conțin aceeași muchie.

Corolar. $\left|\mathcal{T}_{K_n}\right| = n^{n-2}$ **(Cayley)**.

În adevăr,

$$L[K_n] = \begin{pmatrix} n-1 & -1 & \dots & -1 \\ -1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 \\ -1 & \dots & -1 & n-1 \end{pmatrix}$$

și (după un simplu calcul):

$$det(L[K_n]_{11}) = \begin{vmatrix} n-1 & -1 & \dots & -1 \\ -1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 \\ -1 & \dots & -1 & n-1 \end{vmatrix} = n^{n-2}.$$

(Dacă $n = 4$, caz ilustrat în figura precedentă, se obține $\left|\mathcal{T}_{K_4}\right| = 4^2 = 16$, adică au fost generați toți arborii !)

Observație. Teorema oferă un algoritm polinomial de determinare a lui $|\mathcal{T}_G|$.

3. Arbori parţiali de cost minim.

Considerām următoarea problemă:

(P1) *Date* $G = (V, E)$ *graf şi* $c : E \to \mathbf{R}$ *(*$c(e)$ *– costul muchiei* e*), să se determine* $T^* \in \mathcal{T}_G$ *astfel încît*

$$c(T^*) = \min\{c(T) \mid T \in \mathcal{T}_G\},$$

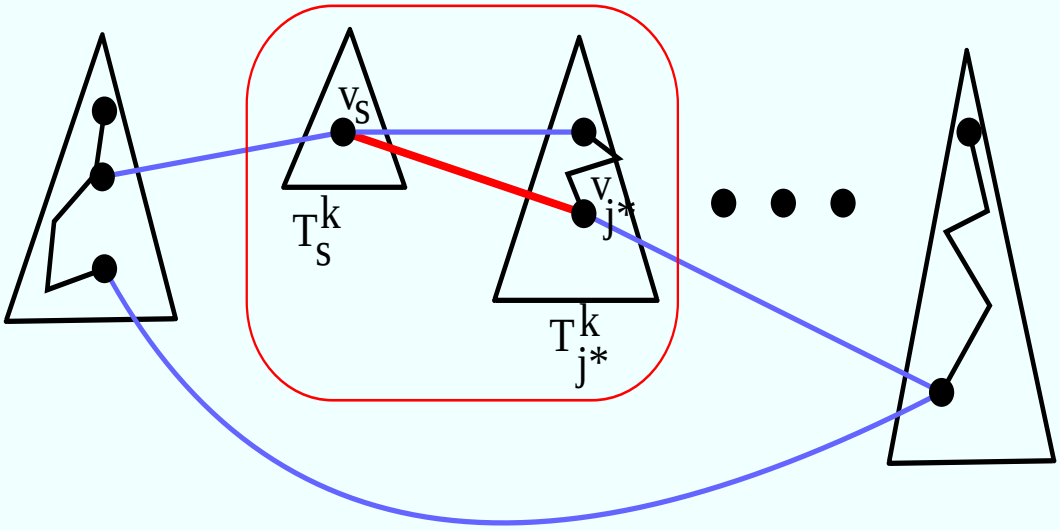
unde $c(T) = \sum_{e \in E(T)} c(e)$.

Algoritmii cunoscuţi pentru rezolvarea problemei (P1) au la bază următoarea ideie:

Se consideră iniţial, familia $\mathcal{T}^0 = (T_1^0, T_2^0, \ldots, T_n^0)$ de arbori disjuncţi $T_i^0 = (\{i\}, \emptyset)$ $i = \overline{1, n}$ (am presupus, ca de obicei că $V = \{1, 2, \ldots, n\}$) .

În pasul general k ($k = \overline{0, n-2}$) al algoritmului se dispune de familia $\mathcal{T}^k = (T_1^k, T_2^k, \ldots, T_{n-k}^k)$ de $n-k$ arbori disjuncţi astfel încît $V(T_i^k)_{i=\overline{1,n-k}}$ constituie o partiţie a lui V şi se construieşte $\mathcal{T}^{k+1}$ astfel:

- se alege T_s^k unul din arborii familiei $\mathcal{T}^k$.
- dintre toate muchiile lui G cu o extremitate în T_s^k și cealaltă în $V - V(T_s^k)$ se alege una de cost minim, $e^* = v_s v_{j^*}$ unde $v_{j^*} \in V(T_{j^*}^k)$ $j^* \neq s$.
- $\mathcal{T}^{k+1} = (\mathcal{T}^k \setminus \{T_s^k, T_{j^*}^k\}) \cup T$ unde T este arborele obținut din T_s^k și $T_{j^*}^k$ la care adăugăm muchia e^*.



Se verifică imediat că noua familie este formată din arbori disjuncți care partiționează mulțimea de vîrfuri ale grafului G. Dacă alegerea muchiei e^* nu este posibilă, atunci rezultă că G nu este conex și deci problema $(P1)$ nu are soluție.

Evident, familia $\mathcal{T}^{n-1}$ construită în pasul $n-2$ este formată dintr-un singur arbore T_1^{n-1}.

Teoremă. 3. *Dacă $G = (V, E)$ este un graf conex cu $V = \{1, 2, \ldots, n\}$ atunci T_1^{n-1} construit de algoritmul descris mai sus este arbore parţial de cost minim.*

Demonstraţie. Vom arăta că $\forall k \in \{0, 1, .., n-1\}$ $\exists T^*$ arbore parţial de cost minim al lui G, astfel încît $E(\mathcal{T}^k) = \cup_{i=1}^{n-k} E(T_i^k) \subseteq E(T^*)$.

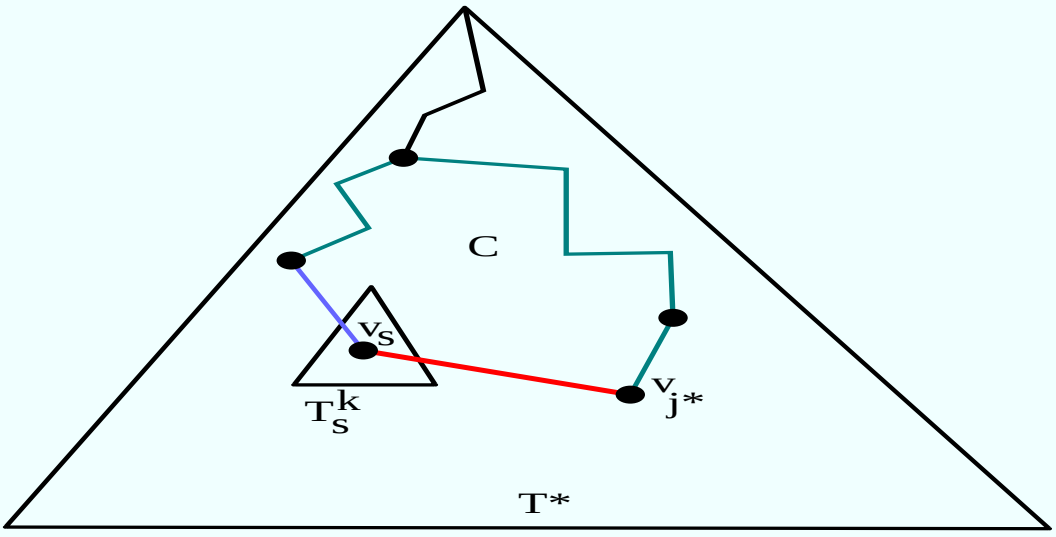
În particular, pentru $k = n-1$, $E(\mathcal{T}^{n-1}) = E(T_1^{n-1}) \subseteq E(T^*)$ va implica $T_1^{n-1} = T^*$ (cei doi arbori avînd același număr de muchii, rezultă că incluziunea are loc cu egalitate) și teorema e demonstrată.

Pentru $k = 0$, afirmaţia este trivial adevărată: $E(\mathcal{T}^0) = \emptyset$ și din conexiunea grafului G, $\mathcal{T}_G$ este nevidă deci există T^* soluţie a problemei $P1$.

Dacă afirmaţia este adevărată pentru $0 \geq k \leq n-2$, atunci avem $E(\mathcal{T}^k) \subseteq E(T^*)$ (T^* arbore parţial de cost minim) şi
$E(\mathcal{T}^{k+1}) = E(\mathcal{T}^K) \cup \{e^*\}$.

Dacă $e^* \in E(T^*)$, atunci, evident, $E(\mathcal{T}^{k+1}) \subseteq E(T^*)$ şi deci afirmaţia are loc pentru $k+1$.

Presupunem, deci, că $e^* \notin E(T^*)$. Atunci $T^* + \{e^*\}$ conţine exact un circuit C ce trece prin muchia $e^* = v_s v_{j^*}$. Cum $v_{j^*} \notin V(T_s^k)$ rezultă că C va conţine o muchie $e_1 \neq e^*$ cu o extremitate în $V(T_s^k)$ şi cealaltă în $V \setminus V(T_s^k)$. Din alegerea muchiei e^*, avem $c(e^*) \leq c(e_1)$ şi $e_1 \in E(T^*) \setminus E(\mathcal{T}^k)$.



Fie $T^1 = (T^* + \{e^*\}) - \{e_1\}$.

$T^1 \in \mathcal{T}_G$ (este conex și are n-1 muchii).

Din construcție, avem că $E(\mathcal{T}^{k+1}) \subseteq E(T^1)$.

În plus, $c(T^1) = c(T^*) + c(e^*) - c(e_1) \leq c(T^*)$ deci $c(T^1) = c(T^*)$, adică T^1 este de cost minim și teorema e demonstratā.

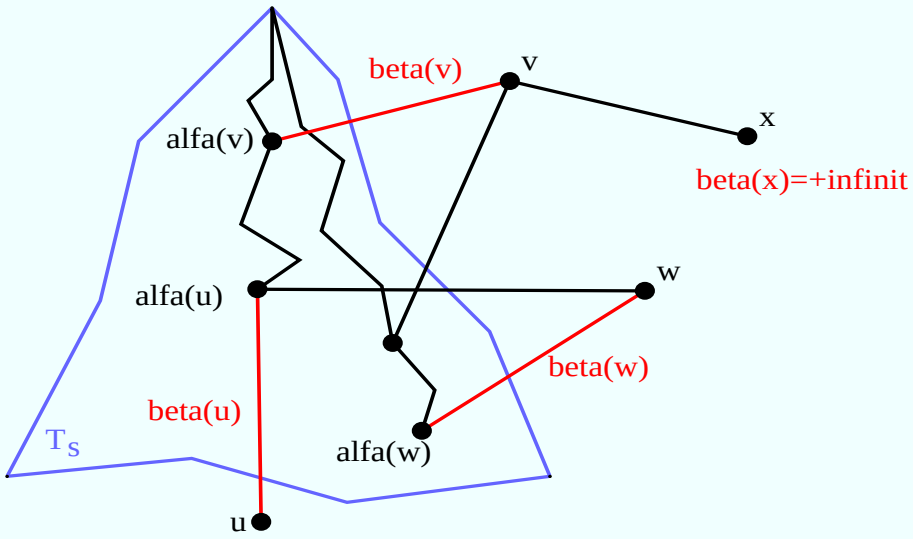
Observații:
1^0 Demonstrația anterioarā rāmîne valabilā pentru funcții de cost $c : \mathcal{T}_G \to \mathbf{R}$ astfel încît: $\forall T \in \mathcal{T}_G, \forall e \in E(T), \forall e' \notin E(T)$

$$c(e') \leq c(e) \Rightarrow c((T + e') - e) \leq c(T).$$

2^0 În algoritmul descris nu s-a precizat modul de alegere al arborelui T^k. Vom considera, în continuare douā strategii de alegere a acestui arbore.

Algoritmul lui Prim(1957) (implementarea este datoratā lui **Dijkstra,** 1961).

Arborele T_s^k va fi întotdeauna arborele cu cele mai multe vîrfuri dintre arborii familiei curente.
Rezultā deci, cā la fiecare pas $k > 0$, vom avea un arbore cu $k + 1$ vîrfuri, ceilalţi $n - k - 1$ avînd cîte un singur vîrf. Notām $T_s = (V_s, E_s)$ arborele curent. Considerām vectorii $\alpha[1..n]$ cu componente din V şi $\beta[1..n]$ cu componente reale, cu următoarea semnificaţie:
(S)
$\forall j \in V - V_s, \quad \beta[j] = c(\alpha[j]j) = \min\{c(ij) \mid i \in V_s, ij \in E\}.$



Descrierea algoritmului:

1. $V_s :\leftarrow \{s\}$; ($s \in V$, oarecare)
$E_s \leftarrow \emptyset$;
for $v \in V \setminus \{s\}$ **do** { $\alpha[v] := s; \beta[v] := c(sv)$};
// dacă $ij \notin E$ atunci $c(ij) = \infty$).

2. **while** $V_s \neq V$ **do**
{ determină $j^* \in V \setminus V_s$ a.î.
$\beta[j^*] = \min\{\beta[j] \mid j \in V - V_s\}$;
$V_s \leftarrow V_s \cup \{j^*\}$;
$E_s := E_s \cup \{\alpha[j^*]j^*\}$;
for $j \in V - V_s$ **do**
if $\beta[j] > c[j^*j]$ **then**
{ $\beta[j] \leftarrow c[j^*j]$;
$\alpha[j] :\leftarrow j^*$
}
}

Se observă că (S) este satisfăcută de inițializările pasului 1, iar în pasul 2, se respectă, pe de o parte, strategia generală de alegere a muchiei de cost minim cu exact o extremitate în V_s (alegerea lui j^*) și pe de altă parte se menține valabilitatea condiției (S) pentru iterația următoare (testul asupra valorii curente a lui $\beta[j]$).

Complexitatea algoritmului este $O(n-1)+O(n-2)+\ldots+O(1)=O(n^2)$ dată de operaţiile din pasul 2 necesare determinării minimului şi actualizării tabloului β. Se poate introduce testul de conexiune a grafului, după determinarea lui $\beta[j^*]$. Algoritmul este recomandat în cazul grafurilor cu multe muchii, $m=O(n^2)$.

Algoritmul lui Kruskal (1956)

În metoda generală prezentată, se va alege la fiecare pas drept arbore T_s^k **unul din cei doi arbori cu proprietatea că sînt "uniţi" printr-o muchie de cost minim printre toate muchiile cu extremităţile pe arbori diferiţi.**

Această alegere, complicată la prima vedere, se realizează simplu prin sortarea muchiilor grafului nedescrescător în raport cu costurile şi apoi prin examinarea în mod "greedy" a listei obţinute.

Dacă notăm cu $T=E(\mathcal{T}^k)$, atunci algoritmul poate fi descris, astfel:

1. Sortează $E = (e_1, e_2, \ldots, e_m)$ astfel încît:
   $c(e_1) \leq c(e_2) \leq \ldots \leq c(e_m)$.
   1.2 $T \leftarrow \emptyset; i \leftarrow 1;$
2. **while** $i \leq m$ **do**
   { **if** $\langle T \cup \{e_i\}\rangle_G$ nu are circuite **then**
   $T \leftarrow T \cup \{e_i\}$;
   $i++$
   }

Evident, pasul 1 necesită $O(m \log n)$ operaţii. Pentru realizarea eficientă a testului din pasul 2 va fi necesar să reprezentăm la fiecare pas k (din metoda generală) $V(T_1^k), V(T_2^k), \ldots, V(T_n^k)$ şi să testăm dacă muchia e_i curentă are ambele extremităţi în aceeaşi muţime. Se vor folosi pentru reprezentarea acestor mulţimi, arbori (care nu sînt în general subarbori ai lui G).

Fiecare astfel de arbore va avea un vîrf, numit *rădăcină*, care va desemna mulţimea de vîrfuri ale lui G pe care o memorează.

Vom folosi o funcţie $find(v)$ care determină în ce mulţime este vîrful v, adică întoarce rădăcina arborelui care memorează mulţimea de vîrfuri la care aparţine v.

Pentru realizarea reuniunilor (disjuncte) de mulţimi de vîrfuri care apar în transformarea familiilor $\mathcal{T}^k$ (din metoda generală) vom folosi o procedură $\mathbf{union(v, w)}$ cu semnificaţia:
"mulţimile de vîrfuri (diferite şi disjuncte) la care aparţin v şi w se reunesc în una singură".
Cu aceste proceduri, pasul 2 al algoritmului se scrie:

2. **while** $i \leq m$ **do**
   { fie $e_i = vw$;
   if $find(v) \neq find(w)$ **then**
   { $union(v, w)$;
   $T \leftarrow T \cup \{e_i\}$
   }
   $i++$
   }

Complexitatea algoritmului va depinde de modul de implementare a funcţiei $find$ şi procedurii $union$.

Soluţia I^a. Considerām tabloul $rad[1..n]$ cu componente din V cu semnificaţia
$rad[v]$ **=rādācina arborelui ce memorează mulţimea la care aparţine vîrful** v**.**

Adāugām pasului 1, iniţializarea

1.3 **for** $v \in V$ **do** $rad[v] \leftarrow v$;

care corespunde familiei $\mathcal{T}^0$. Funcţia $find$ are în acest caz complexitatea $O(1)$ şi este:

function $find(v : V)$**;**
return $rad[v]$

Procedura $union$ necesită $O(n)$ operaţii:

procedure $union(v, w : V)$**;**
    variabilă locală $i : V$;
    for $i \in V$ **do**
        if $rad[i] = rad[v]$ **then** $rad[i] := rad[w]$

Pasul 2 al algoritmului necesită $O(m)$ apeluri ale funcţiei $find$ (exact m aşa cum l-am descris, sau $O(m)$ dacă se introduce şi un test asupra cardinalului mulţimii T curente).
În secvenţa acestor $O(m)$ apeluri ale funcţiei $find$ se vor intercala exact $n - 1$ apeluri ale procedurii $union$.

Rezultă că în total pasul 2 necesită $O(m + (n-1)O(m)) = O(n^2)$ operaţii.

Deci, complexitatea algoritmului este $O(max(m \log n, n^2))$.
Dacă graful este plin, $m = O(n^2)$, se observă că acest algoritm este mai puţin eficient decît cel al lui Prim.

Soluţia a II^a. Considerām $pred[1..n]$ un tablou întreg cu interpretarea :
"$pred[v]$= **vîrful dinaintea lui v de pe drumul unic la v, de la rādācina arborelui care memorează mulţimea la care aparţine v;** $pred[v] = 0 \Leftrightarrow v$ **este rādācina arborelui".**

Adāugām în pasul 1, iniţializarea

1.3 **for** $v \in V$ **do** $pred[v] \leftarrow 0$;

Modificām şi pasul 2 al algoritmului astfel:

2. **while** $i \leq m$ **do**
{ fie $e_i = vw$;
$x :\leftarrow find(v)$; $y \leftarrow find(w)$;
if $x \neq y$ **then** $union(x, y)$**;**
{ $union(x, y)$;
$T \leftarrow T \cup \{e_i\}$
}
$i++$
}

Deci, procedura $union$ va fi apelată numai pentru argumente reprezentînd rădăcini de arbori diferiţi:

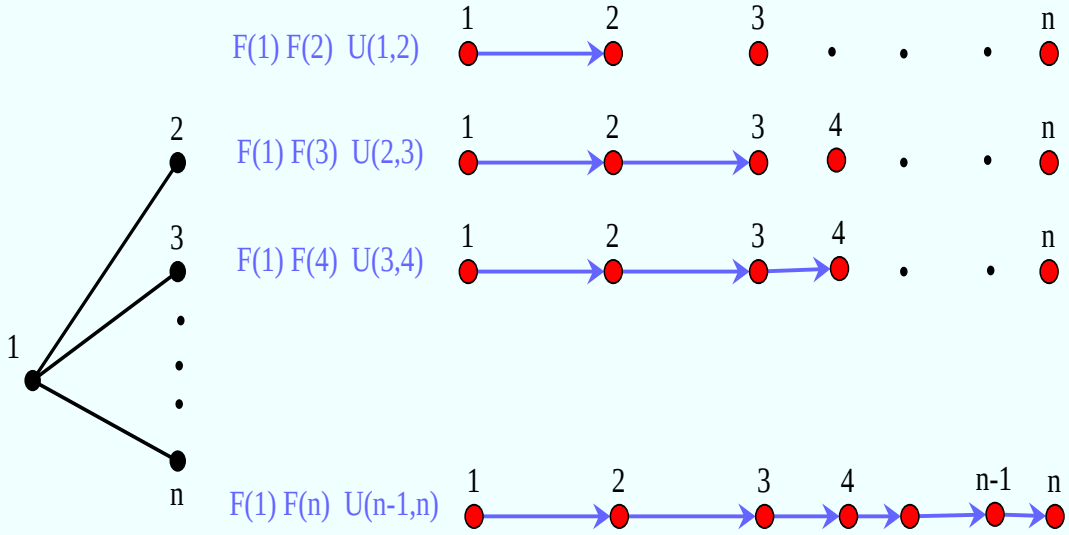
procedure $union(v, w : V)$**;**
    $pred[v] \leftarrow w$

Evident, complexitatea procedurii $union$ este $O(1)$. Funcţia, $find$ este în acest caz mai complicată:

function $find(v : V)$**;**
    variabilă locală $i : V$;
    $i \leftarrow v$;
    while $pred[i] > 0$ **do** $i \leftarrow pred[i]$**;**
    return i

Complexitatea lui $find(v)$ este $O(h(v))$ unde $h(v)$ este lungimea drumului (din arbore) de la vîrful v la rădăcina arborelui care-l conţine pe v.

Dacă graful G este $K_{1,n-1}$ desenat mai jos, și lista ordonată a muchiilor $E = \{12, 13, \ldots, 1n\}$, atunci execuția algoritmului provoacă următorul șir de apeluri ale procedurii $union(U)$ și funcției $find(F)$:
$F(1), F(2), U(1,2), F(1), F(3), U(2,3), \ldots, F(1), F(n), U(n-1,n)$.



Apelurile $F(i)(i > 1)$ și $U(i, i+1)$ $i \geq 1$ necesită în total $O(n)$ operații. Șirul de $F(1)$ necesită însă $O(1) + O(2) + \cdots + O(n-1) = O(n^2)$ operații.

Este deci posibil ca pasul 2 al algoritmului în această implementare să fie de complexitate $\Omega(n^2)$ chiar dacă graful este rar.

Deficienţa acestei implementări este datoratā posibilităţii ca în procedura $union$ sā declarām rādācinā nouā pentru cei doi arbori pe cea a celui cu mai puţine vîrfuri, ceea ce are ca efect posibilitatea ca $h(v)$ sā devinā mare ($O(n)$) pe parcursul algoritmului.

Acest defect poate fi evitat dacā, la execuţia lui $union$ ţinem seama de cardinalul celor douā mulţimi.
Se poate memora cardinalul unei mulţimi în componenta tabloului $pred$ corespunzātoare rādā cinii arborelui care memorează acea mulţime. Mai precis, considerām iniţializarea

1.3 **for** $v \in V$ **do** $predd[v] \leftarrow -1$;

şi modificām procedura $union$ astfel încît sā asigurām îndeplinirea condiţiei

$pred[v] < 0 \Leftrightarrow v$ **este rādācinā a unui arbore şi** $-pred[v]$ **este cardinalul mulţimii memorate în el.**

Procedura $union$ are, în acest caz, tot complexitatea $O(1)$, dar selectează drept nouă rădăcină pe cea care corespunde arborelui cu mai multe vîrfuri:

procedure $union(v, w : V)$**;**
    //v și w sunt rădăcini
    variabila locală întreagă t

    $t \leftarrow pred[v] + pred[w]$;
    if $pred[v] > pred[w]$ **then**
    { $pred[v] \leftarrow w; pred[w] \leftarrow t$ }
    else { $pred[w] \leftarrow v; pred[v] \leftarrow t$ }

Cu această implementare a funcției $find$ și procedurii $union$, pe tot parcursul algoritmului are loc:

$(*)$ $\quad \forall v \in V \quad -pred[find(v)] \geq 2^{h(v)}$

(reamintim că $h(v)$ notează lungimea drumului de la v la rădăcina $find(v)$ a arborelui ce memorează v).

După inițializarea 1.3, $\forall v \in V \ h(v) = 0$ și $find(v) = v$ iar $-pred[v] = 1$, deci (*) are loc cu egalitate.
Dacă, înaintea unei iterații din pasul 2, (*) are loc, atunci, dacă în acea iterație nu se execută $union$, nu se modifică tabloul $pred$ și deci (*) rămîne valabilă și după execuție.
Presupunem prin urmare că se apelează $union(x, y)$ și că se execută $pred[y] := x$.
Aceasta înseamnă că înaintea acestei iterații avem $-pred[x] \geq -pred[y]$. Să observăm că singurele vîrfuri v cărora li se modifică $h(v)$ după execuția iterației curente sînt cele care înaintea iterației satisfăceau $find(v) = y$, pentru care aveam $-pred[y] \geq 2^{h(v)}$.

Dupa execuția iterației avem $h'(v) = h(v) + 1$ iar $find'(v) = x$, și deci trebuie să verificam că $-pred'[x] \geq 2^{h'(v)}$. Avem $-pred'[x] = -pred[x] - pred[y] \geq 2 \cdot (-pred[y]) \geq 2 \cdot 2^{h(v)} = 2^{h(v)+1} = 2^{h'(v)}$.

Rezulta că (*) are loc pe tot parcursul algoritmului, deci, prin logaritmare obţinem
$\forall v \in V \ h(v) \leq \log(-pred[find[v]]) \leq \log n$.

Complexitatea pasului 2 va fi deci $O(n - 1 + 2m \log n) = O(m \log n)$ şi deci tot algoritmul are complexitatea $O(m \log n)$ ceea ce-l face superior algoritmului lui Prim pentru grafuri rare.

Soluţia a III^a. Complexitatea pasului 2, cu implementarea precedentă, este datorată apeluri lor succesive ale lui $find$.

Tarjan (1976) a propus ca fiecare apel al lui $find$ care necesită parcurgerea unui drum de lungime mai mare decît 1, să "distrugă" acest drum, aducîndu-i vîrfurile drept descendenţi imediaţi ai rădăcinii, cu scopul ca apelurile viitoare ale lui $find$ pentru aceste vârfuri să nu mai consume timp. Mai precis, avem

function $find(v : V)$**;**
variabile întregi locale i, j, k;
$i \leftarrow v$;
while $pred[i] > 0$ **do** $i \leftarrow pred[i]$**;**
$j \leftarrow v$;
while $pred[j] > 0$ **do**
$\{\ k \leftarrow pred[j];\ pred[j] \leftarrow i;\ j \leftarrow k;\ \}$
return i

Dacă $A : \mathbf{NxN} \rightarrow \mathbf{N}$ este funcţia lui **Ackermann** dată de:
$A(i, o) = 0\ \forall i \geq 0$;
$A(i, 1) = 2\ \forall i \geq 1$;
$A(0, x) = 2x\ \forall x \geq 0$;
$A(i + 1, x + 1) = A(i, A(i + 1, x))\ \forall i \geq 0\ \forall x \geq 1$,
atunci, considerînd $\forall m \geq n > 0$

$$\alpha(m, n) = \min\{z \mid A\big(z, 4\lceil m/n \rceil\big) \geq \log n, z \geq 1\}$$

avem:

Complexitatea pasului 2, utilizînd $union$ **din soluţia a II-a şi** $find$ **descris mai sus, este** $O(m \cdot \alpha(m, n))$**.**

Notăm că $\alpha(m, n)$ creşte extrem de încet (pentru valorile practice ale lui $n, \alpha(m, n) \leq 3$) şi deci se poate presupune ca această ultimă implementare este practic liniară (în raport cu m).

IV. Cuplaje (de cardinal maxim).

Fie $G = (V, E)$ un (multi)graf. Dacă $A \subseteq E$ și $v \in V$, vom nota cu
$d_A(v) = |\{e \mid e \in A, e \text{ incidentă cu } v\}|$,
adică gradul vîrfului v în graful parțial $< A >_G$.

Definiție. Se numește **cuplaj** (sau **mulțime independentă de muchii)** al grafului G, orice mulțime M de muchii cu proprietatea că $d_M(v) \leq 1, \ \forall v \in V$.

Vom nota cu $\mathcal{M}_G$ familia cuplajelor grafului G:

$$\mathcal{M}_G = \{M \mid M \subseteq E, M \text{ cuplaj în } G\}.$$

Se observă că $\mathcal{M}_G$ satisface proprietățile:

i) $\emptyset \in \mathcal{M}_G$.
ii) $M \in \mathcal{M}_G, \ M' \subseteq M \ \Rightarrow \ M' \in \mathcal{M}_G$.

Dacă $M \in \mathcal{M}_G$ atunci un vîrf $v \in V$ cu $d_M(v) = 1$ se numește **saturat** de cuplajul M. Mulțimea $\mathrm{S(M)}$ **a vîrfurilor saturate de cuplajul** M **în graful** G**,** satisface $|S(M)| = 2|M|$.

Dacă $d_M(v) = 0$, atunci v se numește **expus** față de cuplajul M. Mulțimea $\mathbf{E(M)}$ **a vîrfurilor expuse față de cuplajul** M satisface $E(M) = V - S(M)$ și $|E(M)| = |V| - 2|M|$.

Problema "cuplajului maxim":

P1 *Dat* $G = (V, E)$ *un graf, să se determine* $M^* \in \mathcal{M}_G$ *astfel încît*

$$|M^*| = \max\{|M| \mid M \in \mathcal{M}_G\}.$$

(Vom nota cu $\nu(G) = \max\{|M| \mid M \in \mathcal{M}_G\}$).

Problema cuplajului maxim este strîns legată de **problema acoperirii minime.**
Definiţie. Se numește **acoperire** (a vîrfurilor cu muchii) în graful G orice mulțime $F \subseteq E$ de muchii cu proprietatea că $d_F(v) \geq 1 \ \forall v \in V$.

$\mathcal{F}_G = \{F \mid F \subseteq E, \ F$ acoperire în $G\}$ notează familia acoperirilor grafului G.

$\mathcal{F}_G \neq \emptyset \Leftrightarrow G$ nu are vîrfuri izolate (atunci, măcar E este o acoperire).

Problema acoperirii minime este:
P2 *Dat $G = (V, E)$ un graf, să se determine $F^* \in \mathcal{F}_G$ astfel încît*

$$|F^*| = \min\{|F| \mid F \in \mathcal{F}_G\}.$$

Teoremă. 1. (Norman-Rabin 1959) *Fie $G = (V, E)$ un graf fără vîrfuri izolate, de ordin n. Dacă M^* este un cuplaj de cardinal maxim în G, iar F^* o acoperire de cardinal minim în G, atunci*

$$|M^*| + |F^*| = n.$$

Demonstraţie: a) Fie M^* un cuplaj de cardinal maxim. Considerăm următorul algoritm:

$F \leftarrow M^*$;
for $\forall v \in E(M)$ **do**
{ determină $v' \in S(M^*)$ astfel încît $vv' \in E$;
$F \leftarrow F \cup \{vv'\}$
}

Să observăm că pentru $\forall v \in E(M^*)$, cum G nu are vîrfuri izolate, există o muchie incidentă cu v, și cum M^* este maximal în raport cu incluziunea, această muchie are cealaltă extremitate din $S(M^*)$. Mulțimea de muchii F astfel construită este o acoperire și în plus $|F| = |M^*| + |E(M^*)| = |M^*| + n - 2|M^*| = n - |M^*|$.

Rezultă că $|F^*| \leq |F| = n - |M^*|$.

b) Fie F^* o acoperire de cardinal minim. Considerăm următorul algoritm:

$M \leftarrow F^*$
while $\exists v \in V : \ d_M(v) > 1$ **do**
{ determină $e \in M$ incidentă cu v;
$M \leftarrow M - e$
}

Algoritmul construiește un cuplaj M în G.

Dacă muchia e incidentă cu v, care se înlătură din M într-o iterație while, este $e = vv'$,

atunci $d_M(v') = 1$ și deci în pasul următor $d_M(v')$ va fi zero, adică la orice îndepărtare a unei muchii din mulțimea M curentă de muchii se obține un vîrf expus față de cuplajul final M (dacă vîrful v' ar fi incident cu încă o muchie, atunci din acoperirea inițială F^* se poate înlătura muchia e și să obținem tot o acoperire, contrazicînd alegerea lui F^*).

Deci dacă M este cuplajul construit de algoritm avem: $|F^*| - |M| = |E(M)| = n - 2|M|$, adică

$$|F^*| = n - |M| \geq n - |M^*|.$$

Din (a) și (b) rezultă concluzia teoremei. Demonstrația făcută arată, chiar mai mult, că problemele (P1) și (P2) sînt polinomial echivalente, cuplajul M și acoperirea F construite fiind și ele soluții optime respectiv pentru cele două probleme.

Dacă vom considera matricea de incidență (vîrf - muchie) a grafului G cu n vîrfuri și m muchii $B = (b_{ij})_{n\times m}$ cu $b_{ij} = 1$ dacă vîrful i și muchia j sînt incidente și $b_{ij} = 0$ altminteri (într-o ordonare fixată a vîrfurilor și muchiilor), și dacă notăm cu e_p vectorul p-dimensional cu toate componentele 1, atunci cele două probleme se scriu analitic astfel

P1' $\max\{e_m^T x | Bx \leq e_n, x \geq 0, x_i \in \{0,1\}\ i = \overline{1,m}\}$

P2' $\min\{e_m^T x | Bx \geq e_n, x \geq 0, x_i \in \{0,1\}\ i = \overline{1,m}\}$

și teorema 1 oferă o egalitate min-max interesantă.

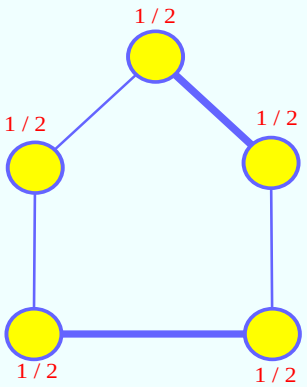
În cele ce urmează ne vom ocupa numai de problema (P1); în plus vom presupune că G nu are vîrfuri izolate.

Dacă s-ar încerca rezolvarea problemei (P1') apelînd la problema de programare liniară asociată

$$(LP1') \qquad \max\{e'_m x \mid Bx \leq e_n, x \geq 0\}$$

se constată că soluțiile optime pot să nu fie cu componente întregi și, chiar mai mult, valoarea maximă determinată de (LP1') să fie superioară lui $\nu(G)$.

Cel mai simplu exemplu în acest sens este $G = C_{2n+1}$. Evident, $\nu(C_{2n+1}) = n$ și totuși $x_i = \frac{1}{2}\ \forall i = 1, 2n+1$ este o soluție optimă a lui (LP1') corespunzătoare, cu valoarea optimă $n + \frac{1}{2} > n$.



Rezultă că existența circuitelor impare în graful G poate provoca dificultăți în rezolvarea problemei (P1). Mai precis, avem următoarea teoremă:

Teoremă. 2. (Balinski 1971) *Vîrfurile politopului* $Bx \leq e_n,\ x \geq 0,\ x \in \mathbf{R}^m$, *au coordonatele* $0, 1$ *și* $\frac{1}{2}$. *Coordonatele* $\frac{1}{2}$ *apar dacă și numai dacă* G *are circuite impare.*

Rezultă de aici că, în cazul grafurilor bipartite, problema (P1) este ușor de rezolvat: se apelează la problema de programare (LP1') și soluția găsită este soluție optimă pentru problema P1 (reprezentînd vectorul caracteristic al unui cuplaj). Adaptarea combinatorie a algoritmului simplex din programarea liniară, direct pe graful bipartit considerat (în scopul unei economii de memorie, tablourile simplex sînt reprezentate implicit) a condus la așa numita "metodă ungară" de rezolvare a problemei (P1) pentru grafurile bipartite. Nu vom prezenta acest algoritm, preferînd descrierea unuia mai performant datorat lui *Hocroft și Karp (1973)*.

Totuși teorema de dualitate din programarea liniară, precum și integritatea soluțiilor optime, pot oferi demonstrații instantanee pentru teoreme de caracterizare a soluțiilor optime ale problemei (P1) în cazul grafurilor bipartite:

Teoremă. 3. (Hall, 1935) *Fie* $G = (R, S; E)$ *un graf bipartit. Există un cuplaj care saturează vîrfurile lui* R *dacă și numai dacă*

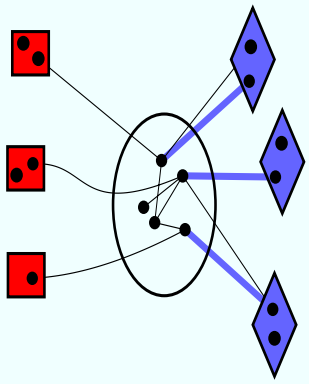
$$|N_G(A)| \geq |A| \qquad \forall A \subseteq R.$$

Teoremă. 4. (Konig,1930) *Fie* $G = (R, S; E)$ *un graf bipartit. Cardinalul maxim al unui cuplaj este egal cu numărul minim de vîrfuri prin îndepărtarea cărora se obține graful nul:*

$$\nu(G) = n - \alpha(G)$$

Revenind la problema (P1) cu G un graf oarecare și observînd că $\nu(G) \leq \frac{1}{2}|V(G)|$, rezultă că este interesant de caracterizat grafurile cu proprietatea că admit un cuplaj M astfel încît $S(M) = V(G)$. Un astfel de cuplaj se numește **cuplaj perfect** sau **1-factor**.

Este evident că un graf care are un cuplaj perfect are în orice componentă conexă un număr par de vîrfuri. Mai mult, dacă $S \subseteq V(G)$ atunci, în ipoteza că G are un cuplaj perfect, va trebui ca pentru fiecare componentă conexă cu un număr impar de vîrfuri a grafului $G - S$ să existe o muchie în cuplajul perfect cu o extremitate în S și cealaltă în componenta conexă impară. Rezultă că numărul componentelor conexe impare ale grafului $G-S$ nu poate depăși $|S|$.



Dacă pentru un graf oarecare H notăm cu $q(H)$ numărul componentelor conexe impare ale lui H, atunci observaţia anterioară arată că o condiţie necesară pentru ca G să aibă un cuplaj perfect este ca $q(G-S) \leq |S|\ \forall S \subseteq V(G)$.

Să observăm că atunci cînd $S = \emptyset$ condiţia anterioară cere ca orice componentă conexă a lui G să aibă un număr par de vîrfuri. Condiţia este și suficientă, așa cum rezultă din următoarea teoremă.

Teoremă. 5. (Tutte, 1947) *Un graf $G = (V, E)$ are un cuplaj perfect dacă și numai dacă*

$$(T) \qquad q(G - S) \leq |S| \quad \forall S \subseteq V.$$

Demonstraţie. Arătăm, prin inducţie după $n = |V|$, că, dacă $G = (V, E)$ satisface (T), atunci G are un cuplaj perfect.

Cum teorema se verifică imediat pentru $n = 1, 2$, vom presupune în pasul inductiv că orice graf G' cu $|G'| < n$ și care satisface (T) are un cuplaj perfect.

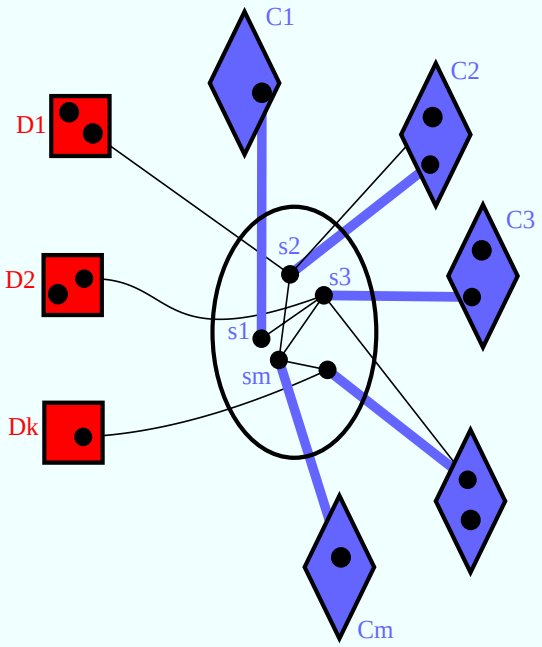
Fie $G = (V, E)$ cu $|V| = n$ și care satisface (T).

Alegem $S_0 \subseteq V(G)$ astfel încît în (T) avem egalitate: $q(G - S_0) = |S_0|$ și oricare ar fi S astfel încît $S_0 \subseteq S,\ S_0 \neq S$ avem $q(G-S) < |S|$.

Cum G satisface T, rezultă că G are numai componente conexe de cardinal par. Într-o astfel de componentă conexă, există întotdeauna un vîrf v_0 care nu este vîrf de articulaţie (orice vîrf pendant al unui arbore parţial) și se verifică imediat că $q(G - \{v_0\}) = 1 = |\{v_0\}|$.

Prin urmare familia $\{S \mid S \subseteq V,\ q(G - S) = |S|\}$ este nevidă și orice element maximal al ei poate fi considerat S_0.

Fie $m = |S_0| > 0$ și $C_1, \ldots, C_m$ componentele conexe de cardinal impar ale lui $G - S_0$, iar $D_1, \ldots, D_k$ componentele conexe de cardinal par ($k \geq 0$) ale lui $G - S_0$.

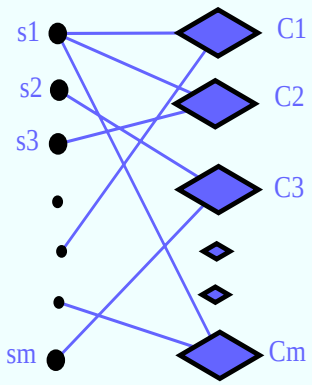


Vom construi un cuplaj perfect al grafului G compus din : **a)** - cîte un cuplaj perfect în fiecare componentă conexă pară D_i;
b) - un cuplaj format din m muchii, muchia e_i $(i = 1, m)$ avînd o extremitate $s_i \in S_0$ și cealaltă $v_i \in C_i$;
c) - cîte un cuplaj perfect în fiecare subgraf $[C_i - v_i]_G$.

a) Pentru orice $i = 1, k$ graful $[D_i]_G$ are un cuplaj perfect.
În adevăr, cum $m > 0$, rezultă că $|D_i| < n$ și deci pentru a dovedi existența cuplajului perfect în $[D_i]_G$ este suficient să dovedim (din ipoteza inductivă) că $G' = [D_i]_G$ satisface (T).

Fie $S \subseteq D_i$. Dacă $q(G' - S) > |S|$ atunci $q(G - S_0 \cup S) = q(G - S_0) + q(G' - S) = |S_0| + q(G' - S) > |S_0| + |S| = |S_0 \cup S|$, contrazicînd faptul că G satisface (T).
Avem $q(G' - S) \leq |S|$ și deci G' satisface T.

b) Fie $H = (S_0, \{C_1, \ldots, C_m\}; E')$ graful bipartit avînd o clasă a bipartiției S_0, cealaltă clasă, mulțimea componentelor conexe impare ale lui $G - S_0$, iar mulțimea muchiilor E' formată din perechile sC_i $(s \in S;\ i = 1, m)$ cu proprietatea că există $v \in C_i$ cu $sv \in E(G)$.



Acest graf are un cuplaj perfect. În adevăr, este suficient să arătăm că H satisface condiția lui Hall de existanță a unui cuplaj M_0 ce saturează vîrfurile lui $\{C_1, \ldots, C_m\}$:
$\forall A \subseteq \{C_1, \ldots, C_m\},\ |N_H(A)| \geq |A|$
și cum $|S_0| = m$ va rezulta că M_0 este cuplaj perfect în H.

Fie $A \subseteq \{C_1, \ldots, C_m\}$.
Observăm că $B = N_H(A) \subseteq S_0$ și din definiția lui H, în graful G nu avem muchii de la un vîrf $v \in S_0 - B$ la un vîrf $v \in C_i$ cu $C_i \in A$. Deci componentele conexe impare C_i din A ale grafului $G - S_0$ vor rămîne componente conexe impare și în $G - B$.

Rezultă că $q(G-B) \geq |A|$. Pe de altă parte G satisface condiția lui Tutte (T) deci $|B| \geq q(G-B)$. Am obținut deci $|B| \geq |A|$ adică $|N_H(A)| \geq |A|$.
Cum A a fost aleasă arbitrar, rezultă că H are un cuplaj perfect $M_0 = \{s_1v_1, s_2v_2, \ldots, s_mv_m\}$ cu $S_0 = \{s_1, \ldots, s_m\}$ și $v_i \in C_i$ $i = 1, m$.

c) Pentru orice $i \in \{1, \ldots, m\}$ graful $G' = [C_i - v_i]_G$ are un cuplaj perfect.
Folosind ipoteza inductivă, afirmația va rezulta dacă dovedim că G' satisface (T).
Fie $S \subseteq C_i - v_i$. Dacă $q(G'-S) > |S|$ atunci cum $q(G'-S) + |S| \equiv 0 \pmod 2$, rezultă că $q(G'-S) \geq |S| + 2$ și atunci considerînd $S' = S_0 \cup \{v_i\} \cup S$, avem $|S'| \geq q(G-S') = q(G - S_0) - 1 + q(G'-S) = |S_0| - 1 + q(G'-S) \geq |S_0| - 1 + |S| + 2 = |S'|$, adică $q(G-S') = |S'|$ ceea ce contrazice alegerea lui S_0 căci $S_0 \subset S'$.
Rezultă că $\forall S' \subseteq C_i - v_i$ $q(G'-S) \leq |S|$ deci G' are un cuplaj perfect.

Evident, cuplajul lui G obţinut prin reuniunea cuplajelor puse în evidenţă în a), b), şi c) de mai sus este un cuplaj perfect şi cu aceasta teorema este demonstratā.

Notām cā *Berge* (1958) a generalizat această teoremā stabilind cā

$$\nu(G) = \frac{1}{2}(|V(G)| - max_{S \subseteq V(G)}[q(G-S) - |S|]\ ).$$

Totuşi, algoritmii care rezolvā problema (P1) se bazează pe o caracterizare mai simplā a cuplajelor de cardinal maxim.

Fie $G = (V, E)$ un graf şi $M \in \mathcal{M}_G$ un cuplaj al sāu.

Definiţie: Se numeşte **drum alternat al lui** G **relativ la cuplajul** M orice drum

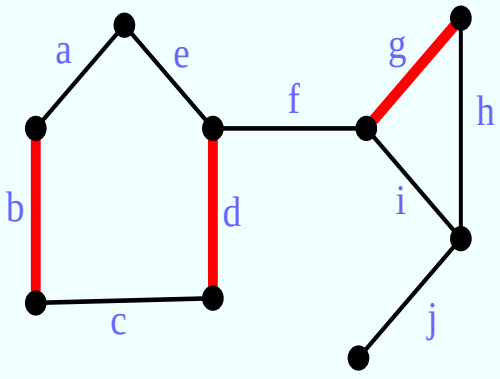
$$P : v_0, v_0v_1, v_1, \ldots, v_{k-1}, v_{k-1}v_k, v_k$$

a. î. $\forall i = \overline{1, k-1}\ \ \{v_{i-1}v_i, v_iv_{i+1}\} \cap M \neq \emptyset$.

Să observăm că, întrucît M este cuplaj, din definiţia dată rezultă că **dintre orice două muchii consecutive ale drumului P exact una aparţine cuplajului**
(muchiile lui P aparţin alternativ la M şi $E - M$).

Vom desemna, în cele ce urmează, prin P mulţimea muchiilor drumului P (pentru simplificarea notaţiilor).

Definiţie: Se numeşte **drum de creştere al lui G relativ la cuplajul M** un drum alternat cu extremităţile vîrfuri distincte, expuse relativ la cuplajul M.

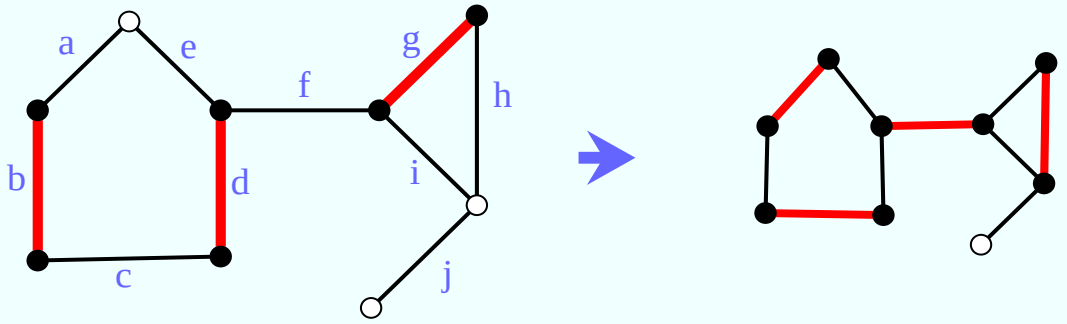


a, b, c, d - drum alternat par
f - drum alternat impar
j - drum de crestere
g, f, d - drum alternat impar
a, b, c, d, e - drum alternat inchis
a, b, c, d, f, g, h - drum de crestere

Observaţie: Din definiţie, rezultă că dacă P este un drum de creştere relativ la cuplajul M atunci $|P - M| = |P \cap M| + 1$.

Teoremă. 6. (Berge 1959) *Un cuplaj M este de cardinal maxim în graful G dacă şi numai dacă nu există în G drumuri de creştere relativ la M.*

Demonstraţie: Dacă M este un cuplaj de cardinal maxim şi P ar fi un drum de creştere în G relativ la M atunci $M' = P\Delta M = (P - M) \cup (M - P)$ este un cuplaj în G. (Construcţia lui $P\Delta M$ revine la interschimbarea muchiilor lui $M - P$ şi $P - M$ pe drumul P). În plus, $|M'| = |P \cap M| + 1 + |M - P| = |M| + 1$, contrazicînd alegerea lui M.

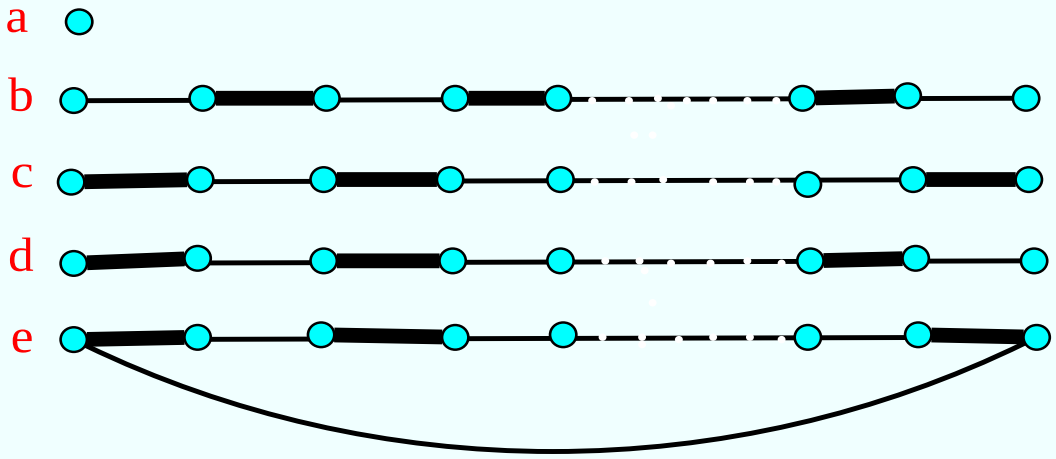


P= a, b, c, d, f, g, h - drum de crestere

(M-P) u (P-M)

Reciproc, dacă M este cuplaj în G și M nu admite drumuri de creștere, considerăm M^* un cuplaj de cardinal maxim. Vom arăta că $|M^*| = |M|$, ceea ce demonstrează teorema.

Fie G' graful parţial al lui G, $G' = (V, M \Delta M^*)$. Evident, $d_{G'}(v) \leq 2 \ \forall v \in V$ și deci componentele conexe ale lui G' sînt drumuri (eventual de lungime 0), sau circuite și avem următoarele posibilităţi (muchiile îngroșate sînt din M^* și cele subţiri, din M):



Situaţia b) nu poate avea loc, pentru că reprezintă un drum de creștere relativ la M^* care-i cuplaj de cardinal maxim. Situaţia c) nu are loc, întrucît M nu admite drumuri de creștere.

Deci, dacă notăm cu $m_M(C)$ numărul muchiilor din M ale componentei conexe C a lui G' și cu $m_{M^*}(C)$ numărul corespunzător, de muchii din M^* avem $m_M(C) = m_{M^*}(C)$. Rezultă că

$$|M - M^*| = \sum_C m_C(M) = \sum_C m_C(M^*) = |M^* - M|$$

(suma se face după toate componentele conexe C ale grafului G') și deci, $|M| = |M^*|$.

Această teoremă justifică următoarea strategie de construire a unui cuplaj de cardinal maxim:

a) fie M un cuplaj oarecare a lui G (eventual $M = \emptyset$);
b) **while** $\exists P$ drum de creștere relativ la M **do**
$M \leftarrow M \triangle P$

La fiecare iterație a ciclului while, cuplajul curent "crește" (cardinalul său se mărește cu o unitate) și deci în cel mult $\frac{n}{2}$ iterații se obține un cuplaj de cardinal maxim, care nu va admite drumuri de creștere.

Neajunsul acestui algoritm este acela că testul de oprire a ciclului - **inexistența drumului de creștere** - poate conduce la un număr exponențial de operații.

Primul care a demonstrat posibilitatea implementării acestui algoritm astfel încît numărul total de operații să fie polinomial în raport cu numărul de vîrfuri ale grafului G, a fost **Edmonds (1965),** obținînd astfel unul din primele rezultate ale teoriei complexității (cantitative) a algoritmilor.

În 1973 **Hopcroft și Karp**, fac o analiză mai detaliată a procesului succesiv de creșteri ale cuplajului curent, cu ipoteza suplimentară că se alege de fiecare dată **un drum de creștere cu număr minim de muchii** (printre toate drumurile de creștere posibile). Această idee stă la baza celui mai eficient algoritm, cunoscut, pentru rezolvarea problemei.

Lemă. 1. *Fie $M, N \in \mathcal{M}_G$, $|M| = r$, $|N| = s$ și $s > r$. Atunci în $M \Delta N$ există cel puțin $s - r$ drumuri de creștere ale cuplajului M, disjuncte ca vîrfuri.*

Demonstrație: Fie $G' = (V, M \Delta N)$ graful parțial secționat în G de $M \Delta N$.
Fie C_i $i = 1, p$ componentele conexe ale lui G'.
Definim pentru fiecare $i \in \{1, \ldots, p\}$

$$\delta(C_i) = |E(C_i) \cap N| - |E(C_i) \cap M|.$$

Observăm că:
$\delta(\mathbf{C_i}) \in \{-\mathbf{1}, \mathbf{0}, \mathbf{1}\}$ (M, N cuplaje, $\Rightarrow C_i$ sînt drumuri sau circuite);

$\delta(\mathbf{C_i}) = \mathbf{1}$ dacă și numai dacă C_i este un drum de creștere relativ la M.
În plus,

$$\sum_{i=1,p} \delta(C_i) = |N - M| - |M - N| = s - r.$$

Rezultă că există măcar $s - r$ C_i cu $\delta(C_i) = 1$, adică există măcar $s - r$ drumuri de creștere disjuncte ca vîrfuri (deci și ca muchii) conținute în $M \Delta N$.

Lemă. 2. *Dacă $\nu(G) = s$ și $M \in \mathcal{M}_G$ $|M| = r < s$, atunci există în G un drum de creștere relativ la M de lungime $\leq 2\lfloor r/(s-r) \rfloor + 1$.*

Demonstrație. Fie $N \in \mathcal{M}_G$ cu $|N| = s = \nu(G)$. Conform lemei precedente, vor exista $s - r$ drumuri de creștere disjuncte pe muchii, conținute în $M \Delta N$. Acestea au împreună cel mult r muchii din M. Rezultă că există unul care conține cel mult $\lfloor r/(s-r) \rfloor$ muchii din M, a cărui lungime este deci $\leq 2\lfloor r/(s-r) \rfloor + 1$.

Definiție Dacă $M \in \mathcal{M}_G$, se numește **drum minim de creștere al lui M în G** , un drum de creștere cu număr minim de muchii printre toate drumurile de creștere ale lui M în G.

Lemă. 3. *Fie $M \in \mathcal{M}_G$, P drum minim de creștere relativ la M, și P' drum de creștere al lui $M \Delta P$. Atunci, $|P'| \geq |P| + 2|P \cap P'|$.*

Demonstraţie: Fie $N = (M \Delta P) \Delta P'$.
Avem $M \Delta N = P \Delta P'$ şi $|N| = |M| + 2$.
Folosind lema 1 obţinem că există P_1, P_2 drumuri de creştere disjuncte ca muchii relativ la M conţinute în $M \Delta N$. Cum P este drum minim de creştere avem: $|P \Delta P'| \geq |P_1| + |P_2| \geq 2|P|$ deci $|P| + |P'| - 2|P \cap P'| \geq 2|P|$.

Considerăm următorul algoritm $(*)$:

$M_0 \leftarrow \emptyset$;
$M_{i+1} \leftarrow M_i \Delta P_i$
(P_i dr. minim de creşt. rel. la M_i; $i \geq 0$.)

Se obţine şirul de drumuri minime de creştere $P_0, P_1, \ldots, P_{\nu(G)-1}$.

Lemã. 4.
a) $\forall i = 1, \nu(G) - 2 \quad |P_i| \leq |P_{i+1}|$;
$|P_i| = |P_{i+1}| \Leftrightarrow P_i$ *şi* P_{i+1} *sînt disjuncte ca vîrfuri.*
b) $\forall i < j < \nu(G) - 1 \quad |P_i| = |P_j|$, *implică* P_i *şi* P_j *sînt disjuncte ca vîrfuri.*

Demonstrţie: a) Considerînd $P = P_i$ şi $P' = P_{i+1}$ în lema 3 se obţine $|P_{i+1}| \geq |P_i| + 2|P_i \cap P_{i+1}| \geq |P_i|$. Egalitatea are loc dacă şi numai dacă P_i şi P_{i+1} sînt disjuncte ca muchii, condiţie care, avînd în vedere alternanţa drumurilor, implică faptul că nu au vîrfuri comune.

b) rezultă aplicînd succesiv a).

Teoremă. 7. (Hopcroft, Karp 1973) *Fie G un graf şi $\nu(G) = s$. Numărul întregilor distincţi din şirul $|P_0|, |P_1|, \ldots, |P_{s-1}|$, construit în algoritmul de mai sus este cel mult $2\lceil\sqrt{s}\,\rceil + 2$.*

Demonstraţie: Fie $r = \lfloor s - \sqrt{s}\rfloor$. Atunci $|M_r| = r$ şi $|P_r| \leq 2\lfloor r/(s-r)\rfloor + 1 = 2\lfloor\lfloor s - \sqrt{s}\rfloor/(s - \lfloor s - \sqrt{s}\rfloor)\rfloor + 1 < 2\lceil\sqrt{s}\,\rceil + 1$.

Rezultă că oricare ar fi $i < r$, $|P_i|$ este unul din cei $\lceil\sqrt{s}\,\rceil + 1$ întregi impari $\leq 2\lceil\sqrt{s}\,\rceil + 1$. În subşirul $|P_r|, \ldots, |P_{s-1}|$ mai sînt cel mult $s - r \leq \lceil\sqrt{s}\,\rceil + 1$ întregi distincţi, deci în total avem cel mult $2\lceil\sqrt{s}\,\rceil + 2$ întregi distincţi.

Dacă algoritmul $(*)$ se descompune în etape, astfel încît la fiecare etapă se determină o mulţime maximală de drumuri minime de creştere disjuncte ca vîrfuri, din lema 4, rezultă că, în etapa următoare, lungimea drumurilor minime de creştere utilizate va creşte strict (altfel s-ar contrzice maximalitatea mulţimii de drumuri alese).

Utilizînd teorema 7, rezultă că numărul fazelor nu va depăşi $2\lceil\sqrt{\nu(G)}\rceil + 2$.
Rezultă că următorul algoritm pentru aflarea unui cuplaj maxim, într-un graf cu măcar o muchie:

0. $M \leftarrow \emptyset$;
1. **repeat**
   Determină $\mathcal{P}$ o familie maximală $(\subseteq)$
   de drumuri minime de creştere;
   for $P \in \mathcal{P}$ **do** $M \leftarrow M \Delta P$
   until $\mathcal{P} = \emptyset$.

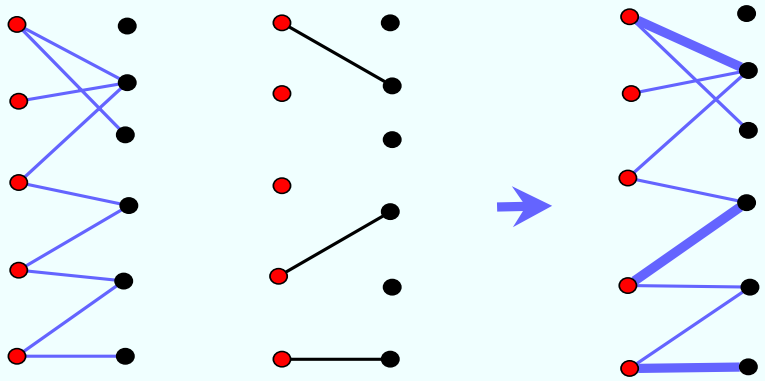
are complexitatea $O(\sqrt{n}A)$ unde A este complexitatea determinării familiei $\mathcal{P}$.

Hopcroft și Karp au arătat cum se poate implementa pasul 1 pentru un graf bipartit, astfel încît $A = O(m+n)$, deci s-a obținut un algoritm de complexitate $O(mn^{1/2})$ pentru aflarea unui cuplaj de cardinal maxim într-un graf bipartit.

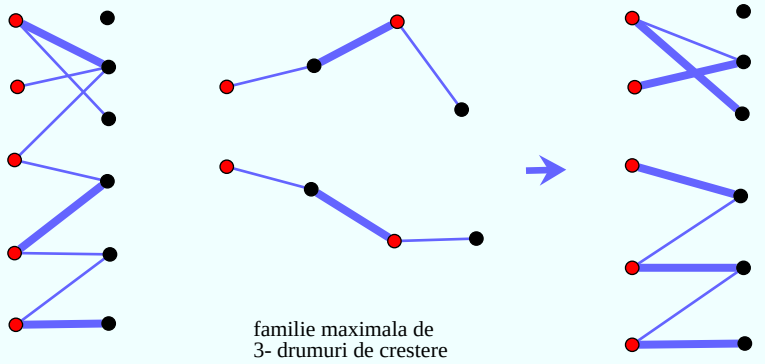
Pentru un graf oarecare, structurile de date necesare obținerii aceleeași complexități sînt mult mai elaborate și au fost descrise de **Micali și Vazirani 1980**. Considerăm, în continuare, numai cazul grafurilor bipartite.

Fie $G = (R, S; E)$ un graf bipartit și $M \in \mathcal{M}_G$. Pornind din una din clase, R de exemplu, se consideră mulțimea extremităților drumurilor de creștere posibile, $R \cap E(M)$, și din fiecare astfel de vîrf, se incepe construcția în paralel, de drumuri alternate.

Prima depistare a unui drum de creştere (sau, concluzia de inexistenţă) opreşte construcţia, oferind lungimea minimă a unui drum de creştere, şi un sistem evident de etichetare va permite depistarea familiei $\mathcal{P}$. Complexitatea $O(m+n)$ rezultă prin utilizarea listelor de adiacenţă. Exemplu:



familie maximala de
1- drumuri de crestere

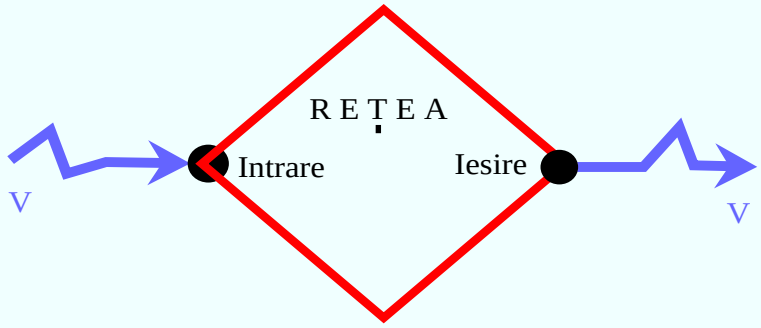


familie maximala de
3- drumuri de crestere

Cuplaj maxim !
toate virfurile rosii
sunt saturate

V. FLUXURI ÎN REȚELE

1. Problema fluxului maxim.



Data Reteaua, care este v MAXIM ?

Numim **rețea** (de transport) cu **intrarea** s și **ieșirea** t, 4-uplul $\mathbf{R} = (\mathbf{G}, \mathbf{s}, \mathbf{t}, \mathbf{c})$ unde:
- $G = (V, E)$ este un digraf,
- $s, t \in V$; $\quad s \neq t$; $\quad d_G^+(s) > 0$; $\quad d_G^-(t) > 0$,
- $c : E \to \mathbf{R}_+$; $c(e)$ este **capacitatea** arcului e.

Vom presupune că
$V = \{1, 2, \ldots, n\}$ $\quad (n \in \mathbf{N}^*)$ și că $|E| = m$.
Extindem funcția c la $c : V \times V \to \mathbf{R}_+$ prin

$$c((i,j)) = \begin{cases} c(ij) & \text{dacă } ij \in E \\ 0 & \text{dacă } ij \notin E \end{cases}$$

și vom nota $c((i,j)) = c_{ij}$.

Definiţie: Numim **flux** în reţeaua $R = (G, s, t, c)$ o funcţie $x : V \times V \to \mathbf{R}$, care satisface

$$(i) \qquad 0 \leq x_{ij} \leq c_{ij} \quad \forall ij \in V \times V$$

$$(ii) \qquad \sum_{j \in V} x_{ji} - \sum_{j \in V} x_{ij} = 0 \quad \forall i \in V - \{s, t\}.$$

Observaţii

1^0 Dacă $ij \in E$ atunci x_{ij} se numeşte **fluxul** (transportat)**pe arcul** ij.

Evident, condiţia (i) cere ca **fluxul pe orice arc să fie nenegativ şi subcapacitar,** iar condiţia (ii) *(legea de conservare a fluxului)* cere ca **suma fluxurilor pe arcele care intră în vîrful** i **să fie egală cu suma fluxurilor pe arcele care ies din vîrful** i**.**

Se putea cere ca fluxul să fie definit numai pe arcele reţelei, dar cu convenţia făcută la extensia funcţiei de capacitate, se observă că pentru perechile (i, j) care nu sînt arce în reţea condiţia (i) impune ca fluxul să fie 0, şi evident cele două definiţii sînt echivalente.

O preferăm pe cea dată, pentru simplitatea notaţiilor, deşi în implementări, structurile de date folosite vor ignora perechile (i, j) care nu sînt arce în reţea.

2^0 Dacă se sumează relaţiile (ii) (pentru $i \in V - \{s,t\}$) se obţine:
$0 = \sum_{i \neq s,t} \left(\sum_{j \in V} x_{ji} - \sum_{j \in V} x_{ij} \right) =$

$$\sum_{i \neq s,t} \sum_{j \neq s,t} x_{ji} - \sum_{i \neq s,t} \sum_{j \neq s,t} x_{ij} +$$

$\sum_{i \neq s,t} x_{si} + \sum_{i \neq s,t} x_{ti} - \sum_{i \neq s,t} x_{is} - \sum_{i \neq s,t} x_{it} =$

$-\left(\sum_i x_{is} - \sum_i x_{si} \right) - \left(\sum_i x_{it} - \sum_i x_{ti} \right)$, adică

$$\sum_{j \in V} x_{jt} - \sum_{j \in V} x_{tj} = -\left(\sum_{j \in V} x_{js} - \sum_{j \in V} x_{sj} \right).$$

Definiţie: Dacă x este un flux în reţeaua $R = (G, s, t, c)$ se numeşte **valoarea fluxului** x numărul

$$v(x) = \sum_{j \in V} x_{jt} - \sum_{j \in V} x_{tj}.$$

$v(x)$ se poate interpreta ca fiind fluxul net care ajunge în ieşirea reţelei sau (conform egalităţii obţinute mai sus) fluxul net care iese din intrarea reţelei.

În orice rețea $R = (G, s, t, c)$ există un flux, fluxul nul $x_{ij} = 0\ \forall ij$, de valoare 0.

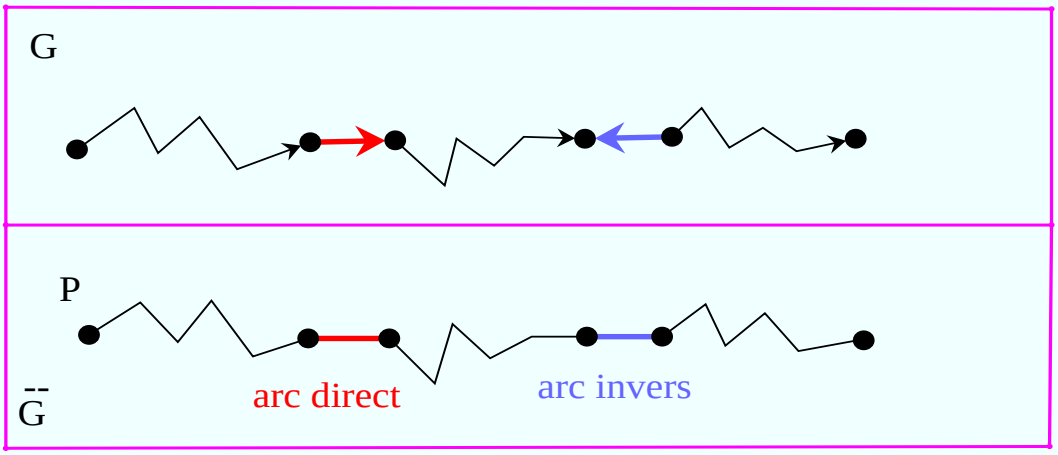
Problema fluxului maxim:
Dată $R = (G, s, t, c)$ o rețea, să se determine un flux de valoare maximă.

Observații: 1^0. Problema se poate formula, evident, ca o problemă de programare liniară:

$$\max\ v$$
$$\sum_j x_{ji} - \sum_j x_{ij} = 0,\ \forall i \neq s, t$$
$$\sum_j x_{js} - \sum_j x_{sj} = -v$$
$$\sum_j x_{jt} - \sum_j x_{tj} = v$$
$$0 \leq x_{ij} \leq c_{ij}\ \forall ij$$

Particularitățile combinatorii ale problemei, numărul mare de restricții și mai ales dificultățile legate de restricțiile de integritate ce s-ar putea impune variabilelor, care uneori în practică sînt esențiale, au condus la dezvoltarea de metode specifice de rezolvare.

Definiţie. Dacă P este un drum în $\overline{G}$, multigraful suport al digrafului G, şi $e = v_i v_j$ este o muchie a lui P atunci:
dacă e corespunde arcului $v_i v_j$ al lui G, e se numeşte **arc direct** al drumului P;
dacă e corespunde arcului $v_j v_i$ al lui G, atunci e se numeşte **arc invers.**



Definiţie. Fie $R = (G, s, t, c)$ şi x flux în R.
Se numeşte **C-drum** (în R relativ la fluxul x) un drum D în $\overline{G}$ cu proprietatea că $\forall ij \in E(D)$:

$x_{ij} < c_{ij}$ dacă ij este arc direct,

$x_{ji} > 0$ dacă ij este arc invers.

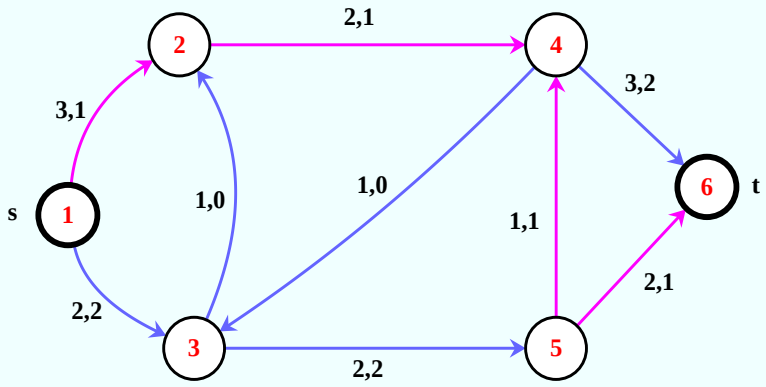
Dacă D este un C-drum și $ij \in E(D)$, se numește **capacitatea reziduală** a lui ij (relativ la C-drumul D) numărul

$$r(ij) = \begin{cases} c_{ij} - x_{ij} & \text{dacă } ij \text{ arc direct în } D \\ x_{ji} & \text{dacă } ij \text{ arc invers în } D \end{cases}.$$

Capacitatea reziduală a drumului D este

$$r(D) = \min_{e \in E(D)} r(e).$$

Exemplu: Fie rețeaua de mai jos, în care pe arce este precizată mai întîi capacitatea și apoi fluxul:



Atunci $D : 1, 12, 2, 24, 4, 45, 5, 56, 6$ este un C-drum de la s la t cu arcele directe $12(x_{12} = 1 < c_{12} = 3); 24(x_{24} = 1 < c_{24} = 2); 56(x_{56} = 1 < c_{56} = 2)$ și arcul invers $45(x_{54} = 1 > 0)$. Capacitatea reziduală a lui D este $r(D) = \min(\min(2, 1, 1), 1) = 1$.

Definiţie. Se numeşte **drum de creştere** a fluxului x, în reţeaua $R = (G, s, t, c)$, un C-drum de la s la t.

Lemã. 1. *Dacă D este un drum de creştere a fluxului x în reţeaua $R = (G, s, t, c)$, atunci $x^1 = x \bigotimes r(D)$ definit prin*

$$x^1_{ij} = \begin{cases} x_{ij} & \text{dacă } \overline{ij} \notin E(D) \\ x_{ij} + r(D) & \text{dacă } ij \in E(D), ij \text{ arc direct în } D \\ x_{ij} - r(D) & \text{dacă } ji \in E(D), ji \text{ arc invers în } D \end{cases}$$

este flux în R şi $v(x^1) = v(x) + r(D)$.

Demonstraţie. Definiţia lui $r(D)$ implică îndeplinirea de către x^1, a condiţiilor (i).
Condiţiile (ii) verificate de x, nu sînt afectate pentru niciun vîrf $i \notin V(D)$.
Dacă $i \neq s, t$ este un vîrf al drumului D, i este incident cu exact două arce ale lui D,fie ele li şi ik.
Avem următoarele cazuri posibile:

a) li și ik arce directe:
$\sum_j x^1_{ji} - \sum_j x^1_{ij} = \sum_{j\neq l} x_{ji} - \sum_{j\neq k} x_{ij} + x^1_{li} - x^1_{ik} =$
$\sum_{j\neq l} x_{ji} - \sum_{j\neq k} x_{ij} + x_{li} + r(D) - x_{ik} - r(D) =$
$\sum_j x_{ji} - \sum_j x_{ij} = 0.$

b) li direct ik invers:
$\sum_j x^1_{ji} - \sum_j x^1_{ij} = \sum_{j\neq l,k} x_{ji} - \sum_j x_{ij} + x^1_{li} + x^1_{ki} =$
$\sum_{j\neq l,k} x_{ji} - \sum_j x_{ij} + x_{li} + r(D) + x_{ki} - r(D) =$
$\sum_j x_{ji} - \sum_j x_{ij} = 0.$

c) li invers, ik direct: similar cu b).

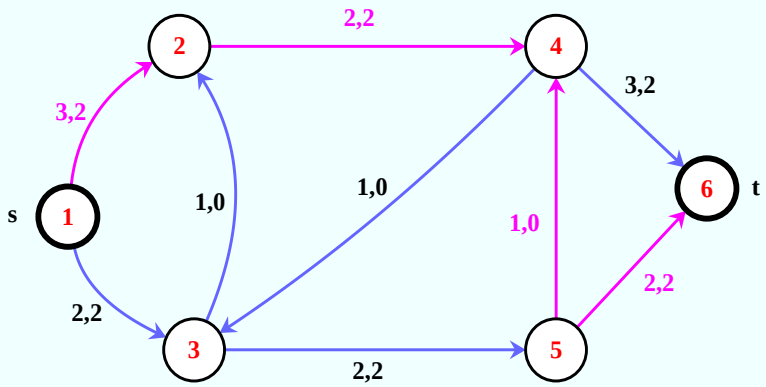
d) li invers, ik invers: similar cu a).

Valoarea fluxului x^1 se obține considerînd lt unicul arc al lui D incident cu t:

Dacă lt direct atunci
$v(x^1) = \sum_j x^1_{jt} - \sum_j x^1_{tj} = \sum_{j\neq l} x_{jt} - \sum_j x_{tj} + x^1_{lt} =$
$\sum_{j\neq l} x_{jt} - \sum_j x_{tj} + x_{lt} + r(D) = v(x) + r(D).$

Daca lt invers atunci
$v(x^1) = \sum_j x^1_{jt} - \sum_j x^1_{tj} = \sum_j x_{jt} - \sum_{j\neq l} x_{tj} - x^1_{tl} =$
$\sum_j x_{jt} - \sum_{j\neq l} x_{tj} - (x_{tl} - r(D)) = v(x) + r(D).$
Deci lema are loc.

Pentru exemplul anterior, fluxul $x^1 = x \otimes r(D)$ de valoare 4 este precizat pe arce:



Observaţii: 1^0 Această lemă justifică denumirea de drum de creștere, precum și cea de capacitate rezidualã.

2^0 Din definiţie, dacă D este drum de creștere, $r(D) > 0$ și deci avem $v(x \otimes r(D)) > v(x)$.

Rezultă că
dacă x admite un drum de creștere atunci x nu este flux de valoare maximă.

Pentru a demonstra că si reciproc este adevărat avem nevoie de o nouă noţiune.

Definiţie. Fie $R = (G, s, t, c)$. Se numeşte **secţiune** în reţeaua R, o partiţie (S, T) a lui V cu $s \in S$ şi $t \in T$.
Capacitatea secţiunii (S, T) este

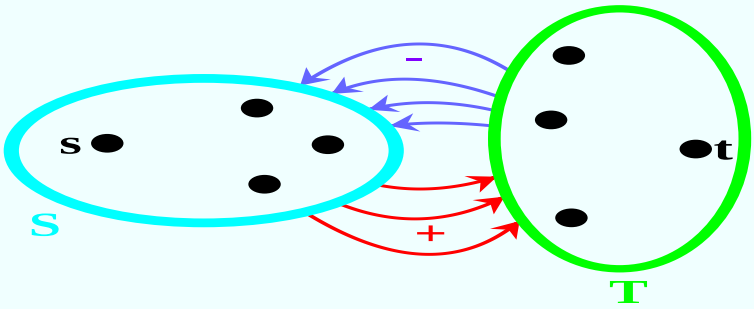
$$c(S, T) = \sum_{i \in S} \sum_{j \in T} c_{ij}$$

(suma capacităţilor arcelor de la S la T).

Lemă. 2. *Daca x este un flux în $R = (G, s, t, c)$ şi (S, T) este o secţiune a reţelei, atunci*

$$v(x) = \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}).$$

(valoarea fluxului este egală cu fluxul net ce trece prin orice secţiune.)



Demonstraţie:

$v(x) = (\sum_j x_{jt} - \sum_j x_{tj}) - 0$

$= -(\sum_j x_{js} - \sum_j x_{sj}) - \sum_{i \in S, i \neq s} (\sum_j x_{ji} - \sum_j x_{ij})$

$= \sum_{i \in S} (\sum_j x_{ij} - \sum_j x_{ji})$

$= \sum_{i \in S} \sum_{j \in S} (x_{ij} - x_{ji}) + \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji})$

$= \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}).$

Lemă. 3. *Dacă x este un flux în $R = (G, s, t, c)$ şi (S, T) este o secţiune, atunci $v(x) \leq c(S, T)$.*

Demonstraţie:

$v(x) = \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji})$ (lema 2)

$\leq \sum_{i \in S} \sum_{j \in T} (c_{ij} - x_{ji})$ $(x_{ij} \leq c_{ij})$

$\leq \sum_{i \in S} \sum_{j \in T} c_{ij}$ $(x_{ji} \geq 0)$.

Observaţii:
1) Dacă $\overline{x}$ este un flux în $R = (G, s, t, c)$ şi $(\overline{S}, \overline{T})$ o secţiune astfel încît $v(\overline{x}) = c(\overline{S}, \overline{T})$, atunci $\forall x$ flux în R $v(x) \leq c(\overline{S}, \overline{T}) = v(\overline{x})$, deci $\overline{x}$ este flux de valoare maximă.

2) În exemplul dat, x^1 este flux de valoare maximă întrucît $v(x^1) = 4 = c(\{1, 2, 3\}, \{4, 5, 6\})$.

Teoremă. 1. *(Teorema drumului de creştere) Un flux x este de valoare maximă într-o reţea R, dacă şi numai dacă, nu există drumuri de creştere a fluxului x în reţeaua R.*

Demonstraţie: O implicaţie este deja stabilită. Reciproc, fie x un flux în R care nu admite drumuri de creştere. Considerăm $S = \{i \mid i \in V \ \wedge \ \exists \ D$ C-drum în R de la s la $i\}$.

Evident $s \in S$ (există D de lungime 0) şi $t \notin S$ (nu există C-drumuri de la s la t). Fie $T = V - S$. Rezultă că (S,T) este o secţiune.

Să observăm că $\forall i \in S$ și $\forall j \in T$ avem:
dacă $ij \in E$ atunci $x_{ij} = c_{ij}$ și
dacă $ji \in E$ atunci $x_{ji} = 0$
(altminteri C-drumul de la s la i se poate extinde la un C-drum de la s la j).

Deci, conform lemei 2, $v(x) = \sum_{i \in S} \sum_{j \in T} (x_{ij} - x_{ji}) = \sum_{i \in S} \sum_{j \in T} (c_{ij} - 0) = c(S, T)$ și prin urmare x este flux de valoare maximă.

Teoremă. 2. *(Teorema fluxului intreg)*
Dacă toate capacităţile sînt întregi, atunci există un flux de valoare maximă cu toate componentele întregi (flux întreg de valoare maximă).
Demonstraţie: Fie algoritmul

1: $x^0 \leftarrow 0$; $i \leftarrow 0$;
2: **while** ($\exists P_i$ drum de creştere relativ la x^i) **do**
{ $x^{i+1} \leftarrow x^i \otimes r(P_i)$;
$i++$
}

Se observă că **"x^i are componente întregi"** este un invariant al algoritmului (din definiţia lui $r(P_i)$, dacă toate capacităţile sînt întregi, rezultă că $r(P_i)$ este întreg în ipoteza că x^i e întreg) şi că la fiecare iteraţie a pasului 2 valoarea fluxului curent creşte cu măcar o unitate, deci pasul 2 se repetă de cel mult $c(\{s\}, V - \{s\}) \in \mathbf{Z}_+$ ori. Fluxul final obţinut este, conform teoremei 1, de valoare maximă.

Observaţie. Algoritmul, descris mai sus, este finit şi în cazul capacităţilor raţionale.

Teoremă. 3. *(Ford-Fulkerson, 1956)*
Valoarea maximă a unui flux în reţeaua $R = (G, s, t, c)$ *este egală cu capacitatea minimă a unei secţiuni a reţelei.*
Demonstraţie: Dacă dispunem de un algoritm care, pornind de la un flux iniţial x^0 (x^0 există întotdeauna, de exemplu $x^0 = 0$), construieşte într-un număr finit de paşi un flux x, care nu admite drumuri de creştere, atunci secţiunea construită în demonstraţia teoremei 1 satisface împreună cu x enunţul teoremei.

Pentru cazul capacităţilor raţionale algoritmul descris în demonstraţia teoremei 2, satisface această condiţie.

Pentru cazul capacităţilor reale vom prezenta, mai tîrziu, un astfel de algoritm, datorat lui Edmonds şi Karp (1972).

Observaţii: i) În demonstraţia teoremei 3 avem nevoie de fapt, doar să arătăm că există un flux x de valoare maximă şi apoi să-i aplicăm construcţia din demonstraţia teoremei 1; existenţa fluxului x maxim rezultă imediat, considerînd transcrierea problemei fluxului maxim ca o problemă de programare liniară; am preferat demonstraţia de mai sus care (deşi va fi completată abia după analiza algoritmului lui Edmonds-Karp) este constructivă.

ii) Importanţa algoritmică a teoremei 3 este evidentă: mulţimea secţiunilor reţelei este finită, pe cînd mulţimea fluxurilor din reţea este infinită.

Algoritmul lui Ford și Fulkerson pentru aflarea unui flux de valoare maximă

Se va folosi un procedeu de etichetare a vîrfurilor rețelei, în vederea depistării drumurilor de creștere a fluxului curent x. Dacă nu există drumuri de creștere, fluxul va fi de valoare maximă.

Eticheta atribuită unui vîrf $j \in V$ are trei componente (e_1, e_2, e_3) unde $e_1 \in V \cup \{0\}$; $e_2 \in \{direct, invers\}$; $e_3 \in \mathbf{R}_+$ și au urmatoarea semnificație:
- dacă $e_2 = direct$ și $e_1 = i$ atunci $\exists$ un C-drum P de la s la j cu ultimul arc ij, arc direct și $r(P) = e_3$;
- dacă $e_2 = invers$ și $e_1 = i$ atunci $\exists$ un C-drum P de la s la j cu ultimul arc ij, arc invers și $r(P) = e_3$.

Inițial, se etichetează sursa s cu eticheta $(0, ., \infty)$. Celelalte vîrfuri primesc etichetă prin *"cercetarea"* vîrfurilor deja etichetate:

Dacă i este un vîrf etichetat, atunci $\forall j \in V$

Dacă j neetichetat, $ij \in E$ și $x_{ij} < c_{ij}$ atunci

j se etichet. $e = (i,\ direct,\ \min(e_3[i], c_{ij} - x_{ij}))$;

Dacă j neetichetat, $ji \in E$ și $x_{ji} > 0$ atunci

j se etichet. $e = (i,\ invers,\ \min(e_3[i], x_{ji}))$.

Evident, în acest fel se respectă semnificaţia celor trei componente ale etichetelor.
Numim această procedură $etichetare(i)$.

Atunci cînd în procedura de etichetare s-a atribuit etichetă vîrfului t, s-a obţinut un drum de creştere P a fluxului curent, de capacitate rezidualã $r(P) = e_3[t]$ și ale cărui vîrfuri se depistează în $O(n)$ explorînd prima componentă a etichetelor. Modificarea fluxului $x \otimes r(P)$ se execută în acest mers înapoi, tot în $O(n)$.

Pentru noul flux se reia procedura de etichetare.

Dacă toate vîrfurile etichetate au fost cercetate și nu s-a reușit etichetarea vîrfului t, rezultă că fluxul curent nu admite drumuri de creștere, este deci de valoare maximă, iar dacă S = mulțimea vîrfurilor etichetate atunci $(S, V - S)$ este o secțiune de capacitate minimă.

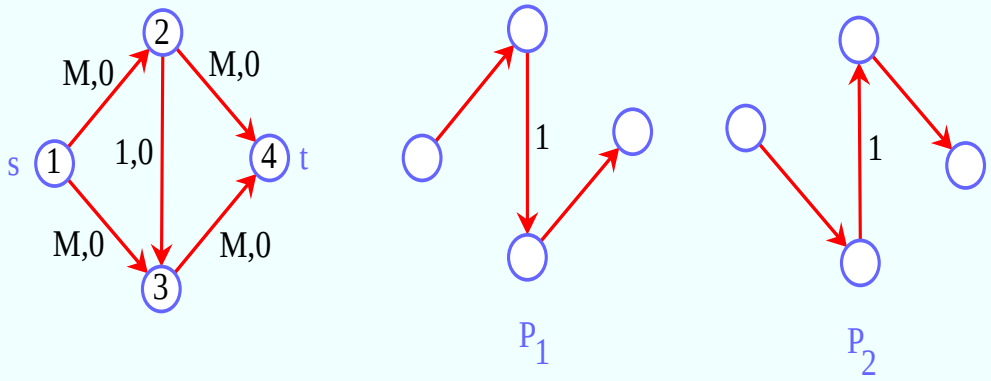
Descrierea algoritmului

1: Se alege $x = (x_{ij})$ flux inițial (de ex. fluxul nul);
Se etichetează s cu $(0, ., \infty)$
2: **while** ($\exists$ vîrfuri etichetate necercetate) **do**
{ "alege" un vîrf etichetat și necercetat i;
$etichetare(i)$;
if (t a primit etichetă) **then**
{ modifică fluxul pe drumul dat de etichete;
șterge toate etichetele;
etichetează s cu $(0, ., \infty)$
}
}
3: $S \leftarrow \{i | i \in V, i$ are etichetă$\}$
$T \leftarrow V - S$

x este flux de valoare maximă
(S, T) este secțiune de capacitate minimă.

Complexitatea algoritmului:
Pentru fiecare creștere a fluxului, sînt necesare cel mult $2m(m = |E|)$ inspecții de arce în vederea etichetării.
Dacă toate capacitățile sînt întregi atunci vor fi necesare cel mult v $(v =$ valoarea fluxului maxim) creșteri succesive. Rezultă că algoritmul are complexitatea $O(mv)$.
Dacă U este o margine superioară a capacităților arcelor atunci $v \leq (n-1)U$ $((n-1)U$ este o margine superioară a capacității secțiunii $(\{s\}, V - \{s\}))$, deci algoritmul are complexitatea $O(nmU)$.

Observații. 1^0. Dezavantajele algoritmului sînt legate de neconvergența în cazul capacităților iraționale (deși practic, în implementări nu este cazul), și de faptul ca mărimile capacităților influențează comportarea sa, acestea neconstituind o măsură a volumului datelor de intrare. Exemplu:



Dacă "alegerea", din pasul 2 al algoritmului, face ca drumurile de creștere succesive (pornind de la fluxul nul) să fie $P_1, P_2, P_1, P_2, \ldots$ unde $P_1 = 1, 2, 3, 4$, $P_2 = 1, 3, 2, 4$ atunci, fiecare creștere a fluxului mărește cu 1 valoarea fluxului curent și, deci, vor fi necesare $2M$ creșteri, ceea ce este inadmisibil pentru $M \in \mathbf{Z}_+$ foarte mari.

2^0. Aceste dezavantaje pot fi evitate dacă alegerile vîrfurilor etichetate supuse cercetării se fac judicios. Primii care au observat acest fenomen, au fost Dinic(1970) și independent, Edmonds și Karp (1972).

Modificarea lui Edmonds și Karp a algoritmului lui Ford & Fulkerson

Numim **drum minim de creștere a fluxului** x în rețeaua R, un drum de creștere de **lungime minimă** printre toate drumurile de creștere.

Fie x un flux oarecare în rețeaua R. Definim șirul de fluxuri x^k în R astfel:

$x^0 \leftarrow x$;
$x^k \leftarrow x^{k-1} \bigotimes r(P_{k-1})$, P_k este drum minim de creștere relativ la x^{k-1}; $k = 1, 2, \ldots$

Vom dovedi că șirul de fluxuri astfel construit este finit.
Notăm, pentru $\forall i \in V$ și $\forall k = 0, 1, 2, \ldots$

$\sigma_i^k =$ lungimea minimă a unui C-drum de la s la i în R relativ la fluxul x^k.
$\tau_i^k =$ lungimea minimă a unui C-drum de la i la t în R relativ la fluxul x^k.

Lemã. 4. *Pentru* $\forall i \in V$ *şi* $\forall k = 0, 1, 2, \ldots$ *avem*

$$\sigma_i^{k+1} \geq \sigma_i^k \text{ şi } \tau_i^{k+1} \geq \tau_i^k.$$

Teoremã. 4. *(Edmonds, Karp)*
Dacă $x = x^0$ *este un flux oarecare în reţeaua* R*, atunci şirul de fluxuri* $x^1, x^2, \ldots$ *obţinut din* x^0 *prin creşteri succesive pe drumuri minime de creştere, are cel mult* $\frac{mn}{2}$ *elemente (în cel mult* $\frac{mn}{2}$ *creşteri succesive, se obţine un flux care nu admite drumuri de creştere).*

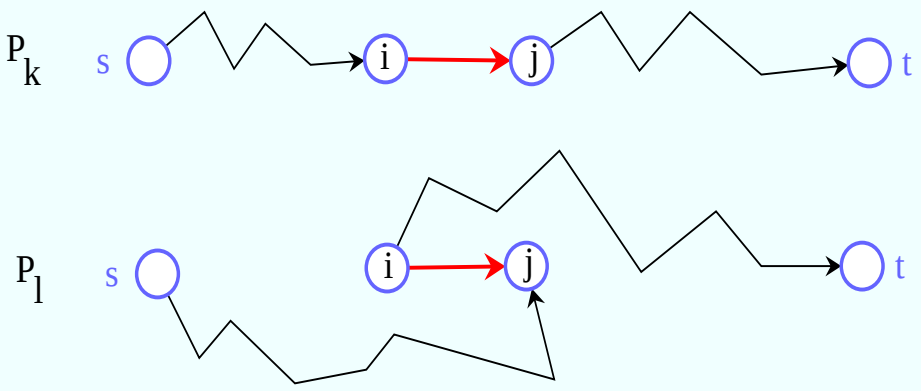
Demonstraţie: Dacă P este un drum de creştere relativ la un flux în reţeaua R, cu capacitatea rezidualã $r(P)$, vom numi **arc critic** în P orice arc $e \in P$ cu $r(e) = r(P)$.
În $x \otimes r(P)$, fluxul pe arcele critice devine sau egal cu capacitatea (pentru arcele directe) sau egal cu 0 (pentru arcele inverse).

Fie ij un arc critic pe drumul minim de creștere P_k relativ la x^k. Lungimea lui P_k este:

$$\sigma_i^k + \tau_i^k = \sigma_j^k + \tau_j^k.$$

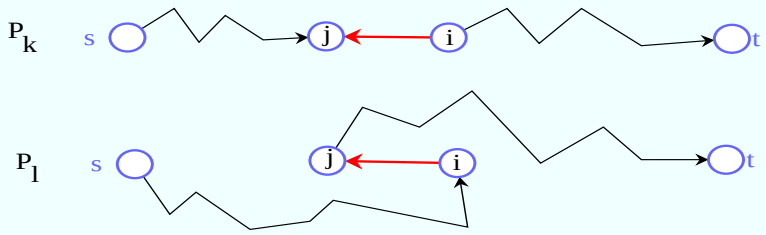
Cum ij este critic în P_k, în x^{k+1} nu va putea fi folosit în aceeași direcție ca în P_k. Prima oară cînd fluxul pe arcul ij se va modifica, el va apare într-un drum de creștere P_l cu $l > k$ relativ la x^l și va fi folosit în direcție opusă. Avem, deci, două cazuri:

i) **ij direct în $\mathbf{P_k}$**. Atunci $\sigma_j^k = \sigma_i^k + 1$. În P_l ij va fi arc invers, deci $\sigma_i^l = \sigma_j^l + 1$.



Rezultă, $\sigma_i^l + \tau_i^l = \sigma_j^l + 1 + \tau_i^l \geq \sigma_j^k + 1 + \tau_i^k = \sigma_i^k + \tau_i^k + 2$ (s-a folosit lema 1). Am obținut că $\lg(P_l) \geq \lg(P_k) + 2$.

ii) **ij arc invers în P_k.** Atunci $\sigma_i^k = \sigma_j^k + 1$. În P_l ij va fi arc direct, deci $\sigma_j^l = \sigma_i^l + 1$.



Rezultă $\sigma_j^l + \tau_j^l = \sigma_i^l + 1 + \tau_j^l \geq \sigma_i^k + 1 + \tau_j^k = \sigma_j^k + \tau_j^k + 2$ Deci, $\lg(P_l) \geq \lg(P_k) + 2$.

Deci **orice drum minim de creștere în care arcul ij este critic este cu măcar două arce mai lung decît precedentul în care ij a fost critic.**

Cum, un drum în G are cel mult $n-1$ arce, rezultă că un arc fixat nu poate fi critic în procesul de creștere mai mult de $\frac{n}{2}$ ori.

Cum orice drum de creștere are cel puțin un arc critic, rezultă că nu putem avea mai mult de $\frac{mn}{2}$ drumuri minime de creștere, în șirul construit. Deci, după cel mult $mn/2$ creșteri, se obține un flux care nu admite drumuri de creștere.

Corolar. *Dacă $R = (G, s, t, c)$ este o rețea, atunci există un flux care nu admite drumuri de creștere.*

Observații: 1^0 Rezultă de aici, că demonstrația teoremei 3 este completă.

2^0 S-ar putea pune întrebarea dacă nu cumva, alegerea drumurilor minime de creștere, mărește complexitatea algoritmului de flux maxim ? Răspunsul este însă banal:

Lemă. 5. *Dacă, în pasul 2 al algoritmului lui Ford și Fulkerson, alegerea vîrfurilor etichetate în vederea cercetării se face după regula "primul etichetat - primul cercetat", atunci drumurile de creștere care se depistează sînt cu număr minim de arce.*

Demonstrație: Fie P un drum de creștere depistat și fie P' un drum minim de creștere. Presupunem că $\lg(P) > \lg(P')$.

Fie

$P : si_1, i_1i_2, \ldots, i_{k-1}i_k, i_ki_{k+1}, i_{k+1}i_{k+2}, \ldots, i_{k+l-1}t$ și

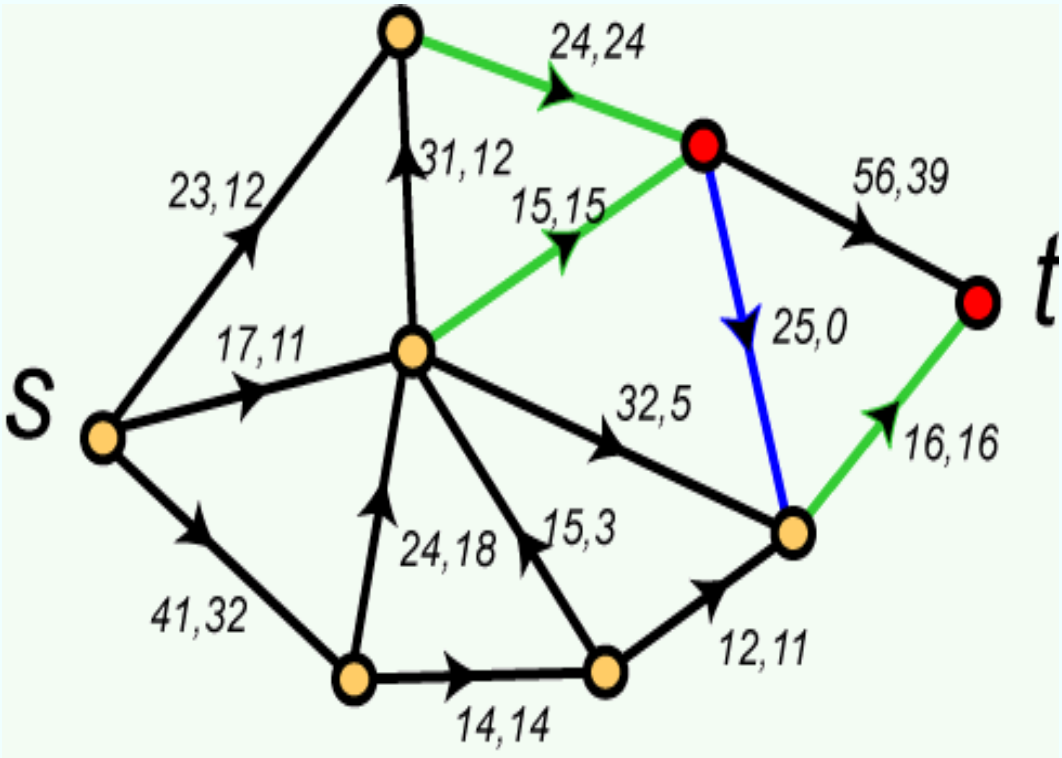
$P' : si_1, i_1i_2, \ldots, i_{k-1}i_k, i_kj_{k+1}, j_{k+1}j_{k+2}, \ldots, j_{k+l'-1}t.$

Deci, $lg(P) = k + l$, $lg(P') = k + l'$, au primele k arce comune și $l' < l$.
Conform regulii de etichetare, i_{k+1} va primi etichetă înaintea lui j_{k+1}, dar j_{k+1} va primi etichetă înaintea lui i_{k+2}; j_{k+2} va primi etichetă înaintea lui i_{k+3} și așa mai departe; obținem inductiv că t primește etichetă înaintea lui $i_{k+l'+1}$, deci t primește etichetă pe drumul P', înainte de a primi etichetă pe drumul P, absurd.

Observație: Regula "primul etichetat - primul cercetat" corespunde unei explorări bfs a vîrfurilor etichetate, ceea ce se poate realiza, utilizînd o coadă pentru memorarea lor. Aceasta nu afectează complexitatea algoritmului, care va necesita tot $O(m)$ operații pentru fiecare creștere a fluxului , si din teorema 4 obținem

Teoremă. 5. *(Edmonds- Karp 1972)*
Dacă se modifică algoritmul lui Ford și Fulkerson cu precizarea alegerii bfs a vîrfurilor etichetate în vederea cercetării, atunci, fluxul maxim se obține în timpul $O(m^2n)$.

Exercițiu. Aplicați algoritmul lui Ford & Fulkerson modificat pentru obținerea unui flux de valoare maximă în rețeaua de mai jos (în care este precizat un flux inițial de valoare 55).



Algoritmi de tip preflux.

Fie $R = (G, s, t, c)$ o reţea.

Definiţie. Se numeşte **preflux** în reţeaua R, o funcţie $x : E \to \mathbf{R}$ astfel încît

(i) $$0 \leq x_{ij} \leq c_{ij} \quad \forall ij \in E$$

(ii) $$\forall i \neq s \; e_i = \sum_{j:ji \in E} x_{ji} - \sum_{j:ij \in E} x_{ij} \geq 0.$$

Numărul $e_i \quad i \in V - \{s, t\}$ se numeşte **excesul** din vîrful i.

Dacă $i \in V - \{s, t\}$ şi $e_i > 0$ atunci i se numeşte **nod activ**.

Dacă $ij \in E$ x_{ij} va fi numit **fluxul pe arcul** ij.

Observaţii: 1^0. Dacă în reţeaua R nu există noduri active, atunci **prefluxul x este flux** de la s la t în R de valoare e_t.

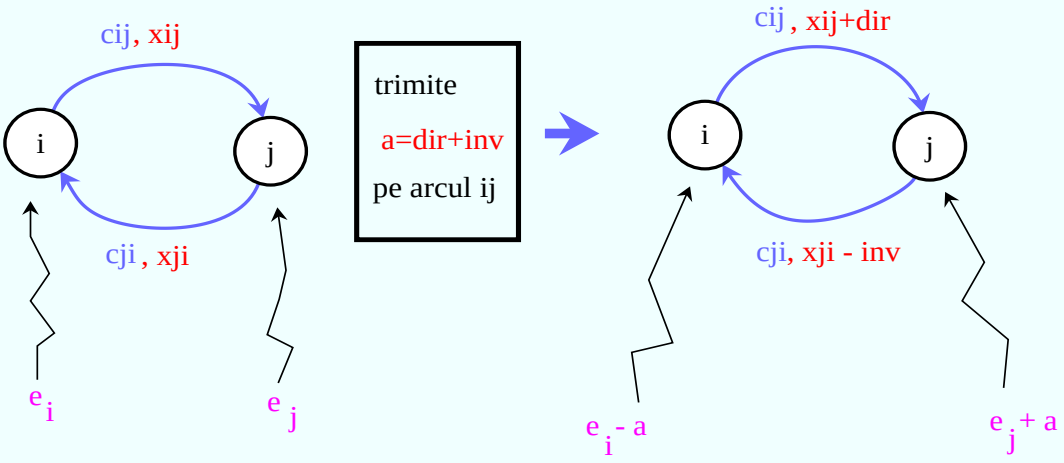
2^0. *Ideea algoritmulor de tip preflux* este: se porneşte cu **un preflux** în R şi **se transformă** prin modificări ale fluxului pe arce **într-un flux** care **nu admite drumuri de creştere.**

3^0. În definiţia unui preflux, nu am mai utilizat convenţia că vom introduce toate perechile de arce din digraful complet simetric de ordin n, întrucît în analiza algoritmilor pe care îi vom prezenta va fi esenţială reprezentarea digrafului G cu ajutorul listelor de adiacenţă. **Totuşi, vom considera că dacă $ij \in E$ atunci şi $ji \in E$ (altminteri, adăugăm arcul ji cu capacitate 0).**

Definiţie: Dacă x este un preflux în R şi $ij \in E$, atunci **capacitatea rezidualã** a arcului ij este

$$r_{ij} = c_{ij} - x_{ij} + x_{ji}$$

(reprezentînd fluxul adiţional ce poate fi *"trimis"* de la nodul i la nodul j utilizînd arcele ij şi ji).



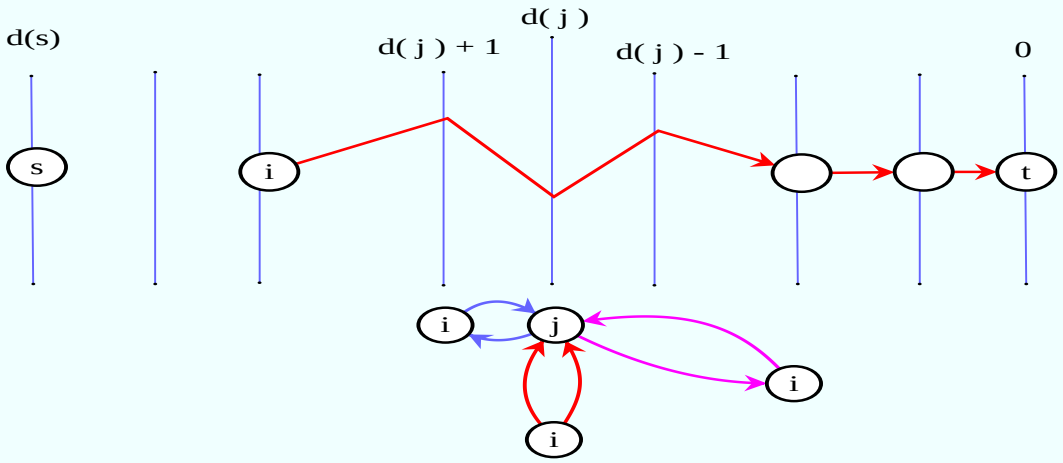
Observaţie. Peste tot, în cele ce urmează, a *"trimite"* flux de la i la j înseamnă să creştem fluxul pe arcul ij sau să micşorăm fluxul pe arcul ji.

Definiţie: Se numeşte **C-drum** în R relativ la prefluxul x, un drum al lui G ale cărui arce au capacitatea rezidualã pozitivã.

Definiţie: Se numeşte *funcţie de distanţă* în R relativ la prefluxul x, o funcţie $d : V \to \mathbf{Z}_+$ care satisface

$(D1)$ $$d(t) = 0$$

$(D2)$ $$\forall ij \in E, r_{ij} > 0 \Rightarrow d(i) \leq d(j) + 1$$



Observaţii: 1^0. Dacă P este un C-drum relativ la prefluxul x în R de la i la t atunci $d(i) \leq \lg(P)$ (arcele unui C-drum au capacitate reziduală pozitivă şi se aplică (D2)).
Rezultă că $d(i) \leq \tau_i$ (lungimea minimă a unui C-drum de la i la t).

2^0. Vom nota cu $A(i)$, pentru orice vîrf i, lista sa de adiacenţă, care conţine arcele $ij \in E$.

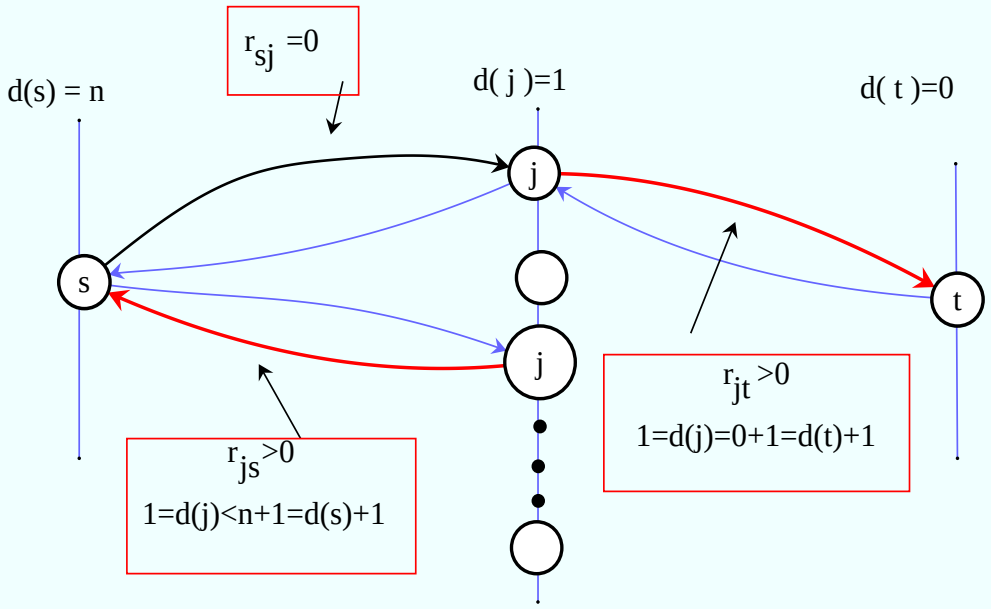
Definiţie. Fie x un preflux în R şi d o funcţie de distanţă relativ la x. Un arc $ij \in E$ se numeşte **admisibil** dacă

$$r_{ij} > 0 \quad \wedge \quad d(i) = d(j) + 1.$$

Dacă R este o reţea, considerăm *iniţializare* următoarea procedură care construieşte în $O(m)$ un preflux x şi o funcţie de distanţă d corespunzătoare acestuia, astfel:

procedure *iniţializare*;
{
    for $\forall ij \in E$ **do**
        if $i = s$ **then** $x_{sj} \leftarrow c_{sj}$ **else** $x_{ij} \leftarrow 0$;
    $d[s] \leftarrow n$; $d[t] \leftarrow 0$;
    for $\forall i \in V - \{s, t\}$ **do** $d[i] \leftarrow 1$
}

Observaţii: 1^0. După execuţia acestei proceduri, $\forall sj \in A(s)$ avem $r_{sj} = 0$, deci alegerea lui $d(s) = n$ nu afectează condiţia $D2$. Pentru arcele cu capacitate reziduală pozitivă de forma js $D2$ este evident verificată.



2^0. Alegerea lui $d(s) = n$ are următoarea interpretare:**" nu există C-drum de la s la t în R relativ la x"** (întrucît, altminteri, ar trebui ca lungimea acestuia să fie $\geq n$).
Dacă, în algoritmii de tip preflux vom păstra acest invariant, atunci cînd x va deveni flux, va rezulta că nu admite drumuri de creştere şi deci x va fi de valoare maximă.

Considerām următoarele proceduri

procedure *pompează* (i);
// i este un vârf diferit de s, t
{
alege $ij \in A(i)$ ij **admisibil;**
"trimite" $\delta = \min(e_i, r_{ij})$ unități de flux de la i la j
}

Dacă $\delta = r_{ij}$ avem o **pompare saturată**, altminteri pomparea este **nesaturată**.

procedure *reetichetare* (i);
// i este un vârf diferit de s, t
{
$d(i) \leftarrow \min\{d(j) + 1 \mid ij \in A(i) \ \wedge \ r_{ij} > 0\}$
}

Schema generală a unui algoritm de tip preflux este:

{
    inițializare;
    while ∃ noduri active în R **do**
    { selectează un nod activ i;
        if ∃ arce admisibile în $A(i)$
            then *pompează*(i)
            else *reetichetare*(i)
    }
}

Lemă. 6. *Algoritmul de tip preflux, de mai sus, are ca invariant "d* **este funcție de distanță relativ la prefluxul** *x". La fiecare apel al lui reetichetare(i), $d(i)$ crește strict.*

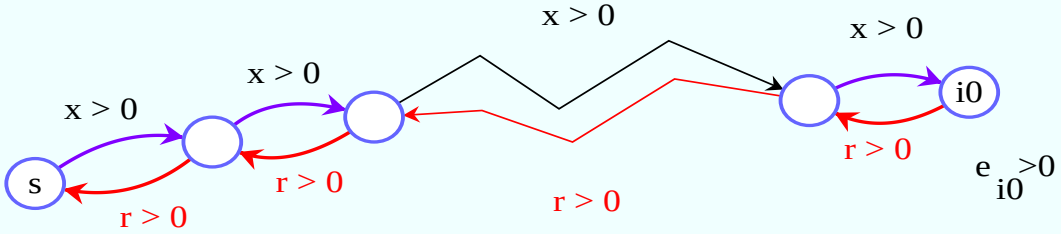
Demonstrație: Procedura inițializare construiește, evident, o funcție de distanță relativ la prefluxul inițial.

Dacă înaintea execuţiei unei iteraţii a lui while, d e funcţie de distanţă relativ la prefluxul curent x, atunci:
a) dacă se execută *pompare(i)*, singura pereche ce poate viola D2 este $d(i)$ şi $d(j)$. Cum arcul ij a fost ales admisibil, avem $d(i) = d(j) + 1$. După pompare, arcul ji poate avea $r_{ji} > 0$ (fără ca înainte să fi fost), dar condiţia $d(j) \leq d(i) + 1$ este, evident, satisfăcută;
b) dacă se execută *reetichetare(i)*, modificarea lui $d(i)$ se face astfel încît D2 să rămînă valabilă pentru orice arc ij cu $r_{ij} > 0$. Cum apelul lui reetichetare implică $d(i) < d(j) + 1 \quad \forall ij$ cu $r_{ij} > 0$, rezultă că după apel $d(i)$ creşte măcar cu o unitate.

Pentru finitudinea algoritmului va trebui să ne asigurăm că , dacă, în timpul execuţiei, avem un nod i activ, atunci în $A(i)$ există măcar un arc ij cu $r_{ij} > 0$. Aceasta rezultă din

Lemă. 7. *Dacă pe parcursul algoritmului, i_0 este un nod activ, atunci există un C-drum de la i_0 la s, în R, relativ la prefluxul curent x.*

Demonstraţie: Dacă x este un preflux în R, atunci x se poate scrie ca o sumă finită $x = x^1 + x^2 + \ldots + x^p$, unde fiecare x^k satisface: mulţimea $A^k = \{ij \mid ij \in E, x^k_{ij} \neq 0\}$ este
a) mulţimea arcelor unui drum de la s la t,
b) m. arcelor unui dr. de la s la un nod activ,
c) mulţimea arcelor unui circuit.
În plus, în situaţiile a) şi c), x^k este flux.
(demonstraţia rezultă algoritmic, construind mai întîi mulţimile a), apoi a celor de tip c) şi b); la fiecare etapă se caută inversul unui drum de tipul a),b), sau c); prefluxul costruit se scade din cel curent; excesele nenegative, permit efectuarea construcţiei; ea este finită, căci după fiecare etapă nr. arcelor cu flux curent nul creşte.)
Cum i_0 este un nod activ în R relativ la x, rezultă că situaţia b) va apare pentru nodul i_0 (întrucît situaţiile b) şi c) nu afectează excesul din nodul i_0). Arcele inverse ale acestui drum au capacitatea reziduală strict pozitivă şi ele formează C-drumul din enunţul lemei.



Corolar. 1. $\forall i \in V \quad d(i) < 2n.$

Demonstraţie: Dacă i nu a fost reetichetat, atunci $d(i) = 1 < 2n$.
Dacă i a fost reetichetat, atunci înainte de reetichetare i este nod activ, deci există C-drum de la i la s de lungime cel mult $n-1$.
Din modul de modificare a lui $d(i)$ și din D2 rezultă că după reetichetare $d(i) \leq d(s) + n - 1 = 2n - 1$, întrucît $d(s) = n$ nu se schimbă pe parcursul algoritmului.

Corolar. 2. *Numărul total de apeluri ale procedurii reetichetare este mai mic decît* $2n^2$.

Demonstraţie: Fiecare din cele $n-2$ vîrfuri ce pot fi supuse etichetării poate fi etichetat de cel mult $2n-1$ ori, avînd în vedere corolarul 1, lema 6 și etichetarea inițială.

Corolar. 3. *Numărul total de pompări saturate este* $\leq nm$.

Demonstraţie: După ce un arc ij devine saturat (situaţie în care $d(i) = d(j) + 1$), pe acest arc nu se va mai putea trimite flux pînă cînd nu se va trimite flux pe arcul ji situaţie în care vom avea $d'(j) = d'(i)+1 \geq d(i)+1 = d(j)+2$; această schimbare de flux nu va avea loc pînă ce $d(j)$ nu creşte cu două unităţi. Deci, un arc nu poate deveni saturat mai mult de n ori şi în total vom avea cel mult mn pompări saturate.

Lemă. 8. *(Goldberg şi Tarjan 1986) Numărul pompărilor nesaturate este cel mult* $2n^2m$.

Lemă. 9. *La terminarea algoritmului* x *este flux de valoare maximă.*

Demonstraţie: Din lemele 6 şi 8 şi corolarul 3 rezultă că algoritmul se termină după cel mult $2n^2m$ iteraţii şi cum $d(s) = n$ nu se modifică pe parcurs, rezultă că fluxul obţinut este fără drumuri de creştere.

Vom prezenta în continuare, algoritmul lui **Ahuja și Orlin** (1988) care, utilizînd o metodă de **scalare,** va mărgini numărul pompărilor nesaturate de la $O(n^2m)$ la $O(n^2 \log U)$.

Vom presupune că toate capacitățile sînt întregi și că $max_{ij \in E}(1 + c_{ij}) = U$. Notăm $\lceil\ log_2 U \rceil = K$. *Ideia algoritmului:*

Se vor executa $K + 1$ etape. Pentru fiecare etapă p, cu p luînd succesiv valorile $K, K - 1, \ldots, 1, 0$ vor fi îndeplinite următoarele condiții:
(a) - la începutul etapei p, $\forall\ i$ satisface $e_i \leq 2^p$
(b) - în timpul etapei p se utilizează procedurile *pompare-etichetare* în vederea eliminării nodurilor active cu $e_i \in (2^{p-1}, 2^p]$.

Din alegerea lui K, în etapa inițială ($p = K$) condiția (a) este satisfăcută și deci, dacă (b) va fi invariant al algoritmului, după $K+1$ etape, excesele nodurilor vor fi $\leq \frac{1}{2}$.

Dacă, toate transformările datelor vor păstra integritatea exceselor, va rezulta că excesul oricărui nod este 0, și, deci, dispunem de un flux de valoare maximă (datoritā proprietăților funcției distanță $d(i)$).

Realizarea lui (b) se poate face astfel:

- se începe etapa p, construind lista $L(p)$ a tuturor nodurilor $i_1, i_2, \ldots, i_{l(p)}$ cu excesele $e_{i_j} > 2^{p-1}$, ordonate crescător după d (avînd în vedere că d poate lua valori între 1 și $2n-1$, o sortare de tip hash rezolvă problema în $O(n)$).

- nodul activ selectat pentru *pompare-reetichetare* va fi pe tot parcursul etapei, **primul nod din $L(p)$.** Va rezulta că, dacă se face o pompare pe arcul ij admisibil, cum $d(j) = d(i) - 1$ și i este primul din $L(p)$, vom avea $e_i > 2^{p-1}$ și $e_j \leq 2^{p-1}$. Dacă, în plus, se va limita δ, fluxul "trimis" de la i la j, în procedura de *pompare* la $\delta = \min(e_i, r_{ij}, 2^p - e_j)$,

atunci, cum $2^p - e_j \geq 2^{p-1}$ va rezulta că orice **pompare nesaturată** trimite cel puţin 2^{p-1} unităţi de flux. După pompare, excesul din nodul j (singurul nod al cărui exces poate creşte) va fi $e_j + \min(e_i, r_{ij}, 2^p - e_j) \leq e_j + 2^p - e_j \leq 2^p$ (deci b) rămîne îndeplinită).

- etapa (p) se termină atunci cînd $L(p)$ devine vidă.

Pentru realizarea eficientă a operaţiilor de depistare a unui arc pe care se face pomparea, sau a examinării arcelor care ies dintr-un vîrf i pentru reetichetare, vom considera listele $A(i)$ organizate astfel:
- fiecare nod al listei conţine: vîrful j; x_{ij}; r_{ij}; pointer către arcul ji (din lista de adiacenţă a vîrfului j); pointer către următorul element din lista $A(i)$.
- parcurgerea listei se face cu ajutorul unui pointer $p(i)$ către elementul curent din listă.

Evident, organizarea acestor liste se face înaintea apelului lui *iniţializare* şi necesită $O(m)$ operaţii.

Algoritmul Ahuja-Orlin

inițializare;
$K \leftarrow \lceil \log_2(U) \rceil$; $\Delta \leftarrow 2^{K+1}$;
for $p = K, K-1, \ldots, 0$ **do**
{ construiește $L(p)$; $\Delta \leftarrow \frac{\Delta}{2}$
while $L(p) \neq \emptyset$ **do**
{ fie i primul element din $L(p)$;
parcurge lista $A(i)$ din locul curent pînă
se determină un arc admisibil sau se
depistează sfîrșitul ei;
if ij *este arcul admisibil găsit* **then**
{ $\delta \leftarrow \min(e_i, r_{ij}, \Delta - e_j)$;
$e_i \leftarrow e_i - \delta$; $e_j \leftarrow e_j + \delta$;
"trimite" δ unități de flux de la i la j;
if $e_i \leq \frac{\Delta}{2}$ **then** șterge i din $L(p)$;
if $e_j > \frac{\Delta}{2}$ **then** adaugă j ca prim nod în $L(p)$
}
else // s-a depistat sfîrșitul listei
{ șterge i din $L(p)$;
parcurge toată lista $A(i)$ pentru calculul lui
$d(i) = \min\{d(j) + 1; ij \in A(i) \wedge r_{ij} > 0\}$;
introdu i în $L(p)$ la locul său (hash);
pune pointerul curent al listei $A(i)$ la început
}
}
}

Operaţiile care dominã complexitatea timp, în cazul cel mai nefavorabil, sînt pompãrile nesaturate. Toate celelalte operaţii sînt dominate de $O(nm)$.

Lemã. 8'. *Numãrul pompãrilor nesaturate este cel mult* $8n^2$ *în fiecare etapã a scalãrii, deci* $O(n^2 \log U)$ *în total.*

Demonstraţie: Fie

$$F(p) = \sum_{i \in V i \neq s,t} \frac{e_i \cdot d(i)}{2^p}.$$

La începutul etapei p, $F(p) < \sum_{i \in V} \frac{2^p \cdot (2n)}{2^p} = 2n^2$.

Dacã în etapa p, atunci cînd se analizează nodul activ i, se apelează *reetichetare*, înseamnã cã nu există arce ij admisibile.

Operaţia de reetichetare măreşte $d(i)$ cu $\epsilon \geq 1$ unitãţi, ceea ce conduce la o creştere a lui F cu cel mult ϵ unitãţi. Cum pentru fiecare i, creşterea lui $d(i)$ pe parcursul întregului algoritm este $< 2n$ rezultã cã F va creşte pînã la cel mult valoarea $4n^2$, la sfîrşitul etapei p.

Dacă, pentru un nod i se execută *pompare*, atunci aceasta se execută pe arcul ij cu $r_{ij} > 0$ și $d(i) = d(j) + 1$.
Cum numărul unităților de flux pompat este $\delta \geq 2^{p-1}$, după pompare $F(p)$ va avea valoarea $F'(p)$, unde

$$F'(p) = F(p) - \frac{\delta \cdot d(i)}{2^p} + \frac{\delta \cdot d(j)}{2^p} = F(p) - \frac{\delta}{2^p} \leq F(p) - \frac{2^{p-1}}{2^p} = F(p) - \frac{1}{2}.$$

Rezultă că această situație nu poate apărea mai mult de $8n^2$ (întrucît $F(p)$ poate crește cel mult la $4n^2$).

Cu atît mai mult, pompările nesaturate nu vor depăși $8n^2$.

Sumarizînd toate rezultatele anterioare obţinem

Teoremă. 6. *(Ahuja-Orlin 1988) Algoritmul de tip preflux cu scalarea exceselor are complexitatea* $O(nm + n^2 \log U)$.

2. Aplicaţii (combinatorii) ale problemei fluxului maxim.

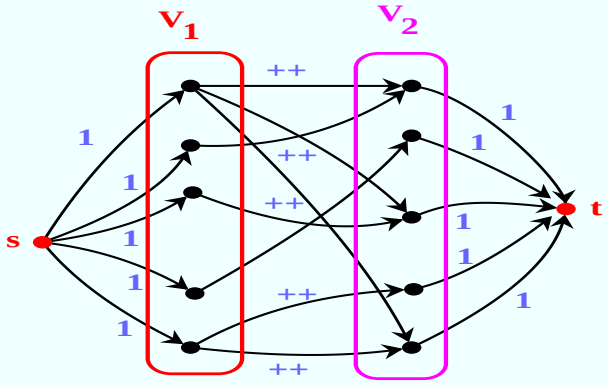
A. Determinarea cuplajului maxim şi a stabilei maxime într-un graf bipartit.

Fie $G = (V_1, V_2; E)$ un graf bipartit cu n vîrfuri şi m muchii.
Considerăm reţeaua $R = (G_1, s, t, c)$, unde

- $V(G_1) = \{s, t\} \cup V_1 \cup V_2$;
- $E(G_1) = E_1 \cup E_2 \cup E_3$
  - $E_1 = \{sv_1 \mid v_1 \in V_1\}$,
  - $E_2 = \{v_2 t \mid v_2 \in V_2\}$,
  - $E_3 = \{v_1 v_2 \mid v_1 \in V_1, v_2 \in V_2, v_1 v_2 \in E\}$;
- $c : E(G_1) \to \mathbf{Z}_+$ definită prin

$$c(e) = \begin{cases} 1 & \text{dacă } e \in E_1 \cup E_2 \\ \infty & \text{dacă } e \in E_3 \end{cases}$$

(vezi figura următoare).

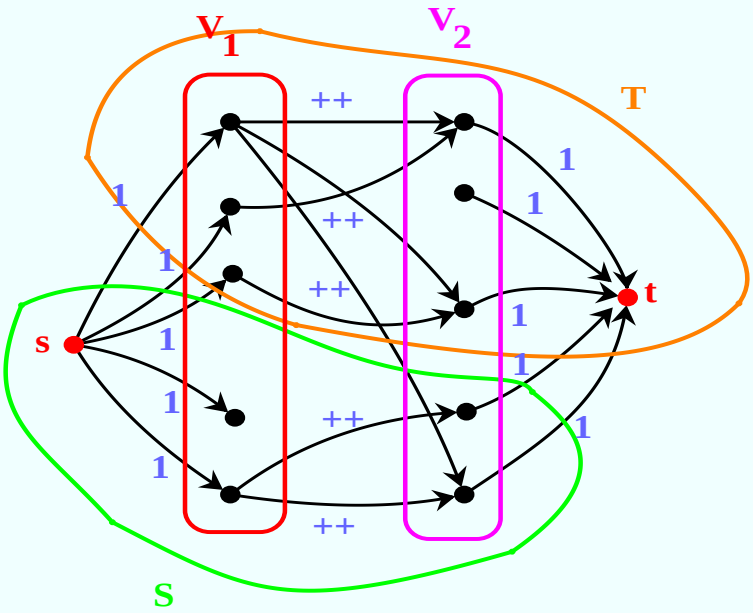


Dacă $x = (x_{ij})$ este un flux cu componente întregi în R atunci se observă că mulțimea de arce $\{ij \mid i \in V_1, j \in V_2 \quad \wedge x_{ij} = 1\}$ induce în graful G bipartit un cuplaj $M(x)$. În plus, $v(x)$ este cardinalul cuplajului $M(x)$.

Reciproc, orice cuplaj din G induce o mulțime de arce neadiacente în G_1; dacă pe fiecare astfel de arc $ij \quad (i \in V_1, j \in V_2)$ se consideră fluxul x_{ij} egal cu 1 și de asemenea $x_{si} = x_{jt} = 1$, și luînd fluxul $x = 0$ pe orice alt arc, atunci fluxul construit are valoarea $|M|$.

Rezultă că rezolvînd problema fluxului maxim pe rețeaua R se determină (pornind de la fluxul nul) în $O(nm + n^2 \log n)$ un cuplaj de cardinal maxim în graful bipartit G.

Fie (S,T) secţiunea de capacitate minimă ce se poate construi, din fluxul maxim obţinut, în $O(m)$ operaţii.
Vom avea, $c(S,T) = \nu(G)$, din teorema fluxului maxim-secţiunii minime.



Cum $\nu(G) < \infty$, rezultă că punînd $S_i = S \cap V_i$ şi $T_i = T \cap V_i$ $(i = 1, 2)$, avem: $|T_1| + |S_2| = \nu(G)$, iar $X = S_1 \cup T_2$ este **mulţime stabilă** în graful G (pentru a avea $c(S,T) < \infty$).
În plus, $|X| = |V_1 - T_1| + |V_2 - S_2| = n - \nu(G)$.
Rezultă că X este stabilă de cardinal maxim, întrucît $n - \nu(G) = \alpha(G)$ (teorema lui König).

B. Recunoașterea secvențelor digrafice.

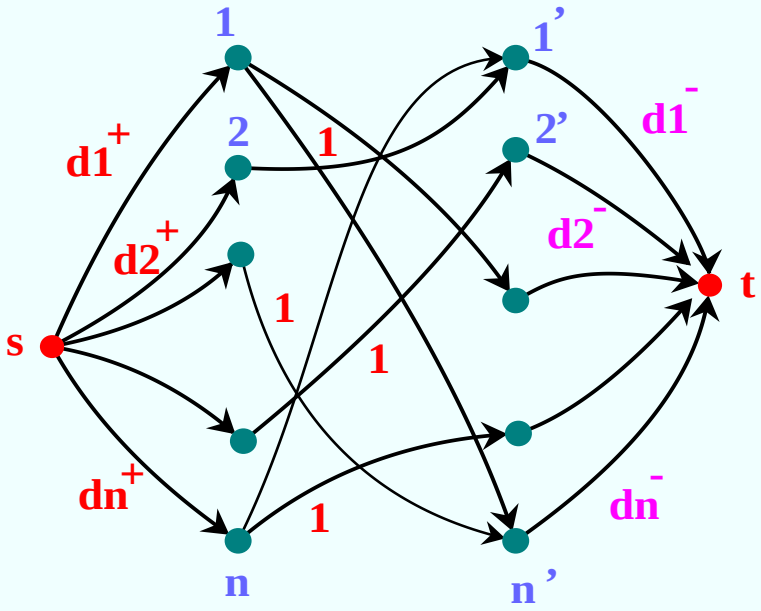
Date $(d_i^+)_{i=1,n}$ și $(d_i^-)_{i=1,n}$, există un digraf G cu n vîrfuri astfel încît $G = (\{1,\ldots,n\}, E)$ și $d_G^+(i) = d_i^+$ și $d_G^-(i) = d_i^-$ $\forall i = 1,n$? (Problema poate apare în proiectarea circuitelor integrate).

Evident, va trebui ca $0 \leq d_i^+ \leq n-1$ și $0 \leq d_i^- \leq n-1$ $\forall i = 1,n$ și $\sum_{i=1,n} d_i^+ = \sum_{i=1,n} d_i^- = m$ (unde, $m = |E|$, iar $d_i^+, d_i^- \in \mathbf{Z}$).

În aceste ipoteze, construim rețeaua bipartită $R = (G_1, s, t, E)$ unde G_1 se obține din $K_{n,n} - \{11', 22', \ldots, nn'\}$ prin orientarea muchiilor ij' $\forall i \neq j \in \{1,\ldots,n\}$, și introducerea arcelor si, $i \in \{1,\ldots,n\}$ și $j't$, $\quad j \in \{1,\ldots,n\}$.

Funcția de capacitate va fi $c(si) = d_i^+$ $\forall i = 1,n$, $\quad c(j't) = d_j^-$ $\forall j = 1,n$ și $c(ij') = 1$.

Dacă în această rețea, există un flux întreg de valoare maximă m, atunci din orice vîrf $i \in \{1, \ldots, n\}$ vor pleca exact d_i^+ arce pe care fluxul este 1, de forma ij' și în fiecare vîrf j' vor intra exact d_j^- arce pe care fluxul este 1, de forma ij'.



Considerînd mulțimea de vîrfuri $\{1, \ldots, n\}$ și introducînd toate arcele ij astfel încît $x_{ij'} = 1$, se obține un digraf G cu secvențele gradelor interioare și exterioare $(d_i^+)_{i=1,n}$ și $(d_i^-)_{i=1,n}$. Reciproc, dacă acest digraf există, atunci procedînd invers ca în construcția anterioară, se obține un flux întreg în rețeaua R de valoare m (deci maxim). Rezultă că recunoașterea secvențelor digrafice (și construcția digrafului în cazul răspunsului afirmativ) se poate face în $O(nm + n^2 \log n) = O(n^3)$.

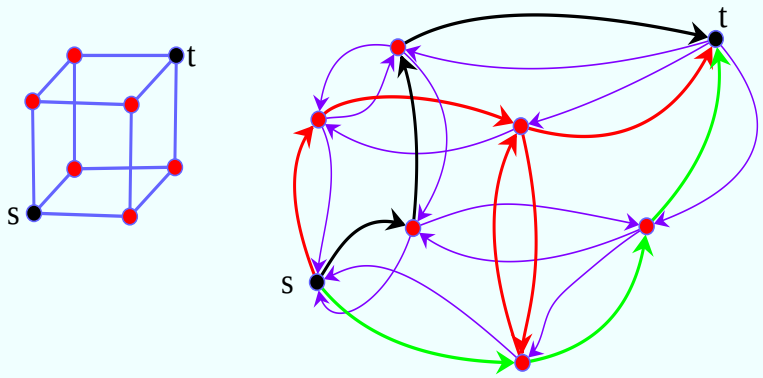
C. Determinarea nr. de muchie-conexiune al unui graf

Fie $G = (V, E)$ un graf. Pentru $s, t \in V, s \neq t$, definim:
-$p_e(s,t)$ = numărul maxim de drumuri cu muchii disjuncte ce unesc s și t în G.
-$c_e(s,t)$ = cardinalul minim al unei mulțimi de muchii, prin îndepărtarea căreia din graf, între s și t nu mai există drumuri.

Teoremă. 7. $p_e(s,t) = c_e(s,t)$.

Demonstrație: Construim din G digraful G_1, înlocuind fiecare muchie a lui G cu o pereche de arce simetrice. Considerăm
$c : E(G_1) \to \mathbf{Z}_+$ prin $c(e) = 1, \forall e \in E(G_1)$.

Fie x^0 un flux întreg de valoare maximă în $R = (G_1, s, t, c)$. Fluxul x^0 se poate scrie ca o sumă de $v(x^0)$ fluxuri x^k întregi de valoare 1, înlocuind, eventual, fluxul pe unele circuite cu 0.



Fluxul pe arcele groase
este 1, pe cele subtiri 0.

Fiecare astfel de flux x^k induce un drum de la s la t în G_1 (considerînd arcele pe care fluxul este nenul), și deci, și în G.
Rezultă că $v(x^0) = p_e(s,t)$, întrucît orice mulțime de drumuri disjuncte pe muchii, generează un flux de la s la t în R de valoare egală cu numărul acestor drumuri.

Fie (S,T) o secțiune de capacitate minimă; avem $c(S,T) = v(x^0)$, din teorema fluxului maxim-secțiunii minime.

Pe de altă parte, $c(S,T)$ este numărul arcelor cu o extremitate în S și cealaltă în T, deoarece $c(e) = 1, \forall e \in E(G_1)$. Această mulțime de arce generează în G o mulțime de muchii de același cardinal și cu proprietatea că decontează, prin îndepărtare, s și t .
Rezultă că avem $c(S,T) = v(x^0) = p_e(s,t)$ muchii în G care deconectează, prin îndepărtare, s și t. Deci $c_e(s,t) \leq p_e(s,t)$.

Cum, inegalitatea $c_e(s,t) \geq p_e(s,t)$ este evidentă, rezultă că teorema este demonstrată.

Corolar. *Dacă G este un graf conex $\lambda(G)$ (valoarea maximă a lui $p \in \mathbf{Z}_+$ astfel încît G este $p-$muchie-conex) este*

$$\min_{\substack{s,t \in V(G) \\ s \neq t}} c_e(s,t). \qquad (*)$$

Rezultă că, pentru a afla $\lambda(G)$, rezolvăm cele $\frac{n(n-1)}{2}$ probleme de flux, descrise în demonstrația teoremei.
Totuși, să observăm că pentru o pereche fixată s și t avem: dacă (S,T) este secțiunea de capacitate minimă, atunci
$\forall v \in S$ și $\forall w \in T$ $c_e(v,w) \leq c(S,T)$ $\quad (**)$.

În particular, dacă (s,t) este perechea pentru care se realizează minimul în (*) vom avea egalitate în (**).
Rezultă că dacă fixăm un vîrf s_0 și rezolvăm $n-1$ probleme de flux cu $t \in V - s_0$ se va obține în mod necesar o pereche s_0, t_0 pentru care $c(s_0,t_0) = \lambda(G)$ (ne asigurăm, astfel, depistarea unui vîrf t_0, care să nu fie în aceeași clasă cu s_0 în partiția (S,T)).
Rezultă că în $O(n \cdot (nm + n^2 c)) = O(n^2 m)$ **se pot determina $\lambda(G)$ și o mulțime separatoare de muchii de cardinal minim în G.**

D. Determinarea numărului de conexiune al unui graf.

Dacă $G = (V, E)$ este un graf și
$s, t \in V, \quad s \neq t, \quad st \notin E$ atunci, notînd
-$\mathbf{p(s,t)} =$ numărul maxim de $st-$drumuri cu mulțimile de vîrfuri disjuncte (cu excepția extremităților),
-$\mathbf{c(s,t)} =$ cardinalul minim al unei mulțimi de vîrfuri $st-$ separatoare,

avem, din teorema lui Menger,

$$p(s,t) = c(s,t) \qquad (*)$$

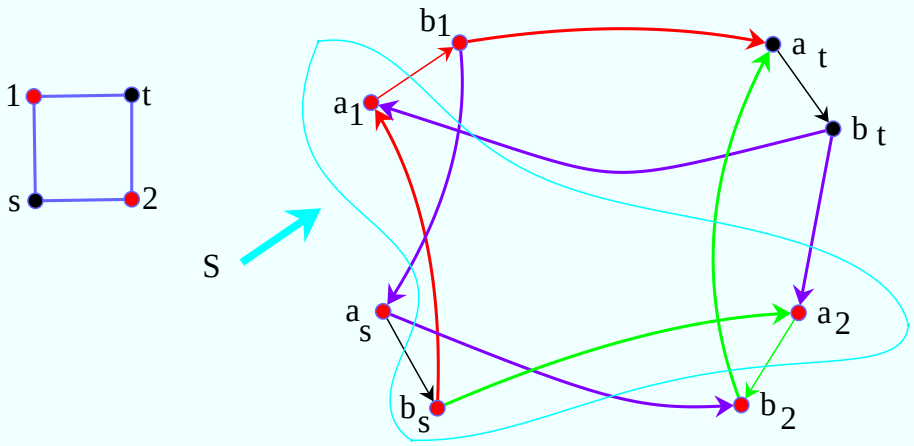
În plus, numărul de conexiune $k(G)$ al grafului G (valoarea maximă a lui $p \in \mathbf{Z}_+$ pentru care G este $p-$conex) este

$$k(G) = \begin{cases} n-1 & \text{dacă } G = K_n \\ \min\limits_{\substack{s,t \in V \\ st \notin E}} c(s,t) & \text{dacă } G \neq K_n \end{cases} \qquad (**)$$

Vom arăta că egalitatea (*) rezultă și din teorema fluxului maxim-secţiunii minime pe o reţea convenabil aleasă.

Fie $G_1 = (V(G_1), E(G_1))$ digraful construit din G astfel:
-$\forall v \in V$ considerăm $a_v, b_v \in V(G_1)$ și $a_v b_v \in E(G_1)$;
- $\forall vw \in E$ considerăm $b_v a_w, b_w a_v \in E(G_1)$.

Exemplu:



Arcele groase au capacitate infinit, cele subtiri 1.

Definim $c : E(G_1) \to \mathbf{Z}_+$ prin

$$c(e) = \begin{cases} 1 & \text{dacă } e = a_v b_v \\ \infty & \text{altfel.} \end{cases}$$

Considerām rețeaua $R = (G_1, b_s, a_t, c)$.

Fie x^0 un flux întreg de la b_s la a_t în R de valoare maximă.

În vîrfurile $b_v (v \in V)$ intrā exact un arc de capacitate 1 și din vîrfurile $a_v (v \in V)$ pleacā exact un arc de capacitate 1. Rezultā cā, pentru ca sā fie satisfācutā legea de conservare a fluxului, $x^0_{ij} \in \{0, 1\} \quad \forall ij \in E(G_1)$.

Aceasta înseamnā cā x^0 se poate descompune în $v(x^0)$ fluxuri x^k, fiecare de valoare 1, și astfel încît arcele pe care x^k sînt nenule corespund la $v(x^0)$ drumuri disjuncte de la b_s la a_t în G_1, care induc o mulțime de $v(x^0)$ drumuri intern disjuncte de la s la t în G.
Cum, pe de altā parte, dintr-o mulțime de p drumuri intern disjuncte de la s la t în G, se pot construi p drumuri intern disjuncte de la b_s la a_t în G_1 pe care se poate transporta cîte o unitate de flux, rezultā cā avem
:

$$\mathbf{v(x^0) = p(s, t)}.$$

Fie (S,T) o secţiune în R astfel încît $v(x^0) = c(S,T)$.
Cum $v(x^0)$ este finit, rezultă că $\forall i \in S, \forall j \in T$ cu $ij \in E(G_1)$,avem $c(ij) < \infty$, deci $c(ij) = 1$, adică $\exists u \in V$ astfel încît $i = a_u$ şi $j = b_u$.

Deci secţiunii (S,T) îi corespunde o mulţime de vîrfuri $A_0 \subseteq V$ astfel încît, $c(S,T) = |A_0|$ şi evident A_0 este $st-$ separatoare.
Cum, pe de altă parte, $\forall A$ $st-$separatoare, $|A| \geq p(s,t) = v(x_0)$ rezultă că
$c(s,t) = |A_0| = c(S,T) = v(x_0) = p(s,t)$.

Demonstraţia de mai sus, arată că pentru a determina $k(G)$ va fi suficient să determinăm minimul din (**) prin rezolvarea a $|E(\overline{G})|$ probleme de flux, unde $\overline{G}$ este graful complementar al lui G.
Deci algoritmul va avea complexitatea

$$O\Big((\frac{n(n-1)}{2} - m)(nm + n^2 \log n))\Big).$$

O simplă observaţie ne conduce la un algoritm mai eficient. Evident,
$k(G) \leq \min_{v \in V} d_G(v) = \frac{1}{n}(n \cdot \min_{v \in V} d_G(v)) \leq \frac{1}{n}(\sum_{v \in V} d_G(v)) = \frac{2m}{n}$.

Dacă A_0 este o mulţime de articulaţie în G cu $|A_0| = k(G)$ atunci $G - A_0$ este neconex şi se poate partiţiona $V - A_0 = V' \cup V''$ astfel încît, $\forall v' \in V'$, $\forall v'' \in V''$ avem $p(v', v'') = k(G)$.

Rezultă că, rezolvînd o problema de flux cu $s_0 \in V'$ şi $t_0 \in V''$ va rezulta că $p(s_0, t_0) =$ valoarea fluxului maxim $= k(G)$.

Vom fi siguri că depistăm o astfel de pereche, dacă procedăm astfel:
considerăm $l = \lceil \frac{2m}{n} \rceil + 1$, alegem l vîrfuri oarecare din G, şi pentru fiecare astfel de vîrf v rezolvăm toate problemele $p(v, w)$ cu $vw \notin E$.

Se vor rezolva în total $O(nl) = O(n(\frac{2m}{n} + 1)) = O(m)$ probleme. Deci complexitatea întregului algoritm va fi $\mathbf{O(m(nm + n^2 \log n))}$.

3. Fluxuri de cost minim

Fie $R = (G, s, t, c)$ o reţea şi x un flux de la s la t în R.
Considerăm $a : E \to \mathbf{R}$ o funcţie de cost care asociază fiecărui arc $ij \in E$ $a(ij) = a_{ij}$ costul (transportului unei unităţi de flux) pe arcul ij.
Costul fluxului x se defineşte ca fiind

$$a(x) = \sum_{i,j} a_{ij} x_{ij}.$$

Problema fluxului de cost minim
Dată R o reţea, $v \in \mathbf{R}^+$ şi $a : E \to \mathbf{R}$ funcţie de cost, să se determine x^0 flux în R astfel încît

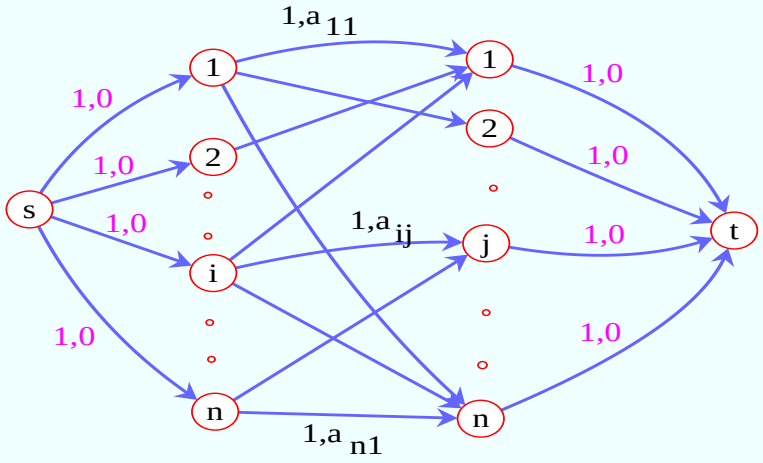
$$a(x^0) = \min\{a(x) \mid x \text{ flux în } R, v(x) = v\}.$$

Observăm că, dacă v nu depăşeşte valoarea fluxului maxim în reţeaua R, atunci problema are întotdeauna soluţii, $a(x)$ fiind liniară, iar mulţimea fluxurilor de valoare dată v fiind mărginită şi închisă în $\mathbf{R}^m$.

Exemple.

1^0. Se dispune de n lucrători și n lucrări. Costul atribuirii lucrătorului i la lucrarea j este $a_{ij}(i, j \in \{1, \ldots, n\})$. **Să se atribuie fiecare dintre cele n lucrări la cîte un lucrător, astfel încît costul total al atribuirii să fie minim.** (*Problema simplă a atribuirii*).

Considerăm rețeaua descrisă mai jos, unde pe fiecare arc este trecută mai întîi capacitatea și apoi costul. Deci $c_{ij} = 1, c_{si} = 1, a_{si} = 0, c_{jt} = 1, a_{jt} = 0 \quad \forall i, j \in \{1, \ldots, n\}$.



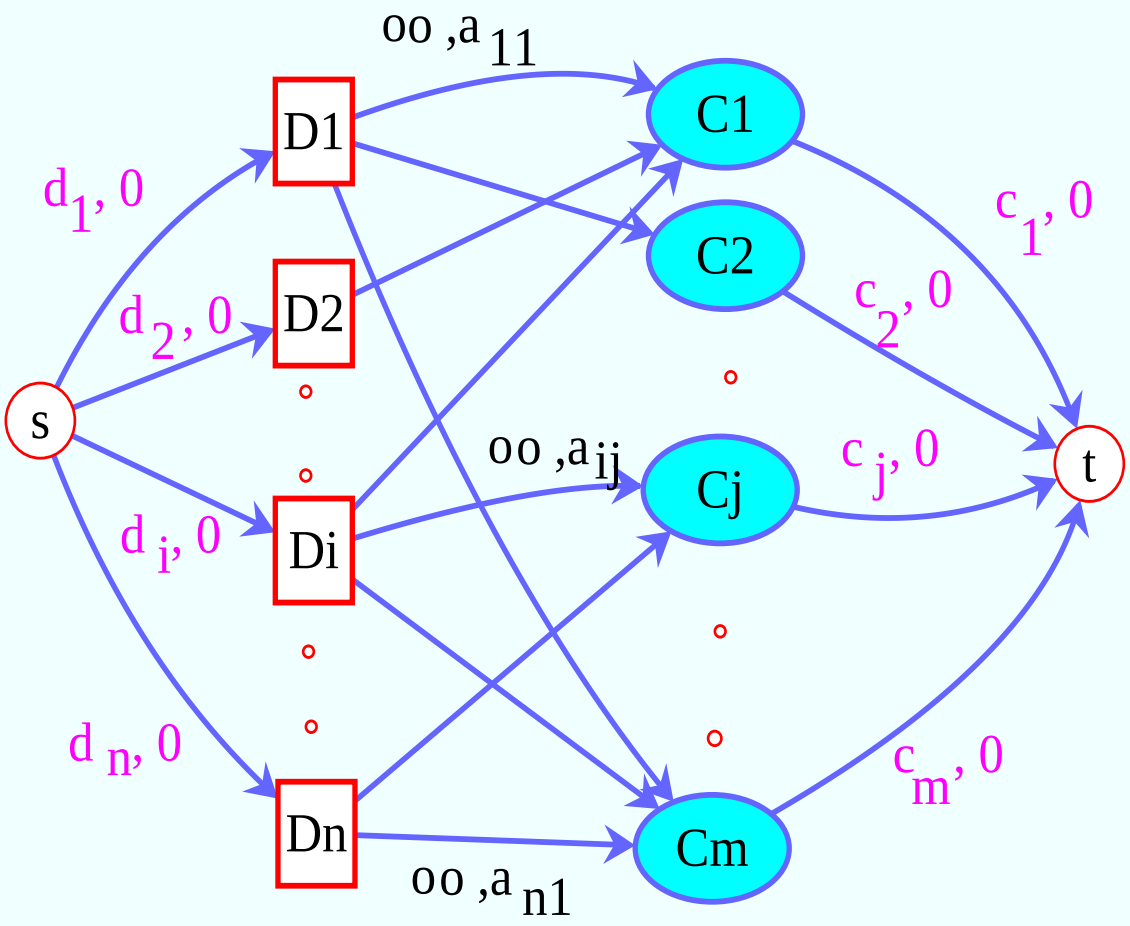
Evident, un flux întreg de valoare n și de cost minim, reprezintă soluția problemei.

Notăm că în mod similar se poate rezolva orice problemă de cuplaj perfect de pondere minimă într-un graf bipartit.

2^0.O marfă disponibilă în depozitele $D_1, \ldots, D_n$ în cantitățile $d_1, \ldots, d_n$ este solicitată în centrele de consum $C_1, C_2, \ldots, C_m$ în cantitățile $c_1, c_2, \ldots, c_m$.
Se cunoaște costul a_{ij} al transportului unei unități de marfă de la depozitul D_i la centrul de consum $C_j (\forall i \in \{1, \ldots, n\}\ \forall j \in \{1, \ldots, m\})$.
Se cere **să se stabilească un plan de transport care să satisfacă toate cererile și să aibă costul total minim** *(problema simplă a transporturilor Hitchcock-Koopmans)*.

Evident, problema are soluție numai dacă $\sum_{i=1,n} d_i \geq \sum_{j=1,m} c_j$.

În acest caz, un flux de cost minim și de valoare $v = \sum_{i=1,m} c_i$ în rețeaua următoare, rezolvă problema.



Definiţie. Fie x un flux în $R = (G, s, t, c)$ şi $a : E \to \mathbf{R}$ o funcţie de cost.
Dacă P este un C-drum în R relativ la fluxul x, atunci **costul drumului** P se defineşte

$$a(P) = \sum_{\substack{ij \in P \\ ij \ direct}} a_{ij} - \sum_{\substack{ij \in P \\ ij \ invers}} a_{ji}.$$

Dacă C este un C-drum închis, $a(C)$ se calculează după aceeaşi formulă, după stabilirea unui sens de parcurgere a lui C (este posibil ca ambele sensuri de parcurgere ale lui C să satisfacă definiţia unui C-drum).

Observaţii: 1^0 Din definiţia dată, rezultă că dacă P este drum de creştere relativ la fluxul x, atunci $x^1 = x \otimes r(P)$ este un flux de valoare $v(x^1) = v(x) + r(P)$ şi de cost $a(x) + r(P) \cdot a(P)$.

2^0 Dacă C este un C-drum închis relativ la x, atunci $x^1 = x \otimes r(C)$ este un flux de valoare $v(x^1) = v(x)$ şi de cost $a(x^1) = a(x) + r(C) \cdot a(C)$.
Dacă $a(C) < 0$ atunci x^1 este un flux de aceeaşi valoare ca şi x, dar de cost strict mai mic.

Teoremă. 8. *Un flux de valoare v este de cost minim dacă și numai dacă nu admite C-drumuri închise de cost negativ.*

Demonstraţie: Necesitatea este evidentă din observaţia anterioară.

Suficienţa. Fie x un flux de valoare v, care nu admite C- drumuri închise de cost negativ. Fie x^* un flux de valoare v, de cost minim (există !) astfel încît

$\Delta(x, x^*) = \min\{\Delta(x, x') \mid x'$ flux de val. v și cost minim$\}$

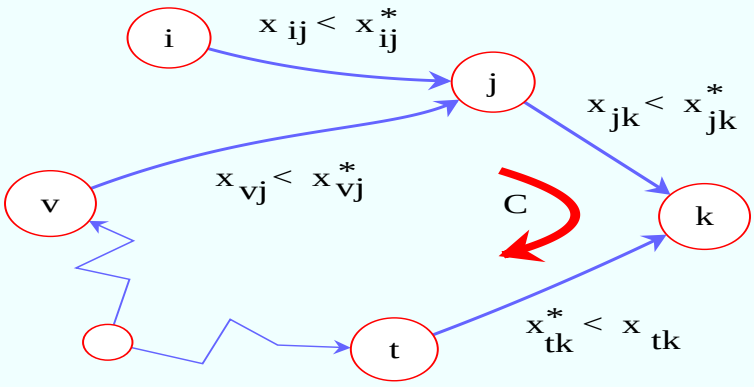
unde $\Delta(x, x') = |\{ij \mid x_{ij} \neq x'_{ij}\}|$.

Dacă $\Delta(x, x^*) = 0$ rezultă $x = x^*$ și deci x este de cost minim.

Dacă $\Delta(x, x^*) > 0$, fie ij astfel încît $x_{ij} \neq x^*_{ij}$. Presupunem $0 \leq x_{ij} < x^*_{ij} \leq c_{ij}$ (altfel, raţionamentul este similar). Din legea de conservare a fluxurilor rezultă că

$\exists\ jk \in E$ astfel încît $0 \leq x_{jk} < x^*_{jk} \leq c_{jk}$, sau

$\exists\ kj \in E$ astfel încît $0 \leq x^*_{kj} < x_{kj} \leq c_{jk}$.



Repetînd acest raţionament, deoarece numărul vîrfurilor este finit, se va obţine C, un C-drum închis relativ la x în R.

Considerînd sensul invers de parcurgere pe C se obţine un C-drum C', închis relativ la x^*.

Deoarece $a(C) \geq 0$ din ipotezā, iar $a(C') = -a(C)$, rezultā, din necesitatea teoremei (x^* este de cost minim), cā $a(C) = 0$.

Modificînd fluxul x^* cu $\delta(C')$ pe C', unde

$$\delta(C') = \min\Big\{ \min_{kj \text{ direct în } C'} x_{kj} - x^*_{kj}, \quad \min_{kj \text{ invers în } C'} x^*_{jk} - x_{jk} \Big\}$$

se obţine un flux x' cu $v(x') = v(x^*) = v$, $a(x') = a(x^*) + \delta(C') \cdot a(C') = a(x^*)$, deci de

cost minim, dar, cu $\Delta(x, x') < \Delta(x, x^*)$, contradicție.

Deci $\Delta(x, x^*) = 0$, și demonstrația este încheiată.

Teoremă. 9. *Dacă x este un flux de valoare v și de cost minim iar P_0 este un drum de creștere, astfel încît*

$a(P_0) = \min\{a(P) \mid P$drum de creștere relativ la $x\}$,

atunci $x^1 = x \otimes r(P_0)$ este un flux de valoare $v(x^1) = v + r(P_0)$ și de cost minim.

Linia demonstrației este următoarea : presupunînd prin reducere la absurd că x^1 nu este de cost minim, atunci x^1 admite un C-drum închis C de cost negativ. Cum x era flux de cost minim rezultă că $E(C) \cap E(P_0) \neq \emptyset$.

Dacă $ij \in E(C) \cap E(P_0)$, atunci va rezulta că $P_0 \cup C - ij$ conține un drum de creștere relativ la x de cost mai mic decît P_0.

Un drum de creștere de cost minim poate fi depistat cu ajutorul algoritmilor de drum minim. Dacă x este un flux în R și $a : E \to \mathbf{R}$ este funcția de cost atunci considerînd $a_{ij} = \infty$ dacă $ij \notin E$ (caz în care $x_{ij} = 0$), construim

$$\overline{a}_{ij} = \begin{cases} a_{ij} & \text{dacă } x_{ij} < c_{ij} \text{ și } x_{ji} = 0, \\ \min\{a_{ij}, -a_{ji}\} & \text{dacă } x_{ij} < c_{ij} \text{ și } x_{ji} > 0, \\ -a_{ji} & \text{dacă } x_{ij} = c_{ij} \text{ și } x_{ji} > 0, \\ +\infty & \text{dacă } x_{ij} = c_{ij} \text{ și } x_{ji} = 0. \end{cases}$$

Un drum de pondere minimă de la s la t în raport cu ponderile $\overline{a}_{ij}$ corespunde unui drum minim de creștere în R relativ la fluxul x.

Un circuit de pondere negativă în raport cu ponderile $\overline{a}_{ij}$ corespunde unui C-drum închis în R relativ la x, de cost negativ.

Rezultă, următorul algoritm pentru rezolvarea problemei fluxului de cost minim, obținut prin combinarea mai multor algoritmi clasici (*Klein,Busacker,Gowan, etc.*).

Algoritm generic de rezolvare a problemei fluxului de cost minim

{
0: Se consideră $x = (x_{ij})$ un flux cu valoarea $v' \leq v$;
{x poate fi fluxul nul sau un flux y determinat cu ajutorul algoritmului de flux maxim și apoi considerînd $x = (\frac{v}{v(y)}y)$}
1: **while** ($\exists$ circuite de pondere < 0 relativ la $\overline{a}_{ij}$) **do**
{ determină un astfel de circuit;
modifică fluxul pe acest circuit
}
2: **while** $v(x) < v$ **do**
{ aplică un algoritm de drum minim în raport cu ponderile $\overline{a}_{ij}$ pentru depistarea unui C-drum P de cost minim;
$x \leftarrow x \bigotimes \min(r(P), v - v(x))$
}
}

Complexitatea pasului 2 este $O(n^3v)$, dacă se pleacă de la fluxul nul. Complexitatea pentru pasul 2 este $O(n^3v)$, dacă se pleacă de la fluxul nul. Se poate dovedi că pasul 1 se poate implementa astfel ca numărul iteraţiilor să fie $O(nm^2logn)$.

VI. Reduceri polinomiale pentru probleme de decizie pe grafuri.

Definiţie. Spunem că problema de decizie $P_1 : I_1 \to \{da, nu\}$ **se reduce polinomial** la problema de decizie $P_2 : I_2 \to \{da, nu\}$ şi notăm aceasta prin $P_1 \propto P_2$, dacă există o funcţie $\Phi : I_1 \to I_2$ polinomial calculabilă, astfel încît, $\forall i \in I_1 \quad P_1(i) = P_2(\Phi(i))$.

Funcţia Φ se va da indicând un algoritm care construieşte pentru orice instanţă $i_1 \in I_1$, în timp polinomial în raport cu $|i_1|$, o instanţă $i_2 \in I_2$ cu proprietatea că $P_1(i_1) = da$ dacă şi numai dacă $P_2(i_2) = da$.

Se observă că relaţia de reducere polinomială $\propto$ este o relaţie tranzitivă pe mulţimea problemelor de decizie (datoritā închiderii mulţimii funcţiilor polinomiale la compunere).

Din punct de vedere algoritmic, construcţia din spatele oricărei reduceri polinomiale este interesantă evidenţiind modul în care prima problemă poate fi rezolvată eficient cu ajutorul unui oracol care rezolvă a doua problemă.

Vom considera cunoscut faptul că $SAT \propto 3SAT$ unde

SAT

Instanță: $U = \{u_1, \ldots, u_n\}$ o mulțime finită de var. booleene.

$C = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ o formulă în formă conjunctivă peste U:

$C_i = v_{i_1} \vee v_{i_2} \vee \ldots \vee v_{i_{k_i}} \quad \forall i = \overline{1,m}$, unde

$\forall i_j \quad \exists \alpha \in \{1, \ldots, n\}$ a. î. $v_{i_j} = u_\alpha$ sau $v_{i_j} = \overline{u}_\alpha$.

Intrebare: Există o atribuire $t : U \to \{A, F\}$ a. î. $t(C) = A$?

$3SAT$ este cazul particular al lui SAT în care fiecare clauză C_i are exact trei literali ($k_i = 3$), un literal v_{i_j} fiind, așa cum este descris mai sus, o variabilă sau negația ei.

Problema SAT este celebră datorită teoremei lui Cook (1971): *SAT este $NP-$completă.*

1. Mulţimi stabile

SM

Instanţă: $G = (V, E)$ graf şi $k \in \mathbf{N}$.

Intrebare: Există S mulţime stabilă în G a. î. $|S| \geq k$?

Teoremă. 1. *(Karp 1972)* $3SAT \propto SM$.

Demonstraţie: Fie $U = \{u_1, u_2, \ldots, u_n\}$, $(n \in \mathbf{N}^*)$, $C = C_1 \wedge \ldots \wedge C_m$ $(m \in \mathbf{N}^*)$ cu $C_i = v_{i_1} \vee v_{i_2} \vee v_{i_3}$ $\forall i = 1, m$, (unde $\forall v_{i_j}$ $\exists \alpha \in \{1, \ldots, n\}$ astfel încît $v_{i_j} = u_\alpha$ sau $v_{i_j} = \overline{u}_\alpha$), reprezentînd datele unei instanţe oarecare a problemei 3SAT.

Vom construi în timp polinomial în raport cu $m + n$, un graf G şi $k \in \mathbf{N}$ astfel încît există o atribuire t a valorilor de adevăr sau fals pentru variabilele booleene din U care să facă adevărată formula C, dacă şi numai dacă există o stabilă S în graful G astfel încît $|S| \geq k$.

Graful G va fi construit astfel:

(1) Pentru orice $i \in \{1, \ldots, n\}$ considerām grafurile disjuncte $T_i = (\{u_i, \overline{u}_i\}, \{u_i\overline{u}_i\})$.

(2) Pentru orice $j = 1, m$ considerām grafurile disjuncte

$Z_j = (\{a_{j1}, a_{j2}, a_{j3}\}, \{a_{j1}a_{j2}, a_{j2}a_{j3}, a_{j3}a_{j1}\})$.

(3) Pentru orice $j = 1, m$ considerām mulţimea de muchii

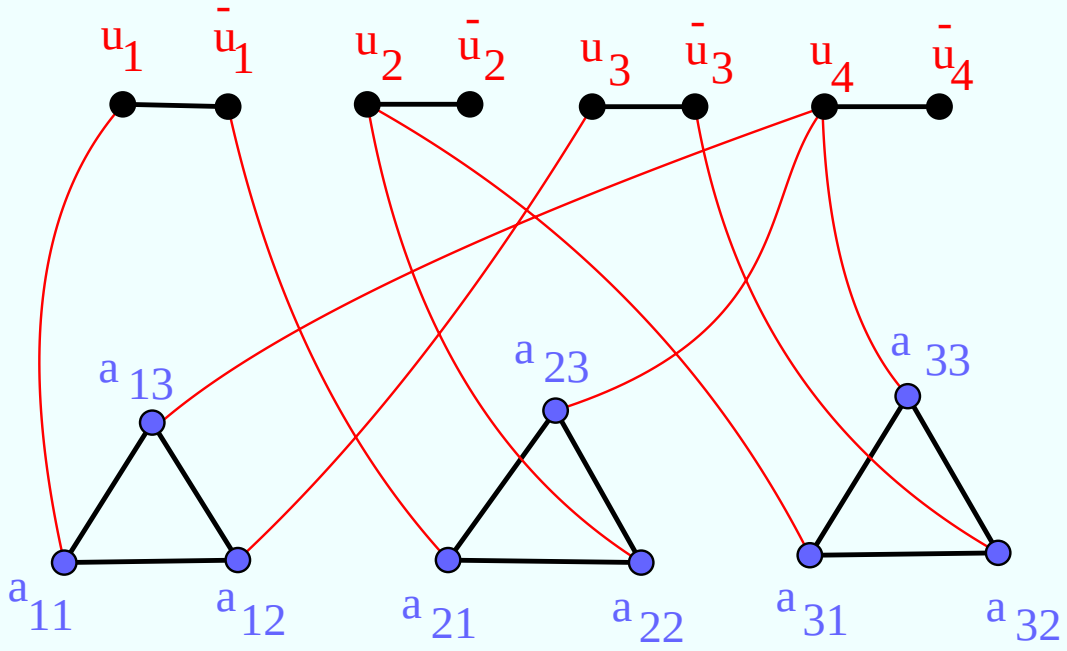
$E_j = \{a_{j1}v_{j1}, a_{j2}v_{j2}, a_{j3}v_{j3}\}$ unde $v_{j1} \vee v_{j2} \vee v_{j3}$ este factorul C_j.

Considerām $V(G) = \cup_{i=1}^{n} V(T_i) \cup \cup_{j=1}^{m} V(Z_j)$ şi $E(G) = \cup_{i=1}^{n} E(T_i) \cup \cup_{j=1}^{m} (E(Z_j) \cup E_j)$.

Evident, construcţia este polinomială în raport cu $m + n$ (G are $2n + 3m$ vîrfuri şi $n + 6m$ muchii).

Considerām $k = n + m$.

Exemplu: $U = \{u_1, u_2, u_3, u_4\}$;
$C = (u_1 \vee u_3 \vee u_4) \wedge (\overline{u}_1 \vee u_2 \vee u_4) \wedge (u_2 \vee \overline{u}_3 \vee u_4)$;
$k = 4 + 3 = 7$.



Să **presupunem că răspunsul la problema SM pentru instanța G și k astfel construite este da.** Deci $\exists S \in \mathcal{S}_G$ astfel încît $|S| \geq k$. Cum orice mulțime stabilă poate avea cel mult un vîrf din orice $V(T_i)$ și din orice $V(Z_j)$ ($i = 1, n$, $j = 1, m$) rezultă că avem $|S| = k$ și deci $|S \cap V(T_i)| = 1$, $|S \cap V(Z_j)| = 1$ $\forall i = 1, n$ și $\forall j = 1, m$.

Considerăm $t : U \to \{A, F\}$ prin

$$t(u_i) = \begin{cases} A & \text{dacă } S \cap V(T_i) = \{\overline{u}_i\} \\ F & \text{dacă } S \cap V(T_i) = \{u_i\}. \end{cases}$$

Atunci, $\forall j = 1, m$ **avem** $t(C_j) = A$ **(și deci** $t(C) = A$**).**

În adevăr, $\forall j = 1, m$ dacă $C_j = v_{j1} \vee v_{j2} \vee v_{j3}$ și $S \cap V(Z_j) = a_{jk}$ $(k \in \{1, 2, 3\})$ atunci, deoarece $a_{jk}v_{jk} \in E$ rezultă că $v_{jk} \notin S$.

Dacă $v_{jk} = u_\alpha$, atunci $u_\alpha \notin S$ deci $\overline{u}_\alpha \in S$ și din definiția lui t avem $t(u_\alpha) = A$, adică $t(v_{jk}) = A$ ceea ce implică $t(C_j) = A$.

Dacă $v_{jk} = \overline{u}_\alpha$, atunci $\overline{u}_\alpha \notin S$ implică $u_\alpha \in S$, deci $t(\overline{u}_\alpha) = A$, adică $t(v_{jk}) = A$, ceea ce implică $t(C_j) = A$.

Reciproc, dacă răspunsul la problema $3SAT$ **este da,** atunci există o atribuire $t : U \to \{A, F\}$ astfel încît $t(C_j) = A$ $\forall j = 1, m$.

Considerām în graful G mulțimea stabilă $\overline{S}_1$, $\overline{S}_1 = \cup_{i=1,n} V_i'$, unde

$$V_i' = \begin{cases} \{\overline{u}_i\} & \text{dacă } t(u_i) = A \\ \{u_i\} & \text{dacă } t(u_i) = F \ . \end{cases}$$

Atunci, $\forall j = 1, m$, cum $t(C_j) = A$, rezultă că există $k_j \in \{1, 2, 3\}$astfel încît $t(v_{jk_j}) = A$. Considerām $\overline{S}_2 = \cup_{j=1,m} \{a_{jk_j}\}$.

Evident, $\overline{S}_2$ este stabilă în G. Am construit $\overline{S}_1 \in \mathcal{S}_G$ cu $|\overline{S}_1| = n$, $\overline{S}_2 \in \mathcal{S}_G$ cu $|\overline{S}_2| = m$.

Considerām $\overline{S} = \overline{S}_1 \cup \overline{S}_2$. Evident, $|\overline{S}| = n + m = k$ (deci $|\overline{S}| \geq k$) și în plus $\overline{S}$ este mulțime stabilă în G (**deci răspunsul la SM pentru intrarea G, k este da**).

Faptul că $\overline{S}$ este mulțime stabilă în G rezultă astfel: dacă $\exists v, w \in \overline{S}$ astfel încît $vw \in E(G)$ atunci o extremitate este din $\overline{S}_1$ și cealaltă din $\overline{S}_2$.

Presupunînd $v \in \overline{S}_1$ avem două cazuri de considerat

a) $v = u_\alpha$, $w = a_{jk_j}$ $\alpha \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$, $k_j \in \{1, 2, 3\}$ și $v_{jk_j} = u_\alpha$. Cum $t(v_{jk_j}) = A$ rezultă $t(u_\alpha) = A$ deci $u_\alpha \notin \overline{S}_1$, contradicție.

b) $v = \overline{u}_\alpha$, $w = a_{jk_j}$ $\alpha \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$, $k_j \in \{1, 2, 3\}$ și $v_{jk_j} = \overline{u}_\alpha$. Cum $t(v_j k_j) = A$, rezultă $t(\overline{u}_\alpha) = A$ deci $t(u_\alpha) = F$ ceea ce implică $\overline{u}_\alpha \notin \overline{S}_1$, contradicție.

Cu aceasta teorema este complet demonstratā.

Sā observām cā reducerea lui SAT la SM este complet similarā, singura deosebire fiind cā grafurile Z_i sunt grafuri complete cu k_i vârfuri.

2. Colorarea vârfurilor.

COL

Instanţă: $G = (V, E)$ graf şi $p \in \mathbf{N}^*$.

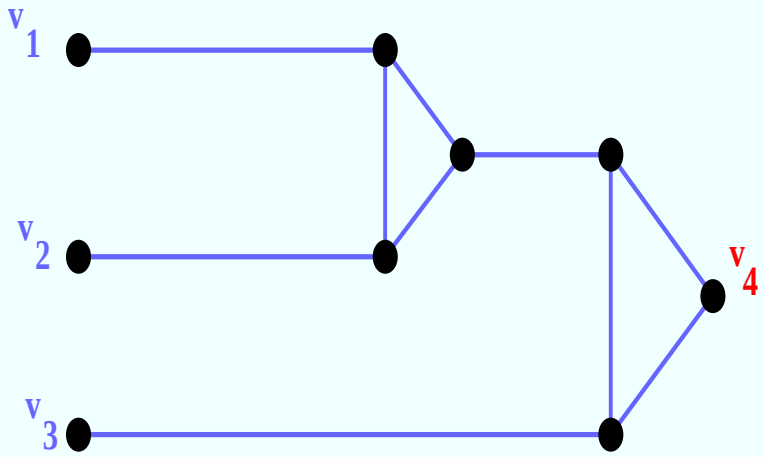
Intrebare: Există o p-colorare a vârfurilor lui G?

Teoremă. 2. $3SAT \propto COL.$

Această teoremă evidenţiază complexitatea problemelor de colorare a vîrfurilor unui graf.

Vom demonstra chiar mai mult: fixînd $p = 3$ în enunţul lui COL, reducerea polinomială a lui 3SAT este încă posibilă !

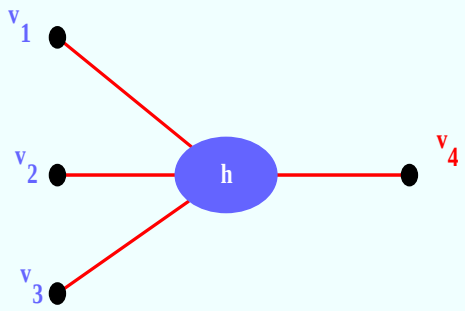
Lemã. 1. *Fie H graful:*



a) *Dacă c este o 3-colorare a lui H astfel încît $c(v_1) = c(v_2) = c(v_3) = a \in \{1,2,3\}$ atunci în mod necesar $c(v_4) = a$.*
b) *Dacă $c : \{v_1, v_2, v_3\} \to \{1,2,3\}$ satisface $c(\{v_1, v_2, v_3\}) \neq \{a\}$ atunci c poate fi extinsă la o 3-colorare c a lui H cu $c(v_4) \neq a$.*

Demonstraţia lemei se poate face examinînd lista 3-colorărilor lui H.

În cele ce urmează, vom desemna (pentru simplitate) graful H astfel:



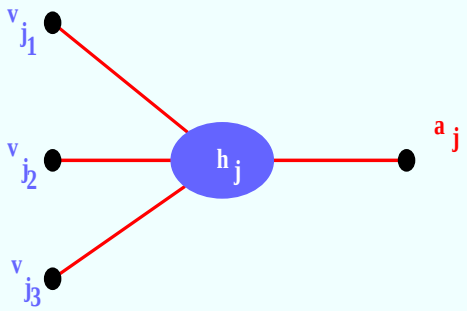
Demonstraţia teoremei 2. Considerām datele unei instanţe ale problemei 3SAT:

$U = \{u_1, \ldots, u_n\}$ o mulţime de variabile booleene ($n \in \mathbf{N}^*$) şi $C = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ ($m \in \mathbf{N}^*$) o formulā astfel încît $\forall i = 1, m$ $C_i = v_{i_1} \vee v_{i_2} \vee v_{i_3}$, unde $\forall j = 1, 3$ $\exists \alpha$ astfel încît $v_{i_j} = u_\alpha$ sau $v_{i_j} = \overline{u}_\alpha$.

Vom construi un graf G, astfel încît, considerînd $p = 3$ în COL, vom obţine cā G este 3-colorabil dacā şi numai dacā rāspunsul la 3SAT este da, adicā există $t : U \to \{A, F\}$, astfel încît $t(C) = A$. În plus, construcţia lui G se va face în timp polinomial, parcurgînd următoarele etape:

1. $\forall i = 1, n$ considerām grafurile disjuncte (V_i, E_i) unde $V_i = \{u_i, \overline{u}_i\}$ și $E_i = \{u_i \overline{u}_i\}$.

2. $\forall j = 1, m$, pentru $C_j = v_{j_1} \vee v_{j_2} \vee v_{j_3}$, considerām grafurile:



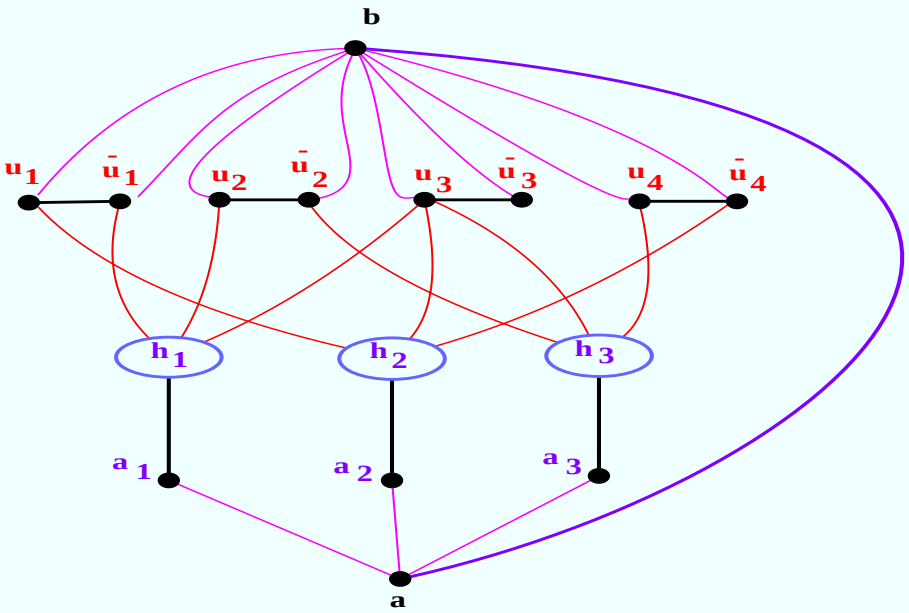
unde v_{j_k} $(k = 1, 3)$ sînt vîrfurile de la pasul 1, corespunzātoare literalilor v_{j_k}, grafurile h_j sînt disjuncte și a_j sînt vîrfuri distincte.

3. Considerām a, un vîrf diferit de toate cele construite în pașii 1 și 2 și unim a cu vîrfurile a_j $j = 1, m$.

4. Considerām b un vîrf nou, unit cu u_i și $\overline{u}_i$ $\forall i = 1, n$ și de asemenea, cu vîrful a.

Graful G astfel construit are un număr liniar de vîrfuri în raport cu $n + m$.

Exemplu: $U = \{u_1, u_2, u_3, u_4\}$,
$C = (\overline{u}_1 \vee u_2 \vee u_3) \wedge (u_1 \vee u_3 \vee \overline{u}_4) \wedge (\overline{u}_2 \vee u_3 \vee u_4)$
Graful G va fi:



Presupunem ca răspunsul la 3SAT este da.
Deci $\exists t : U \to \{A, F\}$ astfel încît $t(C) = A$, deci $t(C_j) = A \ \forall j = 1, m$.

Construim o 3-colorare C a grafului G.

Definim mai întîi $c(u_i)$ și $c(\overline{u}_i)$ $\forall i = 1, n$ astfel:

$c(u_i) = 1$ și $c(\overline{u}_i) = 2$, dacă $t(u_i) = A$ și

$c(u_i) = 2$ și $c(\overline{u}_i) = 1$, dacă $t(u_i) = F$.

Se observă că dacă v este un literal (u_α sau $\overline{u}_\alpha$), atunci vîrful v este colorat $c(v) = 2$ dacă și numai dacă $t(v) = F$.

Deci $\forall j = 1, m$ nu avem $c(v_{j_1}) = c(v_{j_2}) = c(v_{j_3}) = 2$.

Folosind lema 1,b) rezultă că putem extinde în fiecare graf h_j colorarea c astfel încît $c(a_j) \neq 2$, deci $c(a_j) \in \{1, 3\}$.

Rezultă că atribuind $c(a) = 2$ și $c(b) = 3$, c este o 3- colorare a lui G.

Reciproc, **presupunem că G este 3-colorabil.** Putem presupune (eventual renumerotînd culorile) că $c(b) = 3$ și $c(a) = 2$.

Va rezulta că $\{c(u_i), c(\overline{u}_i)\} = \{1, 2\}$ și $c(a_j) \in \{1, 3\}$ $\forall i = 1, n$, $\forall j = 1, m$.

Din lema 1 a) rezultă că nu vom avea $c(v_{j1}) = c(v_{j2}) = c(v_{j3}) = 2$ $\forall j = 1, 3$.

Deci $\forall j = 1, m$ $\exists v_{jk}$ astfel încît $c(v_{jk}) = 1$.

Definim $t : U \to \{A, F\}$ prin

$$t(u_i) = A \Leftrightarrow c(u_i) = 1.$$

Conform observației anterioare, vom avea că $t(C_j) = A$ $\forall j = 1, m$, deci $t(C) = A$, adică **răspunsul la 3SAT este da.**

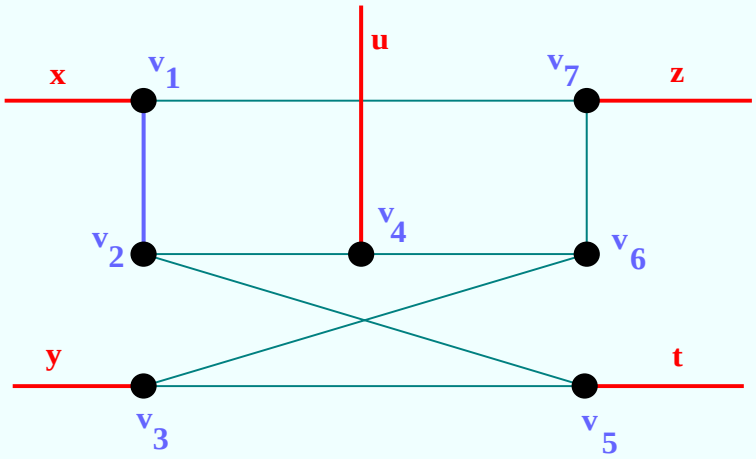
3. Colorarea muchiilor

Considerām următoarea problemă de decizie.

HOL
Instanţă: $G = (V, E)$ graf 3-regulat.
Intrebare: Există o 3-colorare a muchiilor lui G?

Teoremă. 3. *(Holyer 1981)* $3SAT \propto HOL$.

Lemă. 2. *Fie* H *graful desenat mai jos.*



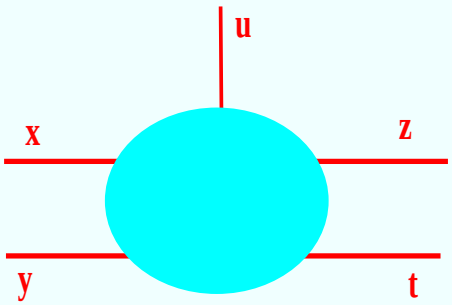
În orice 3-colorare c *a muchiilor lui* H*, avem*
$c(x) = c(y)$, $|c(\{z, t, u\})| = 3$ *sau*
$c(z) = c(t)$, $|c(\{x, y, u\})| = 3$.

Demonstraţie Fie c o 3-colorare a lui H.

a) presupunem $c(x) = c(y)$. Eventual, după o renumerotare a culorilor vom avea $c(x) = c(y) = 1$ şi $c(v_1v_7) = 2$, $c(v_1v_2) = 3$.
a1).$c(u) = 1$. Avem următorul şir de implicaţii: $c(v_2v_4) = 2$,$c(v_2v_5) = 1$,$c(v_4v_6) = 3$,$c(v_3v_6) = 2$,$c(v_3v_5) = 3$, $c(t) = 2$,$c(v_6v_7) = 1$,$c(z) = 3$.
a2).$c(u) = 2$. Avem următorul şir de implicaţii: $c(v_2v_4) = 1$,$c(v_2v_5) = 2$,$c(v_4v_6) = 3$,$c(v_3v_6) = 2$,$c(v_3v_5) = 3$, $c(t) = 1$,$c(v_6v_7) = 1$,$c(z) = 3$.
a3).$c(u) = 3$. Muchia v_2v_4 poate avea culorile 1 sau 2.
a3.1.)$c(v_2v_4) = 1$. Avem atunci, $c(v_2v_5) = 2$,$c(v_4v_6) = 2$,$c(v_3v_6) = 3$, şi nu putem atribui $c(v_3v_5)$.
a3.2.)$c(v_2v_4) = 2$. Avem atunci, $c(v_2v_5) = 1$,$c(v_4v_6) = 1$, $c(v_6v_7) = 3$, $c(z) = 1$,$c(v_3v_6) = 2$,$c(v_3v_5) = 3$, $c(t) = 2$.

b) Argument similar, dacă $c(z) = c(t)$.

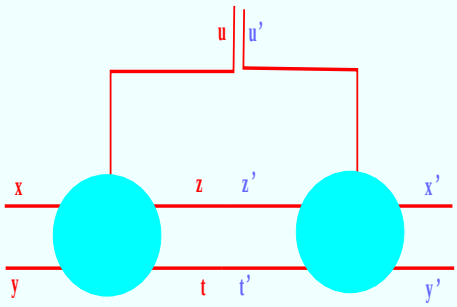
Să observăm că în graful H muchiile etichetate nu au precizată una din extremități. În construcția pe care o vom face, aceste extremități neprecizate vor fi identificate 2 cîte 2; cu alte cuvinte, perechi de astfel de muchii vor fi identificate. Vom reprezenta simplificat graful H astfel:



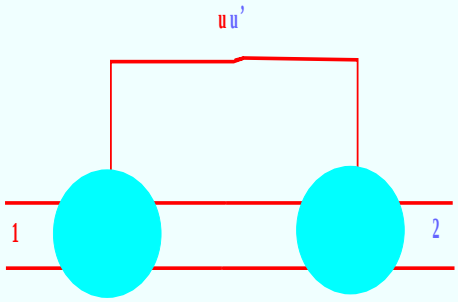
Dacă graful H este subgraf într-un graf care admite o 3- colorare a muchiilor, vom interpreta perechile de muchii (x, y) și (z, t), ca reprezentînd valorea "adevăr", dacă sînt la fel colorate, respectiv, valoarea "fals", dacă sînt colorate diferit.

Lema 2 și această interpretare justifică denumirea de componentă inversoare pentru H (dacă (x, y) este "intrare" în H atunci (z, t) este "ieșire" și H transformă adevăr în fals și fals în adevăr).

Definim graful $2H$, considerînd două componente inversoare H cărora le identificăm o pereche de muchii etichetate.

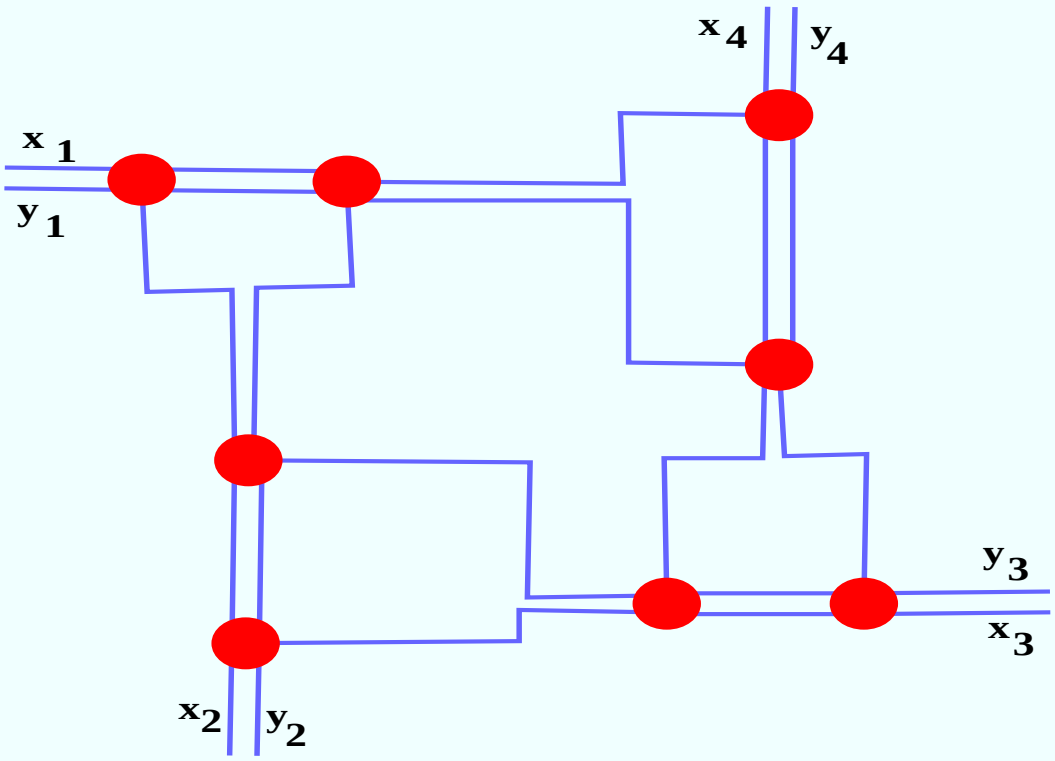


Pentru orice $n \geq 2$, $n \in \mathbf{N}$, definim graful H_n astfel: 1. H_2 se obţine identificînd muchiile u şi u' din $2H$:



2. $\forall n > 2$ H_n se construieşte astfel:

-considerām n copii izomorfe disjuncte ale lui $2H$, $2H^i$ $i = 1, n$, avînd perechile de muchii libere (x_i, y_i), (x'_i, y'_i), (u_i, u'_i).
-pentru $i = 1, n-1$ identificām perechile $(u_i, u'_i) \equiv (x'_{i+1}, y'_{i+1})$.
-identificām $(u_n, u'_n) \equiv (x'_1, y'_1)$.



H_n are n perechi de intrare (x_i, y_i) $i = 1, n$, și este construit din $2n$ componente H.

În orice 3-colorare c a lui H_n avem
a) $c(x_1) = c(y_1) \Rightarrow c(u_1) = c(u_1')$ (din lema 2 și construcția lui $2H$), $\Rightarrow c(x_2') = c(y_2')$ (din identificarea de la construcția lui H_n) $\Rightarrow c(x_2) = c(y_2)$ (lema 2) $\Rightarrow c(u_2) = c(u_2') \Rightarrow \ldots \Rightarrow c(x_n) = c(y_n)$.

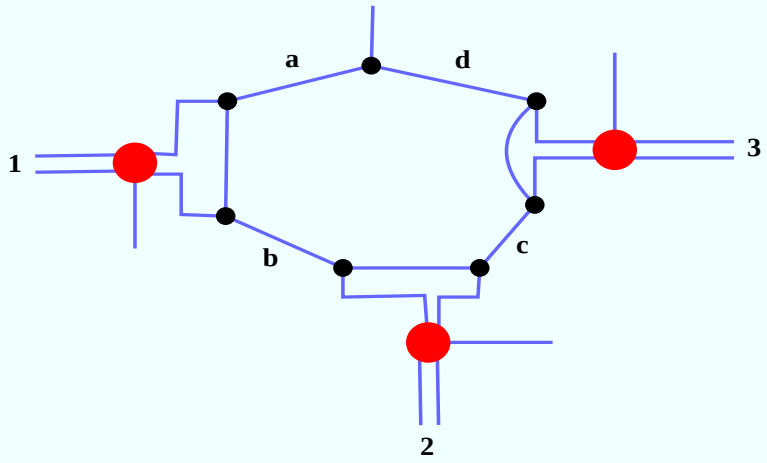
b) $c(x_1) \neq c(y_1) \Rightarrow c(x_1') \neq c(y_1')$ (lema 2) $\Rightarrow c(u_n) \neq c(u_n')$(din identificarea de la construcția lui H_n) $\Rightarrow c(x_n) \neq c(y_n)$(din lema 2, prin reducere la absurd) $\Rightarrow c(x_n') \neq c(y_n') \Rightarrow c(u_{n-1}) \neq c(u_{n-1}') \Rightarrow \ldots \Rightarrow c(x_2) \neq c(y_2)$.

Rezultă de aici că **în orice 3-colorare a muchiilor lui $H_n(n \geq 2)$ toate perechile de muchii de intrare reprezintă aceeași valoare de adevăr.**

Pentru $n = 1$ se poate construi H_1 cu o singură pereche de intrare, care să poată reprezenta oricare din cele 2 valori de adevăr. Am dovedit

Lemã. 3. *Pentru orice $n \in \mathbf{N}^*$ se poate construi în timp polinomial graful H_n avînd n perechi de muchii de "intrare", cu proprietatea cã în orice 3-colorare a muchiilor lui H_n, toate cele n perechi reprezintã aceeaşi valoare de adevãr (sau, în orice pereche de intrare cele douã muchii au culori diferite, sau, în orice pereche de intrare cele douã muchii sînt la fel colorate).*

Lemã. 4. *Fie F graful*



În orice 3-colorare c a muchiilor lui F, mãcar una din cele 3 perechi de muchii de intrare are muchiile la fel colorate (reprezintã valoarea adevãr).

Demonstraţia este imediată şi rezultă prin reducere la absurd: dacă există o 3-colorare a muchiilor lui F, astfel încît în cele 3 perechi muchiile sînt colorate diferit, atunci $c(a) = c(b), c(b) = c(c)$ şi $c(c) = c(d)$ (s-a utilizat lema 2), deci $c(a) = c(d)$ contrazicînd faptul că c este colorare a muchiilor. Notăm că există colorări c cu proprietatea din enunţ.

Demonstraţia teoremei 3. Fie $U = \{u_1, \ldots, u_n\}$ $(n \in \mathbf{N}^*)$, $C = C_1 \wedge \ldots \wedge C_m$ $(m \in \mathbf{N}^*)$, $C_i = v_{i_1} \vee v_{i_2} \vee v_{i_3}$ cu v_{i_j} literali ($v_{i_j} = u_\alpha$ sau $v_{i_j} = \overline{u}_\alpha$) reprezentînd datele unei probleme 3SAT.

Vom construi un graf 3-regulat G cu proprietatea că $\chi'(G) = 3$ dacă şi numai dacă există $t : U \to \{adev, fals\}$ astfel încît $t(C) = adev$.

Construcţia se face în timp polinomial în raport cu n şi m şi se poate descrie astfel:

1. Pentru fiecare variabilă booleană u_i $i = 1, n$ se determină x_i, numărul aparițiilor (negate sau nu) ale lui u_i în C și se consideră cîte o copie disjunctă a grafului H_{x_i}.

2. Pentru fiecare clauză C_j $j = 1, m$ se consideră cîte o copie disjunctă F_j a grafului F.

3. Pentru $j = 1, m$, fie $C_j = v_{j_1} \vee v_{j_2} \vee v_{j_3}$.

Dacă v_{j_i} $(i = 1, 3)$ este u_α și reprezintă intrarea cu numărul y a variabilei u_α atunci perechea de intrare cu numărul y din H_{x_α} se identifică cu perechea de intrare numărul i din componenta F_j.

Dacă v_{j_i} este $\overline{u}_\alpha$ se plasează o componentă inversoare H între perechea i a lui F_j și perechea de intrare cu numărul y din H_{x_α}, și se identifică corespunzător intrările lui H cu cele precizate mai sus.

4. Graful obţinut la 1-3 are anumite muchii pentru care nu s-a precizat o extremitate. Considerām o copie izomorfā disjunctā a sa şi identificām perechile corespunzātoare de astfel de muchii, din cele două grafuri. Se obţine un graf G 3-regulat.

Construcţia lui G şi lemele 3 şi 4 simulează algoritmul de calcul al valorii de adevār a lui C într-o atribuire t fixatā şi deci G **admite o 3-colorare dacā şi numai dacā** C **este satisfiabilā.**

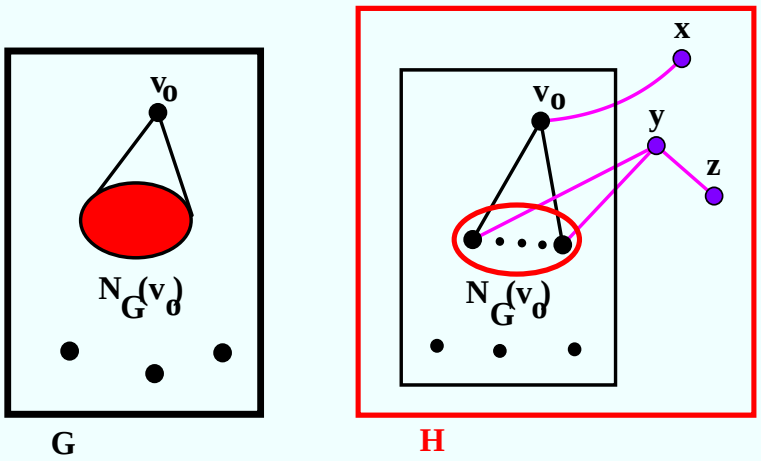
4. Probleme hamiltoniene

Definiție: Fie $G = (V(G), E(G))$ un (di)graf. Un circuit C al lui G se numește **circuit hamiltonian** dacă $V(C) = V(G)$.
Un drum deschis D al lui G se numește **drum hamiltonian** dacă $V(D) = V(G)$.
Un (di)graf care are un circuit hamiltonian se numește **(di)graf hamiltonian**.
Un (di)graf care are un drum hamiltonian se numește **(di)graf trasabil.**

Teoremă. 4. *(Nash-Williams 1969) Problemele următoare sînt polinomial echivalente:*

CH : *Dat G graf. Este G hamiltonian ?*
TR : *Dat G graf. Este G trasabil ?*
DCH: *Dat G digraf. Este G hamiltonian ?*
DTR: *Dat G digraf. Este G trasabil ?*
BCH: *Dat G graf bipartit. Este G hamiltonian ?*

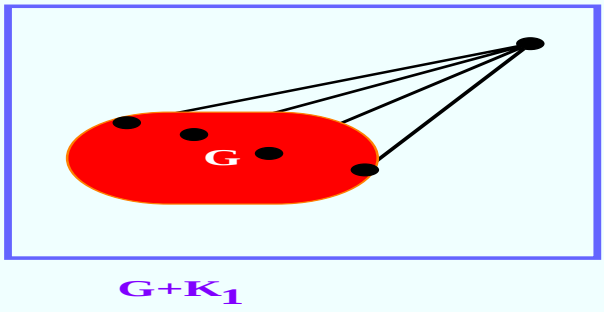
Demonstraţie:
$CH \propto TR$
Fie G un graf şi $v_0 \in V(G)$ un vîrf fixat al său. Construim (în timp polinomial) un graf H astfel încît G este hamiltonian dacă şi numai dacă H este trasabil.
Fie $V(H) = V(G) \cup \{x, y, z\}$ şi $E(H) = E(G) \cup \{xv_0, yz\} \cup \{wy \mid w \in V(G) \quad \wedge \quad wv_0 \in E(G)\}$.



Se observă că H este trasabil dacă şi numai dacă are un drum hamiltonian D cu extremităţile x şi z (care au gradul 1 în H). D există în H dacă şi numai dacă în G există un drum hamiltonian cu o extremitate v_0 şi cealaltă un vecin al lui v_0, deci dacă şi numai dacă G este hamiltonian.

$TR \propto CH$

Fie G un graf. Considerām $H = G + K_1$. H este hamiltonian dacă și numai dacă G are un drum hamiltonian.



Echivalența problemelor DCH și DTR de demonstrează în mod similar.

$CH \propto DCH$

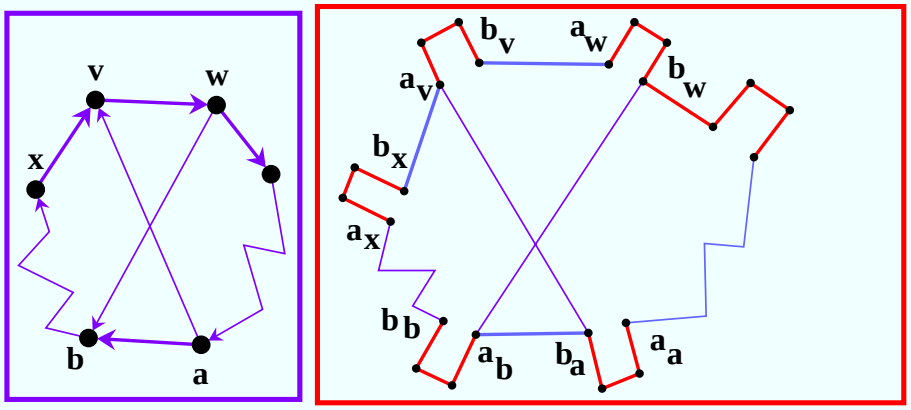
Fie G un graf. Fie D digraful obținut din G înlocuind fiecare muchie cu o pereche de arce simetrice. Orice circuit hamiltonian în G induce un circuit hamiltonian în D și reciproc.

$DCH \propto CH$

Fie $D = (V(D), E(D))$ un digraf. Pentru orice vîrf $v \in V(D)$ asociem un drum de lungime 3, P_v cu extremitățile a_v și b_v

$$P_v = (\{a_v, c_v, d_v, b_v\}, \{a_v c_v, c_v d_v, d_v b_v\}).$$

Pentru orice arc $vw \in E(D)$ considerăm muchia $b_v a_w$. Fie G graful cu $V(G) = \cup_{v \in V(D)} V(P_v)$ și $E(G) = \cup_{v \in V(D)} E(P_v) \cup \{b_v a_w \mid vw \in E(D)\}$. Evident, G se poate construi în timp polinomial în raport cu numărul de vîrfuri ale lui D.



Se observă că orice circuit C al lui D induce un circuit în G și reciproc orice circuit al lui D este generat de un circuit al lui G. Rezultă: D este hamiltonian dacă și numai dacă G este hamiltonian.

Să observăm că dacă C este un circuit al lui G, acesta este generat de un circuit C' al lui D și în plus lungimea circuitului $C, l(C)$ satisface $l(C) = 3l(C') + l(C') = 4l(C')$; deci orice circuit al lui G este de lungime pară, adică G este bipartit.

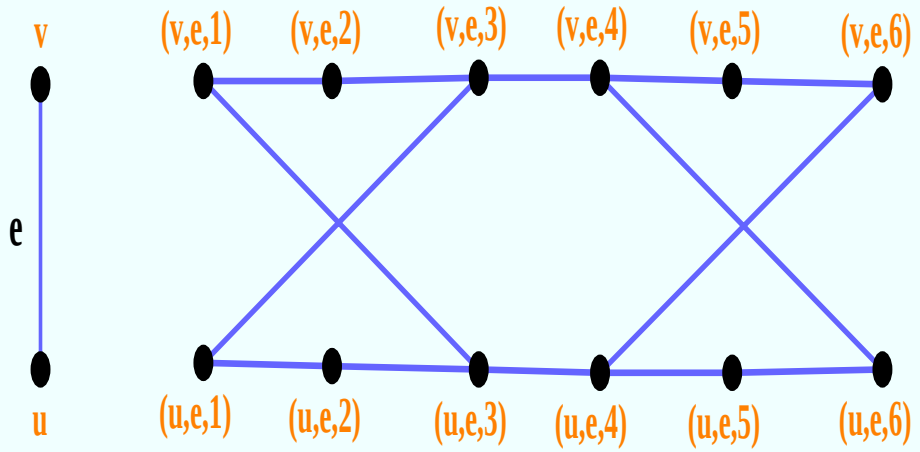
Rezultă că, de fapt, am demonstrat și că $DCH \propto BCH$.

Cum $BCH \propto CH$ este evidentă, rezultă că teorema este demonstrată.

Teoremă. 5. *(Karp 1972)* $SM \propto CH$.

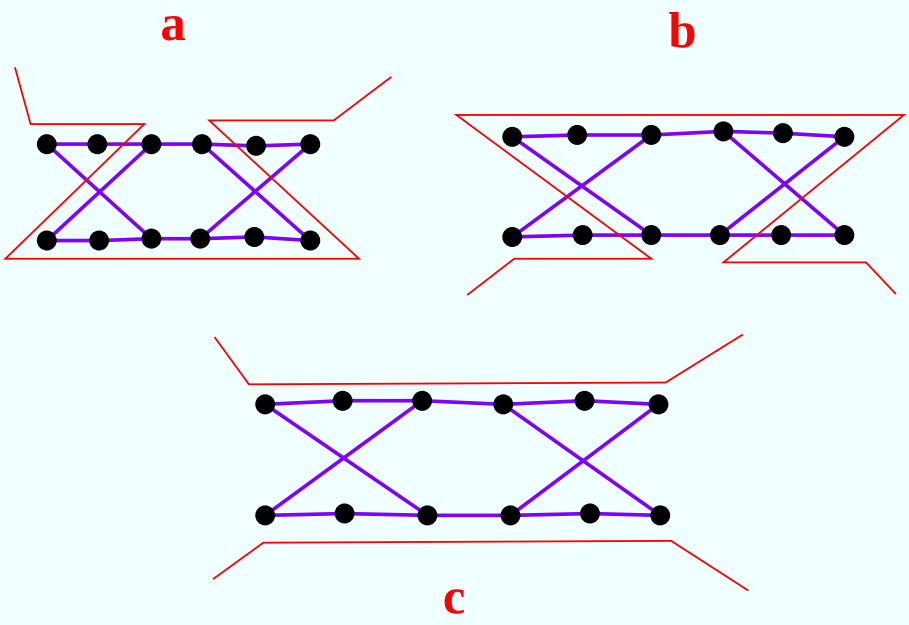
Demonstrație: Fie $G = (V, E)$ graf și $j \in \mathbf{Z}_+$ datele unei probleme SM. Construim în timp polinomial (în raport cu $n = |V|$) un graf H astfel încît: există S stabilă în G cu $|S| \geq j$ dacă și numai dacă H este hamiltonian.

Fie $k = |V| - j$. Putem presupune $k > 0$, altfel luăm drept H orice graf nehamiltonian.
i) Fie $A = \{a_1, a_2, \ldots, a_k\}$ o mulțime de k vîrfuri distincte.
ii) Pentru orice $e = uv \in E(G)$ considerăm graful $G'_e = (V'_e, E'_e)$ cu
$V'_e = \{(w, e, i); w \in \{u, v\}, i = 1, 6\}\}$ și
$E'_e = \{(w, e, i)(w, e, i + 1); w \in \{u, v\}, i = 1, 5\} \cup$
$\{(u, e, 1)(v, e, 3), (u, e, 6)(v, e, 4), (v, e, 1)(u, e, 3),$
$(v, e, 6)(u, e, 4)\}$.



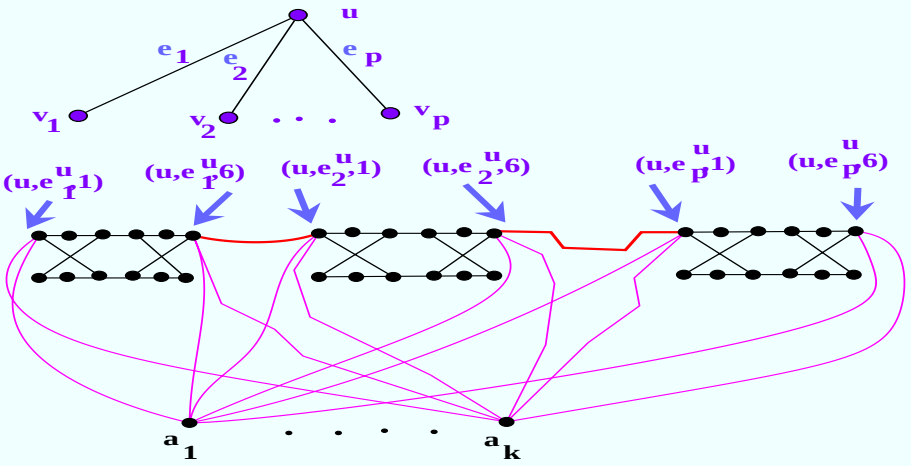
Graful G'_e a fost ales astfel încît, dacă este subgraf al unui graf hamiltonian H și niciunul din vîrfurile (w, e, i) cu $w \in \{u, v\}$ și $i = \overline{2, 5}$ nu are alt vecin în H decît cele din G'_e,

atunci singurele posibilități de traversare de către un circuit hamiltonian al lui H a vîrfurilor din G'_e sînt (a) (b) și (c) indicate în figura următoare (orice altă parcurgere lasă un vîrf netraversat):



Deci, dacă circuitul hamiltonian "intră" în G'_e printr-un vîrf de tip u, ($(u, e, 1)$ sau $(u, e, 6)$) atunci, va "ieși" tot printr-un vîrf corespunzător lui u.

(iii) Pentru fiecare vîrf $u \in V$, se consideră, (într-o ordine oarecare) toate muchiile lui G incidente cu u: $e_1^u = uv_1, e_2^u = uv_2, \ldots, e_p^u = uv_p$, unde, $p = d_G(u)$.
Fie $E''_u = \{(u, e_i^u, 6)(u, e_{i+1}^u, 1); i = 1, p-1\}$ și $E'''_u = \{a_i(u, e_1^u, 1), a_i(u, e_p^u, 6); i = 1, k\}$.

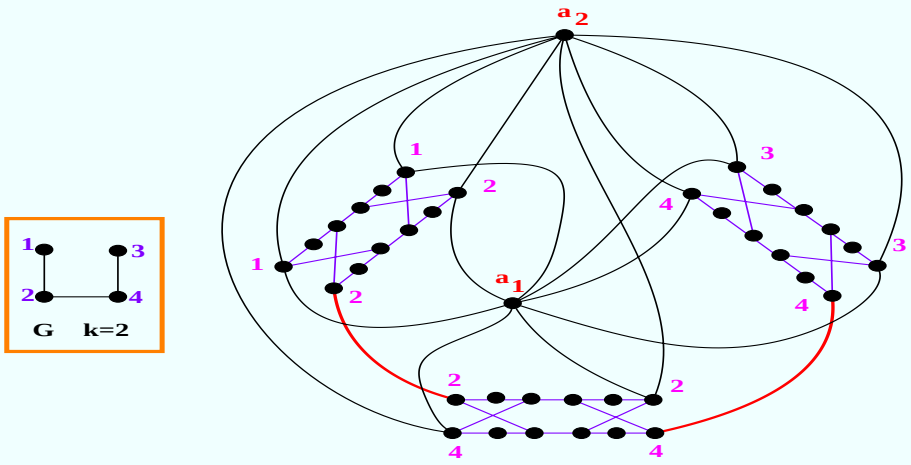


Graful H va avea

$V(H) = A \cup \cup_{e \in E} V'_e$ și
$E(H) = \cup_{e \in E} E'_e \cup \cup_{u \in V} (E''_u \cup E'''_u)$.

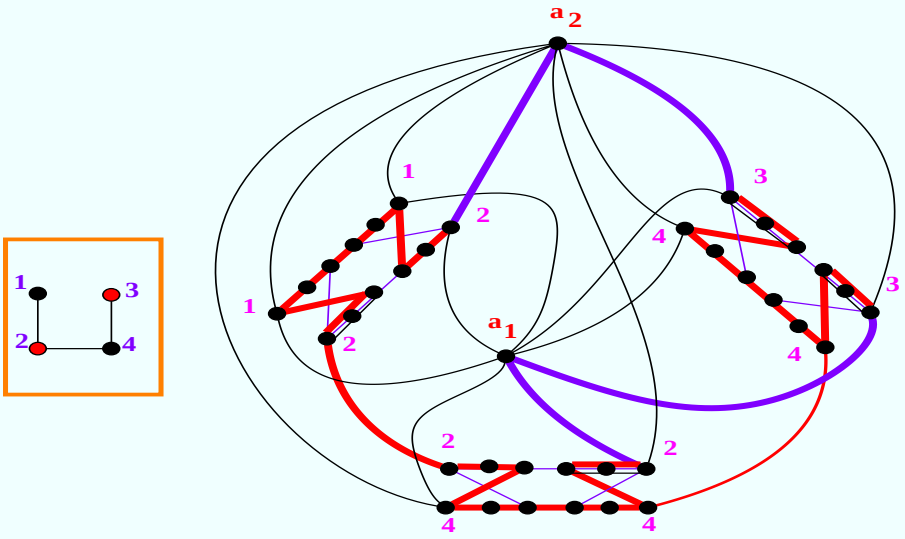
Avem $|V(H)| = k + 12 \cdot |E|$; $|E(H)| = 14 \cdot |E| + 2k \cdot |V_0| + 2 \cdot |E| - |V_0|$, unde $V_0 \subseteq V$ este mulțimea vîrfurilor neizolate din G; deci H se poate construi în timp polinomial în raport cu $|V|$.

Exemplu:



1. Dacă H este hamiltonian, atunci există C un circuit hamiltonian în H. Cum A este o mulțime stabilă în H, A va descompune circuitul C în exact k drumuri disjuncte (cu excepția extremităților): $D_{a_{i_1}a_{i_2}}, D_{a_{i_2}a_{1_3}}, \ldots, D_{a_{i_k}a_{i_1}}$.
Fie $D_{a_{i_j}a_{i_{j+1}}}$ un astfel de drum ($j+1 = 1 + (j(\text{mod } k))$).
Din construcția lui H, rezultă că primul vîrf care urmează lui a_{i_j} pe acest drum va fi $(v_{i_j}, e_1^{v_{i_j}}, 1)$ sau $(v_{i_j}, e_p^{v_{i_j}}, 6)$ unde $p = d_G(v_{i_j}) \quad v_{i_j} \in V$.

În continuare, $D_{a_{i_j}a_{i_{j+1}}}$ va intra în componenta G_{e_1} sau G_{e_p} din care va ieși tot printr-un vîrf corespunzător lui v_{i_j}. Dacă vîrful următor nu este $a_{i_{j+1}}$, se intră într-o componentă corespunzătoare următoarei muchii incidente cu v_{i_j} și va ieși din aceasta, tot printr-un vîrf corespunzător lui v_{i_j}.



Rezultă că drumului $D_{a_{i_j}a_{i_{j+1}}}$ i se poate asocia în mod unic vîrful $v_{i_j} \in V$ și că prima și ultima muchie a acestui drum sînt $a_{i_j}(v_{i_j}, e_t^{v_{i_j}}, x)$, $a_{i_{j+1}}(v_{i_j}, e_{t'}^{v_{i_j}}, x')$ cu $t = 1$ și $t' = d_G(u)$, $x = 1$, $x' = 6$ sau $t = d_G(u)$, $t' = 1$, $x = 6$, $x' = 1$. Aceasta implică faptul că v_{i_j} sînt distincte.

Fie $V^* = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$. Cum C este hamiltonian în H, rezultă că $\forall e \in E$ există un drum $D_{a_{i_j}a_{i_{j+1}}}$ care trece prin G'_e deci există $v \in V^*$ incident cu e. Rezultă că $S = V - V^*$ este stabilă în G și $|S| = j$. Deci am obținut că dacă H este hamiltonian, în G există o mulțime stabilă de cardinal j, prin urmare răspunsul la SM este da.

2. Presupunem că răspunsul la SM este da, deci în G există S_0 stabilă cu $|S_0| \geq j$. Există atunci $S \subseteq S_0$ stabilă cu $|S| = j$.

Fie $V^* = V - S = \{v_1, v_2, \ldots, v_k\}$. Considerām în H pentru fiecare muchie $e = uv \in E$:

- cele două drumuri din G'_e situaţia (a), dacă $u, v \in V^*$,

- drumul din G'_e situaţia (b) dacă $u \in V^*, v \notin V^*$

- drumul din G'_e situaţia (c) dacă $u \notin V^*, v \in V^*$.

Dacă la reuniunea acestor drumuri adăugām muchiile
$a_i(v_i, e_1^{v_i}, 1)$,
$(v_i, e_1^{v_i}, 6)(v_i, e_2^{v_i}, 1), \ldots, (v_i, e_{p-1}^{v_i}, 6)(v_i, e_p^{v_i}, 1)$,
$(v_i, e_p^{v_i}, 6)a_{i+1}$, (cu $p = d_G(v_i)$) pentru $i = 1, k$, se obţine un circuit hamiltonian în H.

5. Problema comisului voiajor.

Dat $G = (V, E)$ un graf și o funcție de ponderе $d : E \to \mathbf{R}_+$, să se determine un circuit hamiltonian H_0 în G astfel încît suma ponderilor muchiilor lui H_0 să fie minimă.

Dacă graful G reprezintă rețeaua rutieră între o mulțime V de localități, iar funcția d avînd interpretarea $d(uv) =$ distanța pe ruta directă dintre localitățile u și v, și se fixează un centru $v_0 \in V$, atunci circuitul hamiltonian H_0 reprezintă cel mai economic mod de vizitare a localităților din V de către un **comis voiajor** ce pleacă din centrul v_0 și se întoarce în același loc, după ce a vizitat fiecare localitate exact o dată.

Problema este interesantă, nu pentru rezolvarea acestei aplicații, mai mult sau mai puțin importantă, ci pentru că ea apare în numeroase probleme de optimizare discretă, motivate, de exemplu, de construirea circuitelor integrate pe scară mare.

În cele ce urmează, vom considera o formă echivalentă a ei

CV Dat $n \in \mathbf{Z}_+ \quad (n \geq 3)$ și $d : E(K_n) \to \mathbf{R}_+$, să se determine H_0 circuit hamiltonian în graful complet K_n cu $d(H_0)$ minim printre toate circuitele hamiltoniene ale lui K_n.

Observații 1^0. $d(H_0) = \sum_{e \in E(H_0)} d(e)$.

2^0. Dacă graful pentru care se cere rezolvarea nu este complet, se introduc muchiile lipsă, atribuindu-le drept ponderi, $M \in \mathbf{R}_+$ cu $M > |V| \cdot \max_{e \in E} d(e)$.
Notăm aici, că în enunţul problemei ne-am limitat numai la "cazul simetric", o problemă similară se poate considera și pentru G digraf oarecare.

3^0. În studiul complexitaţii acestei probleme vom considera că $d(e) \in \mathbf{Z}_+$.

Problema de decizie asociata va fi

DCV

Instanță: $n \in \mathbf{Z}_+ \quad (n \geq 3)$, $d : E(K_n) \to \mathbf{Z}_+$ și $B \in \mathbf{Z}_+$

Intrebare: Există H_0 circuit hamiltonian în K_n
astfel încît $d(H_0) \leq B$?

Teoremă. 6. $CH \propto DCV$.

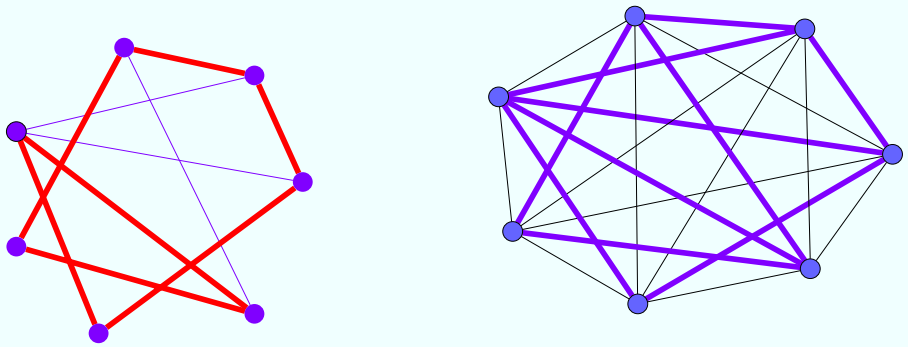
Demonstrație: Fie $G = (V, E), \quad (|V| = n)$ reprezentînd datele unei probleme CH. Construim în timp polinomial o problemă DCV cu proprietatea că în K_n există un circuit hamiltonian de pondere totală care nu depășește B dacă și numai dacă G este hamiltonian.

Fie

$$d(vw) = \begin{cases} 1 & \text{dacă } vw \in E(G) \\ 2 & \text{dacă } vw \in E(\overline{G}). \end{cases}$$

și considerăm $B = n$.

Atunci în K_n există un circuit hamiltonian de pondere $\leq n$ dacă și numai dacă G are un circuit hamiltonian.



Rezultă, de aici, dificultatea rezolvării problemei CV.

O soluţie pentru abordarea problemei CV, ar fi aceea de a considera algoritmi A, care pentru datele unei probleme CV vor oferi în timp polinomial (în raport cu n) un circuit hamiltonian H_A, care va aproxima soluţia optimă H_0.

Măsuri ale eficienţei unei astfel de "euristici" A pot fi considerate numerele:

$$R_A(n) = \sup_{\substack{d:E(K_n)\to\mathbf{R}_+ \\ d(H_0)\neq 0}} \frac{d(H_A)}{d(H_0)}$$

$$R_A = \sup_{n\geq 3} R_A(n).$$

Evident, s-ar dori ca aceste numere să fie finite. În cazul general, condiţia ca R_A să fie finit este la fel de dificilă cu aceea a rezolvării eficiente "exact" a problemei CV. În adevăr, are loc

Teoremă. 8. *Dacă există un algoritm aproximativ A cu timp de lucru polinomial pentru CV, astfel încît $R_A < \infty$, atunci CH se poate rezolva în timp polinomial.*

Demonstraţie: Fie A un algoritm cu timp de lucru polinomial şi cu $R_A < \infty$. Există deci $k \in \mathbf{Z}_+$ astfel încît $R_A \leq k$.

Fie $G = (V, E)$ un graf arbitrar, intrare pentru CH. Dacă $n = |V|$ atunci definim $d : E(K_n) \to \mathbf{Z}_+$ prin

$$d(uv) = \begin{cases} 1 & \text{dacă } uv \in E \\ kn & \text{dacă } uv \notin E \end{cases}$$

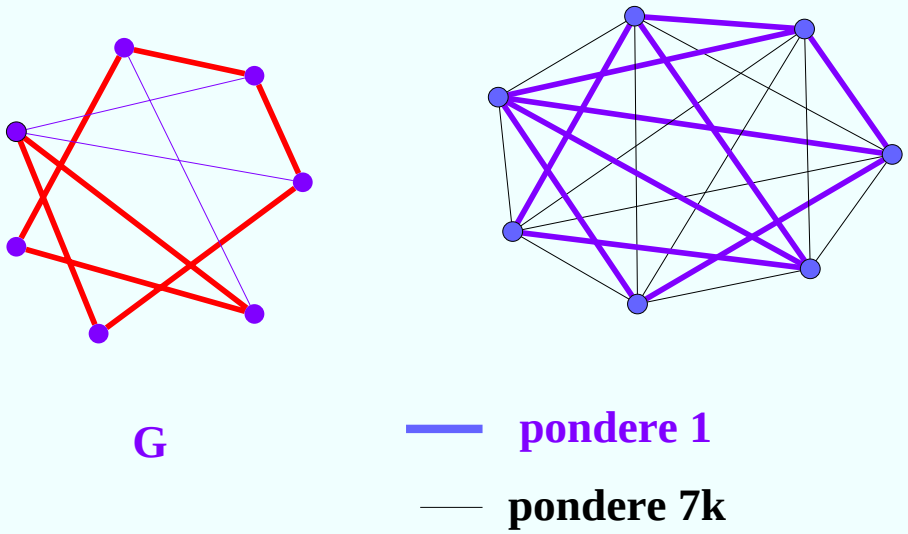
Evident, G este hamiltonian dacă şi numai dacă soluţia optimă H_0 a lui CV astfel construit satisface $d(H_0) = n$.

Aplicăm A pentru rezolvarea problemei CV. Se obţine în timp polinomial o soluţie H_A.

Dacă $d(H_A) \leq kn$, atunci $d(H_A) = n$ şi $H_A = H_0$.

Dacă $d(H_A) > kn$, atunci $d(H_0) > n$. În adevăr, presupunînd că $d(H_0) = n$, avem $\frac{d(H_A)}{d(H_0)} \leq k$, deci $d(H_A) \leq kd(H_0) = kn$, contradicţie.

Rezultă că G este hamiltonian dacă și numai dacă $d(H_A) \leq kn$, și cum timpul de lucru al lui A este polinomial, rezultă că CH se poate rezolva în timp polinomial.



Observaţie. Enunţul teoremei se poate formula echivalent și astfel:

Dacă $\mathbf{P} \neq \mathbf{NP}$, *atunci nu există un algoritm aproximativ* A *polinomial și cu* $R_A < \infty$.

VII. Abordări ale unor probleme NP-dificile pe grafuri.

1. Euristici

Se proiectează algoritmi eficienţi care, deşi nu rezolvă problema, oferă soluţii aproximative, care se pot analiza şi uneori se pot folosi în metaeuristici (de exemplu, cele care imită procese din natură). Pentru unele euristici se precizează şi instanţe ale problemelor pentru care ele funcţionează corect.

Partizanii unor astfel de abordări apelează la aşa numita **no free lunch theorem** (Wolpert and Macready, 1994).

all non-repeating search algorithms have the same mean performance when averaged uniformly over all possible objective functions f : $X \to Y$

1.1 Colorarea vârfurilor unui graf

Algoritmul greedy de colorare

Un mod simplu și natural de a colora un graf, cu nu foarte multe culori, este dat de următorul algoritm:
pornind de la o ordine fixată a vârfurilor lui G, $v_1, v_2, \ldots, v_n$, parcurgem această listă, colorând fiecare vârf v_i cu prima culoare disponibilă, adică cu cea mai mică culoare (număr întreg pozitiv) nefolosită de nici unul dintre vecinii lui v_i , deja întâlniți până în acel moment.

Această metodă poartă denumirea de *algoritmul greedy de colorare*, datoritã alegerii, de fiecare dată,**a celei mai mici culori** care s-ar putea folosi.

Fie $G = (V, E)$, cu $V = \{1, 2, \ldots, n\}$ și π o permutare a lui V (ce corespunde unei ordonări a mulțimii vârfurilor).

Algoritmul construiește colorarea c ce utilizează $\chi(G, \pi)$ culori,
$c : \{1, 2, \ldots, n\} \longrightarrow \{1, 2, \ldots, \chi(G, \pi)\}$.

Algoritmul greedy-color

- $c(\pi_1) \leftarrow 1; \chi(G,\pi) \leftarrow 1; S_1 \leftarrow \{\pi_1\};$
- **for** $i \leftarrow 2$ **to** n **do**
- { $j \leftarrow 0;$
- **repeat**
- $j \leftarrow j+1;$
- determină primul vârf (conform ordonării
- π), v din S_j a. î. $\pi_i v \in E(G);$
- **if** v există **then**
- $first(\pi_i, j) \leftarrow v$
- **else**
- { $first(\pi_i, j) \leftarrow 0;$
- $c(\pi_i) \leftarrow j;$
- $S_j \leftarrow S_j \cup \{\pi_i\};$
- }
- **until** $first(\pi_i, j) = 0$ **or** $j = \chi(G,\pi);$
- **if** $first(\pi_i, j) \neq 0$ **then**
- { $c(\pi_i) \leftarrow j+1;$
- $S_{j+1} \leftarrow \{\pi_i\};$
- $\chi(G,\pi) \leftarrow j+1;$
- }
- }

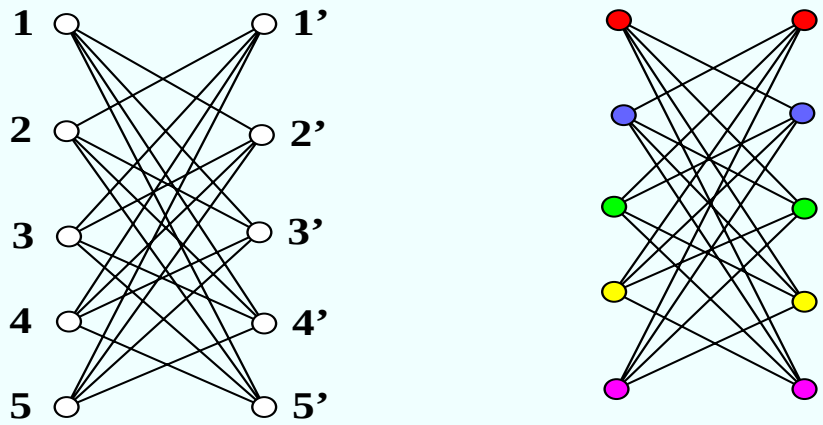
Să observăm că folosind algoritmul greedy nu vom obține niciodată mai mult de $1 + \Delta(G)$ culori.

Rezultă că $\chi(G) \leq 1 + \Delta(G)$.

Însă $\Delta(G)$ se poate îndepărta oricât de mult de $\chi(G)$ și există grafuri și permutări pentru care algoritmul poate să greșească la fel de mult.

Un astfel de exemplu este următorul, unde graful G este obținut din graful bipartit complet $K_{n,n}$, având mulțimea de vârfuri $\{1, 2, \ldots, n\} \cup \{1', 2', \ldots, n'\}$, din care se elimină muchiile $11', 22', \ldots, nn'$.

Fixând ordinea $1, 1', 2, 2', \ldots, n, n'$ algoritmul va obține colorarea c cu $c(1) = c(1') = 1, c(2) = c(2') = 2, \ldots, c(n) = c(n') = n$ având deci n culori; însă, cum G este bipartit, $\chi(G) = 2$.



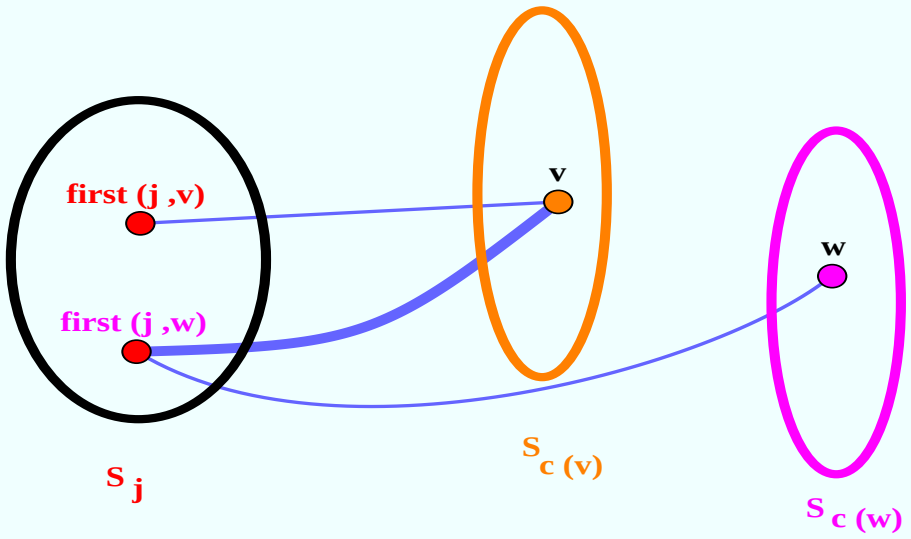
pi: 1, 1', 2, 2', 3, 3', 4, 4', 5, 5'

Este ușor de văzut că, pentru orice graf G, există totuși permutări pentru care algoritmul oferă o colorare optimală.

Într-adevăr dacă $S_1, S_2, \ldots, S_{\chi(G)}$ sunt clasele de colorare corespunzătoare unei colorări optimale, atunci pentru orice permutare care păstrează ordinea claselor (adică dacă $i < j$, $v \in S_i$ și $w \in S_j$ atunci v se află înaintea lui w în permutare), executând algoritmul greedy, se vor obține $\chi(G)$ culori.

Teoremă. 1. *Dacă pentru orice* $vw \in E$ *și orice* $j < \min\{c(v), c(w)\}$ *astfel încât* $first(v, j) < first(w, j)$ *(în ordonarea* π*) avem că* $vfirst(w, j) \in E$ *atunci* $\chi(G, \pi) = \chi(G)$.

Condiția din enunț poate fi verificată în timp liniar în raport cu numărul de muchii ale grafului și poate reprezenta un ultim pas al algoritmului.



În demonstația teoremei se arată că, atunci când condiția este îndeplinită, $\chi(G, \pi) = \omega(G)$ și prin urmare, în acest caz, se obține și numărul de clică al grafului .

Pe de altă parte, există grafuri pentru care această condiție nu va fi îndeplinită de nici o permutare.

Algoritmul Dsatur { *Degree of Saturation*}

Acest algoritm (Brélaz, 1979) este o metodă secvenţială dinamică de colorare.

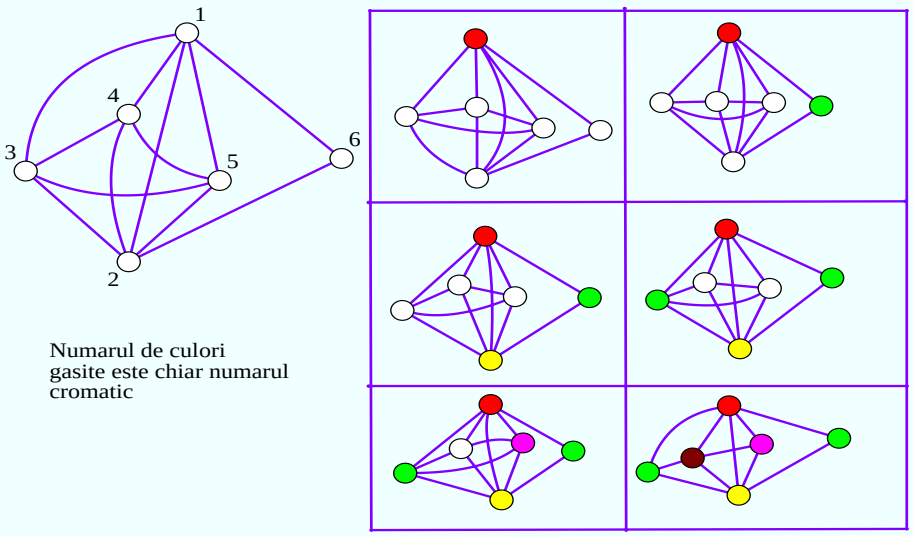
Ideea este de colora vârfurile pe rând, alegând de fiecare dată vârful cu un număr maxim de constrângeri privitoare la culorile disponibile acestuia.

Această abordare este într-un fel opusă primeia (cea geedy) deoarece se aleg vârfuri care formează "clici" mari în raport cu vârfurile deja alese (spre deosebire de mulţimi stabile mari în cazul greedy).

Dacă G este un graf şi c o colorare parţială a vârfurilor lui G, definim *gradul de saturaţie* al unui vârf v, notat $d_{sat}(v)$, ca fiind numărul de culori diferite din vecinătatea acestuia.

Algoritmul DSatur

- ordonează vârfurile în ordinea descrescătoare a gradelor lor;
- atribuie unui vârf de grad maxim culoarea 1;
- **while** există vârfuri necolorate **do**
- { alege un vf. necol. cu gr. de satur. maxim; dacă acesta
- nu-i unic, alege un vf. de grad maxim în subgr.necolorat;
- colorează vârful ales cu cea mai mică culoare posibilă;
- }



Teoremă. 2. *Algoritmul DSatur garantează găsirea numărului cromatic pentru grafurile bipartite.*

1.2 Problema comisului voiajor

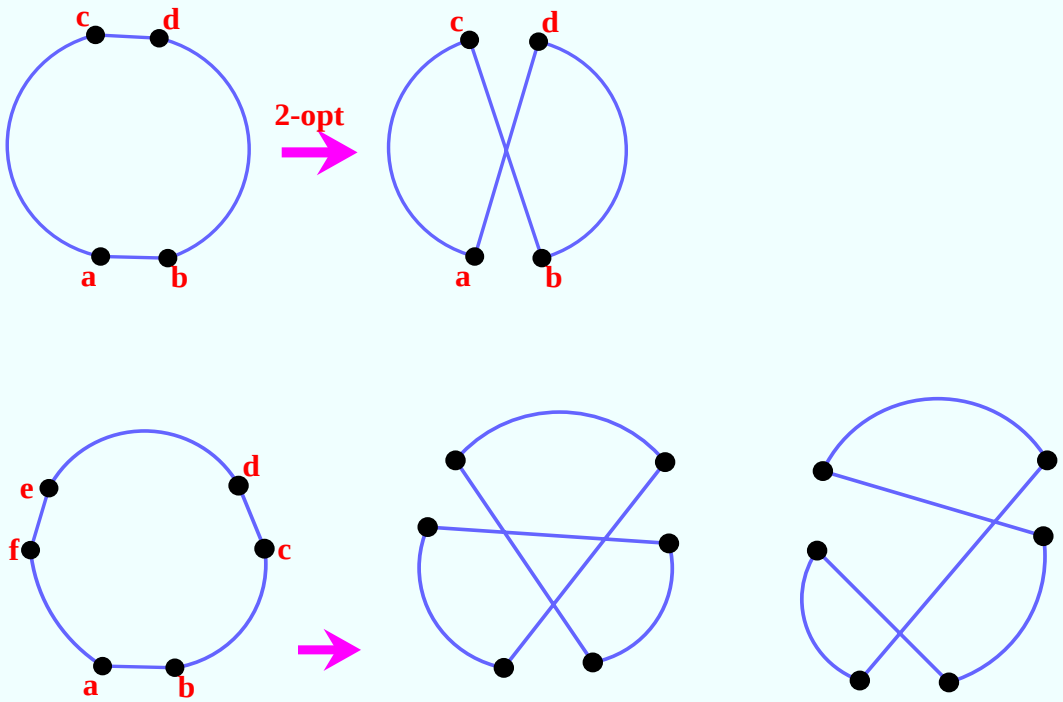
Cele mai performante euristici se bazează pe principiul **optimizării locale:**

- se porneşte cu un circuit hamiltonian ales aleator sau obţinut cu o **euristică de tip construcţie** (tour-construction heuristic), de exemplu greedy;

- se încearcă repetat imbunătăţiri locale, trecând de la soluţia curentă la o soluţie vecină din spaţiul soluţiilor (**neighborhood search**).

Cele mai cunoscute modificări locale sunt aşa numitele **2-opt, 3-opt** sau combinaţii ale acestora care au condus la faimosul algoritm **Lin-Kernighan**.

Figura următoare sugerează cele două operaţii 2-opt şi 3-opt.



doua operatii 3-opt posibile

Pentru reprezentarea circuitului hamiltonian se utilizează structura de date *Tour* care trebuie să suporte (eficient) următoarele 4 operaţii:

$Next(a)$: întoarece vârful următor lui a pe circuit;

$Prev(a)$: întoarece vârful dinaintea lui a pe circuit;

$Between(a, b, c)$: întoarce $true$ dacă la traversarea circuitului din a se întâlneşte mai întâi b şi apoi c;

$Flip(a, b, c, d)$: Parametrii verifică $a = Next(b)$ şi $d = Next(c)$. Se actualizează circuitul prin înlocuirea muchiilor ab şi cd cu bc şi ad. Se actualizează orientarea circuitului (pointerii $Next$ şi $Prev$) în mod corespunzător.

Operaţia 3-opt se realizează cu ajutorul a două sau trei $Flip$. Pentru evitarea formării de circuite disjuncte este nevoie de funcţia $Between$. Operaţia mai complicată λ-opt, utilizată de algoritmul Lin-Kernighan se exprimă ca un 3-opt şi o secvenţă de 2-opt.
Principala problemă în implementarea operaţiei $Flip$ este de a determina eficient şirul de schimbări ale pointerilor $Next$ şi $Prev$ (inversarea unuia din cele două drumuri; care din ele ?).

Un algoritm λ-opt se bazează pe conceptul de λ-*optimalitate*:
Un tur este λ-*optimal* (sau simplu λ-*opt*) dacă nu este posibil să se obţină un tur mai scurt (de cost mai mic) prin înlocuirea unei mulţimi de λ muchii ale sale cu altă mulţime de λ muchii.

Observaţie: Dacă un tur este λ-optimal atunci el este λ'-optimal pentru orice λ', $2 \leq \lambda' \leq \lambda$. De asemenea, un tur cu n vârfuri este optimal dacă şi numai dacă este n-optimal.

Numărul operaţiilor necesăre testării tuturor λ-schimbărilor posibile creşte rapid pentru n mare. Într-o implementare naivă, testarea unei λ-schimbări necesită $O(n^\lambda)$ operaţii. Ca urmare, cele mai folosite valori sunt $\lambda \in \{2, 3\}$ (dar se cunosc şi abordări cu $\lambda = 4, 5$).

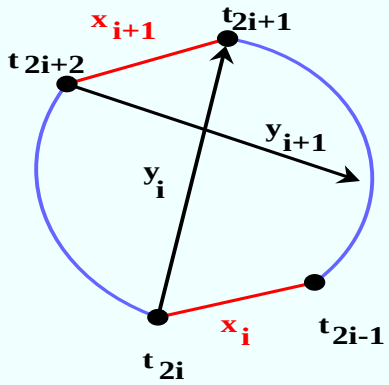
E dificil să se estimeze ce valoare a lui λ trebuie aleasă pentru ca să se obţină cel mai bun compromis între timpul de execuţie şi calitatea soluţiei modificate.

Lin şi Kernighan au abordat această problemă prin considerarea unei metode numită *variable* λ-*opt*: algoritmul va schimba valoarea lui λ în timpul execuţiei.

La fiecare iteraţie se încearcă valori crescătoare pentru λ pentru a obţine un tur mai scurt.

Fie T turul curent. La fiecare iterație a algoritmului se caută mulțimile de muchii $X = \{x_1, \ldots, x_r\}$ și $Y = \{y_1, \ldots, y_r\}$ astfel ca înlocind X cu Y în T, să se obțină un tur mai bun. Această interschimbare o numim *r-opt move.*
Cele 2 mulțimi se construiesc element cu element. Inițial sunt vide și în pasul i se adaugă x_i la X și y_i la Y.

Criteriile de selectare ale acestor muchii, urmăresc eficiența algoritmului și în același timp îmbunătățirea turului curent:
-**Criteriul de schimb secvențial**: x_i și y_i sunt adiacente și la fel x_{i+1} și y_{i+1}. Dacă t_1 este una din extremitățile lui x_1, atunci $x_i = t_{2i-1}t_{2i}$, $y_i = t_{2i}t_{2i+1}$ și $x_{i+1} = t_{2i+1}t_{2i+2}$.



Deci $x_1, y_1, x_2, y_2, x_3, \ldots, x_r, y_r$ formează muchiile unui drum. Pentru **schimbul secvențial** se impune și ca acest drum să fie închis : $y_r = t_{2r}t_1$.

-**Criteriul de admisibilitate**: se va alege $x_i = t_{2i-1}t_{2i}$ astfel ca dacă t_{2i} se va uni cu t_1, configurația care rezultă să fie tur. Se aplică pentru $i \geq 3$ și garantează că este posibil să închidem un tur. Este introdus în algoritm pentru reducerea timpului de execuție și simplificarea implementării.

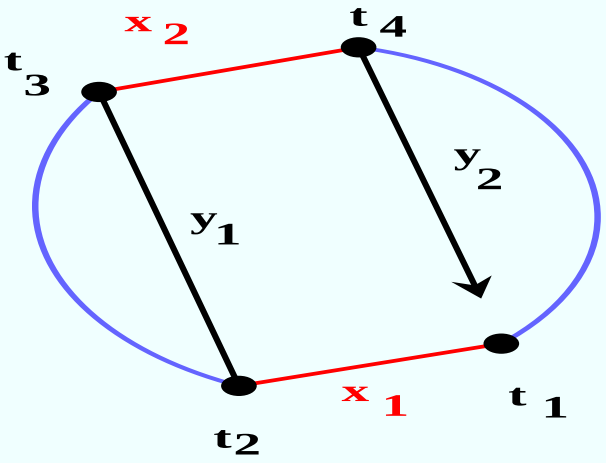
-**Criteriul câștigului**: se cere ca y_i să fie ales astfel ca să existe un **câstig**; G_i trebuie să fie pozitiv; dacă $g_i = d(x_i) - d(y_i)$ este câștigul interschimbării lui x_i cu y_i, atunci $G_i = g_1 + g_2 + \cdots + g_i$. Rațiunea alegerii acestui criteriu este că dacă suma unei secvențe de numere este pozitivă, atunci există o permutare circulară a acestei secvențe astfel ca orice sumă parțială este pozitivă.

-**Criteriul dijunctivității**: se impune ca mulțimile X și Y să fie disjuncte. Aceasta simplifică implementarea, reduce timpul de execuție și dă un criteriu efectiv de oprire.

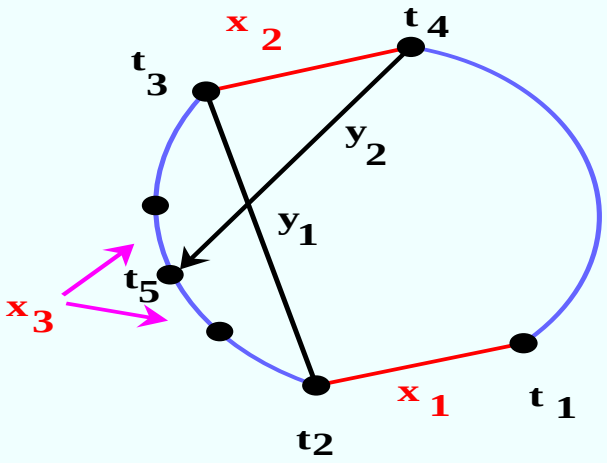
Linia algoritmului Lin & Kernighan

1. Generează (random) un tur inițial T;
2. $i \leftarrow 1$; Alege t_1;
3. Alege $y_1 = t_1t_2 \in T$;
4. Alege $y_1 = t_2t_3 \notin T$ a.î. $G_1 > 0$; **If** $\nexists$ **goto** 12;
5. $i++$;
6. Alege $x_i = t_{2i-1}t_{2i} \in T$ a.î.
   (a) dacă t_{2i} e unit cu t_1 se obține un tur T' și
   (b) $x_i \neq y_s$ pentru toți $s < i$
   If T' e mai bun ca T **then** fie $T = T'$ și **goto** 2;
7. Alege $y_i = t_{2i}t_{2i+1} \notin T$ a.î.
   (a) $G_i > 0$,
   (b) $y_i \neq x_s$ pentru toți $s < i$ și
   (c) x_{i+1} există.
   If y_i există then goto 5;
8. **If** mai $\exists$ alte alegeri pt. y_2 **then** $i \leftarrow 2$ și **goto** 7;
9. **If** mai $\exists$ alte alegeri pt. x_2 **then** $i \leftarrow 2$ și **goto** 6;
10 **If** mai $\exists$ alte alegeri pt. y_1 **then** $i \leftarrow 1$ și **goto** 4;
11 **If** mai $\exists$ alte alegeri pt. x_1 **then** $i \leftarrow 1$ și **goto** 3;
12 **If** mai $\exists$ alte alegeri pt. t_1 **then goto** 2;
13 **Stop** (sau **goto** 1).

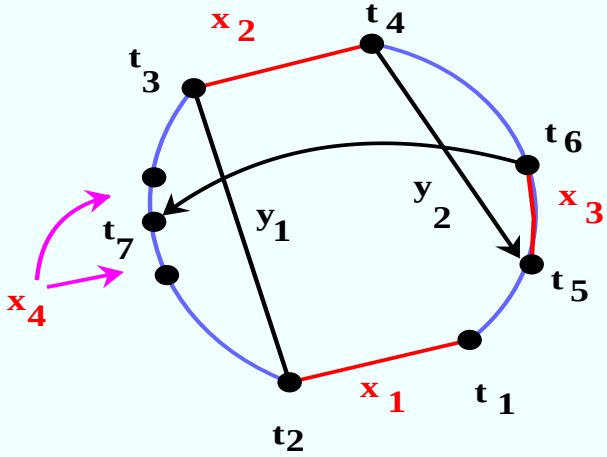
Î n descrierea anterioară "alege" înseamnă selectarea unei alternative neîncercate deja (pentru turul curent). De exemplu în pasul 3 avem două posibilităţi de a alege o muchie de pe tur incidentă cu t_1.
Î n pasul 6 există două alegeri pentru x_i. Pentru un y_{i-1} ($i \geq 2$) numai una dintre acestea va face posibilă închiderea turului (prin adăugarea lui y_i). Cealaltă alegere va conduce la apariţia a două subtururi disjuncte. Totuşi pentru $i = 2$ este permisă o astfel de alegere neadmisibilă:



Dacă y_2 va fi ales astfel ca t_5 să fie între t_2 şi t_3, turul va putea fi închis în următorul pas.



Atunci, t_6 poate fi ales în ambele părți ale lui t_5.
Dacă însă, y_2 va fi ales astfel ca t_5 să fie între t_4 și t_1, atunci există doar o singură alegere pentru t_6 (între t_4 și t_5) și t_7 trebuie să se afle între t_2 și t_3. Dar atunci t_8 trebuie să fie de cealaltă parte a lui t_7; algoritmul alege alternativa cu $d(t_7t_8)$ maximă.



Pașii 8-12 ai algoritmului cauzează backtracking-ul. El este permis numai dacă nu se obține o îmbunătățire și numai la nivelele 1 și 2.

Algoritmul se termină cu un tur după ce toate valorile posibile ale lui t_1 au fost examinate fără îmbunătățire.

Numeroase alte modificări ale metodei de bază au fost examinate. De exemplu, Lin și Kernighan limitează căutarea lui $y_i = t_{2i}t_{2i+1}$ la vecinii de pe tur ai lui t_{2i} la distanță cel mult 5.

Unele reguli au fost alese pentru salvarea timpului de execuție, **limitarea căutării**, iar altele pentru **direcționarea căutării**.

Se poate consulta articolul : **The Traveling Salesman Problem: A Case Study in Local Optimization** David S. Johnson Lyle A. McGeoch, *Local Search in Combinatorial Optimization*, E. H. L. Aarts and J. K. Lenstra (eds.), John Wiley and Sons, London, 1997, pp. 215-310.

O altă euristică populară pentru problemele în care funcţia de distanţă satisface inegalitatea triunghiulară, este dată de *Christofides*.

Spre deosebire de cazul general când am demonstrat că nu se poate spera la o euristică polinomială A cu R_A finită, dacă $\mathbf{P} \neq \mathbf{NP}$, în acest caz se poate demonstra următoarea

Teoremă. *(Christofides,1973) Fie CV cu* d *satisfăcînd*

$\forall v, w, u \in V(K_n)$distincte $d(vw) \leq d(vu)+d(uw)$.

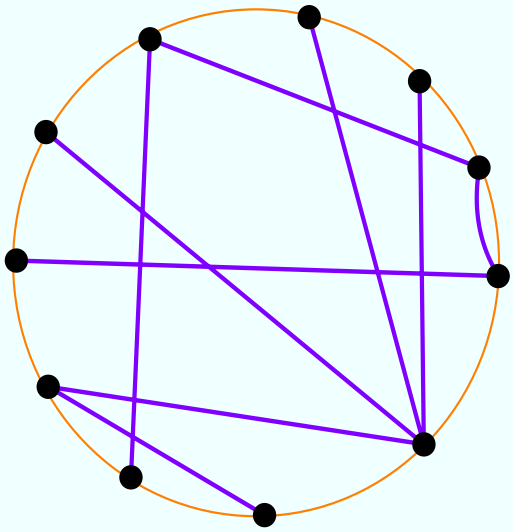
Există un algoritm aproximativ A *pentru CV care satisface* $R_A = \frac{3}{2}$ *şi are timp de lucru polinomial.*

Demonstraţie:
Considerăm următorul algoritm A:

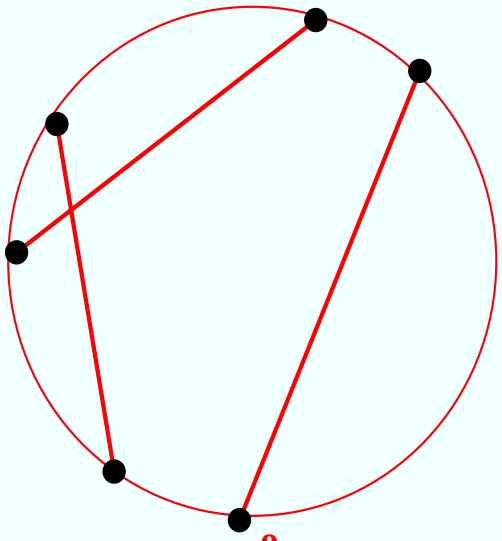
1^0. Se determină T^0 mulţimea muchiilor unui arbore parţial de cost minim în K^n (costul muchiei e fiind $d(e)$). Problema se rezolvă în timp polinomial cu algoritmul lui Prim.

2^0. Se determină M^0 un cuplaj perfect în subgraful indus de vîrfurile de grad impar ale arborelui T^0 şi de cost minim. Problema se rezolvă în timpul $O(n^3)$ utilizînd algoritmul lui Edmonds.

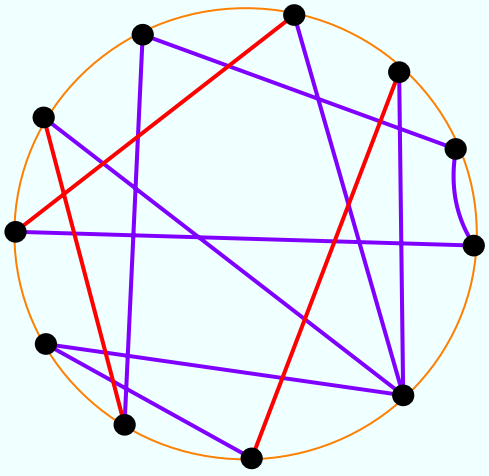
3^0. Se consideră multigraful obţinut din $< T^0 \cup M^0 >_{K_n}$, prin duplicarea muchiilor din $T^0 \cap M^0$. În acest multigraf există un parcurs Eulerian închis, ale cărui vîrfuri sînt $(v_{i_1}, v_{i_2}, \ldots, v_{i_1})$. Eliminînd orice apariţie multiplă a unui vîrf în acest şir cu excepţia primului şi ultimului vîrf, se obţin vîrfurile unui circuit hamiltonian H_A în K_n cu muchiile $H_A = (v_{j_1}v_{j_2}, v_{j_2}v_{j_3}, \ldots, v_{j_n}v_{j_1})$ (Acest pas necesită $O(n^2)$ operaţii).



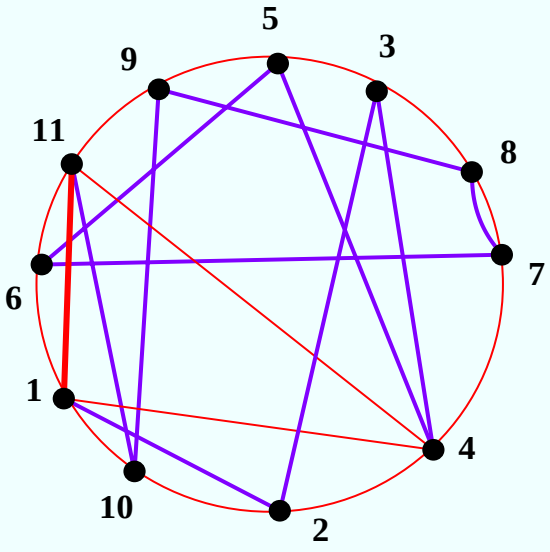
Arborele T^0



Cuplajul M^0



Graf Eulerian



Turul H_A

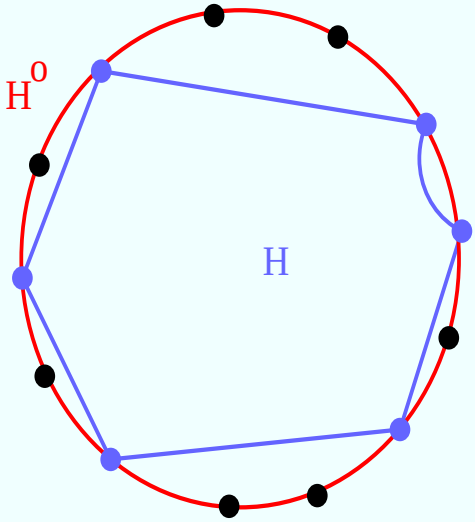
H_A este soluția aproximativă a problemei CV. Fie $m = \lfloor \frac{n}{2} \rfloor$ și H_0 soluția optimă. Vom arăta (cf. *Cornuejols și Nemhauser*) că

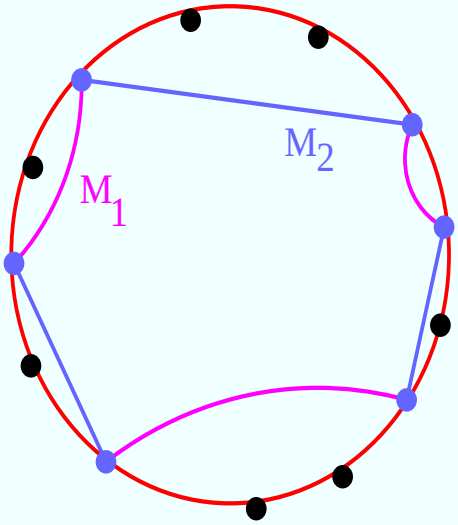
$$\forall n \geq 3 \quad d(H_A) \leq \frac{3m-1}{2m} d(H_0)$$

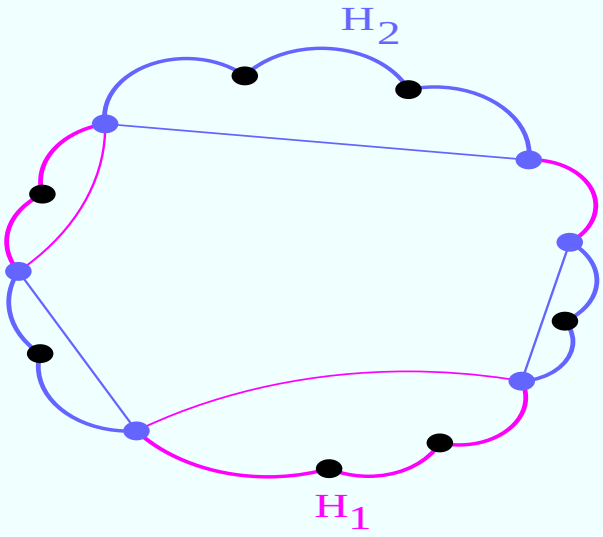
Presupunem că $H_0 = \{v_1v_2, v_2v_3, \ldots, v_nv_1\}$ (eventual, schimbînd numerotarea vîrfurilor). Fie $A = \{v_{i_1}, v_{i_2}, \ldots, v_{i_{2k}}\}$ vîrfurile de grad impar ale lui $< T^0 >_{Kn}$, $i_1 < i_2 < \ldots < i_{2k}$.

Dacă $H = \{v_{i_1}v_{i_2}, v_{i_2}v_{i_3}, \ldots, v_{i_{2k-1}}v_{i_{2k}}, v_{i_{2k}}v_{i_1}\}$, avem din inegalitatea triunghiulară, $d(H) \leq d(H_0)$, prin înlocuirea ponderii fiecărei "corzi" $d(v_{i_j}v_{i_{j+1}})$ cu suma ponderilor muchiilor de pe circuitul H_0 subîntinse de $v_{i_j}v_{i_{j+1}}$.

Pe de altă parte, H este circuit de lungime pară, deci este reuniunea a două cuplaje perfecte în $[A]_{K_n}$, $M^1 \cup M^2$.
Presupunem că $d(M^1) \leq d(M^2)$.


H0
H


M2
M1


H2
H1

Din alegerea lui M^0, avem $d(M^0) \leq d(M^1) \leq \frac{1}{2}(d(M^1) + d(M^2)) = \frac{1}{2}d(H) \leq \frac{1}{2}d(H_0)$.
Fie $\alpha \in \mathbf{R}_+$ astfel încît $d(M^0) = \alpha d(H_0)$.
Evident, $0 < \alpha \leq \frac{1}{2}$.
Partiţionăm H_0 în $H^1 \cup H^2$, punînd în H^i toate muchiile lui H_0 subîntinse de o coardă din M^i ($v_{i_j}v_{i_j+1} \in M^i \Rightarrow v_{i_j}v_{i_j+1}, \ldots, v_{i_{j+1}-1}v_{i_{j+1}} \in H^i$).
Din inegalitatea triunghiulară obţinem:
$d(H^i) \geq d(M^i) \quad i = 1, 2$.

Măcar unul din H^1 sau H^2 are cel mult $m = \lfloor \frac{n}{2} \rfloor$ muchii.
Presupunem că H^1. Deoarece $d(H^1) \geq d(M^1) \geq d(M^0) = \alpha d(H_0)$, rezultă că există $e \in H^1 : d(e) \geq \frac{\alpha}{m}d(H_0)$.
Fie T arborele parţial obţinut din H_0 prin înlătu-rarea unei muchii de pondere maximă. Avem
$d(T) = d(H_0) - \max_{e \in H_0} d(e) \leq d(H_0) - \frac{\alpha}{m}d(H_0)$.
Cum T^0 este arbore parţial de cost minim în K_n, rezultă $d(T^0) \leq d(H_0)(1 - \frac{\alpha}{m})$.
Folosind inegalitatea triunghiulară,
$d(H_A) \leq d(T^0) + d(M^0) \leq d(H_0)(1 - \frac{\alpha}{m}) + \alpha d(H_0) = (1 + \frac{\alpha(m-1)}{m})d(H_0)$.
Cum $\alpha \leq \frac{1}{2}$ se obţine $d(H_A) \leq \frac{3m-1}{2m}d(H_0)$ pentru $n \geq 3$.

2. Metode care imitã natura

In ultimii 20 de ani s-au dezvoltat numeroase metaeuristici inspirate din comportamentul unor sisteme biologice sau fizice ale lumii reale.

Metaeuristica *simulated annealing*.
Una din metaeuristicile populare utilizate pentru rezolvarea problemelor de optimizare NP-dificile este *cālirea simulatā*, metodā inspiratā din termodinamicā , inventatā independent de *Kirkpatrick, Gelatt şi Vechi* în 1983 şi de *Černy* în 1985.

Ca orice metodā Monte Carlo, repetarea algoritmului cu diferiţi parametri de start oferā şansa imbunātāţirii soluţiilor gāsite, care nu sunt în general, şi soluţii optime.

În termodinamică , se poate interpreta călirea (intilnită tradiţional în prelucrarea metalelor în fierăriile cu forjă , baros şi nicovală) ca un proces stohastic ce determină o aranjare a atomilor care minimizează energia totală a unui corp.

La temperaturi înalte, atomii se mişcă liber şi se mută cu rapiditate în poziţii care cresc energia totală. Pe măsură ce temperatura este scăzută, atomii se apropie gradual de o dispunere laticeală regulată şi numai ocazional îşi măresc energia. Aceste creşteri ocazionale de energie joacă un rol crucial în călire : ele permit ieşirea din minimele locale printr-o creştere temporară de energie.

La temperaturi înalte, astfel de salturi apar cu mare probabilitate, iar la temperaturi joase ele apar rar. Temperatura este scazută lent pentru a menţine echilibrul termal. Când atomii sunt în echilibru la temperatura T, probabilitatea ca energia lor totală să fie E este proporţională cu $e^{-\frac{E}{kT}}$, unde k este constanta lui *Bolzmann.*

În consecință, probabilitatea ca energia să fie $E + dE$ poate fi exprimată,

$$Pr(E + dE) = Pr(E) \cdot e^{-\frac{dE}{kT}},$$

adică probabilitatea de creștere a energiei scade odată cu temperatura.

Călirea simulată este o metodă computațională care imită modul natural de determinare a unei configurații care minimizează energia unui sistem. Atunci cnd se dorește minimizarea unei funcții $f : D \to \mathbf{R}$, vom interpreta domeniul de definiție al funcției, D, ca fiind mulțimea configurațiilor posibile ale sistemului, iar funcția f ca fiind energia acestuia.

O variabilă fictivă T, asociată procesului de căutare, va juca rolul temperaturii iar constanta lui Bolzmann va fi considerată 1.

Algoritm de călire simulată

1. Se consideră un **plan de călire:**
   - temperatura inițială T_{start}
   - configurația inițială $x_{start} \in D$
   - temperatura finală T_{min}
   - o funcție de reducere lentă a temperaturii
     decrease (T)
   - nr. maxim de încercări de îmbunătățire a soluției
     la fiecare prag de temperatură *attempts*
   - nr. maxim de schimbări ale soluției
     la fiecare prag de temperatură *changes*.
2. $T \leftarrow T_{start}$; $x_{old} \leftarrow x_{start}$
   while $T > T_{min}$ **do**
   { $na \leftarrow 0$; $nc \leftarrow 0$
     while $na < attempts$ *and* $nc < changes$ **do**
     { generează o soluție nouă x_{new}; $na++$;
       $\Delta E \leftarrow f(x_{old}) - f(x_{new})$;
       If $\Delta E < 0$ **then** { $x_{old} \leftarrow x_{new}$; $nc++$}
       else
       { generează $q \in (0,1)$ un nr. aleator
         if $q < e^{-\frac{\Delta E}{T}}$**then** { $x_{old} \leftarrow x_{new}$; $nc++$}
       }
     }
     $decrease(T)$
   }
3. **return** x_{old}

Planul de călire se stabileşte adeseori prin experimente (*tuning*) asupra clasei de probleme la care se aplică. Modul în care se permite, la un anumit nivel de temperatură , ca unele soluţii noi care nu micşorează valoarea funcţiei să fie considerate (cu scopul părăsirii minimelor locale) poartă denumirea inventatorului : *schema Metropolis.*

Pentru aplicarea călirii simulate pentru rezolvarea problemei comisului voiajor, se porneşte cu un tur ales aleator, iar trecerea de la o soluţie curentă la o soluţie vecină în spaţiul soluţiilor se realizează de obicei cu ajutorul unei **2-move**.

Se poate lua $T_{start} = O(\sqrt{n})$, $attempts = 100n$, $changes = 10n$, $decrease(T) = 0.95T$ şi $T_{min} = O(1)$.

VIII. GRAFURI PLANARE

1. Proprietăți de bază ale grafurilor planare

Definiție. Fie $G = (V, E)$ un graf și S o suprafață în $\mathbf{R}^3$. Spunem că G este **reprezentabil pe S** dacă există $G' = (V', E')$ un graf astfel încît:
a) $G \cong G'$.
b) V' e o mulțime de puncte distincte din S.
c) Orice muchie $e' \in E'$ este o curbă simplă conținută în S care unește cele două extremități.
d) Orice punct al lui S este sau vîrf al lui G', sau prin el trece cel mult o muchie a lui G'.

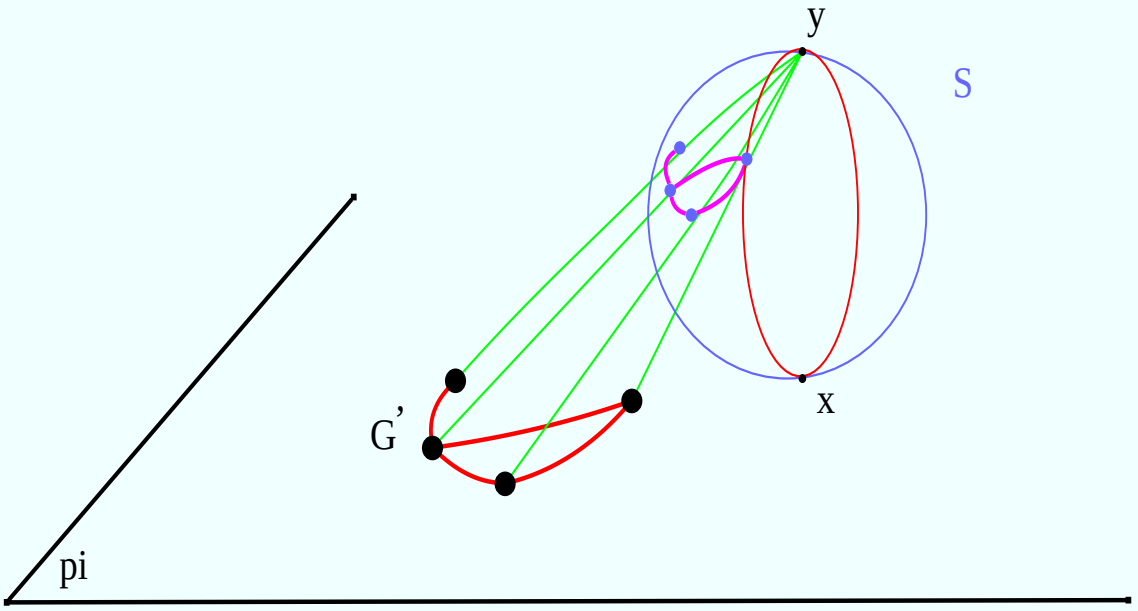
G' se numește **reprezentare a lui** G în S.

Dacă S este un plan atunci G se numește **planar** iar G' o **reprezentare planară** a lui G.

Dacă S este un plan și G' este un graf care satisface b) c) și d) de mai sus atunci G' se numește **graf plan.**

Lemã. 1. *Un graf este planar dacã şi numai dacã este reprezentabil pe o sferã.*

Demonstraţie. Fie G planar şi G' o reprezentare planarã a sa în planul π. Considerãm un punct x al lui π şi S o sferã tangentã la π în x. Fie y punctul diametral opus al lui x pe sferã. Definim $\phi : \pi \to S$ considerînd pentru orice punct M al planului π, $\phi(M)$ cel de-al doilea punct de intersecţie al dreptei My cu sfera S. Evident ϕ este o bijecţie şi deci $\phi(G')$ este reprezentarea lui G pe sferã (ϕ este *proiecţia stereograficã*).



Reciproc, dacă G este reprezentabil pe o sferă S: se alege un punct y pe sferă, se consideră punctul x diametral opus lui y pe sferă, se construiește un plan tangent π sferei S în punctul x și se definește $\psi : S \to \pi$ considerînd pentru orice punct M al sferei, $\psi(M)$ intersecția dreptei yM cu planul π. Imaginea prin ψ a reprezentării lui G pe sferă va fi o reprezentare planară a lui G.

Definiție. Fie G un graf plan. Dacă îndepărtăm punctele lui G (vîrfurile și muchiile sale) din plan se obține o reuniune de regiuni conexe (orice două puncte se pot uni printr-o curbă simplă conținută în regiune) ale planului, care se numesc **fețele** lui G.

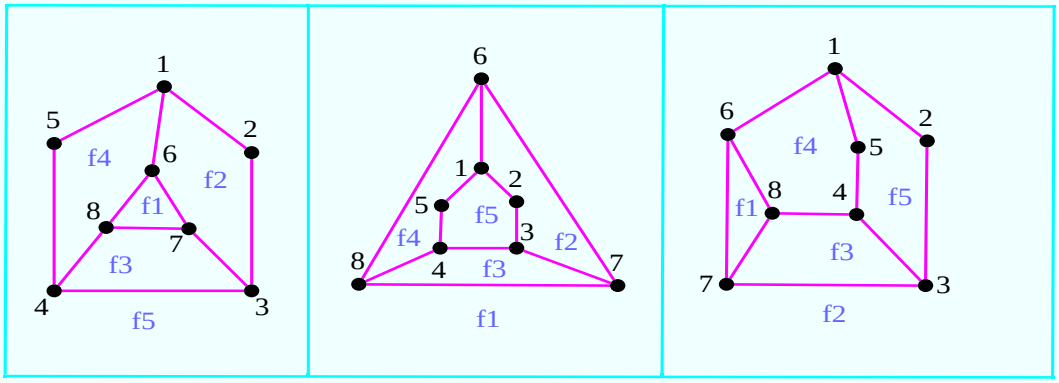
Evident, orice graf plan are un număr finit de fețe, dintre care una singură este nemărginită și se numește **față exterioară** a lui G.

Frontiera unei fețe este mulțimea muchiilor lui G conținute în închiderea acestei fețe.

Orice circuit al lui G împarte punctele planului în două regiuni conexe, deci orice muchie a unui circuit al grafului G se află în frontierele a două fețe.

Un graf plan, împreună cu mulțimea fețelor sale se numește **hartă planară** (uneori, în această definiție se impune ca graful sa fie 2-conex; atunci fețele hărții planare se numesc țări și două țări sînt vecine dacă frontierele lor au măcar o muchie comună; aceasta corespunde aspectelor intuitive de la hărțile geografice).

Unui graf planar i se pot asocia reprezentări planare diferite. În figura alăturată sînt desenate trei reprezentări planare diferite ale aceluiași graf:



Un instrument util în caracterizarea și recunoașterea grafurilor planare este următoarea observație:

Lemă. 2. *Orice reprezentare planară a unui graf poate fi transformată într-o reprezentare diferită astfel încît o față specificată a sa să devină fața exterioară.*

Demonstrație: Fie G' o reprezentare planară a lui G și F o față a lui G'. Considerăm G^0 o reprezentare pe sferă a lui G' (construită, de exemplu, ca în lema 1) și fie F^0 fața corespunzătoare a lui F (proiecția stereografică transformă orice față tot într-o față). Alegem un punct y din interiorul feței F^0, considerăm x punctul diametral opus pe sferă lui y și efectuăm proiecția stereografică ϕ pe planul tangent sferei în punctul x. Graful plan G'' astfel obținut va avea fața exterioară $\phi(F^0)$ care are aceeași frontieră cu fața F din reprezentarea G'.

Este aproape evident că orice două reprezentări planare ale aceluiași graf au același număr de fețe. Acest lucru este mai bine precizat de următoarea teoremă.

Teoremă. 1. (Formula lui Euler) *Fie* $G = (V, E)$ *un graf plan conex cu* $\mathbf{n}$ *vîrfuri,* $\mathbf{m}$ *muchii și* $\mathbf{f}$ *fețe. Atunci*

$$\mathbf{f = m - n + 2}$$

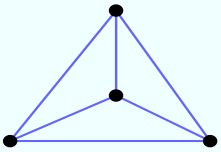
Demonstrație: Inducție după numărul fețelor lui G. Dacă $f = 1$, atunci G nu are circuite și cum este și conex, rezultă că G este arbore, deci $m = n - 1$ și prin urmare teorema are loc.

În pasul inductiv, presupunem teorema adevărată pentru orice graf plan și conex cu mai puțin de $f(\geq 2)$ fețe. Există măcar o muchie e, care nu este punte (altminteri G ar fi arbore și ar avea o singură față). Rezultă că e aparține frontierei a exact două fețe ale lui G. Considerăm $G_1 = G - e$. Din alegerea lui e, G_1 este conex. Evident G_1 este graf plan.

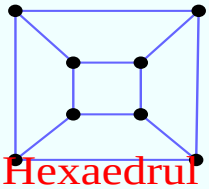
Numărul feţelor lui G_1 este $f_1 = f - 1$ (cele două feţe ale lui G vor genera în G_1 o singură faţă); în plus G_1 are $n_1 = n$ vîrfuri şi $m_1 = m-1$ muchii.
Pentru G_1 are loc ipoteza inductivă, deci $f_1 = m_1 - n_1 + 2$.
Substituind valorile lui f_1, m_1 şi n_1 se obţine $f - 1 = (m - 1) - n + 2$, adică $f = m - n + 2$.

Observaţii: 1^0 Oricărui poliedru convex din $\mathbf{R}^3$ i se poate asocia un graf planar cu acelaşi număr de vîrfuri şi muchii (se reprezinta vîrfurile şi muchiile poliedrului pe o sferă şi se execută proiecţia stereografică).

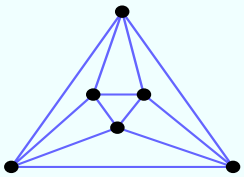
Figura următoare conţine grafurile planare asociate poliedrelor platonice:



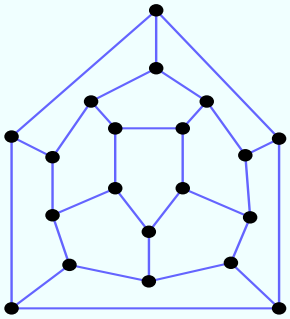
Tetraedrul



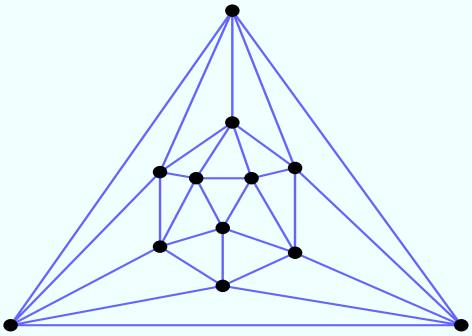
Hexaedrul
(Cubul)



Octoedrul



Dodecaedrul



Icosaedrul

Steinitz și Rademacher (1934) au arătat că un graf planar este graful asociat unui poliedru convex dacă și numai dacă este 3-conex. Relația din teorema 1, este cunoscută ca **formula poliedrală a lui Euler** și precizează numărul fețelor poliedrului convex în funcție de numărul vîrfurilor și muchiilor sale.

2^0. Din punct de vedere algoritmic, teorema 1 are drept consecinţă imediată faptul că orice graf planar este "rar", numărul muchiilor este de ordinul numărului de vîrfuri. Va rezulta că orice traversare în ordinul $O(|V| + |E|)$ a lui G este de fapt în $O(|V|)$ operaţii.

Corolar. 1. *Fie G un graf planar, conex, cu $n(\geq 3)$ vîrfuri şi $m > 2$ muchii. Atunci*

$$m \leq 3n - 6.$$

Demonstraţie. Fie G' o reprezentare planara a lui G. Dacă G' are o singură faţă, atunci G' este arbore, $m = n-1$ (G' are acelaşi număr de vîrfuri şi muchii ca şi G), şi pentru $n \geq 3$ inegalitatea are loc. Daca G' are măcar două feţe, atunci fiecare faţă F are în frontieră muchiile unui circuit $C(F)$, şi fiecare astfel de muchie aparţine la exact două feţe. Orice circuit al grafului are măcar 3 muchii, deci

$2m \geq$

$\sum_F$ față în G' (nrul muchiilor circ. $C(F)) \geq$

$\sum_F 3 = 3f = 3(m - n + 2)$,

inegalitate, evident, echivalentă cu cea din enunț.

Corolar. 2. *Dacă G este un graf bipartit, conex și planar cu $m > 2$ muchii și n vîrfuri, atunci $m \leq 2n - 4$.*

Demonstrație: Același raționament ca mai sus, cu observația că orice circuit are măcar 4 muchii pentru că, graful fiind bipartit, numărul muchiilor oricărui circuit este par.

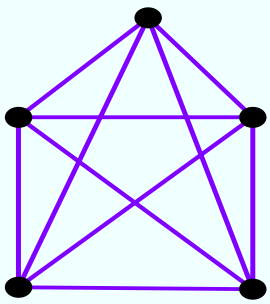
Corolar. 3. *Dacă G este un graf planar conex, atunci G are un vîrf de grad cel mult 5.*

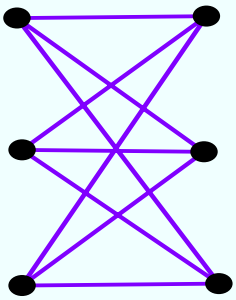
Demonstraţie: Dacă G are mai mult de 2 muchii aplicăm consecinţa 1 astfel: fie G' o reprezentare planară a lui G cu n vîrfuri şi m muchii; notăm cu n_i numărul vîrfurilor de grad i $(1 \leq i \leq n-1)$ din G'; atunci
$\sum_{i=1}^{n-1} i \cdot n_i = 2m \leq 2(3n-6) = 6(\sum_i n_i) - 12$, deci

$$\sum_i (i-6)n_i \quad + 12 \leq 0$$

Cum pentru $i \geq 6$ toţi termenii sumei sînt nenegativi, rezultă că există i_0, $1 \leq i_0 \leq 5$, astfel încît $n_{i_0} > 0$, ceea ce trebuia demonstrat.

Observaţie: Grafurile K_5 şi K_{33} nu sînt planare.





Primul graf violează inegalitatea din corolarul 1, cel de al doilea pe cea din corolarul 2.

Fie $G = (V, E)$ un graf și $v \in V(G)$ astfel încît $d_G(v) = 2$ și $vw_1, vw_2 \in E$, $w_1 \neq w_2$. Considerăm $h(G) = (V \setminus \{v\}, E \setminus \{vw_1, vw_2\} \cup \{w_1w_2\})$. Se observă că G este planar dacă și numai dacă $h(G)$ este planar.

(Dacă $w_1w_2 \notin E(G)$ atunci pe curba simplă ce unește w_1 și w_2 într-o reprezentare planară a lui $h(G)$ se introduce un nou vîrf v; dacă $w_1w_2 \in E(G)$, în una din fețele reprezentării planare a lui $h(G)$ se plasează "suficient" de aproape de curba w_1w_2 un vîrf nou v și se "unește" cu w_1 și w_2. Reciproc, în reprezentarea planară a lui G ștergem punctul v și cele două muchii vw_1 și vw_2 le înlocuim cu reuniunea lor. Dacă $w_1w_2 \in E(G)$ atunci se șterge și curba w_1w_2).

Vom nota cu $h^*(G)$ graful obținut din G aplicîndu-i repetat transformarea h, pînă cînd graful curent nu mai are vîrfuri de grad 2. Din observația anterioară rezultă că G este planar, dacă și numai dacă $h^*(G)$ este planar.

Definiţie. Două grafuri G_1 şi G_2 se numesc *homeomorfe* dacă şi numai dacă $h^*(G_1) \cong h^*(G_2)$.

Teoremă. 2. *(Kuratowski 1930) Un graf este planar dacă şi numai dacă nu are subgrafuri homeomorfe cu K_5 sau K_{33}.*

Necesitatea teoremei este evidentă: dacă un graf este planar atunci orice subgraf al său este planar. Dacă ar exista un subgraf G' al lui G homeomorf cu K_{33} sau K_5, cum aceste două grafuri am arătat că nu sînt planare şi cum $h^*(G')$ este planar dacă şi numai dacă G' este planar rezultă că G' şi deci G nu este planar, contrazicînd alegerea lui G.

Suficienţa se demonstrează prin inducţie după numărul de muchii.

2. Desenarea unui graf planar

Fie G un graf planar. Se pune problema trasării lui în plan. Este aproape evident că se pot aproxima curbele simple ce unesc vîrfurile, cu linii poligonale, astfel încît să nu se violeze condiţia de planaritate. Are loc, totuşi, un rezultat mai tare demonstrat de *Fary în 1948* (şi independent de *Wagner şi Stein*):

Orice graf planar are o reprezentare planară cu toate muchiile segmente de dreaptă (**reprezentarea Fary).**

Această teoremă a deschis o serie de rezultate referitoare la desenarea grafurilor planare. Pentru un mediu grafic, care permite poziţionarea în puncte de coordonate întregi şi trasarea de segmente ce unesc aceste puncte, este desigur important să se demonstreze existenţa unei reprezentări Fary cu **vîrfuri în puncte de coordonate întregi,** şi în acelaşi timp aria suprafeţei ocupate de reprezentare să fie polinomială în raport cu numărul n de vîrfuri ale grafului.

Descriem unul dintre rezultatele de acest tip:

Teoremă. 3. *(Fraysseix, Pach, Pollack (1988)) Orice graf planar cu n vîrfuri are o reprezentare planară cu vîrfuri de coordonate întregi în $[0, 2n-4] \times [0, n-2]$ și cu muchii segmente de dreaptă.*

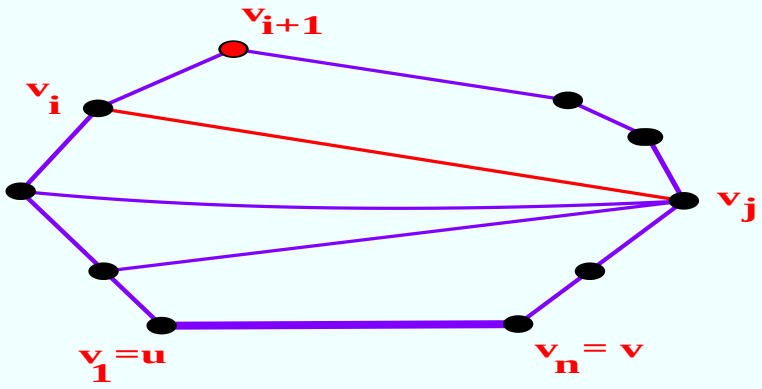
Notăm că demonstrația care urmează poate fi baza unui algoritm de complexitate $O(n \log n)$ pentru obținerea acestei reprezentări.

Lemă. 3. *Fie G un graf planar și G' o reprezentare planară a sa. Dacă C' este un circuit al lui G' ce trece prin muchia $uv \in E(G')$ atunci există $w \in V(C')$ astfel încît $w \neq u, v$ și w nu este extremitatea nici unei corzi interioare a lui C'.*

Demonstraţie: Fie $v_1, v_2, ..., v_n$ vîrfurile circuitului C' într-o parcurgere a sa de la u la v ($v_1 = u, v_n = v$). Dacă C' nu are corzi interioare lema este demonstrată. Altfel, alegem perechea (i, j) astfel încît $v_i v_j$ este coardă interioară a lui C' şi

$j - i = \min\{k - l \mid k > l+1, v_k v_l \in E(G'), v_k v_l \text{ coardă interioară }\}$.

Atunci v_{i+1} nu e incident cu nici o coardă interioară ($v_{i+1} v_p$ cu $i + 1 < p < j$ nu e coardă interioară căci s-ar contrazice alegerea perechii (ij) şi $v_{i+1} v_l$ cu $l < i$ sau $l > j$, nu este coardă interioară datoritā existenţei coardei $v_i v_j$ şi a planarităţii).

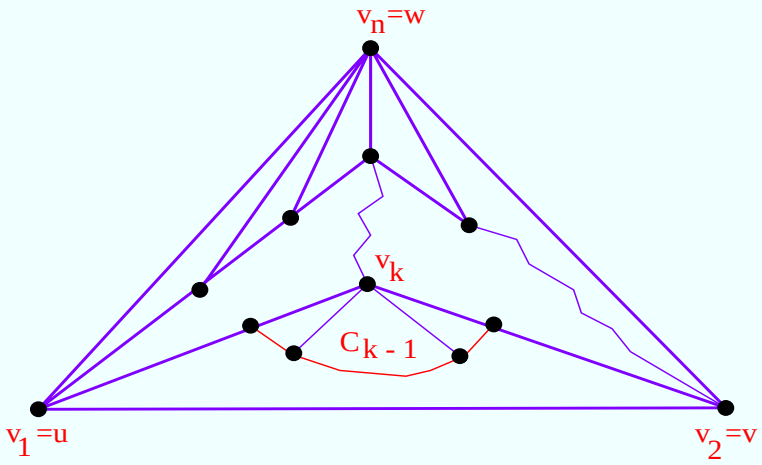


Vom demonstra teorema 4 în ipoteza suplimentară că G este **maximal planar :** orice muchie i s-ar adăuga se obţine un graf neplanar (sau multigraf).
Să observăm că orice faţă a unui graf maximal planar este un C_3 (altminteri în reprezentarea lui G cu faţa exterioară mărginită de un C_n cu $n \geq 4$ se pot introduce muchii fără a pierde planaritatea grafului).
Ipoteza nu este restrictivă: de la o reprezentare a lui G ca o hartă planară (ce se obţine aplicînd de exemplu algoritmul de testare a planarităţii) se trece la o hartă cu toate feţele triunghiului prin inserţia în timp liniar de corzi în circuite. La desenarea grafului obţinut, muchiile fictive introduse nu se vor trasa.

Lemã. 4. *Fie G maximal planar cu $n \geq 4$ vîrfuri şi G' o reprezentare planară a sa cu faţa exterioară u, v, w. Există o etichetare a vîrfurilor lui G' : $v_1, v_2, ..., v_n$ astfel încît $v_1 = u, v_2 = v, v_n = w$ şi pentru orice $k, 4 \leq k \leq n$ avem:*

(i) *subgraful* $G'_{k-1} = [\{v_1, \ldots, v_{k-1}\}]_{G'}$, *este 2-conex şi faţa sa exterioară este determinată de circuitul* C'_{k-1} *ce conţine* uv.
(ii) *în subgraful* G'_k, *vîrful* v_k *este în faţa exterioară a lui* G'_{k-1} *şi* $N_{G'_k}(v_k) \cap \{v_1, \ldots, v_{k-1}\}$ *este un drum de lungime* ≥ 1 *de pe circuitul* $C'_{k-1} \setminus uv$.

Demonstraţie: Fie $v_1 = u, v_2 = v, v_n = w, G'_n = G, G'_{n-1} = G - v_n$. Să observăm că $N_{G'_n}(v_n)$ este un circuit ce conţine uv (este suficient, pentru a demonstra aceasta să ordonăm $N_{G'}(w)$ după abscisele vîrfurilor şi să folosim planaritatea maximală).



Rezultă că pentru $k = n$ (i) şi (ii) au loc.

Dacă v_k a fost ales ($k \leq n$) atunci în $G'_{k-1} = G' - \{v_n, \ldots, v_k\}$ vecinii lui v_k determină un circuit C'_{k-1} ce conține uv și mărginește fața exterioară a lui G'_{k-1}. Din lema 3, rezultă că există v_{k-1} pe C'_{k-1} astfel încît v_{k-1} nu este incident cu o coardă interioară a lui C'_{k-1} .

Din construcție, rezultă că v_{k-1} nu este incident nici cu corzi exterioare lui C'_{k-1} și cu această alegere, se observă că G'_{k-2} va conține un circuit C'_{k-2} cu proprietățile (i) și (ii).

Notăm că etichetarea precizată în lema 4 se poate construi în $O(n)$ considerînd o reprezentare a lui G cu liste de adiacență ordonate circular, prin fixarea vîrfurilor v_1, v_2 și v_n și apoi considerînd pentru fiecare k $k \geq 3$ un vîrf v_k cu proprietatea că arcele cu o extremitate v_k și cealaltă în $\{v_1, \ldots, v_{k-1}\}$ formează un segment continuu în lista circulară a lui $A(v_k)$ (existența unui asemenea vîrf fiind asigurată de lema 4).

Demonstraţia teoremei 4. Aşa cum am mai observat, se poate presupune G maximal planar cu $n(\geq 4)$ vîrfuri. Fie G' o reprezentare planară a lui G, cu vîrfurile $v_1, v_2, \ldots, v_n$, faţa exterioară u, v, w şi etichetarea vîrfurilor satisface condiţiile din lema 4.

Vom construi o reprezentare Fary a lui G cu vîrfurile puncte de coordonate întregi.

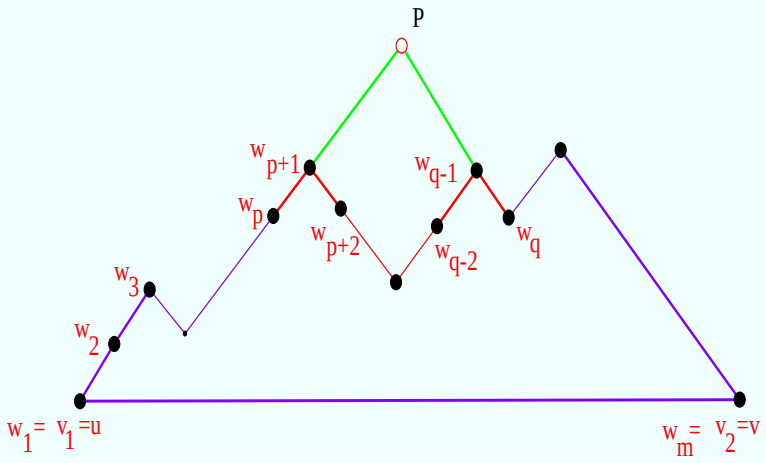
În pasul $k(\geq 3)$ al construcţiei dispunem de o astfel de reprezentare a lui G_k, şi în plus sînt satisfăcute condiţiile:
(1) v_1 are coordonatele $x_{v_1} = 0$ şi $y_{v_1} = 0$; v_2 are coordonatele $x_{v_2} = i \leq 2k - 4, \quad y_{v_2} = 0$;
(2) Dacă $w_1, w_2, \ldots, w_m$ sînt vîrfurile circuitului ce mărgineşte faţa exterioară a lui G_k în ordinea parcurgerii lor de la v_1 la $v_2(w_1 = v_1, w_m = v_2)$ atunci

$$x_{w_1} < x_{w_2} < \ldots < x_{w_m}.$$

(3) Muchiile $w_1w_2, w_2w_3, \ldots, w_{m-1}w_m$ sînt segmente de dreaptă paralele cu prima sau a doua bisectoare a axelor de coordonate.

Condiția (3) asigură că $\forall i < j$ dacă prin w_i se duce o paralelă la prima bisectoare și prin w_j o paralelă la a doua bisectoare,**intersecția lor este un punct de coordonate întregi.**

Construim G'_{k+1}. Fie $w_p, w_{p+1}, \ldots, w_q$ vecinii lui v_{k+1} în G'_{k+1} (cf. lemei 4) $1 \leq p < q \leq m$.



Paralela prin w_p la prima bisectoare intersectează paralela prin w_q la a doua bisectoare, într-un punct P. Dacă din punctul P se pot trasa segmentele $Pw_i \quad p \leq i \leq q$ astfel încît ele să fie distincte, atunci putem lua $v_{k+1} = P$ și obţinem reprezentarea Fary pentru G_{k+1} cu vîrfuri de coordonate întregi, satisfăcînd condiţiile (1) (2) și (3).

Dacă segmentul $w_p w_{p+1}$ este paralel cu prima bisectoare, atunci translăm cu o unitate la dreapta toate vîrfurile grafului G_k, care au abscisa $\geq x_{w_{p+1}}$. Efectuăm apoi o translaţie cu o unitate la dreapta a tuturor vîrfurilor cu abscisa $\geq x_{w_q}$. Să observăm că acum toate segmentele $P'w_i$ cu $p \leq i \leq q$ sînt distincte, segmentele $w_i w_{i+1}$ cu $i = \overline{q, m-1}$ au pantele ± 1 și de asemenea $w_p P'$ și $P'w_q$ (unde P' este punctul obţinut prin intersecţia paralelelor la prima (respectiv a doua) bisectoare, duse prin w_p și w_q). Luăm $v_{k+1} = P'$ și pasul k al construcţiei este terminat.

Condiţiile (1)- (3) sînt evident satisfăcute. Cu aceasta teorema este demonstrată.

Să observăm că determinarea vîrfurilor care se vor transla în pasul k, poate fi organizată astfel încît să nu necesite în total $O(n^2)$ operaţii ci numai $O(n \log n)$, utilizînd structuri convenabile de date.

3. Grafuri plane - versiunea combinatorială.

Vom considera în cele ce urmează doar grafuri conexe cu măcar două vârfuri.

În versiunea combinatorială un graf este un triplet $G = (E, \theta, ^-)$, unde E este o mulţime de cardinal par, $^-$ este o *involuţie* pe E (permutare de ordin 2) fără puncte fixe, şi θ este o permutare pe E.

Elementele lui E sunt gândite ca *arce*; o muchie (neorientată) este reprezentată ca o pereche $e, \overline{e} \in E$ de arce, inverse unul altuia.
Aplicaţia $^-$ inversează direcţia.

Se doreşte ca aplicaţia θ să dea o **orientare a muchiilor din jurul unui vârf** (în sens contrar acelor de ceasornic).

Vârfurile sunt *ciclii* permutării θ. (Un ciclu al permutării θ este o submulţime nevidă a lui E închisă în raport cu θ şi minimală cu această proprietate).

Dacă notăm cu V mulţimea ciclilor permutării θ atunci definim
$t : E \to V$ **prin** $t(e) =$ **unicul ciclu al lui** θ **ce conţine** e **(extremitatea iniţială a arcului** e**)**
$t : E \to V$ **prin** $t(e) =$ **unicul ciclu al lui** θ **ce conţine** $\overline{e}$ **(extremitatea finală a arcului** e**)**
Se observă că $\forall e \; t(e) = h(\overline{e})$ şi $h(e) = t(\overline{e})$.

Dacă vom considera permutarea $\theta^* : E \to E$ definită de $\theta^*(e) = \theta(\overline{e})$, atunci o **faţă** a lui G este un ciclu al permutării θ^*.
Intuitiv, pentru a calcula $\theta^*(e)$, inversăm e pentru a obţine $\overline{e}$ şi apoi ne rotim (în sensul acelor de ceasornic) în jurul extremităţii iniţiale a lui $\overline{e}$. Numărul feţelor lui G se notează cu f.

O **componentă conexă** a lui G este o *orbită* a lui E în grupul de permutări generat de θ şi $^-$: **o mulţime nevidă minimală cu proprietatea că este închisă la θ şi $^-$.**

Fie G un graf cu $m = \frac{1}{2}|E|$ muchii (neorientate), $n = |V|$ vârfuri, f feţe, şi c componente conexe. **Caracteristica Euler** a lui G se defineşte ca fiind

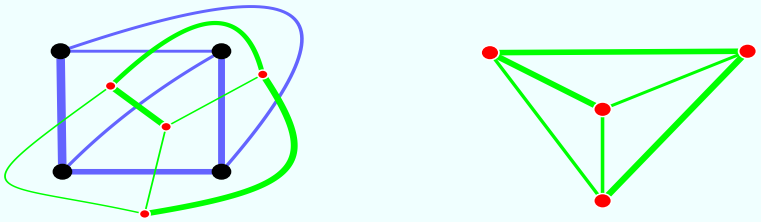
$$\chi(G) = 2c + m - n - f.$$

Un graf G se numeşte **graf plan** dacă $\chi(G) = 0$.

Se poate demonstra că pentru un graf conex în definiţia tradiţională, cele două noţiuni de grafuri plane coincid (graful neorientat construit aşa cum am descris mai sus ataşat unui graf în formă combinatorială este graf plan conform definiţiei tradiţionale şi invers,
dacă pentru un graf tradiţional plan conex se construieşte θ conform unei orientări inverse acelor de ceasornic a muchiilor şi $^-$ corespunzătoare, graful combinatorial obţinut este plan în noua definiţie).

4. Teorema separatorului

Definiţie Dacă $G = (V, E)$ este un graf plan maximal (cu toate feţele triunghiuri) atunci **dualul** său este graful $G^* = (V^*, E)$ unde V^* sunt feţele grafului G iar muchiile lui G^* se obţin din muchiile grafului G, asociindu-i fiecărei muchii $e \in E$ cele două feţe în frontira cărora se află e.



G $\qquad$ G*

Lemă. 1. *Fie $G = (V, E)$ este un graf plan maximal conex , $G^* = (V^*, E)$ dualul său şi $E' \subseteq E$. Atunci (V, E') este arbore parţial al lui G dacă şi numai dacă $(V^*, E - E')$ este arbore parţial al lui G^*.*

Teoremă. 1. *(Tarjan & Lipton, 1979)*
Fie G un graf planar cu n vârfuri. Există o partiţie a lui $V(G)$ în clasele disjuncte A, B, S astfel încât:

1. *S separă A de B în G: $G - S$ nu are muchii cu o extremitate în A şi cealaltă în B.*
2. *$|A| \leq \frac{2}{3}n$, $|B| \leq \frac{2}{3}n$.*
3. *$|S \leq 4\sqrt{n}$.*

Această partiţie se poate afla în timpul $O(n)$.

Demonstraţie. Considerăm graful conex şi de asemenea considerăm că dispunem de o reprezentare planară (obţinute cu un algoritm liniar).

Alegem un vârf s şi executăm o parcurgere bfs din s numerotând vârfurile (în ordinea întâlnirii lor în această parcurgere) şi atribuind fiecărui vârf v nivelul său în arborele bfs construit.

Vom nota cu $L(t)$, $0 \leq t \leq l+1$ mulțimea vârfurilor de pe nivelul t (nivelul $l+1$ va fi introdus în scopuri tehnice și este vid, ultimul nivel este de fapt l).
Evident, fiecare nivel este un separator în G (avem muchii doar între nivele consecutive).
Fie t_1 nivelul de la mijloc, adică nivelul ce conține vârful numerotat bfs cu numărul de ordine $\frac{n}{2}$. Mulțimea $L(t_1)$ are o parte din proprietățile separatorului pe care îl căutăm:

$$\left|\cup_{t<t_1} L(t)\right| < \frac{n}{2} \qquad \left|\cup_{t>t_1} L(t)\right| < \frac{n}{2}.$$

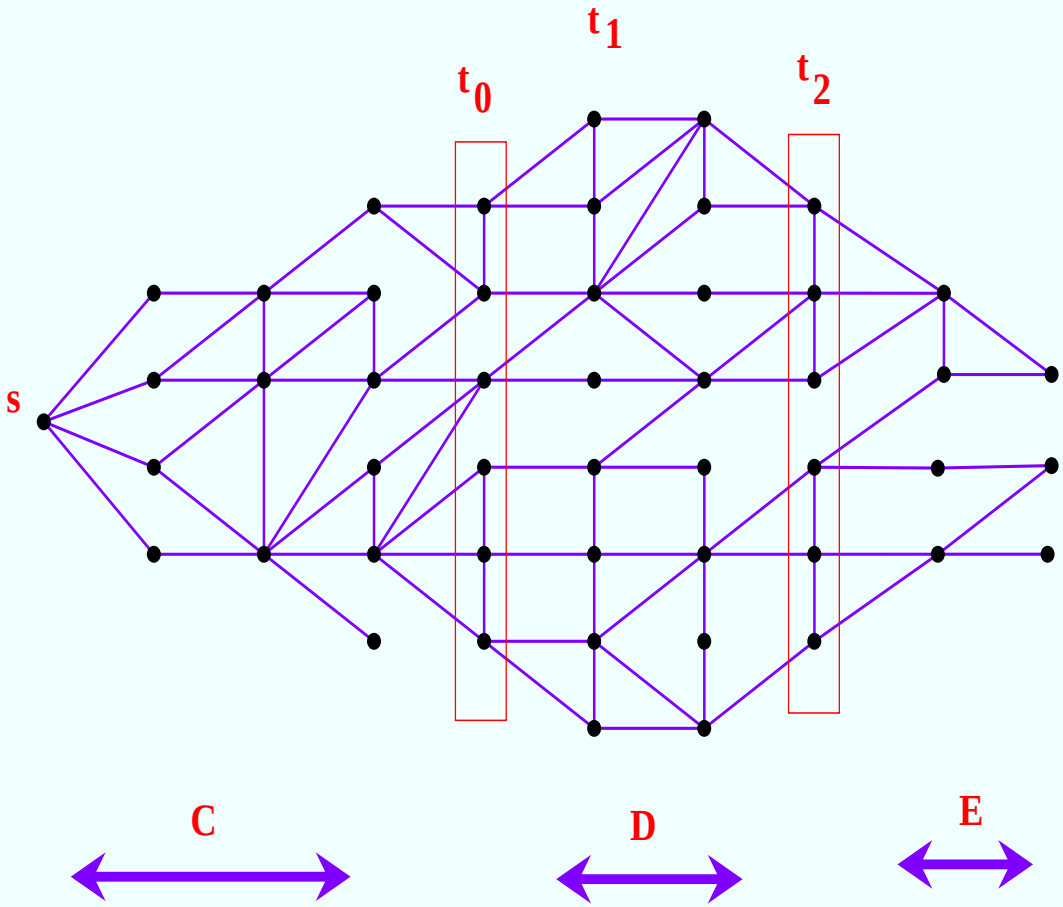
Dacă și $L(t_1) \leq 4\sqrt{n}$, teorema are loc.

Lemă. 2. *Există nivelele $t_0 \leq t_1$ și $t_2 > t_1$ a. încât $|L(t_0)| \leq \sqrt{n}$, $|L(t_2)| \leq \sqrt{n}$ și $t_2 - t_0 \leq \sqrt{n}$.*

Se alege t_0 cel mai mare număr cu proprietățile $t_0 \leq t_1$ și $|L(t_0)| \leq \sqrt{n}$ (există un astfel de nivel pentru că $|L(0)| = 1$). La fel, există t_2 un cel mai mic număr astfel înc at $t_2 > t_1$ și $|L(t_2)| \leq \sqrt{n}$ (de aceea s-a luat $|L(l+1)| = 0$). Orice nivel strict între t_0 și t_2 are mai mult de $\sqrt{n}$ vârfuri deci numărul acestor nivele este mai mic decât $\sqrt{n}$, altfel am avea mai mult de n vârfuri în graf.

Considerām

$C = \cup_{t<t_0} L(t),\ D = \cup_{t_0<t<t_2} L(t),\ E = \cup_{t>t_2} L(t).$



Dacă $|D| \leq \frac{2}{3}n$ atunci teorema are loc cu $S = L(t_0) \cup L(t_2)$, A mulțimea cu cele mai multe elemente dintre C, D, E și B reuniunea celor-lalte două (nu uităm că C și E au cel mult $\frac{n}{2}$ elemente).

Considerām deci cā $n_1 = |D| > \frac{2}{3}n$.
Dacā vom gāsi un separator de tipul $\frac{1}{3} \leftrightarrow \frac{2}{3}$ pentru D cu cel mult $2\sqrt{n}$ vârfuri, atunci , atunci îl vom adāuga la $L(t_0) \cup L(t_2)$ pentru a obţine un separator de cardinal cel mult $4\sqrt{n}$, reunim mulţimea cu cel mai mare număr de elemente dintre C şi E cu partea micā rāmasā din D pentru a obţine A, iar partea mare rāmasā în D o reunim cu cealaltā mulţime (micā) dintre C şi E pentru a obţine B.

Ne ocupām de construcţia separatorului pentru D. Vom şterge toate vârfurile grafului care nu-s în D cu excepţia lui s pe care-l unim cu toate vîrfurile de pe nivelul $t_0 + 1$ (primul nivel rāmas în D). Graful obţinut îl notām cu D şi este evident planar şi conex. În plus are un arbore parţial T de diametru cel mult $2\sqrt{n}$ (orice vârf este accesibil din s pe un drum de lungime cel mult $\sqrt{n}$ aşa cum am arātat în lemā).

Acest arbore se construiește simplu pornind de la ultimul nivel și adăugând pentru fiecare vârf o muchie incidentă cu el și cu cealaltă extremitate pe nivelul precedent (nivelul 0 conține doar s).

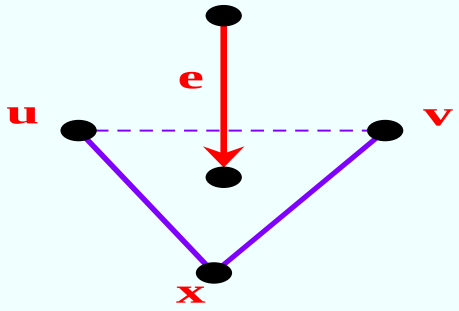
Putem presupune că D este triangulat (altfel se triangularizează în timp liniar). Construim dualul său D^*, tot în timp liniar. Muchiile lui D care nu-s pe arborele T vor fi numite **țepi**. Din lema 1, rezultă că țepii vor secționa un arbore parțial T^* al lui D^*. Considerăm o față a lui D drept rădăcină a lui T^* și considerăm muchiile lui T^* orientate dinspre rădăcină.

Fie $e = uv$ un țep. Există un unic drum de la u la v în T care împreună cu e determină un circuit $c(e)$.

Parcurgem țepii conform unei traversări dfs a lui T^* calculând următoarele informații pentru fiecare țep, recursiv de la frunze în sus:

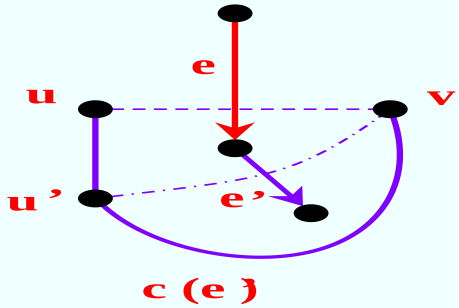
- $I(e)$- numărul vârfurilor din interiorul lui $c(e)$.
- $|c(e)|$- numărul vârfurilor de pe circuitul $c(e)$;
- o reprezentare ca listă a lui $c(e)$.

Sunt posibile următoarele patru cazuri (ce corespund întoarcerilor din parcurgerea dfs).



Cazul 1.

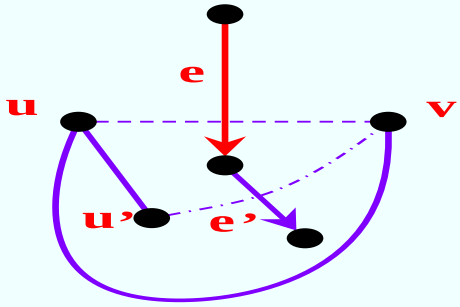
Suntem într-o frunză e a lui T^* (ce se determină numărând vecinii). Atunci:
- $I(e) = 0$;
- $|c(e)| = 3$ (D e triangulat);
- $c(e) = [u, x, v]$.



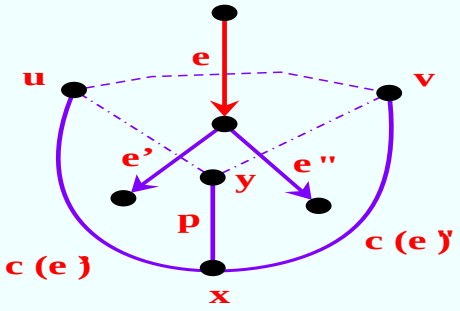
Cazul 2.

Avem calculată informaţia pentru ţepul $e' = u'v$, e este un ţep în acelaşi triunghi ca şi e' şi u' este pe circuitul $c(e)$; ac. rezultă testând dacă u nu-i pe lista $c(e')$. Atunci:

- $I(e) = I(e')$;
- $|c(e)| = c(e') + 1$;
- $c(e) = [u] \cdot c(e')$.



Cazul 3.
Avem calculată informaţia pentru ţepul $e' = u'v$, e este un ţep în acelaşi triunghi ca şi e' şi u' nu este pe circuitul $c(e)$; ac. rezultă testând dacă u e pe lista $c(e')$. Atunci:
- $I(e) = I(e') + 1$;
- $|c(e)| = c(e') - 1$;
- $c(e') = [u] \cdot c(e)$ (deci $c(e)$ se obţine ştergându-l pe u din lista $c(e')$.



Cazul 4.
Avem calculată informaţia pentru ţepii $e' = uy$ şi $e'' = yv$ iar e este un ţep în acelaşi triunghi ca şi e', e''.

Fie p drumul comun al lui $c(e')$ și $c(e'')$ și fie x celălalt capăt al lui p în afară de y. Atunci:
- $I(e) = I(e') + +I(e'') + |p| - 1$ (toate vârfurile lui p cu excepția lui x sunt în interiorul lui $c(e)$);
- $|c(e)| = c(e') + c(e'') - 2|p| + 1$;
- $c(e') = c' \cdot [x] \cdot c''$ unde c' este $c(e')$ cu p șters și c'' este $c(e'')$ cu p șters.

Calculul lui $|p|$ și reprezentarea lui $c(e)$ se obțin scanând $c(e')$ și $c(e'')$ plecând din y până întâlnim ultimul vârf comun care este x. Aceasta nu afectează complexitatea de timp liniară.

Rămâne de arătat că există un țep e astfel încât

$$I(e) \leq \frac{2n_1}{3} \quad \wedge \quad n_1 - (I(e) + c(e)) \leq \frac{2n_1}{3}.$$

Atunci se poate lua $c(e)$ ca separator, vârfurile din interior drept o clasă, iar cele din interior drept cealaltă clasă.

Fie e primul țep întâlnit la întoarcerea din frunzele lui T^* către rădăcină, care satisface proprietatea că $I(e) + |c(e)| \geq \frac{n_1}{3}$.

Atunci mulţimea vârfurilor din exteriorul lui c(e) este de cardinal $n_1 - (I(e) + |c(e)|) \leq \frac{2n_1}{3}$, deci ceea ce rămâne de arătat este că $I(e) \leq \frac{2n_1}{3}$.

Aceasta rezultă analizând cazurile 1-4 de mai sus în care s-a ajuns la e:
1. $I(e) = 0 \leq \frac{2n_1}{3}$.
2. $I(e)+|c(e)| = I(e')+|c(e')|+1 < \frac{n_1}{3}+1 \leq \frac{2n_1}{3}$ (pentru $n_1 \geq 3$).
3. $I(e) + c(e) = I(e') + c(e')$, deci e nu-i prima muchie cu proprietatea că $I(e) + |c(e)| \geq \frac{n_1}{3}$.
4. $I(e) + c(e) = I(e') + I(e'') + |p| - 1 + c(e') + c(e'') - 2|p| + 1 < \frac{2n_1}{3} - |p| < \frac{2n_1}{3}$.

Cu aceasta teorema separatorului este complet demonstrată.

Ilustrăm cum poate fi folosită teorema separatorului împreună cu o metodă de *divide & impera* pentru a obţine algoritmi exacţi cu comportare subexponenţială pentru probleme NP-hard pe grafuri planare.

Considerăm problema testării dacă un graf planar dat admite o 3-colorare a vârfurilor (problemă cunoscută ca fiind NP-completă).

Pentru grafuri cu puține vârfuri (un număr constant c) se poate testa în timpul $O(3^c) = O(1)$ dacă graful are o 3-colorare.

Pentru grafuri planare cu numărul n de vârfuri mai mare decât c, construim în timp liniar $O(n)$, așa cum ne asigură teorema separatorului, partiția A, B, C a mulțimii vârfurilor sale cu $|A|, |B| \leq \frac{2n}{3}$ și $|C| \leq 4\sqrt{n}$.

Pentru fiecare din cele $3^{|C|} = 2^{O(\sqrt{n})}$ funcții posibile definite pe C și cu valori în $\{1, 2, 3\}$ se testează dacă este 3-colorare a subgrafului indus de C și dacă poate fi extinsă la o 3-colorare a subgrafului indus de $A \cup C$ în G și la o 3-colorare a subgrafului indus de $B \cup C$ în G (recursiv).

Timpul de lucru al acestui algoritm, $T(n)$, va satisface recurența

$$T(n) = \begin{cases} O(1) & \text{dacă } n \leq c; \\ O(n) + 2^{O(\sqrt{n})}(O(\sqrt{n}) + 2T(\frac{2n}{3})) & \text{dacă } n > c. \end{cases}$$

Se obține $T(n) = 2^{O(\sqrt{n})}$, destul de bun pentru probleme de dimensiuni rezonabile.

Este posibil însă ca notația $O(.)$ să ascundă constante mari !

Există și alte abordări pentru obținerea de algoritmi performanți pentru problemele dificile pe grafuri planare. Descriem în continuare una din ele.

Definiţie O **t-descompunere** a unui graf $G = (V(G), E(G))$ este o pereche $(\{X_i | i \in V(T)\}, T)$, unde $\{X_i | i \in V(T)\}$ este o familie de submulţimi ale lui $V(G)$ şi T este un arbore, astfel încât:

1. $\cup_{i \in V(T)} X_i = V(G)$;
2. $\forall vw \in E(G)$ $\exists i \in V(T)$ astfel încât $v, w \in X_i$;
3. $\forall v \in V(G)$ mulţimea vârfurilor $\{i | v \in X_i\}$ induce un subarbore în T .

Lăţimea unei t-descompuneri $\{X_i | i \in V(T)\}$ se defineşte ca fiind $\max_{i \in V(T)}(|X_i| - 1)$.

t-lăţimea unui graf G este lăţimea minimă a unei t-descompuneri a lui G şi se notează cu $\mathbf{tw}(G)$.

Definiţie O **b-descompunere** a unui graf $G = (V(G), E(G))$ este o pereche (T, τ), unde T este un arbore cu vârfurile de gradul 1 sau 3 şi τ este o bijecţie de la mulţimea vârfurilor pendante ale lui T la $E(G)$.

Ordinul unei muchii e a lui T este numărul vârfurilor $v \in V(G)$ astfel încât există vârfurile pendante t_1, t_2 ale lui T în componente conexe diferite ale lui $T - e$ cu $\tau(t_1)$ şi $\tau(t_2)$ incidente cu v.

Laţimea b-descompunerii (T, τ) este ordinul maxim al unei muchii a lui T.

b-lăţimea unui graf G este lăţimea minimă a unei b-descompuneri a lui G şi se notează cu $\mathbf{bw}(G)$. (dacă $|E(G)| \leq 1$ se defineşte b-lăţimea lui G ca fiind 0; grafurile nule nu au b-descompuneri; dacă $|E(G)| = 1$ atunci G are o b-descompunere care constă dintr-un arbore cu un nod, a cărui lăţime se consideră 0).

Teoremă. 2. *(Robertson & Seymour, '90) Pentru orice graf conex G cu $E(G) \geq 3$ avem*

$$\mathbf{bw}(G) \leq \mathbf{tw}(G) + 1 \leq \frac{3}{2}\mathbf{bw}(G).$$

Teoremă. 3. *(Fomin & Thilikos, 2003) Pentru orice graf planar G avem*

$$\mathbf{bw}(G) \leq \sqrt{4.5|V(G)|} \leq 2.122\sqrt{|V(G)|}.$$

Teoremă. 4. *(Fomin & Thilikos, 2003) Fie Π o problemă de optimizare care este rezolvabilă pe grafuri cu b-lățimea cel mult l și ordinul n în timpul $f(l)g(n)$. Atunci pe grafurile planare de ordin n problema Π este rezolvabilă în timpul $O(f(2.122\sqrt{n})g(n) + n^4)$.*

Demonstrația rezultă utilizând teorema anterioară și un algoritm de complexitate $O(n^4)$ dat de Robertson & Seymour pentru determinarea unei b-descompuneri optimale a unui graf.

De exemplu, se ştie că problema determinării unei mulţimi stabile de cardinal maxim într-un graf cu t-lăţimea l se poate rezolva în timpul $O(2^l n)$. Deci, pentru grafuri cu b-lăţimea cel mult l se poate rezolva în $O(2^{\frac{3}{2}l} n)$.

Obţinem că problema determinării unei mulţimi stabile de cardinal maxim într-un graf planar se poate rezolva în timpul $O(2^{3.182\sqrt{n}} n + n^4)$.

Timpii obţinuţi sunt mai buni decât metodele bazate pe teoremele de separare !

E N D

Seminarii

Setul de probleme 1

Problema 1. Un graf G se numeşte *rar* dacă numărul său de muchii m este mai mic decit $\frac{n^2}{\log n}$, unde n reprezintă numărul de virfuri. O justificare este aceea că matricea de adiacenţă A a grafului, care ocupă n^2 locaţii de memorie, poate fi intotdeauna reprezentată folosind $O(\frac{n^2}{\log n})$, locaţii de memorie astfel incit răspunsul la o intrebare " $A(i,j) = 1$? " să se facă in $O(1)$. Descrieţi o astfel de schemă de reprezentare. (**4 puncte**)

Problema 2. Diametrul unui graf este lungimea maximă a unui drum de lungime minimă intre două virfuri ale grafului. Două virfuri care sunt extremităţile unui drum minim de lungime maximă in graf se numesc diametral opuse. Demonstraţi că următorul algoritm determină o pereche de virfuri diametral opuse intr-un arbore T:

• dintr-un virf oarecare se execută o parcurgere BFS a lui T; fie u ultimul virf vizitat;din virful u se execută o

parcurgere BFS a lui T; fie v ultimul virf vizitat;

• return u,v.

Este valabil algoritmul pentru un graf conex oarecare ? (**4 puncte**)

Problema 3. Fie T un arbore un arbore binar cu rădăcină. Un algoritm simplu de desenare a lui T poate fi descris recursiv dupa cum urmează.

- Folosim ca suport o grilă (liniatura unui caiet de mate); virfurile se plasează in punctele de intersecţie ale grilei.

- Desenăm subarborele sting; Desenăm subarborele drept.

- Plasăm cele două desene unul lingă altul la distanţă pe orizontală doi şi cu rădăcinile la aceeaşi inălţime.

- Plasăm rădăcina cu un nivel mai sus la jumătatea distanţei pe orizontală dintre cei doi copii.

- Dacă avem doar un copil plasăm rădăcina cu un nivel mai sus la distanţa 1 faţă de copil (la stinga sau la dreapta după cum este acesta).

Descrieţi cum se poate asocia pentru fiecare nod v al arborelui T (folosind algoritmul de mai sus) coordonatele $(x(v), y(v))$ reprezentind punctul de pe grilă unde va fi desenat.(**3 puncte**)

Problema 4. Intr-o sesiune de examene s-au inscris n studenţi care trebuie să susţină examene dintr-o mulţime de m discipline. Intrucit examenele se susţin in scris, se doreşte ca toţi studenţii care dau examen la o disciplină să facă acest lucru simultan. De asemenea, regulamentul de desfăşurare a examenelor interzice ca un student să dea două examene in aceeaşi zi. Pentru fiecare student se dispune de lista disciplinelor la care doreşte să fie examinat.

Să se descrie construcţia unui graf G care să ofere răspunsul la următoarele două intrebări prin determinarea unor parametri asociaţi (care se vor preciza):

- care e numărul maxim de examene ce se pot organiza in aceeaşi zi ?

- care e numărul minim de zile necesare organizării tuturor examenelor? (**3 puncte**)

Setul de probleme 2

Problema 1. Fie $G = (S, T; E)$ un graf bipartit și $X \in \{S, T\}$. Graful G se numește X-lanț dacă vârfurile mulțimii X pot fi ordonate $X = \{x_1, x_2, \ldots, x_p\}$ (unde $p = |X|$) astfel încât $N_G(x_1) \supseteq N_G(x_2) \supseteq \ldots \supseteq N_G(x_p)$.

a) Demonstrați că G este X-lanț dacă și numai dacă este $\overline{X}$-lanț , unde $\overline{X} = \{S, T\} - \{X\}$. (**2 puncte**)

b) Dacă G (bipartit) este reprezentat cu ajutorul listelor de adiacență , are ordinul n și dimensiunea m, descrieți un algoritm cu timpul $O(n + m)$ care să testeze dacă G este S-lanț . (**2 puncte**)

Problema 2. Un graf G se numește *autocomplementar* dacă este izomorf cu complementul său : $G \simeq \overline{G}$.

a) Demonstrați că un graf autocomplementar este conex si că ordinul său este multiplu de 4 sau multiplu de 4 plus 1.(**2 puncte**)

b) Demonstrați că pentru orice graf G există un graf autocomplementar H astfel incit G este subgraf indus in H. (**2 puncte**)

c) Determinați toate grafurile autocomplementare cu cel mult 7 virfuri. (**2 puncte**)

Problema 3. O echipă de doi programatori L(azy) și T(hinky) primește ca sarcină să determine un drum intre 2 noduri date, care să satisfacă anumite cerințe, intr-un graf G dat, despre care se știe că este rar : $|E(G)| = O(|G|)$. Programatorul L propune ca soluție generarea (cu backtracking) a tuturor drumurilor dintre cele două noduri și selectarea celui convenabil, motivând că intr-un astfel de graf nu pot exista prea multe drumuri intre două noduri fixate (sunt puține muchii și deci puține posibilități de ramificare; de ex., intr-un arbore există exact un drum intre orice două noduri fixate). Programatorul T nu-i de acord și dă următorul contraexemplu: se consideră graful $H = K_2 \times P_{n-1}$ (n un intreg mare); o pereche de virfuri de grad 2 adiacente din H se unește cu un virf nou x, iar cealaltă pereche de virfuri de grad 2 adiacente din H se unește cu un virf nou y; graful obținut, G, are proprietățile din problema de rezolvat și totuși numărul drumurilor de la x la y in G este prea mare. Ajutați-l pe L să înțeleagă contraexemplul, desenind graful G, arătind că este rar și estimind numărul drumurilor de la x la y. (**2 puncte**)

Problema 4. Presupunem că un *turneu* (digraf cu proprietatea că orice 2 virfuri sunt unite exact printr-un arc) are un circuit C de lungime $n \geq 4$.

Arătaţi că pentru orice virf x al lui C se pot determina in timpul $O(n)$, incă două virfuri ale lui C y şi z astfel incit (x, y, z) este un circuit de lungime 3. (**2 puncte**)

Setul de probleme 3

Problema 1. Fie $G = (V, E)$ un graf cu n virfuri, m muchii și cu matricea de adiacență A. Dintre cele 2^m orientări posibile ale muchiilor sale considerăm una oarecare și cu ajutorul ei construim matricea de incidență virf-arc $Q \in \{0, 1, -1\}^{n \times m}$ definită prin :
$(Q)_{ve} = -1$, dacă v este extremit. inițială a arcului e,
$(Q)_{ve} = 1$, dacă v este extremitatea finală a arcului e
$(Q)_{ve} = 0$ in toate celelalte cazuri.
Demonstrați că matricea $A + QQ^T$ este o matrice diagonală și precizați semnificația combinatorie a elementelor ei. (**3 puncte**)

Problema 2. Fie G un graf oarecare și notăm cu $b(G)$ graful obținut din G prin inserarea cite unui nou nod pe fiecare muchie. Demonstrați că $b(G)$ este un graf bipartit. (**2 puncte**)
Demonstrați că G și H sunt izomorfe dacă și numai dacă $b(G)$ este izomorf cu $b(H)$. Deduceți că testarea izomorfismului a 2 grafuri oarecare se reduce polinomial la testarea izomorfismului a 2 grafuri bipartite (**2 puncte**)

Problema 3. Graful *paianjen* cu n virfuri este graful care se obține unind unul din virfurile de grad 1 ale grafului P_3 cu toate virfurile unui graf oarecare cu $n-3$ virfuri, disjunct de P_3 (n este un intreg pozitiv mare).

Dacă G este un graf cu n virfuri reprezentat prin matricea de adiacență, arătați că se poate testa dacă este graf paianjen folosind doar $O(n)$ *probe* ale matricii de adiacență.

(o probă este un acces la un element oarecare al matricii, fără a-l memora explicit pentru utilizări ulterioare). (**4 puncte**)

Problema 4. Asociem unui arbore binar T de ordin n cu rădăcina r un drum P_{3n} orientat procedind astfel: fiecărui nod v al lui T i se asociază trei noduri cu același nume v pe care le desemnăm prin v_1, v_2, v_3; dacă v nu are in T descendent sting, atunci se introduce arcul v_1v_2 in P_{3n}; dacă v nu are in T descendent drept, atunci se introduce arcul v_2v_3 in P_{3n}; dacă descendentul sting al lui v in T este w, atunci se introduc in P_{3n} arcele v_1w_1 și w_3v_2; dacă descendentul drept al lui v in T este w, atunci se introduc in P_{3n} arcele v_2w_1 și w_3v_3.

Dacă se parcurge drumul P_{3n} de la extremitatea inițială r_1 la extremitatea finală r_3 și se listează numele virfurilor in ordinea parcurgerii lor se obține un șir in care numele fiecarui virf al lui T apare exact de trei ori.

Demonstrați că :

dacă din acest șir se reține doar prima apariție a fiecărui nume se obține parcurgerea *pre-order* a arbrelui T;

dacă din acest șir se reține doar a doua apariție a fiecărui nume se obține parcurgerea *in-order* a arbrelui T;

dacă din acest șir se reține doar a treia apariție a fiecărui nume se obține parcurgerea *post-order* a arbrelui T.

(**3 puncte**)

Setul de probleme 3'

Problema 1. Fie $G = (V, E)$ un graf de ordin n și dimensiune m. O ordonare $V = \{v_{i_1}, \ldots, v_{i_n}\}$ a vârfurilor lui G se numește *d-mărginită* dacă în digraful $G^{\mapsto}$, obținut din G prin înlocuirea fiecărei muchii $\{v_{i_j}, v_{i_k}\}$ cu arcul $(v_{i_{\min\{j,k\}}}, v_{i_{\max\{j,k\}}})$, avem $\forall v \in V \quad d^+_{G^{\mapsto}}(v) \leq d$.

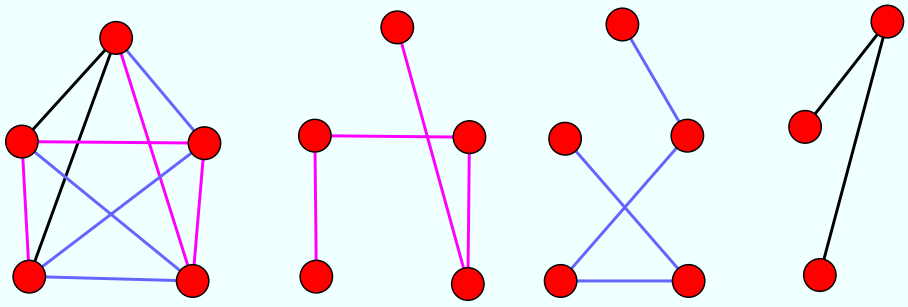
a) Descrieți un algoritm care primind la intrare G reprezentat cu ajutorul listelor de adiacență și $d \in \mathbf{N}^*$, testează în timpul $O(n+m)$ dacă G are o ordonare d-mărginită (se vor argumenta corectitudinea și complexitatea). (**2 puncte**)

b) Utilizați algoritmul de la punctul a) pentru a determina în timpul $O(m \log n)$ parametrul

$o(G) = \min\{d \in \mathbf{N} |\ G$ are o ordonare d-mărginită $\}$. (**2 puncte**)

c) Arătați că orice graf G admite o colorare a vârfurilor cu $o(G) + 1$ culori. (**2 puncte**)

Problema 2. Demonstrați algoritmic că mulțimea muchiilor oricărui graf complet K_n $(n \geq 2)$ poate fi partiționată în $\lceil \frac{n}{2} \rceil$ submulțimi, fiecare dintre acestea reprezentând mulțimea muchiilor unui arbore (subgraf al lui K_n). Exemplu. K_5, $\lceil \frac{5}{2} \rceil = 3$:



(**4 puncte**)

Problema 3. Pentru un graf conex G se execută următorul algoritm:

- Se inițializează o coadă $\mathbf{Q}$ cu graful G.
- Cât timp coada $\mathbf{Q}$ nu-i vidă:
- se extrage în H graful din capul cozii,
- se determină o mulțime de articulație $A \subseteq V(H)$, minimală în raport cu incluziunea (nici o submulțime proprie nu-i mulțime de articulație în H), și dacă $V_1, \ldots, V_k$ ($k \geq 2$) sunt mulțimile de vârfuri ale componentelor conexe ale grafului $H - A$, atunci
- se adaugă la $\mathbf{Q}$ grafurile $[A \cup V_1]_H, \ldots, [A \cup V_k]_H$.

Se observă că dacă graful curent este complet atunci nu se adaugă nimic în coada $\mathbf{Q}$.

a) Arătați că fiecare graf introdus în coadă este conex.(**2 puncte**)
b) Demonstrați că numărul total al grafurilor introduse în coada $\mathbf{Q}$ nu depășește $|G|^2$. (**2 puncte**)

Setul de probleme 3"

Problema 1. Fie $\mathcal{C}$ clasa grafurilor G cu proprietatea că orice arbore dfs al lui G este un drum (pentru orice ordonare a vârfurilor lui G și orice ordonare a listelor de adiacență asociate acestor vârfuri, orice aplicare a unui dfs generează un drum hamiltonian în G).
Demonstrați că

$$\mathcal{C} = \{K_1, K_2\} \cup \cup_{n \geq 3} \{K_n, C_n, K_{n,n}\}.$$

(1+3 puncte)

Problema 2. Fie $D = (V, E)$ un digraf (fără bucle) de ordin n cu mulțimea vârfurilor $V = \{1, 2, \ldots, n\}$. Considerăm următorul algoritm:

1. $SK \leftarrow \emptyset$;
   for $i = 1$ **to** n **do** \\ stânga $\rightarrow$ dreapta
   if ($\nexists j \in SK$ astfel încât $ji \in E$) **then** $SK \leftarrow SK \cup \{i\}$;
2. **for** $i = n$ **to** 1 **do** \\ dreapta $\rightarrow$ stânga
   if $i \in SK \wedge (\exists j \in SK$ astfel încât $ji \in E)$ **then** $SK \leftarrow SK \setminus \{i\}$;
3. **output** SK.

Demonstraţi că SK este un *seminucleu* în D: SK este nevidă, stabilă în $G[D]$ (graful suport al digrafului D) şi orice vârf din $v \in V \setminus SK$ e accesibil în D, dintr-un vârf al lui SK, pe un drum de lungime cel mult 2.
Indicaţi structurile de date şi modul de folosire a acestora pentru o implementare a algoritmului de mai sus în timpul $\mathcal{O}(n+m)$ (m fiind $|E|$).

(2+2 puncte)

Problema 3. Arătaţi că dacă $G = (S, T; E)$ este un graf bipartit cu următoarele proprietăţi:
- $|S| = n; |T| = m$ $(n, m \in \mathbf{N}^*)$;
- $\forall t \in T$ $|N_G(t)| > k > 0$;(pentru un k oarecare — mai mic decât n);
- $\forall t_1, t_2 \in T$ dacă $t_1 \neq t_2$ atunci $N_G(t_1) \neq N_G(t_2)$;
- $\forall t_1, t_2 \in T$ dacă $t_1 \neq t_2$ atunci $|N_G(t_1) \cap N_G(t_2)| = k$,

atunci are loc inegalitatea $m \leq n$. **(2 puncte)**

Problema 4.

Pentru $n \in \mathbf{N}^*$ definim graful $G_n = (V, E)$ astfel:
— $V = \{(i, j) | 1 \leq i \leq n, 1 \leq j \leq n\}$,
— $(i, j)(k, l) \in E$ (pentru două vârfuri (i, j) şi (k, l) distincte din V) dacă şi numai dacă $i = l$ sau $j = k$.

Demonstraţi că G_n este *universal pentru familia arborilor de ordin* n:
oricare ar fi T un arbore de ordin n există $A \subset V$ astfel încât $T \cong [A]_{G_n}$.

(2+2 puncte)

Setul de probleme 4

Problema 1. Prezentați (pe cel mult o pagină) o problemă interesantă din domeniul IT care să necesite rezolvarea eficientă a unei probleme de drum minim intr-un digraf asociat problemei inițiale. (**3 puncte**)

Problema 2. Fie $G = (V, E)$ un graf, $s \in V$ un virf oarecare al lui G iar t un alt virf, accesibil in G printr-un drum din s. O mulțime A de muchii se numește *st-inevitabilă* dacă există $S \subset V$ astfel incit $s \in S$, $t \notin S$ și $A = \{e \in E | e = uv, u \in S\ v \notin S\}$. Arătați că numărul maxim de mulțimi st-inevitabile disjuncte două cite două este egal cu distanța in G de la s la t și că se poate determina o familie de astfel de mulțimi cu ajutorul unui bfs a lui G din s. (**3 puncte**)

Problema 3. Fie $G = (V, E)$ un graf conex și v un virf al său cu proprietatea că $N_G(v) \neq V - \{v\}$. Dacă pentru $A \subset V$ notăm cu $N_G(A) = \cup_{a \in A} N_G(a) - A$, se observă că există mulțimi de virfuri A care satisfac proprietățile : $v \in A$, $[A]_G$ este conex, $N = N_G(A) \neq \emptyset$ și $R = V - (A \cup N) \neq \emptyset$ (de exemplu, $A = \{v\}$).

a) Demonstraţi că dacă se consideră o mulţime A maximală (in raport cu incluziunea) satisfăcind proprietaţile enunţate, atunci orice virf din R este adiacent cu orice virf din N.(**2 puncte**)
b) Dacă, in plus, graful G este $\{C_k\}_{k\geq 4}$-free, atunci mulţimea N de la punctul a) are proprietatea că este clică in graful G.(**2 puncte**)
c) Deduceţi că singurele grafuri $\{C_k\}_{k\geq 4}$-free, regulate şi conexe sunt grafurile complete.(**2 puncte**)

Problema 4. Arătaţi că se poate utiliza o parcurgere dfs pentru a determina un circuit par intr-un graf 3-regulat oarecare. (**2 puncte**)

Setul de probleme 5

Problema 1. Să se arate că un graf G este bipartit dacă și numai dacă orice subgraf indus H al lui G satisface proprietatea $2\alpha(H) \geq |H|$ (**3 puncte**)

Problema 2. Demonstarați că intr-un graf bipartit G cu n virfuri și m muchii avem inegalitatea $4m \leq n^2$. (**2 puncte**)

Descrieți un algoritm care să testeze dacă un graf cu n virfuri și m muchii este complementarul unui graf bipartit in timpul $O(n+m)$ (**3 puncte**)

Problema 3. Arătați că orice graf G cu m muchii are un graf partial H bipartit si cu cel puțin $\frac{m}{2}$ muchii.(**3 puncte**)

Problema 4. Demonstrați că in orice graf conex $G = (V, E)$ există o mulțime stabilă S astfel incit graful bipartit $H = (S, V-S; E')$ este conex, unde $E' = E - \mathcal{P}_2(V-S)$. Deduceți că $\alpha(G) \geq \frac{|G|-1}{\Delta(G)}$ pentru orice graf conex G. (**3 puncte**)

Setul de probleme 6

Problema 1. Pentru $d \in N^*$ se consideră graful $G_d = \underbrace{K_2 \times K_2 \times ... \times K_2}_{d \ factori}$.

Să se determine ordinul, dimensiunea și diametrul lui G_d. (**2 puncte**)

Să se arate că G_d este bipartit și să se determine $\alpha(G_d)$. (**2 puncte**)

Problema 2. Un graf cu cel puțin trei virfuri se numește *confidențial conex* dacă pentru orice trei virfuri distincte a, b, c ale grafului există un drum de la a la b astfel incit niciunul dintre virfurile interne ale acestui drum (dacă există astfel de virfuri) nu este c sau un vecin al lui c. Un exemplu banal de graf confidențial conex este graful K_n cu $n \geq 3$.

Demonstrați că un graf conex $G = (V, E)$, cu cel puțin trei virfuri și care nu-i complet, este confidențial conex dacă și numai dacă au loc următoarele două condiții :

1. Pentru orice virf v mulțimea $\overline{N}(v) = \{w \in V | w \neq v, \ vw \notin E\}$ este nevidă și induce un graf conex.

2. Orice muchie a grafului este conținută intr-un C_4 indus in graf sau este muchia din mijlocul unui P_4 indus in graf.

(**4 puncte**)

Problema 3. In Problema 2-SAT se dau : o mulţime de variabile boolene $U = \{x_1, x_2, ..., x_n\}$ şi o mulţime de clauze $C = \{C_1, C_2, ..., C_m\}$, unde fiecare clauză C_i este disjuncţia a doi literali $C_i = v_i \vee w_i$, literalii reprezentind variabile sau negaţiile acestora. Problemei i se asociază un digraf G cu $V(G) = \{x_1, x_2, ., x_n, \overline{x_1}, \overline{x_2}, ..., \overline{x_n}\}$ (adică toţi literalii posibili) şi in care pentru fiecare clauză $C_i = v_i \vee w_i$ se adaugă arcele $\overline{v_i}w_i$ şi $\overline{w_i}v_i$ (folosind, evident, convenţia referitoare la dubla negare). Demonstraţi că există o atribuire a valorilor de adevăr şi fals pentru variabilele booleene, astfel incit fiecare clauză să fie adevărată, dacă şi numai dacă digraful G are poprietatea că pentru orice $i \in \{1, ..., n\}$ $\overline{x_i}$ şi x_i aparţin la componente tari conexe diferite. (**4 puncte**)
Argumentaţi complexitatea timp de $O(n + m)$ pentru testarea proprietăţii de mai sus. (**2 puncte**)

Setul de probleme 7

Problema 1. Gossip Problem. Intr-un grup de n "doamne", fiecare cunoaşte o parte dintr-o birfă pe care celelalte n-o cunosc. Ele comunică prin telefon şi orice apel telefonic intre orice două doamne are ca efect faptul că fiecare din ele va afla tot ce cunoaşte cealaltă.

(a) Descrieţi o schemă de a da telefoanele astfel incit intr-un numar minim $f(n)$ de apeluri telefonice, fiecare "doamnă" va afla tot ce ştiu celelalte.

Indicaţie: Arătaţi că $f(2) = 1, f(3) = 3, f(4) = 4$ şi pentru $n > 4$ $f(n) = 2n - 4$ (uşor, indicind scheme de telefonare cu aceste numere de apeluri). Incercaţi sa argumentaţi că $2n - 4$ este chiar numărul minim.

(**2 puncte** pentru descrierea schemei, **1 punct** pentru demonstrarea optimalităţii)

(b) Modelaţi problema in limbajul teoriei grafurilor: schemei de telefonare ii va corespunde un şir de muchii iar cunoaşterea comună se va exprima printr-o conditiţie referitoare la existenţa unor drumuri speciale cu elemente din şirul considerat **(1 punct)**

Problema 2. Fie D un digraf și două funcții definite pe multimea arcelor sale, $a : E(D) \longrightarrow R_+$ și $b : E(D) \longrightarrow R_+^*$. Descrieți un algoritm eficient pentru determinarea unui circuit C^* in D astfel incit
$\frac{a(C^*)}{b(C^*)} = \min\{\frac{a(C)}{b(C)}; C \text{ circuit in } D \}$
(**4 puncte**)

Problema 3. Fie $A_1, A_2, ..., A_n$ submulțimi distincte ale unei mulțimi de n elemente S. Demonstrați că există un element x in mulțimea S astfel incit $A_1 - \{x\}, A_2 - \{x\}, ..., A_n - \{x\}$ să fie și ele distincte. (**2 puncte**)

Problema 4. Fie G un graf și $c : E(G) \longrightarrow R_+$ o funcție de capacitate a muchiilor. Oricărui drum din graf cu măcar o muchie i se asociază *locul ingust* ca fiind muchia sa de capacitate minimă . Descrieți un algoritm eficient care să determine pentru două virfuri s și t distincte ale grafului drumul cu locul ingust cel mai mare (dintre toate drumurile de la s la t in graful G).
(**4 puncte**)

Setul de probleme 7'

Problema 1. Fie $G = (V, E)$ un digraf de ordin n, $a : E \to \mathbf{R}_+$ o funcţie de cost nenegativă, şi $s \neq t$ două vârfuri fixate. Pentru rezolvarea problemei **P1** (a determinării unui drum de cost a minim de la s la t în G) se propune următorul algoritm:

1. **for** each $i \in V$ **do** $p_i \leftarrow 0$;
   $i \leftarrow s$;*înainte*$(s) \leftarrow s$;
2. **while** $i \neq t$ **do**
   if $\exists j \in V$ astfel încât $p_i - p_j = a_{ij}$ **then**
   { *înainte*$(j) \leftarrow i$; $i \leftarrow j$; }
   else
   { $p_i \leftarrow \min_{ij \in E}(a_{ij} + p_j)$; $i \leftarrow$*înainte*(i) };
3. Costul unui drum de cost minim de la s la t este $p_s - p_t$ şi un drum de cost minim se obţine din:
   t, *înainte*(t), *înainte*(*înainte*(t)), ..., s.

a)Demonstraţi că dacă pasul 2 se termină atunci afirmaţiile din pasul 3 sunt corecte.(**2 puncte**)
b)Stabiliţi complexitatea timp a algoritmului(**2 puncte**)

Problema 2. Fie $T = (V, E)$ un arbore şi $w : V \to \mathbf{R}_+$ o funcţie de pondere nenegativă. Pentru orice subarbore T' al lui T se defineşte ponderea sa, $w(T')$, ca fiind suma ponderilor vârfurilor sale.

Arătaţi că există un vârf $v_0 \in V$ astfel încât nici unul din subarborii lui $T - v_0$ nu are ponderea mai mare decât $\frac{1}{2}w(T)$.(**1 punct**)
Descrieţi un algoritm cu timpul $O(|V|)$ pentru găsirea lui v_0. (**2 puncte**)

Problema 3. Dacă G și H sunt două grafuri, notația $G \to H$ semnifică faptul că există $f : V(G) \to V(H)$ astfel încât $\forall\ uv \in E(G)$ avem că $f(u)f(v) \in E(H)$ (există un morfism de grafuri de la G la H).
Justificați corectitudinea unui algoritm care să răspundă în timpul $O(1)$ la întrebarea: *"Are loc $C_n \to C_m$? "* ($n, m \in \mathbf{N}, n, m \geq 3$; C_k este graful circuit de ordin k). (**3 puncte**)

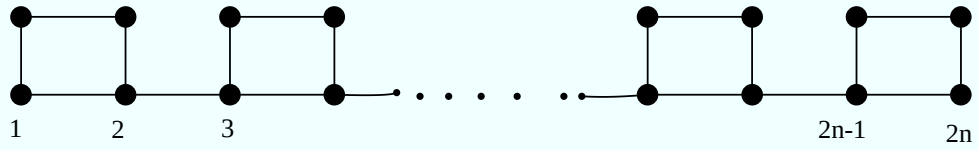
Problema 4. Dacă H este un graf, atunci $q(H)$ notează numărul componentelor conexe de ordin impar ale lui H, iar $\nu(H)$ cardinalul maxim al unui cuplaj al lui H . Demonstrați că pentru orice graf G are loc relația:

$$\max_{S \subset V(G)} (q(G - S) - |S|) = |V(G)| - 2\nu(G) \ .$$

(Se presupune cunoscută teorema lui Tutte)(**4 puncte**)

Setul de probleme 7”

Problema 1. Determinaţi numărul cuplajelor perfecte ale grafului:



(3 puncte)

Problema 2.
a) Fie $D = (V, E)$ un digraf aciclic cu n vârfuri şi m arce şi $A, B \subset V$ două mulţimi disjuncte, stabile în $G(D)$ (graful suport al digrafului). Fie $d(A, B) := \min\{d(a, b) | a \in A, b \in B\}$ ($d(x, y)$=distanţa în D de la x la y = lungimea celui mai scurt drum dintre x şi y, dacă acesta există). Descrieţi un algoritm de complexitate $\mathcal{O}(n + m)$ pentru aflarea unei mulţimi maximale $\mathcal{P}$ de drumuri disjuncte (cu mulţimile de vârfuri disjuncte) de la A la B, fiecare de lungime $d(A, B)$ (maximalitatea lui $\mathcal{P}$ este în raport cu incluziunea, adică nu mai există un alt drum de la A la B care să aibă lungimea $d(A, B)$ şi să fie disjunct de orice drum din $\mathcal{P}$).
b) Arătaţi cum poate fi folosit algoritmul de la a) pentru implementarea algoritmului lui Hopcroft & Karp de aflare a unui cuplaj de cardinal maxim într-un graf bipartit.

(3+2 puncte)

Problema 3. Fie $D = (V, E)$ un digraf cu mulţimea de vârfuri $V = \{1, \ldots, n\}$ şi mulţimea arcelor $E = \{e_1, \ldots, e_m\}$.

Fie $A = (a_{ij}) \in \mathcal{M}_{n \times m}(\{-1, 0, 1\})$ matricea de incidenţă a lui D (dacă arcul e_j iese din i atunci $a_{ij} = 1$, dacă arcul e_j intră în i atunci $a_{ij} = -1$, altfel $a_{ij} = 0$). Arătaţi că pentru orice submatrice pătrată B a lui A are loc:

$$det(B) \in \{-1, 0, 1\}.$$

(2 puncte)

Problema 4. Într-un graf fără vârfuri izolate se construieşte un drum P astfel: se pleacă dintr-un vârf oarecare de start şi apoi, din vârful curent în care ne aflăm, alegem un vecin diferit de vârfurile deja vizitate. Atunci când nu mai este posibilă nici o alegere, construcţia lui P se încheie. Evident, lungimea drumului P este cel puţin 1 şi ea depinde de structura grafului şi de alegerile făcute. Proprietarul grafului solicită o plată pentru folosirea acestuia în procesul de construcţie a drumului P. Această plată se poate face înaintea fiecărei alegeri şi, dacă se plăteşte 1 RON se obţine dreptul de a face această alegere, iar dacă se plătesc $T >> 1$ RONi atunci se obţine dreptul de a face gratuit toate alegerile următoare. După terminarea construcţiei se poate compara suma plătită, $Apriori(P)$, cu cea care s-ar fi făcut dacă s-ar fi cunoscut drumul P, notată $Posteriori(P)$. Găsiţi o strategie de plată astfel încât pentru orice graf şi orice drum construit P să avem $Apriori(P) \leq (2 - 1/T)Posteriori(P)$.

(4 puncte)

Setul de probleme 8

Problema 1. Fie G un graf conex şi o funcţie de cost $c : E(G) \longrightarrow R$. Vom numi *tăietură* orice mulţime A de muchii ale lui G cu proprietatea că există o bipartiţie $(S, V(G) - S)$ a mulţimii virfurilor lui G astfel incit A este mulţimea muchiilor lui G cu extremităţile in clase diferite ale bipartiţiei.

a) Arătaţi că dacă funcţia de cost are proprietatea că orice tăietură are o unică muchie de cost minim, atunci există un unic arbore parţial de cost minim. **(2 puncte)**

b) Deduceţi că, dacă funcţia de cost c este injectivă,atunci G are un unic arbore parţial de cost minim. **(1 punct)**

c) Sunt adevărate reciprocele afirmaţiilor a) si b) ? **(1 punct)**

Problema 2. Considerăm o numerotare fixată a celor $m > 0$ muchii ale unui graf conex $G = (V, E)$ de ordin n. Pentru orice submulţime de muchii A considerăm $x^A \in GF^m$ vectorul m-dimensional cu elemente 0,1 definit prin $x_i^A = 1 \Leftrightarrow e_i \in A$ (vect. caracteristic). GF^m este spaţiul vectorial peste corpul GF (cu elem. 0 şi 1, şi operaţiile de adunare şi inmulţire modulo 2).

a) Demonstraţi că mulţimea vectorilor caracteristici ai tuturor tăieturilor grafului G, la care adăugăm şi vectorul nul, formeaza un subspaţiu vectorial X al lui GF^m.**(1 punct)**

b) Demonstraţi că vectorii caracteristici ai mulţimilor muchiilor circuitelor grafului G generează un subspaţiu vectorial U al lui GF^m ortogonal pe X. **(1 punct)**

c) Arătaţi că $dim(X) \geq n-1$ **(1 punct)**

d) Arătaţi că $dim(U) \geq m-n+1$**(1 punct)**

e) Deduceţi că $dim(X) = n-1$ şi că $dim(U) = m-n+1$. **(1 punct)**

Problema 3. Arătaţi că orice arbore cu gradul maxim $t > 0$ are cel puţin t virfuri pendante. (**2 puncte**)

Problema 4. Fie $T = (V, E)$ un arbore cu rădăcina r (un virf oarecare) şi cu $parent(v)$ părintele nodului $v \in V$, $v \neq r$. Un cuplaj M al lui T se numeşte *propriu* dacă orice virf expus v (relativ la M) in T are un frate w (două virfuri sunt fraţi dacă au acelaşi părinte) astfel incit $w\ parent(v) \in M$. a)Demonstraţi că orice cuplaj propriu este de cardinal maxim. **(1 punct)**

b)Arătaţi că pentru orice arbore cu n virfuri, dat prin listele de adiacenţă, se poate construi in timpul $O(n)$ un cuplaj propriu. **(2 puncte)**

Setul de probleme 9

Problema 1. Fie $G = (S, T; E)$ un graf bipartit. Utilizaţi teorema lui Hall pe un graf convenabil pentru a demonstra că pentru orice intreg k, cu $0 \leq k \leq |S|$, graful G are un cuplaj de cardinal cel puţin $|S| - k$ dacă şi numai dacă $\forall A \subseteq S \ |N_G(A)| \geq |A| - k$. **(2 puncte)**

Problema 2. Numim cuplaj *de grad maxim* in graful G, un cuplaj M cu suma gradelor virfurilor saturate de M maximă printre toate cuplajele grafului.

a) Arătaţi că un cuplaj de grad maxim este de cardinal maxim **(2 puncte)**

b) Dem. că există in graful G un cuplaj care saturează toate virfurile de grad maxim dacă şi numai dacă orice cuplaj de grad maxim are aceeaşi proprietate. **(2 puncte)**

c) Demonstraţi că dacă mulţimea virfurilor de grad maxim ale grafului G induce un graf bipartit, atunci G are un cuplaj care saturează toate virfurile de grad maxim. **(2 puncte)**

d) Deduceţi că mulţimea muchiilor unui graf bipartit G poate fi partiţionată in $\Delta(G)$ cuplaje.**(2 puncte)**

Problema 3. Considerām următoarea problemă de decizie:

Instanţă: $G = (V, E)$ un graf, $k \in N$, $b \in N^*$.
Intrebare : Există in G un subgraf H cu b muchii, fără virfuri izolate şi cu ordinul lui H cel puţin k ?

Arătaţi că problema se poate rezolva in timp polinomial. (**2 puncte**)

Problema 4. Arătaţi, utilizind teorema lui Tutte, că orice graf 2-muchie conex 3-regulat are un cuplaj perfect. **(2 puncte)**

Setul de probleme 10

Problema 1. Fie G un graf conex cu n virfuri și $\mathcal{T}_G$ familia arborilor săi parțiali. Se consideră graful $H = (\mathcal{T}_G, E(H))$ unde $T_1T_2 \in E(H) \Longleftrightarrow |E(T_1) \triangle E(T_2)| = 2$.
a) Demonstrați că H este conex și are diametrul cel mult $n-1$. **(2 puncte)**
b)Demonstrați că pentru orice funcție de cost c pe mulțimea muchiilor grafului G, mulțimea arborilor parțiali de cost c minim induce un subgraf conex in H. **(2 puncte)**

Problema 2. Fie $H = (V, E)$ un digraf și $ts \in E$ un arc fixat al său. Se colorează toate arcele lui H cu galben, roșu și verde arbitrar, cu singura condiție ca arcul ts să fie galben (se poate intimpla ca să nu avem arce roșii sau verzi). Demonstrați algoritmic că are loc exact una din urmatoarele situații:
i) există un circuit in graful $G(H)$ (nu se ține seama de orientare) cu arce galbene sau verzi care conține arcul ts și toate arcele galbene ale sale au aceeași orientare.
ii) există o partiție (S, T) a lui V astfel incit $s \in S, t \in T$, toate arcele de la S la T sunt roșii și toate arcele de la T la S sunt roșii sau galbene.
(2 puncte)

Problema 3. Fie $G = (V, E)$ un graf. O mulțime de virfuri $A \subseteq V$ se numește *m-independentă* dacă există un cuplaj M al lui G astfel incit $A \subseteq S(M)$. Demonstrați că dacă A și B sunt mulțimi m-independente și $|A| < |B|$, atunci $\exists b \in B - A \quad : \quad A \cup \{b\}$ este m-independentă (mulțimile m-independente maximale au același cardinal).(**4 puncte**)

Problema 4. Cuplaje stabile in grafuri bipartite
Fie graful complet bipartit $K_{n,n} = (B, F; E)$, unde $B = \{b_1, b_2, ..., b_n\}$ și $F = \{f_1, f_2, ..., f_n\}$. Dacă M este un cuplaj perfect in $K_{n,n}$ (fiecare b este cuplat cu exact un f), vom folosi notația : $b_i f_j \in M \iff f_j = M(b_i) \iff b_i = M(f_j)$.
Vom presupune că
$\forall b \in B$ are o ordonare a preferințelor sale pe F :
$f_{i_1} <_b f_{i_2} <_b ... <_b f_{i_n}$ și
$\forall f \in F$ are o ordonare a preferințelor sale pe B :
$b_{i_1} <_f b_{i_2} <_f ... <_f b_{i_n}$.
Un cuplaj perfect M al lui $K_{n,n}$ se numește *stabil* dacă :

$\forall b \in B$ dacă $f <_b M(b)$, atunci $M(f) <_f b$ și, de asemenea,
$\forall f \in F$ dacă $b <_f M(f)$, atunci $M(b) <_b f$.
Să se arate că pentru orice ordonări ale preferințelor există un cuplaj stabil și să se construiască unul in $\mathcal{O}(n^3)$. **(4 puncte)**

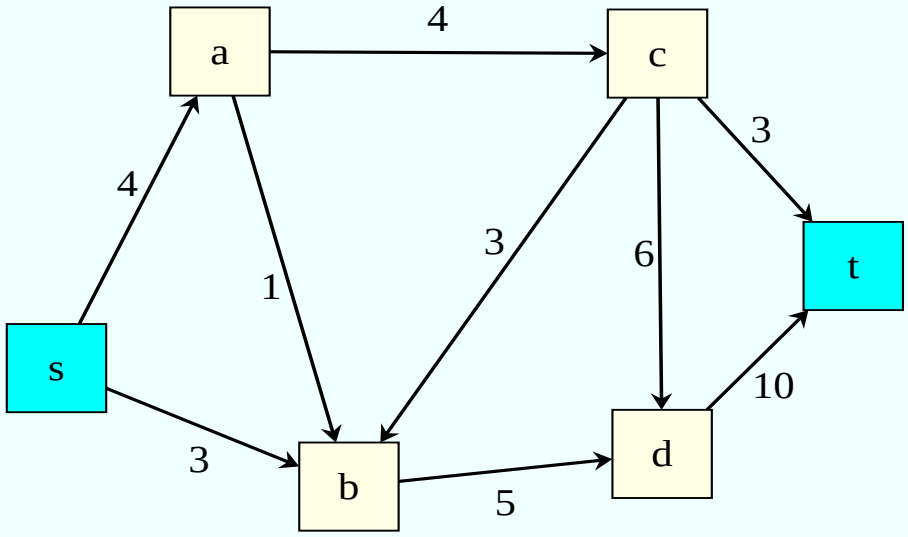
Setul de probleme 11

Problema 1. Se dispune de un algoritm care primind la intrare un graf G și o funcție de pondere nenegativă pe mulțimea muchiilor acestuia, returnează un cuplaj perfect in graful G de pondere minimă (printre toate cuplajele perfecte ale grafului; dacă G nu are cuplaj perfect se anunță acest lucru). Arătați căse poate utiliza acest algoritm pentru determinarea eficientă a cuplajului de cardinal maxim intr-un graf oarecare. **(3 puncte)**

Problema 2. Arătați că se poate determina, intr-o matrice cu elemente 0 și 1 dată, o mulțime de cardinal maxim de elemente egale cu 0 și care să nu se găsească pe aceeași linie sau coloană, cu ajutorul unui algoritm de flux maxim (pe o rețea convenabil definită). **(3 puncte)**

Problema 3. Digraful $G = (V, E)$ descrie topologia interconectarii intr-o rețea de procesoare. Pentru fiecare procesor $v \in V$ se cunoaște incărcarea sa $load(v) \in R^+$. Se cere să se determine (cu ajutorul unei probleme de flux maxim) un plan de *echilibrare statică* a incărcării procesoarelor : se va indica pentru fiecare procesor ce cantitate de incărcare va trimite și la ce procesor astfel incit, in final, toate procesoarele să aibă aceeași incărcare. (**4 puncte**)

Problema 4. Să se determine fluxul de valoare maximă in reteaua din figura de mai jos (explicind funcţionarea algoritmului lui Edmonds-Karp):



(Etichetele arcelor reprezinta capacităţile)

(4 puncte)

Setul de probleme 11'

Problema 1. Considerăm următoarele probleme de decizie:

PERF *Instanță*: G un graf.
Intrebare: Are G un cuplaj perfect?

3PERF *Instanță*: G un graf cu gradul fiecărui vârf ≤ 3.
Intrebare: Are G un cuplaj perfect?

Demonstrați că **PERF** se reduce polinomial la **3PERF** (**4 puncte**)

Problema 2. Demonstrați că nu există nici o permutare $e_1, e_2, \ldots, e_{10}$ a muchiilor grafului complet K_5, astfel încât pentru orice $i \in \{1, \ldots, 9\}$ muchiile e_i și e_{i+1} nu sunt adiacente în K_5 și, de asemenea, e_1 și e_{10} nu sunt adiacente în K_5. (**2 puncte**)

Problema 3. Un organizator al unei conferinţe trebuie să asigure feţe de masă (curate) pentru fiecare din cele D zile cât durează conferinţa.
Se cunoaşte numărul M_i al meselor de care e nevoie în ziua i a conferinţei ($i = 1, D$). Se consideră că toate cele M_i feţe de masă se murdăresc la sfârşitul zilei i ($i = 1, D$).
Organizatorul are de ales între a cumpăra feţe de masă noi, la preţul unitar p, sau, în dimineaţa zilei i, să trimită la curăţat feţe de masă murdare (din zilele precedente; $i \geq 2$). Curăţătoria are două tipuri de servicii: *serviciul rapid*, prin care se returnează feţele de masă curate la începutul zilei $i + 1$ la un cost unitar c_1, şi *serviciul lent* prin care returnează feţele de masă curate la începutul zilei $i + 2$ la un cost unitar c_2. Desigur, $p > c_1 > c_2$.
Problema pe care şi-o pune organizatorul este de a face o planificare a modului de cumpărare şi trimitere la curăţătorie a feţelor de masă, astfel încât să satisfacă toate cererile pe durata conferinţei, la un preţ minim.
(Se presupune că nu există feţe de mese în stoc, la începutul conferinţei, şi că valoarea acestora după terminarea conferinţei e neglijabilă).

Să se formuleze problema organizatorului ca o problemă de flux de cost minim (justificare). (**4 puncte**)

Problema 4. O euristică naturală pentru colorarea vârfurilor unui graf $G = (V, E)$ este următoarea:

a) Se alege o D-ordonare a lui G, adică o ordonare $V = \{v_{i_1}, v_{i_2}, \ldots, v_{i_n}\}$ astfel încât $d_G(v_{i_1}) \geq d_G(v_{i_2}) \geq \ldots \geq d_G(v_{i_n})$.

b) Se colorează *greedy* vârfurile: lui v_{i_1} i se atribuie culoarea 1 şi apoi pentru fiecare vârf v_{i_j}, cu $j = 2, \ldots, n$, se atribuie cea mai mică culoare posibilă (cel mai mic număr natural p cu proprietatea că nu a fost atribuit drept culoare unuia dintre vecinii săi deja coloraţi).

Considerām următoarea problemă de decizie:

3GCOL *Instanţă*: G un graf.
Intrebare: Există o D-ordonare a vârfurilor lui G astfel încât euristica de mai sus dă o 3-colorare a lui G ?

Demonstraţi că problema

3COL *Instanţă*: G un graf.
Intrebare: Admite G o 3-colorare ?

se reduce polinomial la **3GCOL**. (**4 puncte**)

Setul de probleme 11”

Problema 1. Fie $R = (G, s, t, c)$ o rețea (G digraful suport, $s \in V(G)$ intrarea, $t \in V(G)$, $t \neq s$ ieșirea și $c : E(G) \to \mathbf{R}_+$ funcția de capacitate). Presupunem (fără a restrânge generalitatea !) că st și ts nu sunt arce în G. Se dispune și de o funcție *de mărginire inferioară* $m : E(G) \to \mathbf{R}_+$, satisfăcând $m(e) \leq c(e)$ pe orice arc e al lui G. Numim *flux legal în* R orice flux x în R cu proprietatea că $x(e) \geq m(e)\ \forall e \in E(G)$.

a) Demonstrați că pentru orice flux legal x și orice secțiune (S, T) în R are loc

$$v(x) \leq \sum_{i \in S, j \in T, ij \in E(G)} c(ij) - \sum_{i \in S, j \in T, ji \in E(G)} m(ji).$$

b) Se construiește din R rețeaua $\overline{R}$ astfel:

- se adaugă la G o intrare nouă $\overline{s}$ și o ieșire nouă $\overline{t}$;
- pentru $\forall v \in V(G)$ se adaugă arcul $\overline{s}v$ de capacitate $\overline{c}(\overline{s}v) = \sum_{uv \in E(G)} m(uv)$;
- pentru $\forall v \in V(G)$ se adaugă arcul $v\overline{t}$ de capacitate $\overline{c}(v\overline{t}) = \sum_{vu \in E(G)} m(vu)$;
- se adaugă arcele st și ts de capacitate $\overline{c}(st) = \overline{c}(ts) = \infty$;
- se definește $\overline{c}$ pe arcele ij ale lui G ca fiind $\overline{c}(ij) = c(ij) - m(ij)$.

Demonstraţi că există un flux legal în reţeaua R dacă şi numai dacă există un flux de valoare $M = \sum_{e \in E(G)} m(e)$ în reţeaua $\overline{R} = (\overline{G}, \overline{s}, \overline{t}, \overline{c})$ ($\overline{G}$ este digraful construit mai sus, $\overline{c}$ este funcţia de capacitate definită mai sus).

c) Utilizând un flux legal de start (care se poate obţine ca la b)), indicaţi cum se poate adapta algoritmul lui Ford & Fulkerson pentru a obţine un flux legal de valoare maximă într-o reţea în care pe fiecare arc este precizată capacitatea şi marginea inferioară.

(2+2+2 puncte)

Problema 2. Dacă H este un graf conex, $A \subseteq V(H)$ o mulţime nevidă de vârfuri ale sale şi $w : E(H) \to \mathcal{R}_+$, atunci se numeşte *arbore Steiner* corespunzător tripletei (H, A, w) un arbore $T(H, A, w) = (V_T, E_T)$, subgraf al lui H, cu proprietatea că $A \subseteq V_T$ şi suma costurilor muchiilor sale, $s[T(H, A, w)] = \sum_{e \in E_T} w(e)$, este minimă printre toţi arborii subgrafuri ale lui H care conţin A.

a) Justificaţi că determinarea lui $T(H, A, w)$ se poate face în timp polinomial pentru cazul când $A = V(H)$ sau $|A| \leq 2$.

b) Fie $G = (V, E)$ un graf conex cu mulţimea de vârfuri $V = \{1, \ldots, n\}$ şi $A \subseteq V$. Pe mulţimea muchiilor lui G este dată o funcţie de cost $c : E \to \mathcal{R}_+$.
Considerăm şi graful complet K_n cu mulţimea de vârfuri V şi cu funcţia de cost $\overline{c} : E(K_n) \to \mathcal{R}_+$ dată de $\overline{c}(ij) = \min_{[\ P \text{ drum în } G \text{ de la } i \text{ la } j\ ]} c(P)$ pentru orice $ij \in E(K_n)$.
Demonstraţi că $s[T(G, A, c)] = s[T(K_n, A, \overline{c})]$ şi că din orice arbore Steiner $T(K_n, A, \overline{c})$ se poate construi un arbore Steiner $T(G, A, c)$.

c) Arătaţi că există un arbore Steiner $T(K_n, A, \overline{c})$ cu proprietatea că vârfurile sale care nu-s din A au gradul cel puţin 3. Deduceţi (folosind această proprietate) că există întotdeauna un arbore Steiner $T(K_n, A, \overline{c})$ cu cel mult $2|A| - 2$ vârfuri.

((1+1)+(1+2)+(2+1) puncte)

Setul de probleme 12

Problema 1. Fie v valoarea fluxului maxim in reţeaua $R = (G, c, s, t)$. Demonstraţi că există k st-drumuri in G, $P_1, ..., P_k$ $(0 \leq k \leq |E(G)|)$, şi numerele reale nenegative $v_1, ..., v_k$, astfel incit $x : E(G) \longrightarrow \mathbf{R}$, definit pentru orice arc ij prin $x_{ij} = 0 + \Sigma_{t:ij \in P_t} v_t$, este flux in R de valoare maximă v. **(4 puncte)**

Problema 2. Numim *GP-descompunere* a grafului graful complet K_n orice mulţime $\mathcal{A} = \{B_1, ..., B_{k(\mathcal{A})}\}$, unde : fiecare B_i este un subgraf bipartit complet al lui K_n, orice două grafuri B_i si B_j au mulţimile de muchii disjuncte şi $\cup_{i=1,k(\mathcal{A})} E(B_i) = E(K_n)$. Arătaţi că orice GP-descompunere $\mathcal{A}$ a lui K_n satisfce inegalitatea $k(\mathcal{A}) \geq n - 1$. **(4 puncte)**

Problema 3. Fie $G = (V, E)$ un graf şi $f : V \longrightarrow V$ cu proprietatea că $\forall uv \in E : \ f(u)f(v) \in E$. Demonstraţi că $\omega(G) \leq |f(V)|$. Este adevărat că pentru orice graf $G = (V, E)$ există funcţii f cu proprietatea de mai sus şi astfel incit $|f(V)| \leq \Delta(G) + 1$? (**4 puncte**)

Problema 4. Fie $G = (V, E)$ un graf. Numim *partiție specială* orice bipartiție (S, T) a lui V astfel incit subgraful indus de T in G este neconex și subgraful indus de S in complementarul grafului G este neconex.
Arătați că graful circuit C_n ($n \geq 3$) nu are partiții speciale.
Descrieți un algoritm polinomial care să testeze dacă un graf dat are partiții speciale. **(2 puncte)**