Capitolul 8
PROBLEME DE DESCOMPUNERI ÎN GRAFURI

8.1. Tipuri de descompuneri în grafuri

În acest paragraf ne referim la evoluția descompunerii în grafuri, din care prezentăm câteva din rezultatele, deja obținute, despre tipuri de descompuneri în grafuri şi anume: descompunerea în care o anumită caracteristică numerică a grafului are proprietatea de aditivitate, descompunerea în care legea de adiacență între submulțimile partiției este cunoscută, descompunerea în funcție de operația de compoziție, G-descompunerea şi în final, descompunerea substituție şi partiționarea vârfurilor.

Fie G un graf cu mulțimea vârfurilor V(G) = V şi mulțimea muchiilor E(O}= E.

În teoria grafurilor, o clasă foarte largă de probleme se referă la descompunerea (partiția) mulțimii vârfurilor V în submulțimile $X_i, i \in I$, astfel încât proprietăți de următoarele tipuri au loc (Olaru, Antohe):

1) Subgraful indus $[X_i]\,(i \in I)$ trebuie să aibă proprietăți prescrise; de exemplu, problema colorării grafurilor în care X_i trebuie să fie mulțimi stabile şi problema acoperirii grafurilor cu clici;

2) Descompunerea în care o anumită caracteristică numerică ϕ a grafului G trebuie să aibă proprietatea de aditivitate, adică relația $\sum_{i\in I} \phi([X_i]_G) = \phi(G)$ are loc;

3) Legea de adiacență între submulțimile $X_i, X_j, i \neq j$, trebuie să fie cunoscută. Apare ca o problemă de descompunere referitoare la o operație de un tip oarecare, care este, esențial, legea de adiacență; o astfel de descompunere este numită descompunere în funcție de un tip de operație.

Apar de asemenea combinații ale tipului 1) şi 3); în astfel de descompuneri, subgrafurile $[X_i]$ trebuie să satisfacă nişte condiții (de indecompozabilitate, coindecompozabilitate). Există de asemenea probleme care sunt combinații ale tipurilor 1) şi 2) şi acestea sunt în acelaşi timp impuse fiecărui subgraf. Aici este inclusă problema grafurilor perfecte.

8.1.1. Descompunerea de tip 2

Rezultatele următoare au fost obținute de Olaru.

Propoziția 1.

Orice graf care admite o clică drept cutset este α - decompozabil.

Propoziția 2.

Grafurile α - critice nu au cutset care să fie clică.

Propoziția 3.

Dacă G este un graf α - decompozabil atunci există o α - descompunere în componente α - indecompozabile care sunt tare stabile.

Teorema 1.

Pentru un graf G cu $\alpha(G) \geq 2$, următoarele afirmații sunt echivalente:

(i) G este minimal imperfect (adică este minimal critic);

(ii) G este minimal tare stabil;

(iii) G este minimal cu: $\chi(G - x) < \chi(G), \forall x \in V(G)$.

{(i) echivalent (ii): E. Olaru, (i) echivalent (iii): W. Wessel }.

Propoziția 4.

Un graf G este perfect dacă şi numai dacă singurele subgrafuri α - indecompozabile ale lui G sunt clici.

Propoziția 5.

Un graf G este perfect α - decompozabil dacă şi numai dacă el conține un graf parțial α - critic şi perfect α - decompozabil.

Propoziția 6.

Un graf G α - critic este perfect α - decompozabil dacă şi numai dacă componentele sale conexe sunt clici.

Remarcăm că orice graf admite cel puțin un graf parțial α - critic. Un criteriu de decompozabilitate este următorul.

Propoziția 7.

Un graf G este α - decompozabil dacă şi numai dacă el are cel puțin un graf parțial α - critic decompozabil.

8.1.2. Descompunerea de tip 3

Descompunerea de tip 3 înseamnă descompunerea mulțimii vârfurilor V în submulțimile $X_i, i \in I$ astfel încât $\forall i, j \in I, i \neq j$, X_i şi X_j sunt total adiacente în G sau $\overline{G}$ (adică $X_i \sim X_j$ în G sau $\overline{G}$).

Acest tip de descompunere coincide cu descompunerea cu respectarea operației graf-adiacență (X-join), introdusă de Sabidussi. O astfel de descompunere se numeşte S-descompunere, iar factorii $X_i, i \in I$, S-componente.

Orice graf admite o S-descompunere în care toate S-componentele au numai un element, numită banală.

Definiție. Un graf G se numeşte S-indecompozabil dacă el admite numai S-descompunerea banală, altfel se numeşte S-decompozabil.

Observăm că un graf decompozabil poate fi prezentat în diverse forme ca S-descompunere de subgrafuri indecompozabile. Este, deci necesar a considera altfel S-descompunerea, proprie (Olaru, Antohe).

Definiție. O S-descompunere în subgrafuri indecompozabile este proprie dacă ea conține un număr minim de S-componente indecompozabile banale.

Teorema 2. (Olaru, Antohe).

Orice graf finit decompozabil admite o S-descompunere proprie, unică până la un izomorfism.

Definiție. Un subgraf [A] a unui graf G este numit coindecompozabil dacă A este omogenă şi este maximală cu această proprietate.

Teorema 3. (Olaru, Antohe).

Orice graf (finit sau nu) admite o unică S-descompunere în care S-componentele sunt coindecompozabile.

8.1.3. Descompunerea în funcție de compoziție

Această descompunere este în funcție de operația booleană numită compoziție, care a fost introdusă de Harary şi investigată de Sabidussi, care se mai numeşte şi produs lexicografic. Reamintim definiția acestei operații. Date fiind două grafuri $G_1 = (V_1,E_1)$ şi $G_2 = (V_2,E_2)$, compoziția, notată $G = G_1[G_2]$ se defineşte astfel: $V(G) = V_1 \times V_2$ şi pentru $\forall u = (u_1, u_2) \in V_1 \times V_2$, $\forall v = (v_1, v_2) \in V_1 \times V_2$, $uv \in E(G)$ dacă şi numai dacă $u_1v_1 \in E_1$ sau ($u_1 = v_1$ şi $u_2v_2 \in E_2$).

Rezultatele următoare se găsesc în (Golumbic, Shamir).

Fie H_0 un graf neorientat cu n vârfuri $v_1,...,v_n$ şi fie $H_1,...,H_n$ n grafuri disjuncte. Graful compoziție $H = H_0[H_1,...,H_n]$ este format astfel.

Pentru i, j = 1, …, n, înlocuim vârful v_i din H_0 cu graful H_i şi luăm fiecare vârf din H_i adiacent la fiecare vârf din H_j ori de câte ori v_i este adiacent lui v_j în H_0.

Formal, pentru $H_i = (V_i,E_i)$ definim H = (V,E) astfel:

$$V = \bigcup_{i \geq 1} V_i$$

şi

$$E = \bigcup_{i \geq 1} E_i \cup \{xy \mid x \in V_i, y \in V_j \text{ şi } v_iv_j \in E_0\}.$$

Mai notăm $E = E_0[E_1,...,E_n]$.

H_0 se numeşte factor extern a lui H, H_i, i = 1, …, n se numesc factori interni ai lui H.

Un graf neorientat G = (V,E), care prin orientarea fiecărei muchii obținem graful orientat (V,F) care satisface următoarea condiție: dacă $ab \in F$ şi $bc \in F$ implică $ac \in F$ $(\forall a, b, c \in V)$, se numeşte graf de comparabilitate (Golumbic, Shamir).

Teorema 4.

Fie $G = G_0[G_1, ..., G_n]$, unde G_i sunt grafuri neorientate disjuncte. Atunci G este graf de comparabilitate dacă şi numai dacă G_i i = 0, l, ..., n este graf de comparabilitate.

Definiție. Un graf se numeşte decompozabil dacă el poate fi exprimat ca o compoziție nebanală de subgrafuri induse; altfel, el se numeşte indecompozabil.

Pentru orice graf există compoziția banală: $G = K_1[G]$.

Formal, G = (V,E) este decompozabil dacă există o partiție $V = V_1 + ... + V_n$ a vârfurilor în submulțimi nevide disjuncte două câte două cu 1 < r < |V| astfel încât $G = G_R[G_{V_1}, ..., G_{V_v}]$ pentru orice mulțime de reprezentanți $R = \{x_1, ..., x_n\}$, x_i din V_i. O astfel de partiție induce o descompunere proprie a lui G.

G are o mulțime partitivă nebanală dacă şi numai dacă G este decompozabil.

8.1.4. G-descompunerea

Un graf orientat este o pereche G = (V,E), unde V este o mulțime nevidă, iar E este o mulțime de perechi ordonate de elemente distincte ale lui V. Elementele lui E se numesc arce.

Pentru un graf orientat G = (V,E), notăm cu $G^{-1} = (V, E^{-1})$, graful invers, unde $E^{-1} = \{(x, y) \mid (y, x) \in E\}$.

Fie G = (V,E) un graf. Un graf orientat H =(V,F) (având aceeaşi mulțime de vârfuri ca şi G) cu proprietatea că $F \cap F^{-1} = \emptyset$ şi $F \cup F^{-1} = E$ se numeşte orientarea grafului G. (H se obține asociind o orientare fiecărei muchii a lui G).

În acest paragraf prezentăm un algoritm pentru G - descompunere.

Fie R o relație binară pe mulțimea muchiilor unui graf neorientat G = (V,E), definită astfel:

abRxy dacă şi numai dacă ori a = x şi $by \notin E$ ori b = y şi $ax \notin E$.

Închiderea reflexivă şi tranzitivă R' a lui R defineşte o relație de echivalență pe E, ale cărei clase de echivalență se numesc clase de implicații. Dacă A este o clasă de implicație atunci $\hat{A} = A \cup A^{-1}$ este închiderea simetrică a lui A.

Pentru G = (V ,E), un graf, o partiție a lui E în k mulțimi: $E = \hat{B}_1 + ... + \hat{B}_k$ se numeşte G-descompunere dacă B_i sunt clase de implicații din graful $G = (V, \hat{B}_i + \hat{B}_{i+1} + ... + \hat{B}_k)$, $\forall i = \overline{1,k}$.

O secvență de muchii $[x_1y_1, ..., x_ny_n]$ se numeşte schemă de descompunere pentru G dacă există o G-descompunere $E = \hat{B}_1 + ... + \hat{B}_k$ astfel încât $x_iy_i \in B_i$, $i = \overline{1,k}$.

Are loc următorul algoritm de descompunere:

```
Input:              Un graf G = (V,E).
Output:       O G-descompunere; Un număr k;
              k mulțimi de muchii din G: B̂1, . . . , B̂k

begin
       fie    E1 := E
              i := 1
loop:
       Alege arbitrar o muchie ei = xiyi ∈ Ei .
       Determină clasele de implicații Bi din Ei conținând xiyi
       Fie Ei+1 := Ei – B̂i
       If Ei+1 = ∅  then
                           Fie k := i
                           stop
                    else
                           Incrementează i cu 1
                           go to loop
       end if

end
```

8.1.5. Descompunerea substituție şi partiționarea vârfurilor

Un graf conex, nenul, care nu conține vârfuri separatoare se numeşte bloc (Behzad, Chartrand, Foster). Un bloc al unui graf G este un subgraf indus al lui G care este el însuşi un bloc şi care este maximal cu respectarea acestei proprietăți.

Operația numită partiționarea vârfurilor este definită în cele ce urmează.

Presupunem că sunt date un graf G = (V,E) şi o partiție inițială P a lui V în blocurile disjuncte $B_1, ..., B_k$. Problema partiționării grafului cere a găsi partiția având puține blocuri a lui V, fie P' = (E_1, ..., E_j), astfel încât:

1) Fiecare E_i este o submulțime a unui B_h;
2) $x, y \in E_i$ implică $N(x) - E_i = N(y) - E_i$.

Condiția a doua poate fi reformulată astfel. Fiecare bloc E_i verifică: pentru fiecare i ori $x \sim E_i$ ori $x \not\sim E_i$ pentru $\forall x \in V - E_i$, adică E_i este modul.

R. McConnell şi Spinrad au descoperit un algoritm O(m+n) pentru partiționarea vârfurilor.

Vârful v separă (splitează) un bloc B ori de cate ori v este utilizat pentru a divide B în blocurile $B \cap N(v)$ şi $B - N(v)$.

Descompunerea modulară (Jamison, Olariu) sau descompunerea substituție (descoperită independent de mai mulți autori printre care R. H. Mohring şi F. J. Radamacher) înseamnă partiționarea unui graf G în subgrafuri, fiecare din ele este un modul într-un subgraf al lui G.

În particular, graful însuşi şi mulțimile cu un singur vârf sunt considerate module.

Algoritmul următor partiționează un graf în O(mlogn) timp. Algoritmul alege un vârf v care ori

i) nu a fost folosit înainte ca element de splitare ori

ii) $v \in B, |B| \leq \frac{|B'|}{2}$, B' care a conținut v ultima dată când v a fost ales ca element de separare ori

iii) este într-un bloc care nu va fi separat în viitor.

Vârful v este folosit să separe fiecare bloc care nu conține v. Separarea cauzată de v poate fi executată în O(|N(v)|).

Pentru a separa pe B, prima dată verificăm dacă v nu este în B; apoi loca1izăm vecinii lui v într-un nou bloc B', care este păstrat "legat" de B până la sfârşitul separării cauzată de v.

Dacă nici un vârf nu satisface criteriul i) sau ii) pentru alegerea elementului de separare, orice vârf din cel mai mare bloc activ C rămas, va satisface criteriul iii). Orice alt vârf v a încercat să separe C, deoarece v a fost separat de C la un moment dat şi este acum într-un bloc care este cel mult de lungimea lui C.

Algoritmul propriu-zis:

```
Fie partiția inițială (B1,...,Bj);
Pentru fiecare bloc Bi execută
       lastused[Bi] = cea mai mare valoare;
       { folositultimadata[Bi] = cea mai mare valoare}
       READY = (B1,...,Bj);
       Cât timp (G este nevid) execută
       Cât timp (READY≠ ∅) execută
              C = orice bloc din READY;
              lastused[C] = |C|;
              pentru fiecare vârf x din C execută
                     separă toate blocurile exceptând C în funcție de
                     adiacența la x;
```

```
            pentru fiecare bloc B separat în B, B2 prin x execută
                  lastused[B2] = lastused[B];
                  dacă  |B| ≤ lastused[B]/2 atunci
                        adaugă B la READY;
                  dacă |B2| ≤ lastused[B2]/2 atunci
                        adaugă B2 la READY;
      C = cel mai mare bloc rămas;
      returnează C ca parte a partiţiei finale;
      foloseşte  fiecare  vârf  din  C  pentru  a  separa  celelalte
blocuri;
      G = G – C.
```

Prezentăm în continuare un algoritm de complexitate O(mlogn) dat de McConnell pentru găsirea descompunerii substituţie a unui graf. Această metoda de descompunere se bazează pe algoritmul de partiţionare de mai sus. Algoritmul lucrează în două faze. Prima fază găseşte o submulţime de module din arborele de descompunere utilizând partiţionarea şi produce o ordine pe vârfuri astfel încât celelalte module vor fi găsite (specificate) prin alegerea diferitelor vârfuri ca elemente de separare.

Descriem procedura de bază. Ori de câte ori procesul de partiţionare returnează o mulţime S de lungime mai mare decât 1, S este modul al lui G. Alegem orice vârf s din S, împărţim S în s şi S – s şi reîncepem procedura de partiţionare. Păstram un arbore a mulţimii partiţiilor care se produc în timpul acestei proceduri. Iniţial, arborele are o rădăcină, s ca descendentul direct stâng şi V – s ca descendentul direct din dreapta. Ori de cate ori o mulţime este separată în vecini şi nevecini, luăm un nou nod intern cu vecinii drept descendentul direct stâng şi nevecinii drept descendentul direct drept. Dacă S a fost specificat (menţionat, găsit) ca a fi un modul, marcăm nodul intern corespunzător lui S.

Notăm că această procedură va găsi modulele care nu conţin un vârf arbitrar ales s, dar nu va găsi modulele conţinând s. Acestea vor fi găsite de faza a doua. Ideea este de a ordona vârfurile astfel încât întotdeauna alegem vârfuri pentru separare pentru faza a doua care nu apar împreună cu s într-un submodul. Numărăm vârfurile din timpul traversării standard postordine, în care mulţimea descendenţilor copilului (descendentului direct) drept dintr-un nod dat va fi mai mare decât a descendenţilor nodului stâng al aceluiaşi copil (descendent direct).

Faza a doua a procedurii este aceeaşi cu prima, excepţie fiind atunci când avem un modul S, întotdeauna alegem un vârf x cu cel mai mare număr din S cu respectarea procedurii de ordonare din prima fază, separând S în x şi S – x şi continuăm ca mai sus. Timpul total luat pentru găsirea celui mai mare număr de vârfuri este O(nlogn), astfel se termină cele două faze în O(mlogn) timp.

Arborele de descompunere este produs prin combinarea modulelor găsite în timpul celor două faze.

Fie un modul M. Daca există un descendent direct c al lui M care nu formează muchii cu nici un alt descendent direct al lui M din G, ştergem c din M şi adăugăm un nod etichetat P cu c drept un descendent direct şi M drept celălalt descendent direct. Similar, dacă există un descendent direct c' al lui M, extremitate a tuturor muchiilor cu toți ceilalți descendenți direcți ai lui M, ştergem c' din M şi adăugăm un nod S cu c' drept un descendent direct şi M drept celălalt descendent direct. Aceasta ia O(n+m) timp. Unim arborii găsiți în fiecare din cele două stagii astfel. Traversăm arborele în ordinea primul în adâncime. Dacă întâlnim un nod serie (care corespunde unui modul M astfel încât $[M]_{\overline{G}}$ este neconex) sau paralel (care corespunde unui modul astfel încât $[M]_G$ este neconex) x, care are un ascendent direct p de acelaşi tip, luăm toți descendenții direcți ai lui x descendenți direcți ai lui p.

În final, combinăm modulele găsite în timpul fiecăreia din cele două faze într-un singur arbore de descompunere, T. Fie T_1 şi T_2 arborii găsiți în timpul celor două faze. Fie toate nodurile lui T_1 şi T_2 în ordine nedescrescătoare a numărului de descendenți direcți vârfuri pendante. Când un modul M este întâlnit, adăugăm M dacă M nu este deja parte a arborelui combinat. Pentru a cunoaşte dacă un modul M cu k descendenți vârfuri pendante este deja în arbore este suficient a verifica dacă cel mai mic număr de vârfuri din M este deja într-un modul de lungime k. Dacă memorăm cel mai mic număr de vârfuri din fiecare modul şi modificăm variabila i pentru un vârf v când v este vârful cu cel mai mic număr dintr-un modul de lungime i, acest pas ia timp constant pe modul. Adăugarea unui modul M la T se face astfel. Orice descendent direct c al lui M se află în T şi pentru orice rădăcină r din arborele curent care conține un descendent direct al lui M, r devine un descendent direct al lui M în arbore. Orice metodă naturală de a marca un arbore, pentru ca un drum să nu fie traversat de două ori, ne permite să combinăm doi arbori în O(n) timp.

La sfârşitul acestui paragraf, prezentăm următorul rezultat (Olariu).

Pentru un graf G următoarele două afirmații sunt echivalente:

(i) G nu are nici un subgraf indus izomorf cu grafurile cu secvența de grade (2,2,2,3,3), (1,1,2,3,3), (2,2,3,3,4,4);

(ii) pentru fiecare subgraf indus H a lui G, cel puțin una din următoarele afirmații sunt adevărate:

(a) H conține o mulțime omogenă;

(b) $\omega(H) \leq 2$.

8.2. Descompunerea slabă a grafurilor

8.2.1. Introducere

Fie $G = (V,E)$ un graf neorientat, fără bucle şi muchii multiple. În diverse probleme de teoria grafurilor, cu precădere în construcţia unor algoritmi de recunoaştere apare frecvent un anumit tip de partiţie a mulţimii vârfurilor în trei clase *A, B, C* astfel încât *A* să inducă graf conex, iar *C* să fie total adiacent cu *B* şi total neadiacent cu *A*. Aşa se întâmplă, de exemplu, cu construcţia cografurilor pornind de la $K_{1,2}$ şi substituind vârfurile cu cografuri. Introducerea noţiunii de descompunere slabă (C. Croitoru) şi studiul proprietăţilor ei ne permite să obţinem alte rezultate de acest tip, cum ar fi: caracterizarea cografurilor cu coarbori (rezultat cunoscut, obţinut de Lerchs, dar pentru care obţinem demonstraţie mai uşoară). De asemenea, caracterizăm grafurile $K_{1,3}$ *–free* şi dăm un algoritm de recunoaştere şi un altul pentru determinarea unui cuplaj de cardinal maxim în această clasă de grafuri. De asemenea, alte proprietăţi se obţin pentru grafuri triangulate, garfuri $\{P_4, C_4\}$*- free*, grafuri paw – free, grafuri confidenţial conexe.

O parte din acest capitol a fost prezentat la ROSYCS 2000 Iaşi.

8.2.2. Descompunerea slabă a unui graf

Înainte de a da definiţia centrală din acest capitol, reamintim că pentru un graf *G=(V,E)* şi $A \subseteq V$, am notat:

$$N_G(A) = \{v \mid v \in V - A, \exists w \in A \text{ si } v \sim w\}$$

$$N_G[A] = A \cup N_G(A)$$

şi

$$\bar{N}_G(A) = V - N_G[A].$$

(Dacă nu sunt posibile confuzii, indicele *G* poate fi omis).

Fie $G_1 = (V_1, E_1)$ şi $G_2 = (V_2, E_2)$, două grafuri oarecare. Notăm cu G_1+G_2, K_2 - adiacenţa (K_2 – join) a grafurilor G_1 şi G_2, adică, graful obţinut din K_2 prin substituţia vârfurilor sale cu G_1 şi G_2 şi orice vârf din G_1 este adiacent cu orice vârf din G_2. Graful G_1+G_2 va fi numit suma grafurilor G_1 cu G_2.

Definiţia 1. *Fie G=(V,E) un graf. O mulţime de vârfuri , A, se numeşte mulţime slabă dacă* $N_G(A) \neq V - A$ *şi subgraful indus de A este conex. Dacă A este mulţime slabă, maximală în raport cu incluziunea, subgraful indus de A se numeşte componentă slabă. Pentru simplificare, componenta slabă G(A), se va nota cu A.*

Denumirea de componentă slabă este justificată de următorul rezultat.

Propoziţia 2.

Orice graf conex şi incomplete G=(V,E) admite o componentă slabă A astfel încât $G(V - A) = G(N(A)) + G(\bar{N}(A))$.

Demonstraţie. Deoarece graful G este incomplet, $\alpha(G) \geq 2$, există vârfurile $x \neq y$, neadiacente. Fie

$$A_0 = \{x\}; B_0 = N(x); C_0 = \overline{N}(x).$$

Avem $y \in C_0$. Dacă $N(x) \sim \overline{N}(x)$ atunci $A = A_0$. Dacă nu, fie $x_1 \in N(x), y_1 \in \overline{N}(x)$ astfel încât $x_1 \aleph y_1$.

Pentru $A_1 = A_0 \cup \{x_1\}, [A_1]$ este conex, deoarece $[A_0]$ este conex şi $x_1 \in N(x)$.

$$N(A_1) = (N(A_0) - \{x_1\}) \cup (N(x_1) \cap C_0). y_1 \in \overline{N}(A_1)$$

(deoarece $x_1 \aleph y_1$ şi $y_1 \in \overline{N}(x)$ şi $y_1 \aleph A_0$).

Deci

$$\overline{N}(A_1) \neq \varnothing.$$

Dacă

$$N(A_1) \sim \overline{N}(A_1)$$

atunci $[V - A_1] = [N(A_1)]_G + [\overline{N}(A_1)]_G$. Presupunem că s-a construit A_i , $B_i=N(A_i)$, $C_i = \overline{N}(A_i)$.

Dacă $B_i \sim C_i$ atunci $A = A_i$. Dacă nu, fie $x_{i+1} \in B_i$ şi $y_{i+1} \in C_i$ cu $x_{i+1} \aleph y_{i+1}$.

Notăm

$$A_{i+1} = A_i \cup \{x_{i+1}\}; B_{i+1} = N(A_{i+1}); C_{i+1} = \overline{N}(A_{i+1}).$$

[A_{i+1}] este conex, deoarece *[A_i]* este conex şi $x_{i+1} \in N(A_i)$. $y_{i+1} \in \overline{N}(A_{i+1})$ (deoarece $x_{i+1} \aleph y_{i+1}$ şi $y_{i+1} \in \overline{N}(A_i)$ şi $y_{i+1} \aleph A_i$).

Deci

$$\overline{N}(A_{i+1}) \neq \varnothing.$$

Dacă $B_{i+1} \sim C_{i+1}$ atunci $A=A_{i+1}$ şi

$$[V - A_{i+1}]_G = [N(A_{i+1})]_G + [\overline{N}(A_{i+1})]_G.$$

Deoarece

$$A_0 \subset A_1 \subset \ldots \subset A_i \subset \ldots \subset V$$

şi $|V| < \infty$ rezultă $\exists p \in \mathbf{N}$ astfel încât $N(A_p) \sim \overline{N}(A_p)$ şi deci $A = A_p$ este componenta slabă cu proprietatea din enunț.

Propoziția 3.

Fie $G = (V,E)$ un graf conex şi incomplet şi $A \subset V$. Atunci A este componentă slabă a lui G dacă şi numai dacă $G(A)$ este conex şi $N(A) \sim \overline{N}(A)$.

Demonstraţie. Presupunem că există $n \in N(A)$ şi $\overline{n} \in \overline{N}(A)$ astfel încât $n\overline{n} \notin E(G)$.

Fie $A' = A \cup \{n\}$ şi $N' = (N(A) - \{n\}) \cup (N(n) \cap \overline{N}(A))$.

$[A']_G$ este conex, $N(A') = N'$ şi $V(G) - (A' \cup N(A')) \supseteq \{\overline{n}\}$. Deci $N(A') \neq V(G) - A'$, contrazicând maximalitatea lui A.

Invers.

Fie *G(A)* conex şi $N(A) \sim \overline{N}(A)$. Arătăm că *G(A)* este componentă slabă. Fie $A' \supset A$, componentă slabă.

$\varnothing \neq A' - A \subseteq N(A)$ (deoarece $A \aleph \overline{N}(A)$ şi $G(A')$ conex).

Fie $n \in A' - A$. Atunci $\overline{N}(A) \subseteq N(n)$. Deci $\overline{N}(A') \neq \varnothing$, contrazicând definiţia componentei slabe.

Definiţia 4. *O partiţie de forma* $(A, N(A), V - N[A])$*, unde A este mulţime slabă o numim descompunere slabă în raport cu A a grafului G. Numim A componentă slabă, N(A) mulţime separatoare minimală, iar V-N[A] mulţime îndepărtată (remote set).*

Propoziţia 5.

Dacă G=(V,E) este un graf conex şi incomplet atunci mulţimea vârfurilor V admite o descompunere slabă (A,B,C) astfel încât G(A) este componentă slabă şi G(V-A)=G(B)+G(C).

{Demonstraţia este cea dată în propoziţia 2}.

Observaţia 6. *Fie G=(V,E) un graf conex şi incomplet. Dacă A este mulţime slabă atunci* $A \neq V$.

{ A mulţime slabă şi G conex rezultă $A \neq V$, altfel $N_G(A) = \varnothing, V - A = \varnothing$, adică $N_G(A) = V - A$}.

Fie G un graf conex şi W_G={*A: A* mulţime slabă în G}.

Observaţia 7. *Fie G=(V,E) un graf conex.* W_G *are cel puţin un element dacă şi numai dacă G nu-i complet.*

{Dacă G nu este complet atunci $\exists\, v \in V(G)$ astfel încât $N_G(A) \neq V - A$. Deci *{v}* este mulţime slabă, adică $\{v\} \in W_G$. Arătăm implicaţia inversă. Presupunem că există A mulţime slabă în G. Atunci $N_G(A) \neq V - A$. Deci $\overline{N}_G(A) \neq \varnothing$. Deci $\exists a \in A \exists b \in \overline{N}_G(A)$ astfel încât $ab \notin E(G)$. Deci G nu este complet.}

Fie $W_G^0 = \{A \mid A \in W_G, A$ maximală în raport cu incluziunea $\}$.

Mai numim componentele slabe ale unui graf G şi frunze. Mulţimea frunzelor o notăm cu L_G. Deci $L_G = W_G^0$.

Remarca 8. *Fie G=(V,E) un graf conex şi incomplet. Dacă* $A \in W_G^0$ *atunci A este cutset (mulţime separatoare) în* $\overline{G}$.

{În $\overline{G} - A, R = \overline{N}(A)$ şi N sunt mulţimi nevide de vârfuri total neadiacente}

Remarca 9. *Fie G=(V,E) un graf conex şi incomplet. Dacă* $A \in W_G^0$ *atunci* $N_G(A)$ *este cutset (mulţime separatoare) în G.*

{În $G - A, R = \overline{N}(A)$ şi A sunt mulţimi nevide de vârfuri total neadiacente}

Demonstraţia dată în propoziţia 3. oferă un algoritm polinomial pentru construirea unei descompuneri slabe pentru un graf conex şi incomplet.

Algoritm de descompunere slabă a unui graf

Input: Un graf conex şi cu cel puţin două vârfuri neadiacente, G=(V,E).

Output: O partiţie V=(A,N,R) astfel încât G(A) conex, N=N(A), $A \aleph R = \overline{N}(A)$.

begin

A := o mulţime arbitrară de vârfuri astfel încât $A \cup N(A) \neq V$

N:=N(A)

$R := V - A \cup N(A)$

while ($\exists n \in N, \exists r \in R$ astfel încât $nr \notin E$) do

$A := A \cup \{n\}$

$N := (N - \{n\}) \cup (N(n) \cap R)$

$R := R - (N(n) \cap R)$

end.

Se observă că $[A]_G$ este conex, $N=N_G(A)$, $R \neq \varnothing$ este un invariant al algoritmului.

Consecinţa 10. *Dacă G este conex şi cu cel puţin două vârfuri neadiacente şi* $A \in W_G^0$ *atunci*

$$\alpha(G) = \max\{\alpha([A]_G) + \alpha(\overline{N}_G(A)), \alpha(N_G(A) \cup A)\}.$$

{Într-adevăr, orice mulţime stabilă de cardinal maxim ori intersectează $\overline{N}_G(A)$ şi atunci are cardinalul $\alpha([A]_G) + \alpha(\overline{N}_G(A))$ ori nu intersectează $\overline{N}_G(A)$ şi atunci are cardinalul $\alpha(N_G[A])$.}

Observaţia 11. *Dacă G este un graf conex cu* $\alpha(G) = 2$ *atunci A este clică şi R este clică, pentru orice descompunere slabă (A,N,R) a lui G cu* $A \in W_G^0$.

$\{2 = \alpha(G) \geq \alpha([A]) + \alpha([R]) \geq 1 + 1 = 2.\}$.

În continuare reamintim următorul rezultat.

Lema 12.

Dacă G este un graf conex, C este cutset, $K_1, K_2, \ldots, K_p$ ($p \geq 2$) sunt componentele conexe ale lui G-C atunci:

(i) $N(K_i) \subseteq C, \forall i = 1,..., p$;

(ii) *C este cutset minimal dacă şi numai dacă* $N(K_i) = C, \forall i = 1,..., p$

În încheierea acestei secțiuni caracterizăm grafurile *G* cu proprietatea $A \sim N$, unde *(A,N,R)* este descompunere slabă.

Fie *G=(V,E)* un graf, $X \subset V$ şi $k \in \mathbf{N}^*$.

Numim *vecinătatea de ordin k a lui X*, mulțimea

$N_G^k(X) = \{ a \mid a \notin X, \exists$ un drum indus în *G, P,* de lungime *k* de la $x \in X$ la *a* astfel încât $V(P) \cap X = \{x\} \}$.

Evident,

$$N'_G(A) = N_G(A).$$

Lema 13.

Fie G conex şi (A,N,R) o descompunere slabă cu $A \in W_G^0$. *Atunci* $N_G^3(R) = \varnothing$ *dacă şi numai dacă* $A \sim N$.

Demonstrație.

Fie $N_G^3(R) = \varnothing$. Presupunem, totuşi, că $\exists a \in A$ şi $n \in N$ cu $a \not\sim n$. Deoarece $[A \cup \{n\}]$ este conex atunci $\exists$ un drum *P* de lungime ≥ 2 de la *a* la *n*. Întrucât $N \sim R$ şi $R \not\sim A$ atunci P', obținut din *P* la care adăugăm muchia $nr(\forall r \in R)$, este un drum indus în *G* de lungime ≥ 3. Obținem că $N_G^3(R) \neq \varnothing$, o contradicție.

Dacă $A \sim N$, având şi $N \sim R$ rezultă că pentru $\forall r \in R$, nu există drum indus de lungime ≥ 3 cu o extremitate în *r* şi restul vârfurilor în *V-R*. Deci $N_G^3(R) = \varnothing$.

8.3. Teorema celor patru culori

Teorema celor patru culori (cunoscută şi sub numele de *teorema de colorare a hărții cu ajutorul a patru culori*) afirmă următoarele: fiind dat un plan separat în regiuni, regiunile pot fi colorate folosind nu mai mult de patru culori, astfel încât două regiuni adiacente nu sunt colorate cu aceeaşi culoare. Două regiuni se numesc *adiacente* doar dacă „împart" un segment de „graniță".

Teorema celor patru culori a fost prima teoremă majoră ce a necesitat computerul pentru a fi demonstrată, iar această demonstrație nu este acceptată de toți matematicienii deoarece ar fi omeneşte imposibil de

demonstrat un astfel de lucru. În ultimă instanță, pentru a da crezare demonstrației, trebuie să se încreadă în corectitudinea compilării şi a execuției hardware.

Istoric

Se ridică problema folosirii a cinci sau mai multe culori.

În anul 1890, Heawood a demonstrat că toate grafurile planare sunt *cinci-colorabile*.

Între 1960 şi 1970 matematicianul german Heinrich Heesch a dezvoltat metode de utilizare a calculatorului în scopul găsirii acelei demonstrații atât râvnite.

Nu mai devreme de anul 1976 s-a demonstrat ipoteza celor patru culori a lui, de către Kenneth Appel şi Wolfgang Haken la Universitatea din Illinois. Au fost asistați, în ceea ce priveşte algoritmica, de John Koch.

Dacă s-ar fi întâmplat ca ipoteza privind cele patru culori să fie falsă, ar fi existat cel puțin o hartă cu cele mai puține regiuni posibile care să necesite cinci culori pentru colorare. Demonstrația a arătat că un astfel de contraexemplu minimal nu poate exista.

Odată cu demonstrația teoremei, au fost elaborați algoritmi eficienți de 4-colorare a hărților, necesitând doar $O\left(n^2\right)$ timp de rulare, unde n reprezintă numărul de noduri. În anul 1996, Neil Robertson, Daniel P. Sanders, Paul Seymour şi Robin Thomas au creat un algoritm cu timpul pătratic.

În anul 2004 Benjamin Werner şi Georges Gonthier au formalizat o demonstrație a teoremei.

Determinarea suficienței folosirii a trei culori în colorarea optimă a unei hărți, este de complexitate NP, ceea ce indică faptul că nu vom avea o soluție „rapidă”. Determinarea posibilității de 4-colorare a unui graf general (posibil ne-planar) are de asemenea complexitatea NP.

Expunere formală în teoria grafurilor

Pentru a expune formal teorema, este mai simplu să o reformulăm în teoria grafurilor. Teorema afirmă că vârfurile fiecărui graf planar pot fi colorate cu ajutorul a cel mult patru culori, astfel ca două vârfuri adiacente oarecare să nu aibă aceeaşi culoare. Sau, pe scurt: „orice graf planar este patru-colorabil”. În cazul acesta, fiecare regiune a hărții este înlocuită cu un vârf, iar două vârfuri sunt conectate prin intermediul unei muchii dacă şi numai dacă regiunile corespunzătoare „împart” un segment de graniță.

Generalizări

S-ar putea pune problema colorării suprafețelor ce nu sunt neapărat plane. Problema corespunzătoare sferei este echivalentă cu cea din plan. Pentru suprafețele închise (orientate sau neorientate), de clasă pozitivă,

numărul maxim p de culori necesare depinde de caracteristica Euler χ a suprafețelor conform formulei:

$$p = \left[\frac{7 + \sqrt{49 - 24\chi}}{2} \right],$$

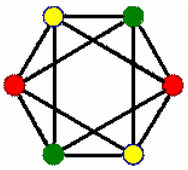
unde parantezele indică partea întreagă a funcției. Singura excepție în ceea ce privește această formulă o reprezintă „sticla lui Klein", a cărei caracteristică a lui Euler are valoarea 0 și necesită 6 culori. Aceasta a fost denumită inițial *ipoteza Heawood*, fiind demonstrată ca și *Teorema colorării hărții* de către Gerhard Ringel și J. T. W. Youngs în anul 1968.

Alternativ, pentru o suprafață orientată, formula poate depinde de parametrul g (parametru ce caracterizează clasa (genul) suprafeței):

$$p = \left[\frac{7 + \sqrt{1 + 48g}}{2} \right].$$

8.3.1. Colorarea grafurilor

În teoria grafurilor, **colorarea grafurilor** constă în atribuirea de „*culori*" vârfurilor unui graf astfel ca oricare două vârfuri adiacente să nu aibă aceeași culoare. În mod analog, **colorarea muchiilor** constă în atribuirea unei culori fiecărei muchii în parte, astfel încât oricare două muchii incidente să nu aibă aceeași culoare, iar **colorarea fețelor** unui graf planar atribuie o culoare fiecărei fețe in parte, ținându-se cont de faptul că oricare două astfel de fețe, ce au o „frontieră" comună, nu pot avea aceeași culoare.



O 3-colorare este suficientă acestui graf, însă, dacă s-ar folosi mai puține culori ar rezulta noduri (vârfuri) adiacente de aceeași culoare.

Găsirea numărului minim de culori necesare colorării unui graf oarecare are dificultate NP

8.3.2. Colorarea vârfurilor

O colorare ce folosește cel mult k culori se numește **k-colorare** și este echivalentă cu problema partiționării mulțimii vârfurilor în k sau mai puține mulțimi independente. Problema colorării grafurilor și-a găsit aplicații, cum ar fi în planificarea calendaristică, înregistrarea alocărilor în compilatoare, distribuția frecvențelor radiourilor mobile, respectiv compatibilitatea modelelor.

8.3.3. Numărul cromatic

Cel mai mic număr de culori necesare colorării unui graf se numeşte **numărul cromatic** χ corespunzător. De exemplu, numărul cromatic al unui graf complet K_n cu n vârfuri (un graf cu o muchie între oricare două vârfuri), este $\chi(K_n) = n$.

Un graf căruia îi poate fi atribuită o k-colorare (corespunzătoare) este ***k*-colorabil**, şi este ***k*-cromatic** dacă numărul său cromatic este chiar k.

Problema găsirii unei colorări minime a unui graf are o dificultate-NP. Problema deciziei corespunzătoare (există o colorare care foloseşte cel mult k culori?) are o complexitate de ordinul NP, reprezentând, la origine, una din cele 21 de probleme ale lui Karp NP-complete. Rămâne NP-completă chiar şi în cazul grafurilor planare de grad cel mult 4, aşa cum a fost demonstrat de către Garey şi Johnson în 1974, chiar dacă în cazul grafurilor planare este trivială (n. demonstrația) pentru k>3 (acest lucru datorându-se teoremei celor patru culori). Există, totuşi, unii algoritmi de aproximare eficienți, care folosesc programarea semidefinită.

Proprietăți ale $\chi(G)$:

1. $\chi(G) = 1$ dacă şi numai dacă G este total neconex.
2. $\chi(G) \geq 3$ dacă şi numai dacă G are un *ciclu impar* (echivalent, dacă G nu este bipartit).
3. $\chi(G) \geq \omega(G)$.
4. $\chi(G) \leq \Delta(G)+1$.
5. $\chi(G) \leq \Delta(G)$ pentru G conex, doar dacă nu cumva G este un *graf complet* sau un ciclu impar (*Teorema lui Brook*).
6. $\chi(G) \leq 4$, pentru orice graf planar G. Acest faimos rezultat este numit *Teorema celor patru* culori.

Aici, $\Delta(G)$ reprezintă gradul maxim, iar $\omega(G)$, numărul clică.

8.3.4. Aspecte algoritmice

Colorarea optimală

Colorarea vârfurilor, în general, este o problemă NP-completă. În loc să cerem cel mai mic număr culori necesare colorării unui graf, putem să ridicăm probleme mult mai simple, cum ar fi „Putem colora graful cu cel mult k culori?”.

Cazul corespunzător lui $k = 2$ este echivalent determinării bipartiției, respectiv a non-bipartiției grafului. Aceasta se poate realiza într-un timp de ordin polinomial. Pentru $k \geq 3$ problema este NP-Completă. Există un rezultat remarcabil, al lui László Lovász, care enunță că este permis să se afirme că numărul cromatic al unui graf perfect implică timpul polinomial.

Algoritmi predefiniți

Algoritmii de colorare pot fi divizați în două categorii:

- ✓ Algoritmi optimali de colorare (de exemplu, *Algoritmul lui Zykov, metoda ramificării şi mărginirii,* etc.)
- ✓ Algoritmi care nu asigură un rezultat cu cele mai puține culori posibile. În această categorie se pot încadra *algoritmii secvențiali*, *algoritmii euristici*, *algoritmi globali aleatori, algoritmi metaeuristici*, respectiv *algoritmii genetici*.

8.3.5. Algoritmul Welsh – Powell

Algoritmul Welsh-Powell de colorare a grafurilor foloseşte o îmbunătățire euristică a unui algoritm greedy. Se demonstrează uşor că o astfel de abordare foloseşte cel mult $\Delta(G)+1$ culori, unde $\Delta(G)$ este gradul maxim al grafului. Algoritmul este următorul:

1. Sortarea nodurilor în ordine descrescătoare a gradului, inițial considerându-se toate nodurile necolorate.
2. Traversarea (Parcurgerea) nodurilor în ordine, atribuindu-i unui nod *culoarea 1* dacă este necolorat şi nu are vecini colorați cu aceeaşi culoare.
3. Se repetă acest proces şi în cazul culorilor 2, 3, etc. până când toate vârfurile vor fi fost colorate.

Acest algoritm nu găseşte neapărat o colorare $\chi(G)$.

8.3.6. Polinomul cromatic

Polinomul cromatic contorizează numărul posibilităților de colorare a unui graf , utilizând un număr prestabilit de culori.

Polinomul cromatic este o funcție $P(G,t)$ ce contorizează numărul *t*-colorărilor lui *G*. Aşa cum indică şi numele, pentru un graf *G* dat funcția este într-adevăr polinomială în *t*.

Polinomul cromatic conține cel puțin la fel de multe informații legate de colorabilitatea lui G ca şi *numărul cromatic*.

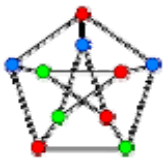
Într-adevăr, χ este cel mai mic întreg pozitiv care nu este rădăcină a polinomului cromatic,

$$\chi(G) = \min\{k : P(G,k) > 0\}$$

A fost folosit pentru prima dată de către Birkhoff şi Lewis în demersul lor împotriva *teoremei celor patru culori*.

A rămas o problemă nerezolvată, în ceea ce priveşte eficiența caracterizării grafurilor ce au acelaşi polinomul cromatic.

Exemple



Graful Petersen are numărul cromatic 3

Polinomul cromatic pentru grafurile

K_3	$t(t\text{-}1)(t\text{-}2)$
Graful complet K_n	$t(t\text{-}1)(t\text{-}2)...(t\text{-}n\text{+}1)$
Arbore cu n noduri	$t(t\text{-}1)^{n\text{-}1}$
Ciclul C_n	$(t\text{-}1)^n + (\text{-}1)^n(t\text{-}1)$
Graful Petersen	$t(t\text{-}1)(t\text{-}2)(t^7\text{-}12t^6 + 67t^5\text{-}230t^4 + 529t^3\text{-}814t^2 + 775t\text{-}352)$

Proprietăți

- $P(G,0) = 0$
- $P(G,1) = 0$ dacă G conține o muchie
- $P(G,t) = 0$, dacă $t < \chi(G)$.
- $P(G,-1)$ este numărul orientărilor aciclice ale lui G
- Dacă G are n noduri, m muchii, şi k componente $G_1,G_2,\ldots,G_k$, atunci
  - $P(G,t)$ are gradul n.
  - Coeficientul lui t^n în $P(G,t)$ este 1.
  - Coeficientul lui t^{n-1} în $P(G,t)$ este $-m$.
  - Coeficienții corespunzători : $t^0,t^1, \ldots t^{k-1}$ sunt toți zero.
  - Coeficientul lui t^k este diferit de zero.
  - $P(G,t) = P(G_1,t)P(G_2,t)\cdots P(G_k,t)$
- Coeficienții fiecărui polinomul cromatic în parte alternează în ceea ce priveşte semnele.
- Un graf G cu n vârfuri este *arbore* dacă şi numai dacă
$$P(G,t) = t(t-1)^{n-1}.$$
- Derivata evaluată în 1, $P'(G,1)$, reprezintă *invariantul cromatic* θ(G).

Calcularea polinomului cromatic

De fiecare dată când G conține o buclă, acesta nu poate fi colorat, astfel că $P(G,t)=0$.

Dacă e nu este o buclă, atunci polinomul cromatic satisface relația de recurență $P(G,t)=P(G-e,t)-P(G/e,t)$ unde $G-e$ reprezintă graful G din care a fost înlăturată muchia e, iar G/e reprezintă graful pentru care nodurile finale ale muchiilor sale sunt *contractate* într-un singur vârf.

Cele două expresii dau naştere unei proceduri recursive, numită (`n.procedura`) *algoritmul suprimare (ştergere) – contractare*.