

ALGORITMICA GRAFURILOR

Săptămâna 1

C. Croitoru

croitoru@info.uaic.ro

FII

October 1, 2014

- 1 Descrierea cursului
- 2 Interesul pentru grafuri în Informatică
- 3 Vocabular al teoriei grafurilor
- 4 Problemele pentru seminarul 1

Pagina cursului

<http://thor.info.uaic.ro/~croitoru/ag/>

Pagina cursului

<http://thor.info.uaic.ro/~croitoru/ag/>

Obiective

Studentii vor fi familiarizați cu noțiunile și rezultatele de bază ale **Teoriei Algoritmice a Grafurilor**, care vor fi aplicate în proiectarea de algoritmi eficienți pentru diverse probleme de optimizare combinatorică.

Pagina cursului

<http://thor.info.uaic.ro/~croitoru/ag/>

Obiective

Studentii vor fi familiarizați cu noțiunile și rezultatele de bază ale **Teoriei Algoritmice a Grafurilor**, care vor fi aplicate în proiectarea de algoritmi eficienți pentru diverse probleme de optimizare combinatorică.

Tematică Generală

Vocabular al Teoriei Grafurilor, Probleme de drum (parcurgeri, drumuri minime, conexiune), Arbori parțiali de cost minim (union-find, complexitate amortizată), Cuplaje, Fluxuri, Reduceri polinomiale pentru probleme de decizie pe grafuri, Abordări ale problemelor NP-dificile, Grafuri Planare.

Competențe acumulate

Utilizarea grafurilor ca limbaj de modelare formală. Cunoașterea algoritmilor de bază pentru problemele clasice pe grafuri. Recunoașterea complexității de calcul pentru probleme de optimizare.

Competențe acumulate

Utilizarea grafurilor ca limbaj de modelare formală. Cunoașterea algoritmilor de bază pentru problemele clasice pe grafuri. Recunoașterea complexității de calcul pentru probleme de optimizare.

Metode de predare

Prezentari video ale slide-urilor (conținând notele de curs) disponibile în format pdf la începutul semestrului.

http://thor.info.uaic.ro/~croitoru/ag/ag_14-15_allinone.pdf

Competențe acumulate

Utilizarea grafurilor ca limbaj de modelare formală. Cunoașterea algoritmilor de bază pentru problemele clasice pe grafuri. Recunoașterea complexității de calcul pentru probleme de optimizare.

Metode de predare

Prezentari video ale slide-urilor (conținând notele de curs) disponibile în format pdf la începutul semestrului.

http://thor.info.uaic.ro/~croitoru/ag/ag_14-15_allinone.pdf

Seminar: **lect. dr. F.E. OLARIU**

Fiecare seminar dezbate câteva probleme (unele dintre ele dificile !) pentru a aprofunda subiectele introduse la curs. Toate problemele sunt postate la începutul semestrului astfel încât studenții interesați să caute soluții originale sau să studieze probleme similare în bibliografia înrudită.

Bibliografie

- CROITORU C., *Tehnici de bază în optimizarea combinatorie*, Editura Univ. Al. I. Cuza Iasi, Iasi, 1992.
- CROITORU C., *Introducere în proiectarea algoritmilor paraleli*, Editura Matrix Rom, Bucuresti, 2002.
- TOMESCU I., *Probleme de combinatorică și teoria grafurilor*, Editura did. și ped., Bucuresti, 1981.
- DIESTEL R., *Graph Theory*, Electronic Edition.
- CORMEN T.H., Leiserson C.E., Rivest R.L., Stein C., *Introduction to Algorithms*, MIT Press 2001.

Bibliografie

- CROITORU C., *Tehnici de bază în optimizarea combinatorie*, Editura Univ. Al. I. Cuza Iasi, Iasi, 1992.
- CROITORU C., *Introducere în proiectarea algoritmilor paraleli*, Editura Matrix Rom, Bucuresti, 2002.
- TOMESCU I., *Probleme de combinatorică și teoria grafurilor*, Editura did. și ped., Bucuresti, 1981.
- DIESTEL R., *Graph Theory*, Electronic Edition.
- CORMEN T.H., Leiserson C.E., Rivest R.L., Stein C., *Introduction to Algorithms*, MIT Press 2001.

Suplimentar

[http://thor.info.uaic.ro/~croitoru/ag/resurse bibliografice \(optionale\)](http://thor.info.uaic.ro/~croitoru/ag/resurse_bibliografice_(optionale))

EVALUARE

Punctajul minim de promovare: 50 puncte.

EVALUARE

Punctajul minim de promovare: 50 puncte.

FORME:

- Activitatea de la seminar (prezența, participare la dezbateri): 0-18 puncte.
- Teme pentru acasă (3 teme, în săptămânile 5, 9,13), maxim 14 puncte fiecare: 0-42 puncte.
- Testul final scris (open books): 0-60 puncte.

EVALUARE

Punctajul minim de promovare: 50 puncte.

FORME:

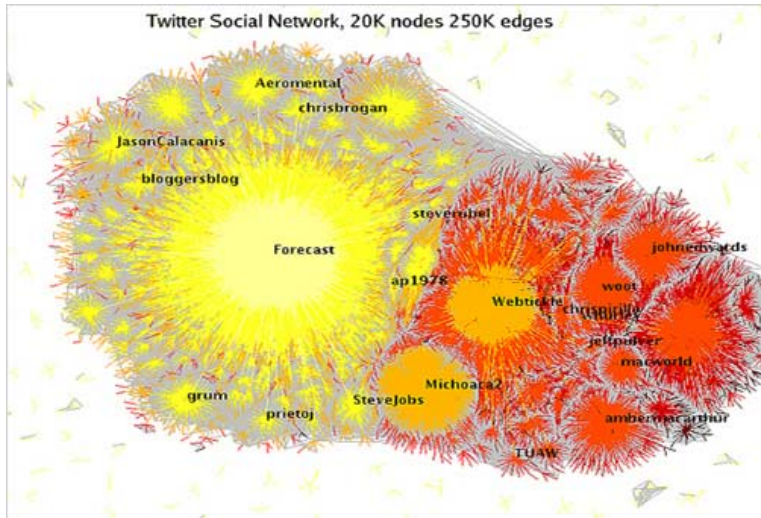
- Activitatea de la seminar (prezența, participare la dezbateri): **0-18 puncte.**
- Teme pentru acasă (3 teme, în săptămânile 5, 9,13), maxim 14 puncte fiecare: **0-42 puncte.**
- Testul final scris (open books): **0-60 puncte.**

Nota finală

Studentii care au obținut minim **50** puncte, sunt sortați descrescător după punctajul final și clasificați după "regulile ETCS" cu adaptările precizate de FII.

Bonus: Seminar Special.

INTERESUL PENTRU GRAFURI ÎN INFORMATICĂ



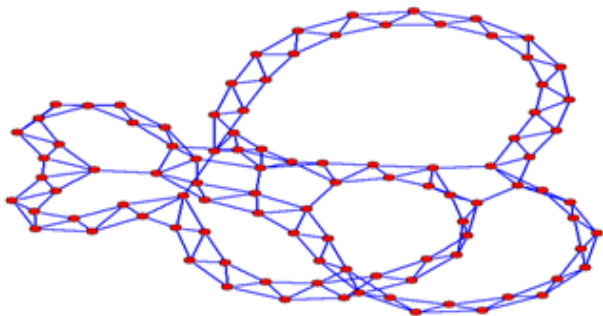
A nice visualization by Akshay Java of network analysis of Twitter.



INTERESUL PENTRU GRAFURI ÎN INFORMATICĂ



Interest in scale-free networks started in 1999 with work by Albert-László Barabási and colleagues at the University of Notre Dame.

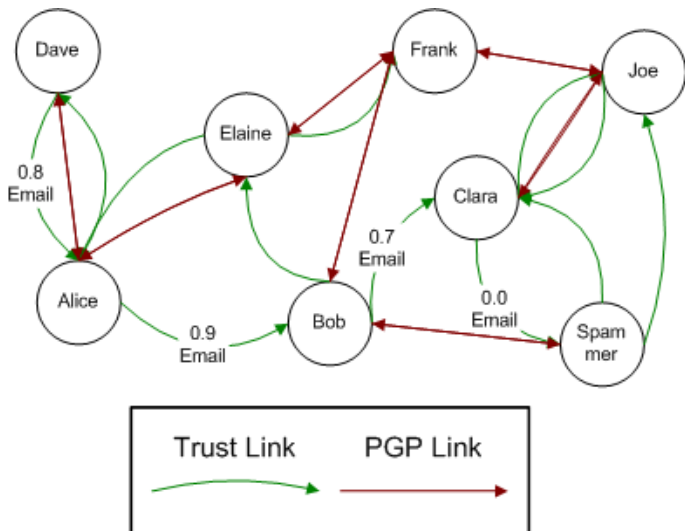


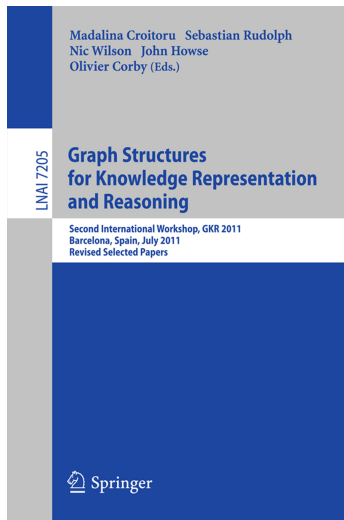
World.png

A small-world network is a type of mathematical graph in which most nodes are not neighbors of one another, but most nodes can be reached from every other by a small number of hops or steps.



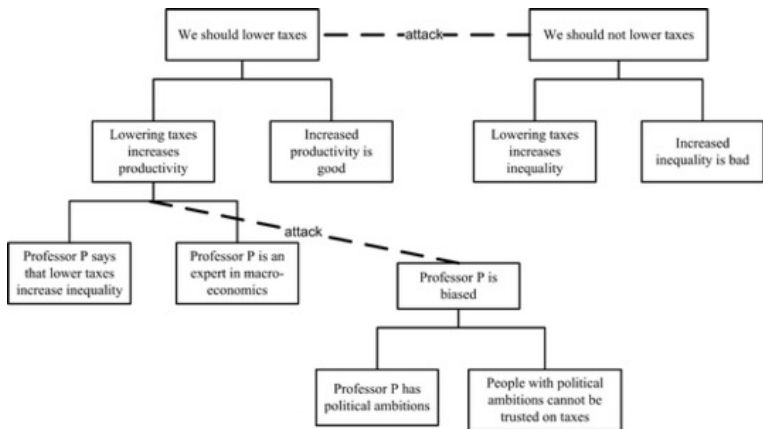
INTERESUL PENTRU GRAFURI ÎN INFORMATICĂ





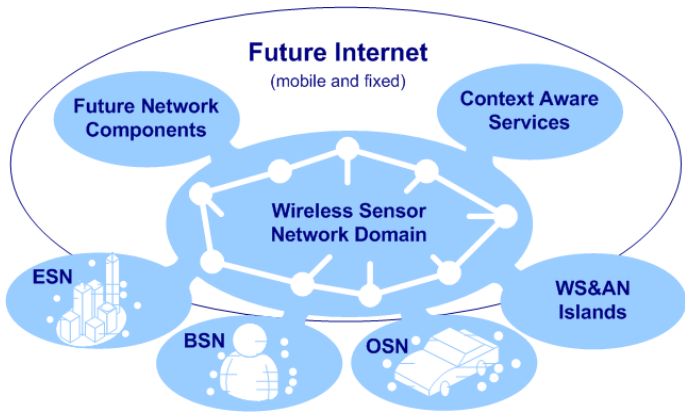
Graph-based knowledge representation formalisms: Bayesian Networks (BNs), Semantic Networks (SNs), Conceptual Graphs (CGs), Formal Concept Analysis (FCA), CP-nets, GAI-nets, etc.

INTERESUL PENTRU GRAFURI ÎN INFORMATICĂ



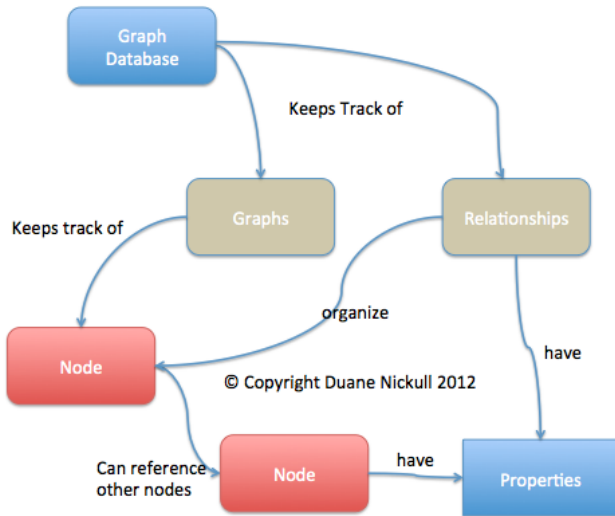
Argumentation Frameworks.

INTERESUL PENTRU GRAFURI ÎN INFORMATICĂ



Environmental Sensor Networks (ESN), Object Sensor Networks (OSN) or Body Sensor Network (BSN) operate a variety of different protocols for the specific application environment.

INTERESUL PENTRU GRAFURI ÎN INFORMATICĂ



Shot.png

Graph-based Data Basis.



Visualization systems.

INTERESUL PENTRU GRAFURI ÎN INFORMATICĂ

simbología

- Transbordo entre líneas de Metro
- Transbordo largo entre líneas de Metro
- Estación con horario restringido
- Estación con acceso para personas con movilidad reducida. Ascensor
- Acceso con rampa
- Estación de Cercanías + Rente
- Estación de largo recorrido + Rente

- Terminal de autobús interurbano
- Aeropuerto de Madrid + Barajas
- Aparcamiento Libre en estación
- Aparcamiento de Pago en estación
- Oficina de información al Cliente

B1 B2 B3

Cambio tarifario exclusivamente para abonos mensuales y anuales, y títulos de 10 viajes



Mayo 2003

Comunidad de Madrid

MetroSur



TFM



leyenda

- 1 Plaza de Castilla / Congosto
- 2 Ventas / Cuatro Caminos
- 3 Legazpi / Moncloa
- 4 Argüelles / Parque de Santa María
- 5 Castillejas / Casa de Campo
- 6 Circular
- 7 Las Musas / Pitis
- 8 Nuevos Ministerios / Barajas
- 9 Herrera Oria / Arganda del Rey
- 10 Fuencarral / Puerta del Sur
- 11 Plaza Elíptica / Pan de Azúcar
- 12 Carabanchel / Casa de Campo
- 13 Circular
- 14 Opera / Príncipe Pío

Madrid-Metro.



INTERESUL PENTRU GRAFURI ÎN INFORMATICĂ



"The **flower** has a **color** of **pink**."

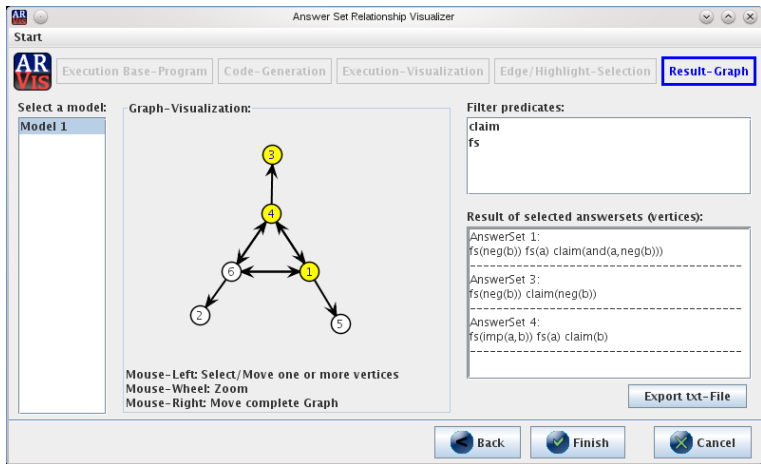
"**Shakespeare** **married** **Anne Hathaway**."

"**John's** **age** is **24**."

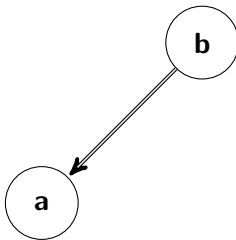
"The **sun** **rises** in the **east**."

A set of such triples is called an RDF graph.

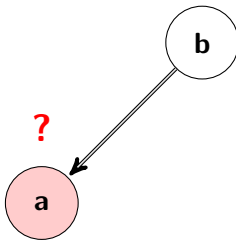
INTERESUL PENTRU GRAFURI ÎN INFORMATICĂ

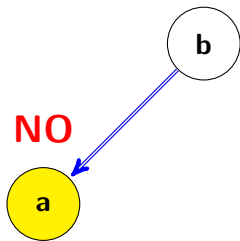


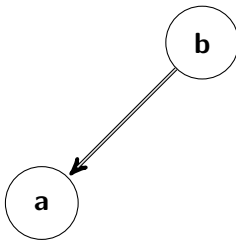
Utilizing ASP for Generating and Visualizing Argumentation Frameworks.



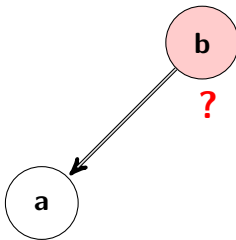
INTERESUL PENTRU GRAFURI ÎN INFORMATICĂ

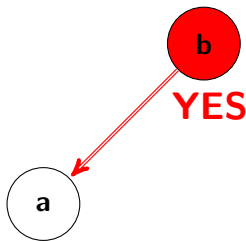




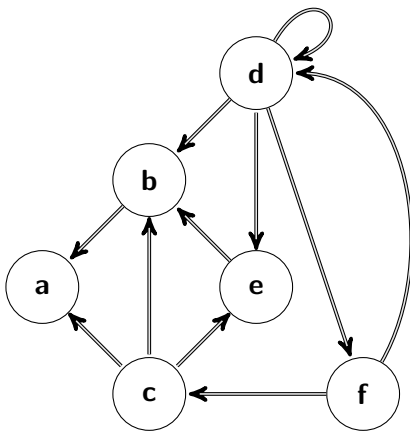


INTERESUL PENTRU GRAFURI ÎN INFORMATICĂ

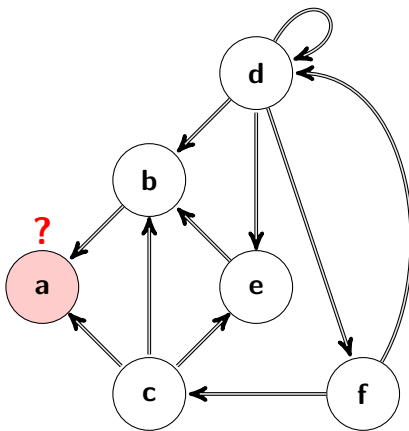




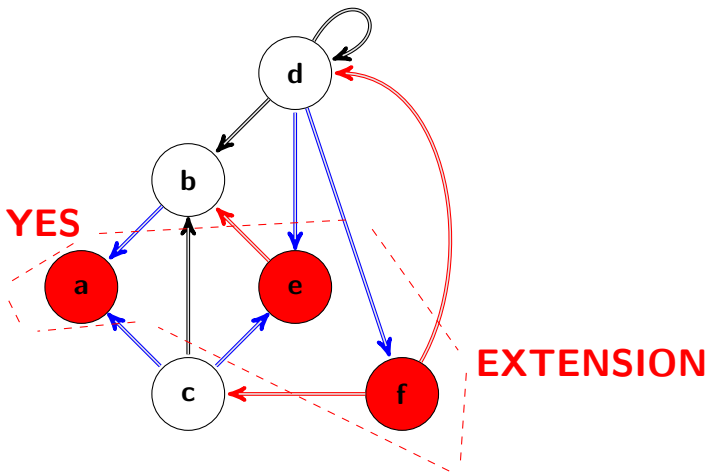
INTERESUL PENTRU GRAFURI ÎN INFORMATICĂ



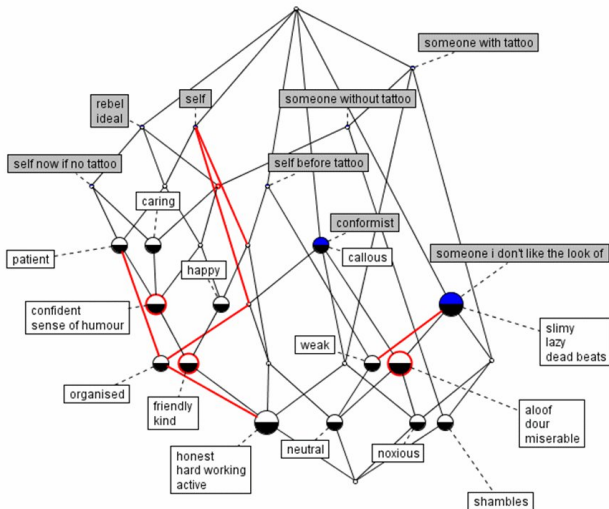
INTERESUL PENTRU GRAFURI ÎN INFORMATICĂ



INTERESUL PENTRU GRAFURI ÎN INFORMATICĂ



INTERESUL PENTRU GRAFURI ÎN INFORMATICĂ



Visualizing a FCA lattice.

Mulțimi stabile

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

NP-completă (Karp, 1972).

Vocabular-Definiția unui graf

Mulțimi stabile

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

NP-completă (Karp, 1972).

Cuplaje

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

Polinomial rezolvabilă (Edmonds, 1965).



Colorarea vârfurilor

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

Este NP-completă chiar pentru $k = 3$.

Vocabular-Definiția unui graf

Colorarea vârfurilor

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

Este NP-completă chiar pentru $k = 3$.

Colorarea muchiilor

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

Este NP-completă chiar dacă e ușor aproximabilă.

Izomorfism

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

Nu se știe dacă e din **P** dar nici nu s-a arătat că e **NP**-completă.

Izomorfism

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

Nu se știe dacă e din **P** dar nici nu s-a arătat că e **NP**-completă.

Izomorfism de subgrafuri

SISO Intrare: G, H grafuri.
 Intrebare: Are G un subgraf G' astfel ca $G' \cong H?$

Este NP-completă.

Problemele pentru seminarul 1

Se vor discuta (cel puțin) patru probleme dintre următoarele:

- ① **Problema 1, Setul 1**
- ② **Problema 3, Setul 1**
- ③ **Problema 4, Setul 1**
- ④ **Problema 3, Setul 3**
- ⑤ **Problema 4, Setul 3**
- ⑥ **Problema 1, Setul 4'**
- ⑦ **Problema 1, Setul 7**
- ⑧ **Problema 1, Setul 8'**

ALGORITMICA GRAFURILOR

Săptămâna 2

C. Croitoru

croitoru@info.uaic.ro

FII

October 8, 2014

① Vocabular al teoriei grafurilor

(ag 14-15 [allinone.pdf](#) pag. 16 →)

② Problemele pentru seminarul 2

- 1 Variații în definiția unui graf
- 2 Grade
- 3 Subgrafuri
- 4 Operații cu grafuri
- 5 Clase de grafuri
- 6 Drumuri și circuite
- 7 Conexiune
- 8 Matrici asociate
- 9 Structuri de date

Se vor discuta (cel puțin) patru probleme dintre următoarele:

- ① **Problema 2, Setul 1**
- ② **Problema 2, Setul 2**
- ③ **Problema 3, Setul 2**
- ④ **Problema 4, Setul 2**
- ⑤ **Problema 2, Setul 3'**
- ⑥ **Problema 3, Setul 3'**
- ⑦ **Problema 4, Setul 3''**
- ⑧ **Problema 4, Setul 7''**

ALGORITMICA GRAFURILOR

Săptămâna 3

C. Croitoru

croitoru@info.uaic.ro

FII

October 15, 2014

- ① **Vocabular al teoriei grafurilor**
(ag 14-15 allinone.pdf pag. 49 → 51)
- ② **Probleme de drum în (di)grafuri**
(ag 14-15 allinone.pdf pag. 52 →)
- ③ **Problemele pentru seminarul 3**

- 1 Variații în definiția unui graf
- 2 Grade
- 3 Subgrafuri
- 4 Operații cu grafuri
- 5 Clase de grafuri
- 6 Drumuri și circuite
- 7 Conexiune
- 8 Matrici asociate
- 9 **Structuri de date**

① Parcurgeri sistematice

- BFS
- DFS

**Componente conexe, tari conexe !!! (pentru examen;
algoritmică ușoară !!!)**

② Probleme de drum minim

Drumuri de cost minim

P1 Date G digraf; $a : E(G) \rightarrow \mathbf{R}$; $s, t \in V(G), s \neq t$.

Să se determine $D_{st}^* \in \mathcal{D}_{st}$, astfel încât

$$a(D_{st}^*) = \min\{a(D_{st}) \mid D_{st} \in \mathcal{D}_{st}\}.$$

P2 Date G digraf; $a : E(G) \rightarrow \mathbf{R}$; $s \in V(G)$.

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; $a : E(G) \rightarrow \mathbf{R}$.

Să se determine $D_{ij}^* \in \mathcal{D}_{ij} \forall i, j \in V(G)$, a.î.

$$a(D_{ij}^*) = \min\{a(D_{ij}) \mid D_{ij} \in \mathcal{D}_{ij}\}.$$

Rezolvarea problemei P2

Teorema 1. Fie $G = (V, E)$ digraf, $V = \{1, \dots, n\}$, $s \in V$ și $a : E \rightarrow \mathbf{R}$, astfel încât

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

Atunci (u_1, \dots, u_n) este o soluție a sistemului

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

dacă și numai dacă

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

Rezolvarea problemei P2 dacă G este digraf fără circuite

O **numerotare aciclică** a (vârfurilor) digrafului $G = (V, E)$ este un vector $ord[v] \quad v \in V$, (cu interpretarea $ord[v] =$ numărul de ordine al vârfului v) a. î.

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

G este un digraf fără circuite dacă și numai dacă admite o numerotare aciclică .

Sortare topologică $\mathcal{O}(n + m)$.

Rezolvarea sistemului $(*)$ prin substituție.

Rezolvarea problemei P2 dacă $\forall ij \in E(G)$ avem $a_{ij} \geq 0$!!!

Algoritmul lui Dijkstra

1. $S \leftarrow \{s\}$; $u_s \leftarrow 0$; $\hat{inainte}[s] \leftarrow 0$;
 for $i \in V \setminus \{s\}$ **do**
 { $u_i \leftarrow a_{si}$; $\hat{inainte}[i] \leftarrow s$ }
 // 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**
 { $u_j \leftarrow u_{j^*} + a_{j^*j}$; $\hat{inainte}[j] \leftarrow j^*$ }
 }

Complexitatea timp a algoritmului, în descrierea dată este $O(n^2)$.

Algoritmul lui Dijkstra

Este posibilă organizarea unor cozi cu prioritate (de exemplu heap-urile) pentru obținerea unui algoritm cu complexitatea $O(m \log n)$ (unde $m = |E|$) *Johnson, 1977*).

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

Problemele pentru seminarul 3

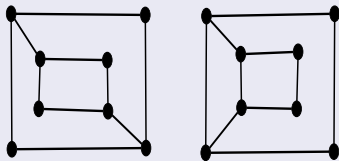
Acesta este un seminar mai deosebit, cu probleme foarte ușoare, având ca singur scop fixarea unor noțiuni.

1

Fie G_1 , G_2 , G_3 trei grafuri. Se știe că $G_1 \not\cong G_2$ și că $G_2 \not\cong G_3$. Rezultă că $G_1 \not\cong G_3$? (justificare)

2

Sunt cele două grafuri desenate mai jos izomorfe ? (justificare)



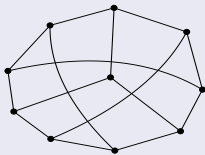
Problemele pentru seminarul 3

3

Demonstrați că dacă un graf conex G are exact un circuit atunci $|G| = |E(G)|$.

4

Determinați numărul de stabilitate al grafului desenat mai jos. (justificare)



5

Fie G un graf conex cu $|G| > 1$ și fără vîrfuri de grad 1. Demonstrați că $|E(G)| \geq n$.

6

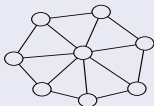
Fie $G = (V, E)$ un graf conex cu $|G| \geq 2$. Demonstrați că există un vârf $v_0 \in V$ astfel încât $G - v_0$ este conex.

7

Este posibil ca numărul arborilor parțiali ai unui graf să fie 1? Dar 2 ? (justificare)

8

Să se determine $L(L(\overline{G}))$, unde graful G este:



9

Dacă G este graful desenat mai jos, este $L(G)$ –graful reprezentativ al muchiilor sale– hamiltonian? (justificare)



10

Precizați numărul cromatic (argumentare) al complementarului grafului de mai sus.

11

Precizați numărul de conexiune (argumentare) al grafului de la problema 9.

12

Este graful următor autocomplementar ? (argumentare)



13

Are graful de mai sus doi arbori parțiali fără muchii comune?
(argumentare)

14

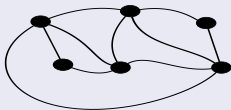
Stabiliți numărul arborilor parțiali ai complementarului grafului de la problema 12. (argumentare)

15

Stabiliți cardinalul maxim al unei multimi stabile din graful $K_2 \times G$, unde G este tot graful de la problema 12.

16

Dacă G este graful desenat mai jos, este $L(G)$ –graful reprezentativ al muchiilor sale– hamiltonian? (justificare)



17

Pentru graful G desenat mai sus să se determine numărul cromatic $\chi(G)$ (argumentare).

18

Este graful de la problema 16 izomorf cu complementarul său ?
(justificare)

19

Determinați numărul de conexiune al grafului de la problema 16.
(justificare)

20

Determinați diametrul grafului de la problema 16. (justificare)

21

Determinați $\chi'(G)$, indicele cromatic al grafului de la problema 16.
(justificare)

Problemele pentru seminarul 3

22

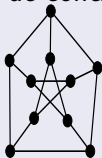
Să se arate că dacă G este graful reprezentativ al muchiilor unui graf H ($G = L(H)$), atunci G este un graf $K_{1,3}$ -free.

23

Desenați graful $P_4 \times K_2$ și determinați-i numărul cromatic (argumentare).

24

Este adevărată inegalitatea $\alpha(G) \leq k(G)$ pentru graful G desenat mai jos? (justificare; $\alpha(G)$ este numărul de stabilitate al lui G , iar $k(G)$ este numărul său de conexiune)



Problemele pentru seminarul 3

25

Fie G un graf conex cu proprietatea că are un vârf din care dacă se execută cele două tipuri de parcurgere (dfs și bfs) arborii dfs și bfs construiți sunt aceși. Poate avea G circuite? (justificare)

26

Demonstrați că dacă $\forall u, v \in V(G), u \neq v$ avem $N_G(u) \cup N_G(v) = V(G)$, atunci graful G este complet.

27

Să se arate că dacă un graf G are exact două vârfuri de grad impar atunci în G există un drum între aceste două vârfuri.

28

Fie $G = (V, E)$ un graf cu proprietatea că $\forall v, w \in V, v \neq w$ are loc $d_G(v) + d_G(w) \geq |V| - 1$. Demonstrați că diametrul lui G este cel mult 2.



29

Fie G un graf conex cu toate vârfurile de grad par. Demonstrați că $\forall e \in E$ graful $G - e$ este conex.

30

Fie $G = (V, E)$ un graf cu cel puțin 3 vârfuri. Demonstrați că G este conex dacă și numai dacă există două vârfuri $u, v \in V$ ($u \neq v$) astfel încât grafurile $G - u$ și $G - v$ sunt conexe.

31

Dacă $H = (V(H), E(H))$ este un graf, notăm numărul muchiilor lui H cu $e(H)$ ($e(H) = |E(H)|$). Demonstrați că pentru orice graf $G = (V, E)$ cu cel puțin 3 vârfuri are loc egalitatea

$$e(G) = \frac{\sum_{v \in V} e(G-v)}{|V|-2}.$$

ALGORITMICA GRAFURILOR

Săptămâna 4

C. Croitoru

croitoru@info.uaic.ro

FII

October 22, 2014

-
- ① Probleme de drum în (di)grafuri
(ag 14-15 [allinone.pdf](#) pag. 94 → ...)
 - ② Problemele pentru seminarul 4

Drumuri de cost minim

P1 Date G digraf; $a : E(G) \rightarrow \mathbf{R}$; $s, t \in V(G), s \neq t$.

Să se determine $D_{st}^* \in \mathcal{D}_{st}$, astfel încât

$$a(D_{st}^*) = \min\{a(D_{st}) \mid D_{st} \in \mathcal{D}_{st}\}.$$

P2 Date G digraf; $a : E(G) \rightarrow \mathbf{R}$; $s \in V(G)$.

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; $a : E(G) \rightarrow \mathbf{R}$.

Să se determine $D_{ij}^* \in \mathcal{D}_{ij} \forall i, j \in V(G)$, a.î.

$$a(D_{ij}^*) = \min\{a(D_{ij}) \mid D_{ij} \in \mathcal{D}_{ij}\}.$$

Rezolvarea problemei P2

Teorema 1. Fie $G = (V, E)$ digraf, $V = \{1, \dots, n\}$, $s \in V$ și $a : E \rightarrow \mathbf{R}$, astfel încât

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

Atunci (u_1, \dots, u_n) este o soluție a sistemului

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

dacă și numai dacă

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

Rezolvarea problemei P2 dacă $\forall ij \in E(G)$ avem $a_{ij} \geq 0$!!!

Algoritmul lui Dijkstra

1. $S \leftarrow \{s\}$; $u_s \leftarrow 0$; $\hat{inainte}[s] \leftarrow 0$;
 for $i \in V \setminus \{s\}$ **do**
 { $u_i \leftarrow a_{si}$; $\hat{inainte}[i] \leftarrow s$ }
 // 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**
 { $u_j \leftarrow u_{j^*} + a_{j^*j}$; $\hat{inainte}[j] \leftarrow j^*$ }
 }

Complexitatea timp a algoritmului, în descrierea dată este $O(n^2)$.

Algoritmul lui Dijkstra

Este posibilă organizarea unor cozi cu prioritate (de exemplu heap-urile) pentru obținerea unui algoritm cu complexitatea $O(m \log n)$ (unde $m = |E|$) *Johnson, 1977*).

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

Strategii de implementare

- Estimator consistent
- PSP

Rezolvarea problemei P2 în cazul general.

Algoritmul lui *Bellman, Ford, Moore* (~ 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**
 $u_i^{m+1} \leftarrow \min(u_i^m, \min_{j \neq i}(u_j^m + a_{ji}))$

Complexitatea $O(n^3)$, dacă determinarea minimului din pasul 2 necesită $O(n)$ operații.

Testarea în $O(n^3)$ a existenței unui circuit C de cost negativ în digraful G !

Rezolvarea problemei P3

P3 Date G digraf; $a : E(G) \rightarrow \mathbf{R}$.

Să se determine $D_{ij}^* \in \mathcal{D}_{ij} \ \forall i, j \in V(G)$, a.î.

$a(D_{ij}^*) = \min\{a(D_{ij}) \mid D_{ij} \in \mathcal{D}_{ij}\}.$

Probleme de drum minim

Rezolvarea problemei P3

P3 Date G digraf; $a : E(G) \rightarrow \mathbf{R}$.

Să se determine $D_{ij}^* \in \mathcal{D}_{ij} \ \forall i, j \in V(G)$, a.î.

$$a(D_{ij}^*) = \min\{a(D_{ij}) \mid D_{ij} \in \mathcal{D}_{ij}\}.$$

$O(n^4)$

Iterarea algoritmului lui Bellman-Ford pentru $s = \overline{1, n}$

Probleme de drum minim

Rezolvarea problemei P3

P3 Date G digraf; $a : E(G) \rightarrow \mathbf{R}$.

Să se determine $D_{ij}^* \in \mathcal{D}_{ij} \ \forall i, j \in V(G)$, a.î.

$a(D_{ij}^*) = \min\{a(D_{ij}) \mid D_{ij} \in \mathcal{D}_{ij}\}.$

$O(n^4)$

Iterarea algoritmului lui Bellman-Ford pentru $s = \overline{1, n}$

$O(n^3)$

Iterarea algoritmului lui Dijkstra, **după preprocesare!**

Probleme de drum minim

Rezolvarea problemei P3

P3 Date G digraf; $a : E(G) \rightarrow \mathbf{R}$.

Să se determine $D_{ij}^* \in \mathcal{D}_{ij} \ \forall i, j \in V(G)$, a.î.

$$a(D_{ij}^*) = \min\{a(D_{ij}) \mid D_{ij} \in \mathcal{D}_{ij}\}.$$

$O(n^4)$

Iterarea algoritmului lui Bellman-Ford pentru $s = \overline{1, n}$

$O(n^3)$

Iterarea algoritmului lui Dijkstra, după preprocesare!

$O(n^3 \log n)$

Înmulțiri matriciale!

Probleme de drum minim

Rezolvarea problemei P3

P3 Date G digraf; $a : E(G) \rightarrow \mathbf{R}$.

Să se determine $D_{ij}^* \in \mathcal{D}_{ij} \ \forall i, j \in V(G)$, a.î.

$a(D_{ij}^*) = \min\{a(D_{ij}) \mid D_{ij} \in \mathcal{D}_{ij}\}.$

$O(n^4)$

Iterarea algoritmului lui Bellman-Ford pentru $s = \overline{1, n}$

$O(n^3)$

Iterarea algoritmului lui Dijkstra, după preprocesare!

$O(n^3 \log n)$

Înmulțiri matriciale!

$O(n^3)$

Algoritmul lui **Floyd-Warshall**

Algoritmul lui **Floyd-Warshal**

```
1: for  $i := 1$  to  $n$  do  
  for  $j := 1$  to  $n$  do  
    {  $\hat{inainte}(i, j) \leftarrow i$ ;  
      if  $i = j$  then {  $a_{ii} \leftarrow 0$ ;  $\hat{inainte}(i, i) \leftarrow 0$  }  
    }  
  
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  
        {  $a_{ij} \leftarrow a_{im} + a_{mj}$ ;  
           $\hat{inainte}(i, j) \leftarrow \hat{inainte}(m, j)$   
          if  $(i = j \wedge a_{ij} < 0)$  then  
            return "circuit negativ"  
        }  
    }
```

Teorema lui Menger

Fie $G = (V, E)$ (di)graf și $X, Y \subseteq V$. Atunci numărul maxim de XY -drumuri disjuncte este egal cu cardinalul minim al unei mulțimi XY -separatoare.

Fie $G = (V, E)$ un (di)graf și $s, t \in V$, astfel încât $s \neq t$, $st \notin E$. Există k drumuri intern disjuncte de la s la t în G dacă și numai dacă îndepărtând mai puțin de k vârfuri diferite de s și t , în (di)graful rămas există un drum de la s la t .

Consecință 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 la determinarea lui

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

(care se poate obține în timp polinomial.)

Teorema lui König

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

Consecință: Dacă G e graf bipartit, atunci :

$$\nu(G) = |G| - \alpha(G).$$

Teorema lui Hall

Dacă $\mathcal{A} = (A_i; i \in I)$ este o familie de submulțimi ale lui S , o funcție $r_{\mathcal{A}} : I \rightarrow S$ 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 \mathcal{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*.

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

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

- ① Problema 1, Setul 3”
- ② Problema 3, Setul 6
- ③ Problema 2, Setul 20
- ④ 3-4 probleme din lista următoare :)

Probleme pentru seminarul 4

1

Să se construiască o funcție care să recunoască un turneu. La intrare aceasta va primi un digraf $G = (\{1, \dots, n\}, E)$ reprezentat cu ajutorul listelor de adiacență și va returna *true* sau *false*. Complexitatea timp?

2

Să se construiască o funcție care primind la intrare un digraf $G = (\{1, \dots, n\}, E)$ reprezentat cu ajutorul listelor de adiacență să returneze inversul lui G reprezentat cu ajutorul listelor de adiacență. Complexitatea timp trebuie să fie $\mathcal{O}(n + |E|)$.

3

Se consideră un graf $G = (\{1, \dots, n\}, E)$ reprezentat cu ajutorul matricii de adiacență. Mulțimea de $n - 1$ muchii A este astfel ca $T = (V, A)$ este arbore parțial al lui G . Construiți un algoritm care să listeze circuitele care se formează prin adăugarea muchiilor din $E - A$ la T . Reprezentarea lui T trebuie să permită depistarea fiecărui astfel de circuit în timpul $\mathcal{O}(n)$.



Probleme pentru seminarul 4

4

Să se construiască o funcție care să determine gradul maxim al unui vârf al unui graf. La intrare aceasta va primi un graf $G = (\{1, \dots, n\}, E)$ reprezentat cu ajutorul listelor de adiacență și va returna $\Delta(G)$. Stabiliți complexitatea timp a algoritmului folosit.

5

Construiți o funcție care primind la intrare graful $G = (V, E)$ reprezentat cu ajutorul listelor de adiacență și k , un număr întreg pozitiv, returnează graful $G^{(k)}$ cu aceeași mulțime de virfuri ca și G , în care două virfuri distincte sunt adiacente dacă și numai dacă în graful inițial sunt conectate printr-un drum de lungime cel mult k . Care este complexitatea timp?

6

Graful conex $G = (V, E)$ cu n vârfuri și m muchii, este reprezentat cu ajutorul listelor de adiacență. Dați un algoritm care să construiască în timpul $O(n + m)$ listele de adiacență ale unui arbore parțial al lui G .



Probleme pentru seminarul 4

7

Fie $G = (V, E)$ un graf cu ordinul $|V| \geq 2$ și $T = (V, E_T)$ un arbore parțial al lui G , dat de tabloul $(p[v])_{v \in V}$, unde $p[v]$ este părintele lui v în T : vârful dinaintea lui v de pe drumul unic de la o rădăcină fixată r , la v , în T ($p[r] = r$). Dați un algoritm care să determine, în timpul $\mathcal{O}(|V|)$, un vârf v_0 pendant (frunză) în T și apoi demonstrați că $G - v_0$ este conex.

8

Digraful $G = (V, E)$ este dat prin listele de adiacență. Să se decidă în $\mathcal{O}(|V| + |E|)$ dacă se pot ordona vârfurile sale: $v_{i_1}, \dots, v_{i_{|V|}}$, astfel încât dacă v_{i_j} apare în lista de adiacență a lui v_{i_k} atunci $k < j$.

9

Fie $T = (\{1, \dots, n\}, E_T)$ un arbore ($n \geq 2$), dat de tabloul $(p[v])_{v \in V}$, unde $p[v]$ este părintele lui v în T : vârful dinaintea lui v de pe drumul unic de la o rădăcină fixată r , la v , în T ($p[r] = r$). Descrieți un algoritm care să construiască, în timpul $\mathcal{O}(n)$, listele de adiacență ale lui T .



10

Se consideră un graf $G = (V, E)$ ($V = \{1, \dots, n\}$), izomorf cu graful circuit (cu cel puțin 3 vârfuri), $G \cong C_n$. Fiecare muchie $e \in E$ are asociat un cost real $c(e)$. Aceste informații sunt disponibile în tablourile *dreapta* și *cost* de dimensiune n cu semnificația: $dreapta[v] = \text{vecinul din dreapta al lui } v$, iar costul muchiei $\{v, dreapta(v)\}$ este $cost[v]$. Descrieți un algoritm cât mai eficient pentru aflarea unui arbore parțial al lui G de cost minim.

11

Se consideră un graf $G = (V, E)$ ($V = \{1, \dots, n\}$), reprezentat cu ajutorul listelor de adiacență. Se știe că graful are gradul minim $\delta(G)$ mărginit de o constantă $c \in \mathbf{N}$. Descrieți un algoritm cu complexitatea timp $O(n)$ pentru determinarea lui $\delta(G)$ și a unui vârf $v_0 \in V$ cu gradul în G egal cu $\delta(G)$.



ALGORITMICA GRAFURILOR

Săptămâna 5

C. Croitoru

croitoru@info.uaic.ro

FII

October, 29, 2014

- ① **Arbori** (ag 14-15 allinone.pdf pag. 134 → 166)
- ② **Problemele pentru seminarul 5**
- ③ **Prezentarea temei pentru acasă**

Teoreme de caracterizare

Teoremă. Fie $G = (V, E)$ un graf.

Următoarele afirmații sînt echivalente:

- (i) G este arbore (*este conex și fără circuite*).
- (ii) G este *conex* și este minimal cu această proprietate.
- (iii) G este *fără circuite* și este maximal cu această proprietate.

Teoremă. 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.

Generarea arborilor parțiali ai unui graf

generare-arbori-parțiali(*int i*);

// se generează toți arborii parțiali ai lui G
avînd drept prime $i - 1$ muchii, elementele
 $T(1), \dots, T(i - 1)$
ale tabloului E (ordonate crescător).

variabile locale:

$j \in \{1, \dots, m\}$; S , listă de vîrfuri; $x \in V$;

if $i = n$ **then**

// $\{T(1), \dots, T(n - 1)\}$ formează un
arbore parțial ;
prelucrează T (*listează, memorează etc.*)

Generarea arborilor parțiali ai unui graf (continuare)

else

if $i = 1$ **then**

for $j := 1$ **to** $d_G(v_0)$ **do**

{ $T[i] \leftarrow j$;

A:

generare-arbori-parțiali($i + 1$);

B:

}

else

for $j := T[i - 1] + 1$ **to** $m - (n - 1) + i$ **do**

if $\langle \{T[1], \dots, T[i - 1]\} \cup \{j\} \rangle_G$ nu are circuite

then

{ $T[i] \leftarrow j$;

A:

generare-arbori-parțiali($i + 1$);

B: }

Numărarea arborilor parțiali ai unui graf

Fie $G = (V, E)$ un multigraf cu $V = \{1, 2, \dots, n\}$.

Considerăm $A = (a_{ij})_{n \times n}$ matricea de adiacență a lui G (a_{ij} = multiplicitatea muchiei ij dacă $ij \in E$, altfel 0).

Fie $D = \text{diag}(d_G(1), d_G(2), \dots, d_G(n))$.

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

Teoremă (Kirchoff-Trent; Matrix Tree Theorem)

Dacă G este un multigraf cu mulțimea de vârfuri $\{1, \dots, n\}$ și $L[G]$ matricea Laplace, atunci

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

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

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

Arbori parțiali de cost minim.

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

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

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

Algoritm generic

Inițial, $\mathcal{T}^0 = (T_1^0, T_2^0, \dots, T_n^0)$, $T_i^0 = (\{i\}, \emptyset)$, $i = \overline{1, n}$ ($V = \{1, 2, \dots, n\}$).

În pasul k ($k = \overline{0, n-2}$), din familia $\mathcal{T}^k = (T_1^k, T_2^k, \dots, T_{n-k}^k)$ de $n-k$ arbori a.î $V(T_i^k)_{i=\overline{1, n-k}}$ este partiție a lui V , se construiește \mathcal{T}^{k+1} :

- se alege T_s^k unul din arborii familiei \mathcal{T}^k .
- dintre 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 $v_{j^*} \in V(T_{j^*}^k)$.
- $\mathcal{T}^{k+1} = (\mathcal{T}^k \setminus \{T_s^k, T_{j^*}^k\}) \cup \mathcal{T}$, unde \mathcal{T} este arborele obținut din T_s^k și $T_{j^*}^k$ la care adăugăm muchia e^* .

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

Algoritmul lui Prim (implementare Dijkstra)

Arborele T_s^k va fi întotdeauna arborele cu cele mai multe vîrfuri dintre arborii familiei curente.

La fiecare pas $k > 0$ avem un arbore $T_s = (V_s, E_s)$ cu $k + 1$ vîrfuri, ceilalți $n - k - 1$ avînd cîte un singur vîrf.

Fie $\alpha[1..n]$ cu componente din V și $\beta[1..n]$ cu componente reale a.î. :
 $\forall j \in V - V_s, \beta[j] = c(\alpha[j]j) = \min\{c(ij) \mid i \in V_s, ij \in E\}$

1. $V_s \leftarrow \{s\}; (s \in V, \text{oarecare})$
 $E_s \leftarrow \emptyset;$
for $v \in V \setminus \{s\}$ **do** $\{ \alpha[v] := s; \beta[v] := c(sv) \};$
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**
 $\{ \beta[j] \leftarrow c[j^*j]; \alpha[j] \leftarrow j^* \}$
 $\}$

Algoritmul lui Kruskal

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

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

1. Sortează $E = (e_1, e_2, \dots, e_m)$ astfel încît:

$$c(e_1) \leq c(e_2) \leq \dots \leq c(e_m).$$

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**

$T \leftarrow T \cup \{e_i\};$

$i++$ }

Pasul 1 necesită $O(m \log n)$ operații.

Pentru realizarea eficientă a testului din pasul 2 va fi necesar să reprezentăm la fiecare pas k , $V(T_1^k), V(T_2^k), \dots, V(T_n^k)$ și să testăm **dacă muchia e_i curentă are ambele extremități în aceeași mulțime.**

Se vor folosi pentru reprezentarea acestor mulțimi, arbori (care nu sînt în general subarbori ai lui G).

Union-Find - Tarjan

$pred[1..n]$:

$pred[v]$ = vârful dinaintea lui v de pe drumul la v de la rădăcina arborelui care memorează mulțimea la care aparține v ;

$pred[v] = 0 \Leftrightarrow v$ este rădăcina arborelui;

$pred[v] < 0 \Leftrightarrow v$ este rădăcină a unui arbore și $-pred[v]$ este cardinalul mulțimii memorate în el”.

Adăugăm în pasul 1 inițializarea 1.3 și modificăm pasul 2 al algoritmului astfel:

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

2. while $i \leq m$ do

```
{  fie  $e_i = vw$ ;
    $x \leftarrow find(v)$ ;  $y \leftarrow find(w)$ ;
   if  $x \neq y$  then {  $union(x, y)$ ;  $T \leftarrow T \cup \{e_i\}$  }
    $i++$  }
```

Union-Find - Tarjan

Procedura union și funcția find sunt:

```
procedure union( $v, w : V$ );
    //  $v$  și  $w$  sunt rădăcini, variabila locală întreagă  $t$ 
     $t \leftarrow \text{pred}[v] + \text{pred}[w]$ ;
    if  $\text{pred}[v] > \text{pred}[w]$  then {  $\text{pred}[v] \leftarrow w$ ;  $\text{pred}[w] \leftarrow t$  }
    else {  $\text{pred}[w] \leftarrow v$ ;  $\text{pred}[v] \leftarrow t$  }

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

Complexitatea pasului 2 este $O(m \cdot \alpha(m, n))$, unde $\alpha(m, n)$ este inversa funcției lui Ackermann.

Se vor discuta (cel puțin) patru probleme dintre următoarele:

- ① **Problema 1, Setul 13**
- ② **Problemele 1,3,4 Setul 5**
- ③ **Problema 1, Setul 6**
- ④ **Problema 2 Setul 4**
- ⑤ **Problema 3 Setul 20**
- ⑥ **Problema 4 Setul 4'**

ALGORITMICA GRAFURILOR

Săptămâna 6

C. Croitoru

croitoru@info.uaic.ro

FII

November 5, 2014

① **Cuplaje** (ag 14-15 allinone.pdf pag. 167 → 195)

② **Problemele pentru seminarul 6**

Problema cuplajului maxim

Fie $G = (V, E)$ un (multi)graf. Dacă $A \subseteq E$ și $v \in V$, vom nota $d_A(v)$ gradul vârfului v în graful parțial $\langle A \rangle_G$.

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

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

$M \in \mathcal{M}_G, v \in V$:

Dacă $d_M(v) = 1$, atunci v se numește **saturat** de cuplajul M .

S(M): mulțimea vîrfurilor saturate de cuplajul M în graful G .

Dacă $d_M(v) = 0$, atunci v se numește **expus** față de cuplajul M .

E(M): mulțimea vîrfurilor expuse față de cuplajul 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 $\nu(G) = \max\{|M| \mid M \in \mathcal{M}_G\}$.

Problema cuplajului maxim

Definiție. Se numește **acoperire** (a vîrfurilor cu muchii) în graful G orice mulțime $F \subseteq E$ de muchii cu proprietatea că $d_F(v) \geq 1 \ \forall v \in V$.

$\mathcal{F}_G = \{F \mid F \subseteq E, F \text{ acoperire în } G\}$ 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 $|F^*| = \min\{|F| \mid F \in \mathcal{F}_G\}$.*

Teorema 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*

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

Dem. **P1 echivalentă polinomial cu P2**

Grafuri bipartite

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

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

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

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

Cuplaje perfecte

Un cuplaj M în graful G astfel încât $S(M) = V(G)$ se numește **cuplaj perfect** sau **1-factor**.

Pentru un graf oarecare H notăm cu $q(H)$ numărul componentelor conexe de ordin impar ale lui H .

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

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

Berge (1958) a generalizat această teoremă stabilind că

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

Problema cuplajului maxim

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

Definiție: Se numește **drum alternat al lui G relativ la cuplajul M** orice drum $P : v_0, v_0v_1, v_1, \dots, v_{k-1}, v_{k-1}v_k, v_k$ a. î.

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

Vom desemna prin P mulțimea muchiilor drumului P .

Definiție: Se numește **drum de creștere al lui G relativ la cuplajul M** un drum alternat cu extremitățile vîrfuri distincte, expuse relativ la cuplajul M .

Teoremă. (Berge 1959) Un cuplaj M este de cardinal maxim în graful G dacă și numai dacă nu există în G drumuri de creștere relativ la M .

Strategie de construire a unui cuplaj de cardinal maxim:

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

Problema cuplajului maxim

Hopcroft, Karp (1973)

0. $M \leftarrow \emptyset$;
1. **repeat**
 Determină \mathcal{P} o familie maximală (\subseteq)
 de drumuri minime de creștere;
 for $P \in \mathcal{P}$ **do** $M \leftarrow M \Delta P$
until $\mathcal{P} = \emptyset$.

Complexitatea $O(\sqrt{n}A)$ unde A este timpul găsirii familiei \mathcal{P} .

Hopcroft și Karp au arătat cum se poate implementa pasul 1 pentru un graf bipartit, astfel încât $A = O(m + n)$: \Rightarrow algoritm $O(mn^{1/2})$ pentru aflarea unui cuplaj de cardinal maxim într-un graf bipartit.

Pentru un graf oarecare, structurile de date necesare obținerii aceleiași complexități sînt mult mai elaborate și au fost descrise de **Micali și Vazirani 1980**.

Se vor discuta (cel puțin) patru probleme dintre următoarele:

- ① Problema 1, Setul 3
- ② Problemele 2,3,4 Setul 7
- ③ Problema 3, Setul 7''
- ④ Problema 1, Setul 8
- ⑤ Problema 3, Setul 8'
- ⑥ Problema 1, Setul 21
- ⑦ Problema 2, Setul 21

ALGORITMICA GRAFURILOR

Săptămâna 7

C. Croitoru

croitoru@info.uaic.ro

FII

November 12, 2014

① **Fluxuri** (ag 14-15 [allinone.pdf](#) pag. 196 → ...)

② **Problemele pentru seminarul 7**

Problema fluxului maxim

Numim **rețea** (de transport) cu **intrarea** s și **ieșirea** t , 4-uplul

$R = (G, s, t, c)$ unde: - $G = (V, E)$ este un digraf,

- $s, t \in V$; $s \neq t$; $d_G^+(s) > 0$; $d_G^-(t) > 0$,

- $c : E \rightarrow \mathbf{R}_+$; $c(e)$ este **capacitatea** arcului e .

$V = \{1, 2, \dots, n\}$ ($n \in \mathbf{N}^*$) și $|E| = m$. Extindem funcția c la

$c : V \times V \rightarrow \mathbf{R}_+$ prin $c((i, j)) = \begin{cases} c(ij) & \text{dacă } ij \in E \\ 0 & \text{dacă } ij \notin E \end{cases} = c_{ij}.$

Numim **flux în rețeaua** $R = (G, s, t, c)$ o funcție $x : V \times V \rightarrow \mathbf{R}$, a.î.:

$$(i) \quad 0 \leq x_{ij} \leq c_{ij} \quad \forall ij \in V \times V$$

$$(ii) \quad \sum_{j \in V} x_{ji} - \sum_{j \in V} x_{ij} = 0 \quad \forall i \in V - \{s, t\}.$$

Dacă $ij \in E$ atunci x_{ij} se numește **fluxul** (transportat) **pe arcul** ij . Evident, condiția (i) cere ca **fluxul pe orice arc să fie nenegativ și subcapacitar**, iar condiția (ii) (*legea de conservare a fluxului*) cere ca **suma fluxurilor pe arcele care intră în vârful** i **să fie egală cu suma fluxurilor pe arcele care ies din vârful** i .

Problema fluxului maxim

Definiție: Dacă x este un flux în rețeaua $R = (G, s, t, c)$, se numește **valoarea fluxului** x numărul

$$v(x) = \sum_{j \in V} x_{jt} - \sum_{j \in V} x_{tj}.$$

$v(x)$ este fluxul net care ajunge în ieșirea rețelei și este egal cu fluxul net care iese din intrarea rețelei.

Problema fluxului maxim:

Dată $R = (G, s, t, c)$ o rețea, să se determine un flux de valoare maximă.

Problema fluxului maxim

Definiție. Dacă P este un drum în \overline{G} , multigraful suport al digrafului G , și $e = v_i v_j$ este o muchie a lui P atunci: **dacă e corespunde arcului $v_i v_j$ al lui G , e se numește arc direct al drumului P ; dacă e corespunde arcului $v_j v_i$ al lui G , atunci e se numește arc invers.**

Definiție. Fie $R = (G, s, t, c)$ și x flux în R . Se numește **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 $ij \in E(D)$, **capacitatea reziduală** a lui ij

(relativ la C-drumul D) este $r(ij) = \begin{cases} c_{ij} - x_{ij} & \text{dacă } ij \text{ arc direct în } D \\ x_{ji} & \text{dacă } ij \text{ arc invers în } D \end{cases}$.

Capacitatea reziduală a drumului D este $r(D) = \min_{e \in E(D)} r(e)$.

Definiție. Se numește **drum de creștere** a fluxului x , în rețeaua $R = (G, s, t, c)$, un C-drum de la s la t .



Problema fluxului maxim

Lema 1. Dacă D este un drum de creștere a fluxului x în rețeaua $R = (G, s, t, c)$, atunci $x^1 = x \otimes r(D)$ definit prin

$$x_{ij}^1 = \begin{cases} x_{ij} & \text{dacă } \overline{ij} \notin E(D) \\ x_{ij} + r(D) & \text{dacă } ij \in E(D), ij \text{ arc direct în } D \\ x_{ij} - r(D) & \text{dacă } ji \in E(D), ji \text{ arc invers în } D \end{cases}$$

este flux în R și $v(x^1) = v(x) + r(D)$.

Observăm că dacă x admite un drum de creștere atunci x nu este flux de valoare maximă.

Definiție. Se numește **secțiune** în rețeaua $R = (G, s, t, c)$, o partiție (S, T) a lui V cu $s \in S$ și $t \in T$. **Capacitatea secțiunii** (S, T) este

$$c(S, T) = \sum_{i \in S} \sum_{j \in T} c_{ij}$$

(suma capacităților arcelor de la S la T).

Problema fluxului maxim

Lema 2. Dacă x este un flux în $R = (G, s, t, c)$ și (S, T) este o secțiune a rețelei, 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 secțiune.)

Teorema 1. (Teorema drumului de creștere)

Un flux x este de valoare maximă într-o rețea R , dacă și numai dacă, nu există drumuri de creștere a fluxului x în rețeaua R .

Teorema 2. (Teorema fluxului întreg)

Dacă toate capacitățile sînt întregi, atunci există un flux de valoare maximă cu toate componentele întregi (flux întreg de valoare maximă).

Teorema 3. (Ford-Fulkerson, 1956)

Valoarea maximă a unui flux în rețeaua $R = (G, s, t, c)$ este egală cu capacitatea minimă a unei secțiuni a rețelei.

Problema fluxului maxim

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 $e_1 \in V \cup \{0\}$; $e_2 \in \{direct, invers\}$; $e_3 \in \mathbb{R}_+$ și au următoarea 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 etichetează $e = (i, direct, \min(e_3[i], c_{ij} - x_{ij}))$;

Dacă j neetichetat, $ji \in E$ și $x_{ji} > 0$ atunci

j se etichetează $e = (i, invers, \min(e_3[i], x_{ji}))$.

Algoritmul lui Ford și Fulkerson

- 1: Se alege $x = (x_{ij})$ flux inițial (de ex. fluxul nul);
Se etichetează s cu $(0, \cdot, \infty)$
- 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 $(0, \cdot, \infty)$
 }
 }
 }
- 3: $S \leftarrow \{i | i \in V, i \text{ are etichetă}\}$
 $T \leftarrow V - S$

x este flux de valoare maximă

(S, T) este secțiune de capacitate minimă.

Algoritmul are complexitatea $O(mv)$, unde v este valoarea fluxului maxim iar $m = |E|$.

Modificarea lui Edmonds & Karp a alg. 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:

$$x^0 \leftarrow x;$$

$$x^k \leftarrow x^{k-1} \otimes r(P_{k-1}), \quad P_k \text{ este drum minim de creștere} \\ \text{relativ la } x^{k-1}; \quad k = 1, 2, \dots$$

Teorema 4. (Edmonds, Karp) Dacă $x = x^0$ este un flux oarecare în rețeaua R , atunci șirul de fluxuri x^1, x^2, \dots obținut din x^0 prin creșteri succesive pe drumuri minime de creștere, are cel mult $\frac{mn}{2}$ elemente (în cel mult $\frac{mn}{2}$ creșteri succesive, se obține un flux care nu admite drumuri de creștere).

Teorema 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 $O(m^2n)$.

Se vor discuta problemele restante din săptămâna 6 și cele din tema 1.

ALGORITMICA GRAFURILOR

Săptămâna 9

C. Croitoru

croitoru@info.uaic.ro

FII

November 26, 2014

① **Fluxuri** (ag 14-15 allinone.pdf pag. 223 → ...)

② **Problemele pentru seminarul 9**

③ **Prezentarea temei pentru acasă**

Problema fluxului maxim

Algoritmi de tip preflux

Se numește **preflux** în rețeaua R , o funcție $x : E \rightarrow \mathbf{R}$ astfel încât

$$(i) \quad 0 \leq x_{ij} \leq c_{ij} \quad \forall ij \in E$$

$$(ii) \quad \forall i \neq s \quad e_i = \sum_{j:ji \in E} x_{ji} - \sum_{j:ij \in E} x_{ij} \geq 0.$$

Numărul e_i $i \in V - \{s, t\}$ se numește **excesul** din vârful i . Dacă $i \in V - \{s, t\}$ și $e_i > 0$ atunci i se numește **nod activ**. Dacă $ij \in E$ x_{ij} va fi numit **fluxul pe arcul ij** .

Dacă în rețeaua R nu există noduri active, atunci **prefluxul x este flux** de la s la t în R de valoare e_t . *Ideea algoritmilor de tip preflux*: se pornește cu **un preflux** în R și **se transformă** prin modificări ale fluxului pe arce **într-un flux care nu admite drumuri de creștere**.

Reprezentarea digrafului G cu ajutorul listelor de adiacență. **Totuși, vom considera că dacă $ij \in E$ atunci și $ji \in E$ (altminteri, adăugăm arcul ji cu capacitate 0).**

Problema fluxului maxim

Algoritmi de tip preflux

x preflux în R , $ij \in E$. **Capacitatea reziduală** a arcului ij este

$$r_{ij} = c_{ij} - x_{ij} + x_{ji}$$

(reprezentînd fluxul adițional ce poate fi "*trimis*" de la nodul i la nodul j utilizînd arcele ij și ji).

A "*trimite*" flux de la i la j înseamnă să creștem fluxul pe arcul ij sau să micșorăm fluxul pe arcul ji .

Se numește **C-drum** în R relativ la prefluxul x , un drum al lui G ale cărui arce au capacitatea reziduală pozitivă.

Se numește *funcție de distanță* în R relativ la prefluxul x , o funcție $d : V \rightarrow \mathbb{Z}_+$ care satisface

$$(D1) \quad d(t) = 0$$

$$(D2) \quad \forall ij \in E, r_{ij} > 0 \Rightarrow d(i) \leq d(j) + 1$$

P C-drum relativ la prefluxul x în R de la i la $t \Rightarrow d(i) \leq \lg(P)$ (arcele lui P au capacitate reziduală pozitivă și se aplică (D2)).

Rezultă că $d(i) \leq \tau_i$ (lungimea minimă a unui C-drum de la i la t).

Problema fluxului maxim

Algoritmi de tip preflux

Fie x un preflux în R și d o funcție de distanță relativ la x . Un arc $ij \in E$ se numește **admisibil** dacă

$$r_{ij} > 0 \quad \wedge \quad d(i) = d(j) + 1.$$

Dacă R este o rețea, considerăm *inițializare* procedura care construiește în $O(m)$ un preflux x și o funcție de distanță d corespunzătoare acestuia:

```
procedure inițializare;  
  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  $d[i] \leftarrow 1$ 
```

Alegerea lui $d(s) = n$ are interpretarea: "**nu există C-drum de la s la t în R relativ la x** " (altfel, ar trebui ca lungimea acestuia să fie $\geq n$).

Dacă, în algoritmi de tip preflux vom păstra acest invariant, atunci când x va deveni flux, va rezulta că nu admite drumuri de creștere și deci x va fi de valoare maximă.



Problema fluxului maxim

Algoritmi de tip preflux

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**, altfel pomparea e **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 \exists noduri active în R **do**

{ selectează un nod activ i ;

if \exists arce admisibile în $A(i)$

then *pompează*(i)

else *reetichetare*(i) }

Problema fluxului maxim

Algoritmi de tip preflux

Lemă 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.

Lemă 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.

Corolar 1. $\forall i \in V \quad d(i) < 2n$.

Corolar 2. Numărul total de apeluri ale procedurii reetichetare este mai mic decât $2n^2$.

Corolar 3. Nr. total de pompări saturate este $\leq nm$.

Lemă (Goldberg și Tarjan 1986) Numărul pompărilor nesaturate este cel mult $2n^2m$.

Lemă La terminarea algoritmului x este flux de valoare maximă.

Problema fluxului maxim

Algoritmi de tip preflux

Algoritmul lui **Ahuja și Orlin (1988)** – cu o metodă de **scalare**, va mărgini numărul pompărilor nesaturate de la $O(n^2m)$ la $O(n^2 \log U)$.

Capacitățile sunt întregi, $\max_{ij \in E}(1 + c_{ij}) = U$. Fie $\lceil \log_2 U \rceil = K$.

Ideia algoritmului:

Se vor executa $K + 1$ etape. Pentru fiecare etapă p , cu p luînd succesiv valorile $K, K - 1, \dots, 1, 0$ 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** (datorită proprietăților funcției distanță $d(i)$).



Problema fluxului maxim

Algoritmul Ahuja-Orlin

inițializare;

$K \leftarrow \lceil \log_2(U) \rceil$; $\Delta \leftarrow 2^{K+1}$;

for $p = K, K - 1, \dots, 0$ **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 $L(p)$; pune pointerul curent al listei $A(i)$ la început }

}

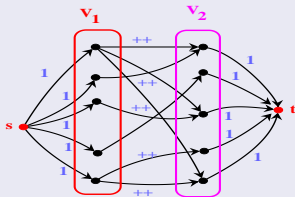
}

Algoritmul Ahuja-Orlin

Lemă. Numărul pompărilor nesaturate este cel mult $8n^2$ în fiecare etapă a scalării, deci $O(n^2 \log U)$ în total.

Teoremă. (Ahuja-Orlin 1988) Algoritmul de tip preflux cu scalarea exceselor are complexitatea $O(nm + n^2 \log U)$.

Aflarea cuplajului maxim și a stabilei maxime într-un graf bipartit.



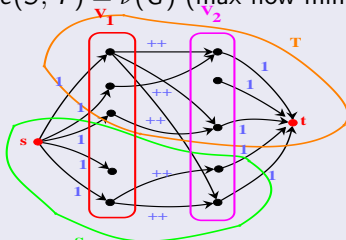
Dacă $x = (x_{ij})$ este un flux cu componente întregi în R atunci se observă că mulțimea de arce $\{ij \mid i \in V_1, j \in V_2 \wedge x_{ij} = 1\}$ induce în graful G bipartit un cuplaj $M(x)$. În plus, $v(x)$ este cardinalul cuplajului $M(x)$.

Reciproc, orice cuplaj din G induce o mulțime de arce neadiacente în G_1 ; dacă pe fiecare astfel de arc ij ($i \in V_1, j \in V_2$) 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 $|M|$.

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 .

Aflarea cuplajului maxim și a stabilei maxime într-un graf bipartit.

Fie (S, T) secțiunea de capacitate minimă ce se obține în $O(m)$, din fluxul maxim aflat. Avem, $c(S, T) = \nu(G)$ (max-flow min-cut).



Cum $\nu(G) < \infty$, rezultă că punînd $S_i = S \cap V_i$ și $T_i = T \cap V_i$ ($i = 1, 2$), avem: $|T_1| + |S_2| = \nu(G)$, iar $X = S_1 \cup T_2$ este **mulțime 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).

Se vor discuta (cel puțin) patru probleme dintre următoarele:

- ① Problemele 4,2 Setul 11
- ② Problemele 1,4 Setul 16
- ③ Problema 2, Setul 15
- ④ Problema 2 Setul 18
- ⑤ Problema 1, Setul 19
- ⑥ Problema 2 Setul 22

ALGORITMICA GRAFURILOR

Săptămâna 10

C. Croitoru

croitoru@info.uaic.ro

FII

December 3, 2014

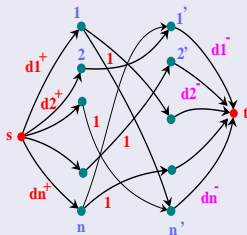
① **Fluxuri** (ag 14-15 allinone.pdf pag. 244 → 264)

② **Problemele pentru seminarul 10**

Recunoașterea secvențelor digrafice.

Date $(d_i^+)_{i=1,n}$ și $(d_i^-)_{i=1,n}$, **există un digraf** G cu n vîrfuri astfel încît $G = (\{1, \dots, n\}, E)$ și $d_G^+(i) = d_i^+$ și $d_G^-(i) = d_i^- \ \forall i = 1, n$?

Dacă $0 \leq d_i^+ \leq n-1$ 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 $d_i^+, d_i^- \in \mathbf{Z}$),
construim rețeaua bipartită:



Răspunsul este da dacă și numai dacă în această rețea, există un flux întreg de valoare maximă m .

Aplicații combinatorii ale fluxurilor

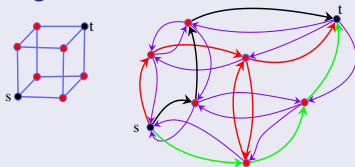
Determinarea numărului de muchie-conexiune al unui graf.

Fie $G = (V, E)$ un graf. Pentru $s, t \in V, s \neq t$, definim:

- $p_e(s, t)$ = nr. maxim de drumuri cu muchii disjuncte ce unesc s și t .
- $c_e(s, t)$ = cardinalul minim al unei mulțimi de muchii, prin îndepărtarea căreia din graf, între s și t nu mai există drumuri.

Construim din G digraful G_1 , înlocuind fiecare muchie cu o pereche de arce simetrice. Considerăm $c : E(G_1) \rightarrow \mathbf{Z}_+, c(e) = 1, \forall e \in E(G_1)$.

Fie x^0 un flux întreg de valoare maximă în $R = (G_1, s, t, c)$.



Fluxul pe arcele groase
este 1, pe cele subțiri 0.

Rezultă că $v(x^0) = p_e(s, t)$. Dacă (S, T) e secțiune de capacitate minimă, avem $c(S, T) = v(x^0) = c_e(s, t)$.

Corolar G conex:

$$\lambda(G) = \min_{\substack{s, t \in V(G) \\ s \neq t}} c_e(s, t).$$

(*)



Determinarea numărului de muchie-conexiune al unui graf.

Pentru a afla $\lambda(G)$ ar trebui să rezolvăm cele $\frac{n(n-1)}{2}$ probleme de flux din (*).

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 $c(s_0, t_0) = \lambda(G)$.

Rezultă: în $O(n \cdot (nm + n^2c)) = O(n^2m)$ se pot determina $\lambda(G)$ și o mulțime separatoare de muchii de cardinal minim în G .

Aplicații combinatorii ale fluxurilor

Determinarea numărului de conexiune al unui graf.

$G = (V, E)$ este un graf și $s, t \in V$, $s \neq t$, $st \notin E$

- $p(s, t)$ = numărul maxim de st -drumuri cu mulțimile de vârfuri disjuncte (cu excepția extremităților),

- $c(s, t)$ = cardinalul minim al unei mulțimi de vârfuri st -separatoare,

Teorema lui Menger \Rightarrow

$$p(s, t) = c(s, t) \quad (*)$$

Numărul de conexiune $k(G)$ al grafului G este

$$k(G) = \begin{cases} n-1 & \text{dacă } G = K_n \\ \min_{\substack{s, t \in V \\ st \notin E}} c(s, t) & \text{dacă } G \neq K_n \end{cases} \quad (**)$$

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 $b_v a_w, b_w a_v \in E(G_1)$.

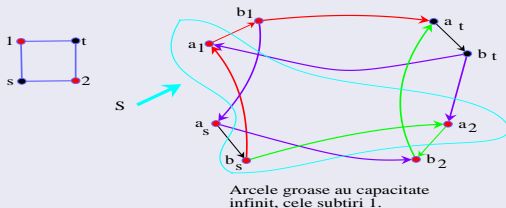
Definim $c : E(G_1) \rightarrow \mathbf{Z}_+$ prin

$$c(e) = \begin{cases} 1 & \text{dacă } e = a_v b_v \\ \infty & \text{altfel.} \end{cases}$$

Aplicații combinatorii ale fluxurilor

Determinarea numărului de conexiune al unui graf.

Considerăm rețeaua $R = (G_1, b_s, a_t, c)$. Exemplu:



x^0 flux întreg de la b_s la a_t în R de valoare maximă $\Rightarrow v(x^0) = p(s, t)$.

(S, T) o secțiune în R a.î. $v(x^0) = c(S, T)$. \Rightarrow

$c(s, t) = |A_0| = c(S, T) = v(x_0) = p(s, t)$, unde A_0 este o mulțime de vîrfuri st -separatoare de cardinal minim.

Pentru a determina $k(G)$ determinăm minimul din $(**)$ prin rezolvarea a $|E(\bar{G})|$ probleme de flux. Se poate construi un algoritm de complexitate $O(m(nm + n^2 \log n))$.

Flux de cost minim.

Fie $R = (G, s, t, c)$ o rețea și x un flux de la s la t în R .

Considerăm $a : E \rightarrow \mathbf{R}$ o funcție de cost care asociază fiecărui arc $ij \in E$ $a(ij) = a_{ij}$ costul (transportului unei unități de flux) pe arcul ij . **Costul fluxului** x se definește ca fiind

$$a(x) = \sum_{i,j} a_{ij}x_{ij}.$$

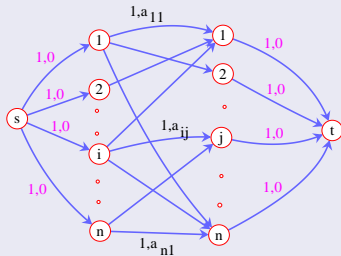
Problema fluxului de cost minim

Dată R o rețea, $v \in \mathbf{R}^+$ și $a : E \rightarrow \mathbf{R}$ funcție de cost, să se determine x^0 flux în R astfel încât

$$a(x^0) = \min\{a(x) \mid x \text{ flux în } R, v(x) = v\}.$$

Problema simplă a atribuirii Se dispune de n lucrători și n lucrări. Costul atribuirii lucrătorului i la lucrarea j este $a_{ij}(i, j \in \{1, \dots, n\})$. **Să se atribuiască fiecare dintre cele n lucrări la câte un lucrător, astfel încât costul total al atribuirii să fie minim.**

Problema atribuirii.

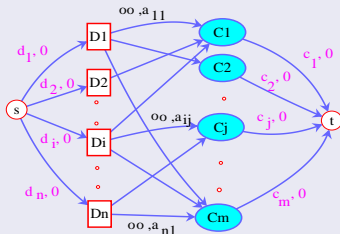


Un flux întreg de valoare n și de cost minim, oferă soluția problemei atribuirii.

Problema simplă a transporturilor (Hitchcock-Koopmans): O marfă disponibilă în depozitele D_1, \dots, D_n în cantitățile d_1, \dots, d_n este solicitată în centrele de consum C_1, C_2, \dots, C_m în cantitățile c_1, c_2, \dots, c_m . Se cunoaște costul a_{ij} al transportului unei unități de marfă de la depozitul D_i la centrul de consum C_j ($\forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, m\}$). Se cere să se stabilească un plan de transport care să satisfacă toate cererile și să aibă costul total minim

Problema transporturilor

Dacă $\sum_{i=1,n} d_i \geq \sum_{j=1,m} c_j$, un flux de cost minim și de valoare $v = \sum_{i=1,m} c_i$ în rețeaua următoare, rezolvă problema.



Definiție. Fie x un flux în $R = (G, s, t, c)$ și $a : E \rightarrow \mathbf{R}$ o funcție de cost. P C-drum în R relativ la fluxul x , \Rightarrow **costul drumului P** este

$$a(P) = \sum_{\substack{ij \in P \\ \text{direct}}} a_{ij} - \sum_{\substack{ij \in P \\ \text{invers}}} a_{ji}.$$

Dacă C este un C-drum închis, $a(C)$ se calculează după aceeași formulă, după stabilirea unui sens de parcurgere a lui C .

Soluția problemei fluxului de cost minim.

Dacă P este drum de creștere relativ la fluxul x , atunci $x^1 = x \otimes r(P)$ este un flux de valoare $v(x^1) = v(x) + r(P)$ și de cost $a(x) + r(P) \cdot a(P)$.

Dacă C este un C-drum închis relativ la x , atunci $x^1 = x \otimes r(C)$ este un flux de valoare $v(x^1) = v(x)$ și de cost $a(x^1) = a(x) + r(C) \cdot a(C)$.

Dacă $a(C) < 0$ atunci x^1 este un flux de aceeași valoare ca și x , dar de cost strict mai mic.

Teoremă. *Un flux de valoare v este de cost minim dacă și numai dacă nu admite C-drumuri închise de cost negativ.*

Teoremă. *Dacă x este un flux de valoare v și de cost minim iar P_0 este un drum de creștere, astfel încât*

*$a(P_0) = \min\{a(P) \mid P \text{ drum de creștere relativ la } x\}$,
atunci $x^1 = x \otimes r(P_0)$ este un flux de valoare $v(x^1) = v + r(P_0)$ și de cost minim.*

Algoritm generic de rezolvare a problemei fluxului de cost minim.

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 $\bar{a}_{ij})$ **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 \bar{a}_{ij} pentru depistarea unui
 C-drum P de cost minim;
 $x \leftarrow x \otimes \min(r(P), v - v(x)) \}$

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 iterațiilor să fie $O(nm^2 \log n)$.

Se vor discuta (cel puțin) patru probleme dintre următoarele:

- ① **Problema 2 Setul 8'**
- ② **Problemele 1 și 4 Setul 9**
- ③ **Problemele 3 și 4, Setul 10**
- ④ **Problema 1 Setul 11'**
- ⑤ **Problema 1, Setul 11''**
- ⑥ **Problema 4 Setul 25**

ALGORITMICA GRAFURILOR

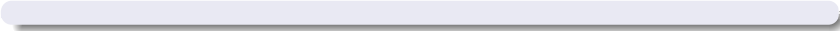
Săptămâna 11

C. Croitoru

croitoru@info.uaic.ro

FII

December 10, 2014

- 
- ① Reduceri polinomiale pentru probleme de decizie pe grafuri (ag 14-15 [allinone.pdf](#) pag. 265 → 308)
 - ② Problemele pentru seminarul 11

Stabila maximă.

SM

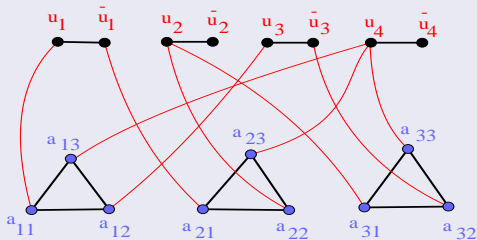
Instanță : $G = (V, E)$ graf și $k \in \mathbf{N}$.

Întrebare: Există S mulțime stabilă în G a. î. $|S| \geq k$?

Teoremă. (Karp 1972) $3SAT \propto SM$.

Exemplu: $U = \{u_1, u_2, u_3, u_4\}$;

$C = (u_1 \vee u_3 \vee u_4) \wedge (\bar{u}_1 \vee u_2 \vee u_4) \wedge (u_2 \vee \bar{u}_3 \vee u_4)$; $k = 4 + 3 = 7$.



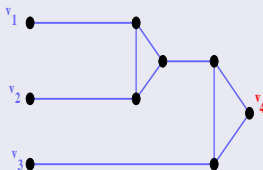
Colorarea vârfurilor.

COL

Instanță: $G = (V, E)$ graf și $p \in \mathbf{N}^*$.

Intrebare: Există o p -colorare a vârfurilor lui G ?

Lemă. Fie H graful:



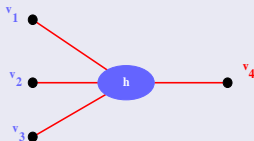
a) Dacă c este o 3-colorare a lui H astfel încât

$c(v_1) = c(v_2) = c(v_3) = a \in \{1, 2, 3\}$ atunci $c(v_4) = a$.

b) Fie $c : \{v_1, v_2, v_3\} \rightarrow \{1, 2, 3\}$ a.î. $c(\{v_1, v_2, v_3\}) \neq \{a\}$.

Atunci c poate fi extinsă la o 3-colorare a lui H cu $c(v_4) \neq a$.

Vom desemna (pentru simplitate) graful H astfel:



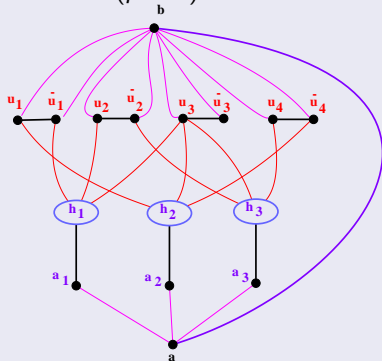
Colorarea vârfurilor.

Teoremă. $3SAT \propto COL$.

Exemplu: $U = \{u_1, u_2, u_3, u_4\}$,

$$C = (\bar{u}_1 \vee u_2 \vee u_3) \wedge (u_1 \vee u_3 \vee \bar{u}_4) \wedge (\bar{u}_2 \vee u_3 \vee u_4)$$

Graful G va fi ($p = 3$):



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. (Nash-Williams 1969) Problemele următoare sînt polinomial echivalente:

CH : *Dat G graf. Este G hamiltonian ?*

TR : *Dat G graf. Este G trasabil ?*

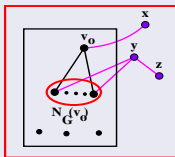
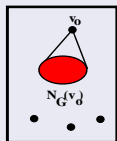
DCH: *Dat G digraf. Este G hamiltonian ?*

DTR: *Dat G digraf. Este G trasabil ?*

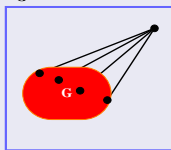
BCH: *Dat G graf bipartit. Este G hamiltonian ?*

Probleme hamiltoniene.

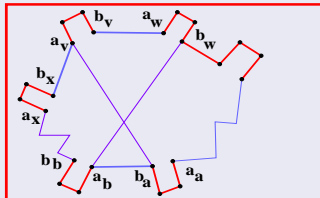
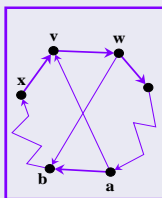
$CH \propto TR$



$TR \propto CH$



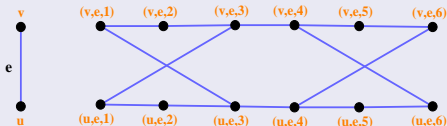
$DCH \propto CH$



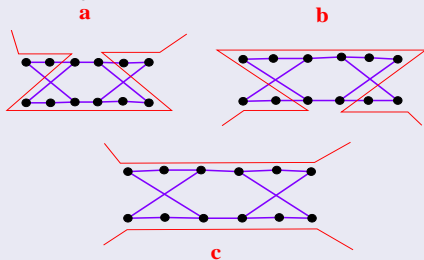
Probleme hamiltoniene.

Teoremă. (Karp 1972) $SM \propto CH$.

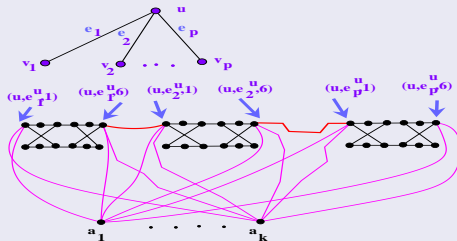
Pentru orice muchie a grafului G (intrare în SM) asociem graful



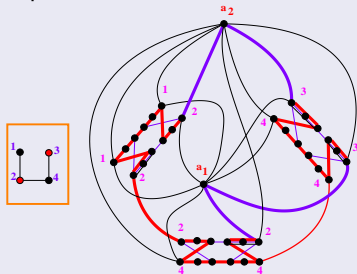
Singurele posibilități de traversare de către un circuit hamiltonian al lui H a vîrfurilor din G'_e sînt (a) (b) și (c) indicate în figura următoare:



Probleme hamiltoniene.



Exemplu:



Problema comisului voiajor

Algoritmi de aproximare.

CV Dat $n \in \mathbf{Z}_+$ ($n \geq 3$) și $d : E(K_n) \rightarrow \mathbf{R}_+$, 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 .

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 soluția optimă H_0 .

Măsuri ale eficienței unei astfel de "euristici" A pot fi considerate numerele:

$$R_A(n) = \sup_{\substack{d: E(K_n) \rightarrow \mathbf{R}_+ \\ d(H_0) \neq 0}} \frac{d(H_A)}{d(H_0)}$$

$$R_A = \sup_{n \geq 3} R_A(n).$$

Teoremă. Dacă există un algoritm aproximativ A cu timp de lucru polinomial pentru CV, astfel încât $R_A < \infty$, atunci CH se poate rezolva în timp polinomial.

$\mathbf{P} \neq \mathbf{NP} \Rightarrow$ nu există algoritm aproximativ A polinomial cu $R_A < \infty$.



Se vor discuta problemele restante din săptămâna 10 și cele din tema 2.



ALGORITMICA GRAFURILOR

Săptămâna 12

C. Croitoru

croitoru@info.uaic.ro

FII

December 17, 2014

- 1 Abordări ale unor probleme NP-dificile pe grafuri
(ag 14-15 allinone.pdf pag. 309 → 337)
- 2 Grafuri planare (ag 14-15 allinone.pdf pag. 338 →)
- 3 Problemele pentru seminarul 12
- 4 **Prezentarea temei pentru acasă**

Colorarea vârfurilor unui graf

Algoritmul greedy de colorare Fie $G = (V, E)$, cu $V = \{1, 2, \dots, n\}$ și π o permutare a lui V . Se construiește colorarea c ce utilizează $\chi(G, \pi)$ culori, $c : \{1, 2, \dots, n\} \longrightarrow \{1, 2, \dots, \chi(G, \pi)\}$.

```
-  $c(\pi_1) \leftarrow 1; \chi(G, \pi) \leftarrow 1; S_1 \leftarrow \{\pi_1\};$   
- for  $i \leftarrow 2$  to  $n$  do  
- {  $j \leftarrow 0;$   
-   repeat  
-      $j \leftarrow j + 1;$   
-     fie  $v$  primul vârf (cf. ordonării  $\pi$ ), 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  
-     {  $c(\pi_i) \leftarrow j + 1;$   
-        $S_{j+1} \leftarrow \{\pi_i\};$   
-        $\chi(G, \pi) \leftarrow j + 1;$   
-     }  
- }
```


Colorarea vârfurilor unui graf

Algoritmul Dsat *{ Degree of Saturation }*

Acest algoritm (Brélaz, 1979) este o metodă secvențială dinamică de colorare.

Ideea este de a 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 greedy) deoarece se aleg vârfuri care formează "clici" mari în raport cu vârfurile deja alese (spre deosebire de mulțimi stabile mari în cazul greedy).

Dacă G este un graf și c o colorare parțială a vârfurilor lui G , definim *gradul de saturație* al unui vârf v , notat $d_{sat}(v)$, ca fiind numărul de culori diferite din vecinătatea acestuia.

- 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;
- }

Algoritmul DSatur garantează găsirea numărului cromatic pentru grafurile bipartite.

Problema comisului voiajor

O euristică populară pentru problemele în care funcția de distanță satisface inegalitatea triunghiulară, este dată de *Christofides*. Spre deosebire de cazul general când nu se poate spera la o euristică polinomială A cu R_A finită, dacă $P \neq NP$, în acest caz se poate demonstra următoarea

Teoremă. (Christofides,1973) Fie CV cu d satisfăcând $\forall v, w, u \in V(K_n)$ distincte $d(vw) \leq d(vu) + d(uw)$. Există un algoritm aproximativ A pentru CV care satisface $R_A = \frac{3}{2}$ și are timp de lucru polinomial.

- 1 Se determină T^0 mulțimea muchiilor unui arbore parțial de cost minim în K^n (costul muchiei e fiind $d(e)$). *Algoritmul lui Prim*.
- 2 Se determină M^0 un cuplaj perfect în subgraful indus de vîrfurile de grad impar ale arborelui T^0 și de cost minim. *Alg. lui Edmonds*.
- 3 Se consideră multigraful obținut din $\langle T^0 \cup M^0 \rangle_{K_n}$, prin duplicarea muchiilor din $T^0 \cap M^0$. Există un parcurs Eulerian închis, cu vîrfurile $(v_{i_1}, v_{i_2}, \dots, v_{i_1})$. Eliminînd aparițiile multiple a unui vîrf, cu excepția primului și ultimului, se obține un circuit hamiltonian H_A în K_n cu muchiile $H_A = (v_{j_1} v_{j_2}, v_{j_2} v_{j_3}, \dots, v_{j_n} v_{j_1})$ ($O(n^2)$ operații).

Metode care imită natura

Metaheuristica *simulated annealing*.

Inspirată din termodinamică, inventată independent de *Kirkpatrick, Gelatt și Vechi* în 1983 și de *Černý* în 1985.

Probabilitatea de creștere a energiei scade odată cu temperatura:

$$Pr(E + dE) = Pr(E) \cdot e^{-\frac{dE}{kT}}$$

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 $f : D \rightarrow \mathbf{R}$, 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 Boltzmann va fi considerată 1.

Algoritm de călire simulată

1. **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}

Pentru problema comisului voiajor, se pornește cu un tur ales aleator, iar soluțiile vecine se obțin cu 2-move. Se

poate lua $T_{start} = O(\sqrt{n})$, $attempts = 100n$, $changes = 10n$, $decrease(T) = 0.95T$ și $T_{min} = O(1)$.

Proprietăți de bază.

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 încât:

- 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**.

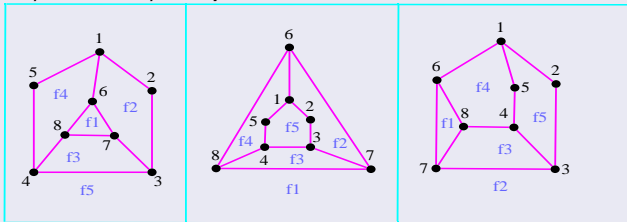
Proprietăți de bază.

Lemă. *Un graf este planar dacă și numai dacă este reprezentabil pe o sferă.*

proiecția stereografică !

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 .

Orice graf plan are un număr finit de fețe, dintre care una singură este nemărginită și se numește **față exterioară** a lui G .



Proprietăți de bază.

Lemă. *Orice reprezentare planară a unui graf poate fi transformată într-o reprezentare diferită astfel încât o față specificată a sa să devină fața exterioară.*

Teoremă. (Formula lui Euler) *Fie $G = (V, E)$ un graf plan conex cu n vîrfuri, m muchii și f fețe. Atunci*

$$f = m - n + 2$$

Din punct de vedere algoritmic, teorema are drept consecință 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 $O(|V|)$ operații.

Corolar 1. *Fie G un graf planar, conex, cu $n(\geq 3)$ vîrfuri și $m > 2$ muchii. Atunci*

$$m \leq 3n - 6.$$

Graful K_5 nu este planar !

Proprietăți de bază.

Corolar 2. Dacă G este un graf bipartit, conex și planar cu $m > 2$ muchii și n vârfuri, atunci $m \leq 2n - 4$.

Graful K_{33} nu este planar.

Corolar 3. Dacă G este un graf planar conex, atunci G are un vârf de grad cel mult 5.

Fie $G = (V, E)$ un graf și $v \in V(G)$ astfel încât $d_G(v) = 2$ și $vw_1, vw_2 \in E$, $w_1 \neq w_2$. Fie $h(G) = (V \setminus \{v\}, E \setminus \{vw_1, vw_2\} \cup \{w_1w_2\})$. Se observă că G este planar dacă și numai dacă $h(G)$ este planar.

Vom nota cu $h^*(G)$ graful obținut din G aplicîndu-i repetat transformarea h , pînă cînd graful curent nu mai are vârfuri de grad 2.

Rezultă că G este planar, dacă și numai dacă $h^*(G)$ este planar.

Definiție. Două grafuri G_1 și G_2 se numesc *homeomorfe* dacă și numai dacă $h^*(G_1) \cong h^*(G_2)$.

Teoremă. (Kuratowski 1930) Un graf este planar dacă și numai dacă nu are subgrafuri homeomorfe cu K_5 sau K_{33} .

Problemele pentru seminarul 12

- 1 Problema 4 Setul 10
- 2 Problema 3 Setul 11
- 3 Problema 4, Setul 11'
- 4 Problema 4 Setul 15
- 5 Problema 3, Setul 22
- 6 Problema 2 Setul 19
- 7 Problema 4 Setul 22
- 8 Problema 2, Setul 25

LA MULȚI ANI!



ALGORITMICA GRAFURILOR

Săptămâna 13

C. Croitoru

croitoru@info.uaic.ro

FII

January 7, 2015

LA MULȚI ANI!



-
- 1 **Grafuri planare** (ag 14-15 [allinone.pdf](#) pag. 338 → 374)
 - 2 **Problemele pentru seminariile 13 și 14**

Proprietăți de bază.

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 încât:

- 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 vârf 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**.

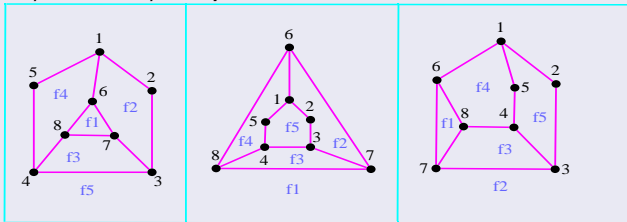
Proprietăți de bază.

Lemă. *Un graf este planar dacă și numai dacă este reprezentabil pe o sferă.*

proiecția stereografică !

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 .

Orice graf plan are un număr finit de fețe, dintre care una singură este nemărginită și se numește **față exterioară** a lui G .



Proprietăți de bază.

Lemă. *Orice reprezentare planară a unui graf poate fi transformată într-o reprezentare diferită astfel încât o față specificată a sa să devină fața exterioară.*

Teoremă. (Formula lui Euler) *Fie $G = (V, E)$ un graf plan conex cu n vîrfuri, m muchii și f fețe. Atunci*

$$f = m - n + 2$$

Din punct de vedere algoritmic, teorema are drept consecință 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 $O(|V|)$ operații.

Corolar 1. *Fie G un graf planar, conex, cu $n(\geq 3)$ vîrfuri și $m > 2$ muchii. Atunci*

$$m \leq 3n - 6.$$

Graful K_5 nu este planar !

Proprietăți de bază.

Corolar 2. Dacă G este un graf bipartit, conex și planar cu $m > 2$ muchii și n vârfuri, atunci $m \leq 2n - 4$.

Graful K_{33} nu este planar.

Corolar 3. Dacă G este un graf planar conex, atunci G are un vârf de grad cel mult 5.

Fie $G = (V, E)$ un graf și $v \in V(G)$ astfel încât $d_G(v) = 2$ și $vw_1, vw_2 \in E$, $w_1 \neq w_2$. Fie $h(G) = (V \setminus \{v\}, E \setminus \{vw_1, vw_2\} \cup \{w_1w_2\})$. Se observă că G este planar dacă și numai dacă $h(G)$ este planar.

Vom nota cu $h^*(G)$ graful obținut din G aplicîndu-i repetat transformarea h , pînă cînd graful curent nu mai are vârfuri de grad 2.

Rezultă că G este planar, dacă și numai dacă $h^*(G)$ este planar.

Definiție. Două grafuri G_1 și G_2 se numesc *homeomorfe* dacă și numai dacă $h^*(G_1) \cong h^*(G_2)$.

Teoremă. (Kuratowski 1930) Un graf este planar dacă și numai dacă nu are subgrafuri homeomorfe cu K_5 sau K_{33} .

Desenarea unui graf planar.

Fary 1948 (independent Wagner și Stein):

Orice graf planar are o reprezentare planară cu toate muchiile segmente de dreaptă (reprezentarea Fary).

Existența unei reprezentări Fary cu **vîrfuri în puncte de coordonate întregi** și, în același timp, aria suprafeței ocupate de reprezentare să fie polinomială în raport cu numărul n de vîrfuri ale grafului !

Teoremă. (Frayssseix, 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.*

Demonstrația: algoritm de complexitate $O(n \log n)$ pentru obținerea acestei reprezentări.

Vom demonstra teorema în ipoteza suplimentară că G este **maximal planar** : orice muchie i s-ar adăuga se obține un graf neplanar (sau multigraf). Să observăm că orice față a unui graf maximal planar este un C_3 (altminteri în reprezentarea lui G cu fața 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 obține aplicînd de exemplu algoritmul de testare a planarității) se trece la o hartă cu toate fețele triunghiului prin inserția în timp liniar de corzi în circuite. La desenarea grafului obținut, muchiile fictive introduse nu se vor trasa.

Grafuri plane - versiunea combinatorială.

În versiunea combinatorială un graf este un triplet $G = (E, \theta, -)$, unde E este o mulțime de cardinal par, $-$ este o *involuție* pe E (permutare de ordin 2) fără puncte fixe, și θ este o permutare pe E .

Elementele lui E sunt gândite ca *arce*; o muchie (neorientată) este reprezentată ca o pereche $e, \bar{e} \in E$ de arce, inverse unul altuia. Aplicația $-$ inversează direcția.

Se dorește ca aplicația 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 submulțime nevidă a lui E închisă în raport cu θ și minimală cu această proprietate).

Dacă notăm cu V mulțimea ciclilor permutării θ atunci definim
 $t : E \rightarrow V$, $t(e) =$ **unicul ciclu al lui θ ce conține e** (extrem. inițială a lui e)
 $h : E \rightarrow V$, $h(e) =$ **unicul ciclu al lui θ ce conține \bar{e}** (extrem. finală a lui e)
Se observă că $\forall e$ $t(e) = h(\bar{e})$ și $h(e) = t(\bar{e})$.

Dacă $\theta^* : E \rightarrow E$ definită de $\theta^*(e) = \theta(\bar{e})$, atunci o **față** a lui G este un ciclu al permutării θ^* . Intuitiv, pentru a calcula $\theta^*(e)$, inversăm e pentru a obține \bar{e} și apoi ne rotim (în sensul acelor de ceasornic) în jurul extremității inițiale a lui \bar{e} . Numărul fețelor lui G se notează cu f .

Grafuri plane - versiunea combinatorială.

O **componentă conexă** a lui G este o *orbită* a lui E în grupul de permutări generat de θ și \neg : **o mulțime nevidă minimală cu proprietatea că este închisă la θ și \neg .**

Fie G un graf cu $m = \frac{1}{2}|E|$ muchii (neorientate), $n = |V|$ vârfuri, f fețe, și c componente conexe. **Caracteristica Euler** a lui G se definește ca fiind

$$\chi(G) = 2c + m - n - f.$$

Un graf G se numește **graf plan** dacă $\chi(G) = 0$.

Se poate demonstra că pentru un graf conex în definiția tradițională, cele două noțiuni de grafuri plane coincid

(graful neorientat construit așa cum am descris mai sus atașat unui graf în formă combinatorială este graf plan conform definiției tradiționale și invers,

dacă pentru un graf tradițional plan conex se construiește θ conform unei orientări inverse acelor de ceasornic a muchiilor și \neg corespunzătoare, graful combinatorial obținut este plan în noua definiție).

Teorema Separatorului.

Teoremă. (Tarjan & Lipton, 1979) Fie G un graf planar cu n vârfuri. Există o partiție 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 de la A la B .
2. $|A| \leq \frac{2}{3}n$, $|B| \leq \frac{2}{3}n$.
3. $|S| \leq 4\sqrt{n}$.

Această partiție se poate afla în timpul $O(n)$.

Demonstrație. Considerăm graful conex și de asemenea considerăm că dispunem de o reprezentare planară.

Alegem un vârf s și executăm o parcurgere bfs din s numerotând vârfurile (în ordinea întâlnirii lor în această parcurgere) și 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).

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:

$$|\cup_{t < t_1} L(t)| < \frac{n}{2} \quad \wedge \quad |\cup_{t > t_1} L(t)| < \frac{n}{2}.$$

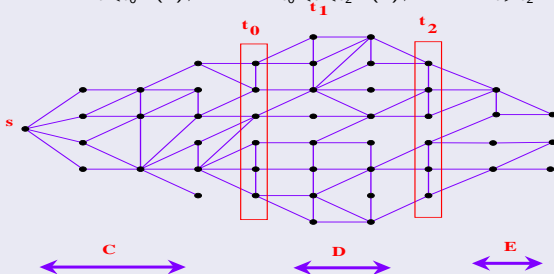
Dacă și $|L(t_1)| \leq 4\sqrt{n}$, teorema are loc.

Teorema Separatorului.

Lemă. 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. Considerăm

$$C = \cup_{t < t_0} L(t), \quad D = \cup_{t_0 < t < t_2} L(t), \quad E = \cup_{t > t_2} L(t).$$



Teorema Separatorului.

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 celorlalte două (nu uităm că C și E au cel mult $\frac{n}{2}$ elemente).

Considerăm deci că $n_1 = |D| > \frac{2}{3}n$.

Dacă vom găsi un separator de tipul $\frac{2}{3} \leftrightarrow \frac{2}{3}$ pentru D cu cel mult $2\sqrt{n}$ vârfuri, atunci

îl vom adăuga la $L(t_0) \cup L(t_2)$ pentru a obține un separator de cardinal cel mult $4\sqrt{n}$,

reunim mulțimea cu cel mai mare număr de elemente dintre C și E cu partea mică rămasă din D pentru a obține A ,

iar partea mare rămasă în D o reunim cu cealaltă mulțime (mică) dintre C și E pentru a obține B .

Construcția separatorului pentru D . Vom șterge toate vârfurile grafului care nu-s în D cu excepția lui s pe care-l unim cu toate vîrfurile de pe nivelul $t_0 + 1$ (primul nivel rămas în D). Graful obținut îl notăm cu D și este evident planar și conex. În plus are un arbore parțial T de diametru cel mult $2\sqrt{n}$ (orice vârf este accesibil din s pe un drum de lungime cel mult \sqrt{n} așa cum am arătat în lemă). Acest arbore se parcurge dfs și se construiește separatorul dorit.

Teorema Separatorului. Aplicație.

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 $O(\cdot)$ să ascundă constante mari !



Problemele pentru seminariile 13 și 14

① <http://profs.info.uaic.ro/~croitoru/ag/Examen/>

ALGORITMICA GRAFURILOR

Săptămâna 14

C. Croitoru

croitoru@info.uaic.ro

FII

January 14, 2015



-
- ① **Tree decomposition and its uses**
 - ② **Anunțuri**

Definition

A **tree decomposition** of a graph $G = (V, E)$ is a pair $(T, \{V_t : t \in T\})$, where T is a tree and $\{V_t : t \in T\}$ denotes a family of subsets of the **vertices** of G , $V_t \subseteq V$ for every **node** $t \in T$ such that:

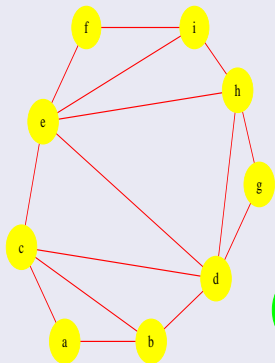
- (*Node coverage*) For every $v \in V$, there is some node t in T such that $v \in V_t$.
- (*Edge coverage*) For every $e \in E$, there is some node t in T such that V_t contains both endpoints of e .
- (*Coherence*) Let t_1, t_2, t_3 be three nodes in T such that t_2 lies on the path between t_1 and t_3 in T . Then, if $v \in V$ belongs to both V_{t_1} and V_{t_3} , v must also belong to V_{t_2} .

Definition - Comment

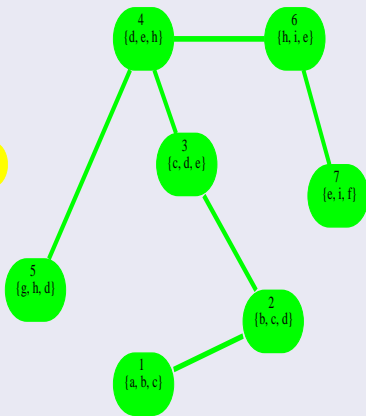
- (*Coherence*) Let t_1, t_2, t_3 be three nodes in T s. t. t_2 lies on the path between t_1 and t_3 in T . Then, if $v \in V$ belongs to both V_{t_1} and V_{t_3} , v must also belong to V_{t_2} .
- (*Coherence'*) Let t_1, t_2, t_3 be three nodes in T s.t. t_2 lies on the path between t_1 and t_3 in T . Then, $V_{t_1} \cap V_{t_3} \subseteq V_{t_2}$.
- (*Coherence''*) For every $x \in V$, the subgraph of T induced by $\{t \in T : x \in V_t\}$ is connected.

The sets V_t are called the **bags** of the tree decomposition.

Definition - Example



Graful G



Arborele T

Definition

Let $(T, \{V_t : t \in T\})$ be a tree decomposition of G .
The **width** of tree decomposition $(T, \{V_t : t \in T\})$ is

$$\text{width}(T, \{V_t : t \in T\}) = \max_{t \in T} |V_t| - 1.$$

Definition

The **tree-width** of G , denoted $\text{tw}(G)$, is *the minimum width of a tree decomposition of G .*

Observation. $tw(G) = 0$ if and only if $E(G) = \emptyset$.

Proposition. *If G is a forest with $E(G) \neq \emptyset$, then $tw(G) = 1$.*

Proof: $tw(G) \geq 1$, by the above observation.

If G is a tree then let T be obtained from G by renaming t_v each vertex $v \in V(G)$ and, after that, inserting on each edge $t_u t_v$ ($uv \in E(G)$) a new vertex t_{uv} .

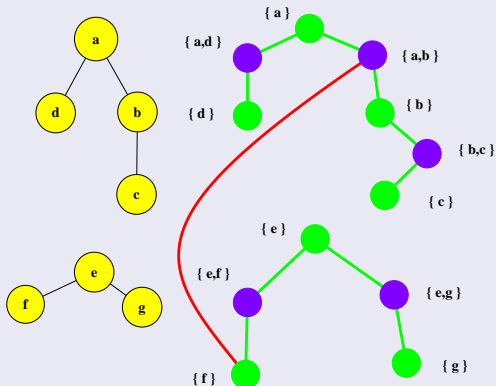
Set $V_{t_u} = \{u\}$ for all t_u associated to $u \in V(G)$, and $V_{t_{uv}} = \{u, v\}$ for all $t_{uv} \in T$ associated to $uv \in E(G)$.

$(T, \{V_t : t \in T\})$ is a tree decomposition of G with width 1.

A tree decomposition of a forest with k components can be obtained by adding $k - 1$ arbitrary edges to tree decompositions for the components (without creating circuits). □

Tree-width

Tree decomposition with width 1 of a forest



Small tree decompositions

A tree decomposition $(T, \{V_t : t \in T\})$ is **small** if there are no distinct $t_1, t_2 \in T$ such that $V_{t_1} \subseteq V_{t_2}$.

Proposition. *Given a tree decomposition of G , a small tree decomposition of G with the same width can be constructed in polynomial time.*

Proof. Let $(T, \{V_t : t \in T\})$ be a tree decomposition of G with $V_{t_1} \subseteq V_{t_2}$ for $t_1, t_2 \in T$. We can suppose that $t_1 t_2 \in E(T)$ (otherwise, we find adjacent nodes with this property, by considering a path from t_1 to t_2).

Contracting $t_1 t_2$ into a **new node** t_{12} with $V_{t_{12}} = V_{t_2}$, gives a smaller tree decomposition of G . Repeat this reduction until a small tree decomposition is obtained. □

Small tree decompositions

Proposition. *If $(T, \{V_t : t \in T\})$ is a small tree decomposition of G , then $|T| \leq |G|$.*

Proof. By induction over $n = |G|$. If $n = 1$ then $|T| = 1$.

For $n \geq 2$, consider a leaf t_1 of T with neighbor t_2 .

$(T - t_1, \{V_t : t \in T - t_1\})$ is a small tree decomposition of $G' = G - (V_{t_1} - V_{t_2})$. Since $V_{t_1} - V_{t_2} \neq \emptyset$, by induction

$$|T| = |T - t_1| + 1 \leq |G'| + 1 \leq |G|.$$



Minors

Observations.

- If the graph H is obtained from G by contracting an edge uv into z , then $tw(H) \leq tw(G)$. In a tree decomposition of G , insert z in every bag containing u or v , and then remove u and v from every bag to obtain a tree decomposition of H .
- If H is a subgraph G , then $tw(H) \leq tw(G)$.

Definition. H is a **minor** of a graph G if it can be obtained from G by iteratively **deleting and contracting** edges.

Corollary. *If H is a minor of a graph G then $tw(H) \leq tw(G)$.*

Let $\mathbf{TW}(k)$ be the class of graphs G such that $tw(G) \leq k$.

Tree-Width (Decision Version)

Input: Graph G and integer k .

Question: $G \in \mathbf{TW}(k)$?

Theorem

Tree-width (decision version) is NP-complete.

Tree-Width is FPT

Lemma. *For every positive integer k , $\mathbf{TW}(k)$ is minor closed.*

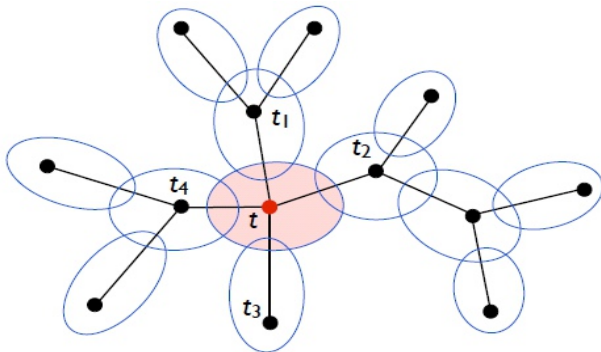
Theorem (Bodlaender). *For every fixed k , the problem of determining whether or not $G \in \mathbf{TW}(k)$ can be solved in $\mathcal{O}(f(k) \cdot n)$ time.*

Notation. Let $(T, \{V_t : t \in T\})$ be a tree decomposition of G . Then, if T' is a subgraph of T , $G_{T'}$ denotes the subgraph of G induced by the set $\bigcup_{t \in T'} V_t$.

Tree decomposition properties

Node Separation Property

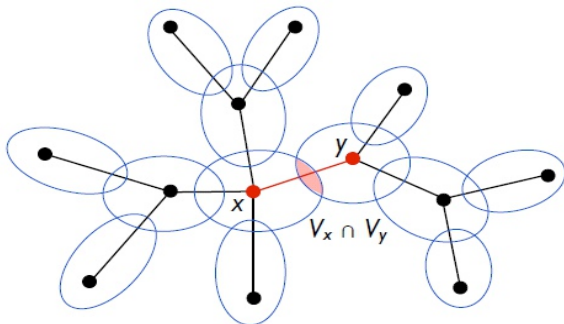
Theorem. Suppose $T - t$ has components T_1, \dots, T_d . Then, the subgraphs $G_{T_1} - V_t, G_{T_2} - V_t, \dots, G_{T_d} - V_t$ have no nodes in common, and there are no edges between them.



Tree decomposition properties

Edge Separation Property

Theorem. Let X and Y be the two components of T after the deletion of edge xy . Then, deleting $V_x \cap V_y$ disconnects G into two subgraphs $H_X = G_X - (V_x \cap V_y)$ and $H_Y = G_Y - (V_x \cap V_y)$. That is, H_X and H_Y share no nodes and there is no edge in G with one endpoint in H_X and the other in H_Y .



Exercises

1. Let G be a connected graph with $tw(G) = p$. Prove that $|V(G)| = p + 1$ or G has a p -cut.
2. Prove that if $tw(G) = 1$, then G is a forest.
3. Prove that $tw(P_k \times P_l) = \min(k, l)$.
4. Prove that $tw(K_n) = n - 1$.

Definition

A **rooted tree decomposition** of G is a tree decomposition $(T, \{V_t : t \in T\})$ of G , where some node r in T is declared to be the root.

Notations. Let t be a node in a rooted tree decomposition.

- T_t is the subtree of T rooted at t .
- $G[t]$ is the subgraph of G induced by the vertices in $\bigcup_{x \in T_t} V_x$ (i.e. $G[t] = G_{T_t}$).

Vertex coloring

- Recall: A k -vertex coloring (k -coloring) of a graph $G = (V, E)$ is a function $\alpha : V \rightarrow \{1, \dots, k\}$ such that for all $uv \in E$, $\alpha(u) \neq \alpha(v)$.
- Let H_1 and H_2 be two subgraphs of G , with k -colorings α_1 and α_2 respectively. α_2 is α_1 -compatible if for all $v \in V(H_1) \cap V(H_2)$, $\alpha_1(v) = \alpha_2(v)$.
- Let $(T, \{V_t : t \in T\})$ be a rooted tree decomposition of G . For every $t \in T$ and every k -coloring α of G_t , define

$$\mathbf{Prev}_t(\alpha) = \begin{cases} 1 & \text{if } G[t] \text{ has an } \alpha\text{-compatible } k\text{-coloring } \beta, \\ 0 & \text{otherwise.} \end{cases}$$

Vertex coloring

Proposition. $\text{Prev}_u(\alpha) = 1$ if and only if for all children v of u , there is an α -compatible coloring β of G_v with $\text{Prev}_v(\beta) = 1$.

Proof. \Rightarrow If γ is an α -compatible coloring of $G[u]$, since G_v is a subgraph of $G[u]$, the restriction of γ to G_v gives the required coloring β .

\Leftarrow Suppose that u has exactly two children v and w of u , and having α -compatible colorings β and γ respectively (the proof is similar for more children). Since $(T, \{V_t : t \in T\})$ is a tree decomposition, $V(G[v]) \cap V(G[w]) \subseteq V_u$, so β is γ -compatible.

Combining β and γ now gives $\delta : V(G[u]) \rightarrow \{1, \dots, k\}$. Since $(T, \{V_t : t \in T\})$ is a tree decomposition, there are no edges $xy \in E(G)$ with $x \in V(G[v]) - V_u$ and $y \in V(G[w]) - V_u$, so δ is a k -coloring of $G[u]$. \square

Vertex coloring

Theorem. *If G , a graph of order n , has a small tree decomposition $(T, \{V_t : t \in T\})$ of width w , we can decide if G is k -colorable in time $k^{w+1} \cdot n^{O(1)}$.*

Proof. Transform $(T, \{V_t : t \in T\})$ in a rooted tree decomposition. For every $v \in T$ and every k -coloring α of G_v , we compute **Prev** $_v(\alpha)$: start at the leaves of T , and use the above proposition for the other nodes, in the right order.

$G = G[r]$ is k -colorable iff **Prev** $_v(\alpha) = 1$ for some α .

Testing whether α is a G_v coloring and computing **Prev** $_v(\alpha)$ can be done in polynomial time $n^{O(1)}$, so the total complexity is mainly determined by the number of candidates for α , which is $k^{|V_v|}$.

Complexity: $|V(T)| \cdot k^{w+1} \cdot n^{O(1)} = k^{w+1} \cdot n^{O(1)}$. □

Similar approaches (more advanced dynamic programming)

Theorem. *If G , a graph of order n , has a small tree decomposition $(T, \{V_t : t \in T\})$ of width w , the size of a minimum vertex cover of G can be computed in time $2^{w+1} \cdot n^{O(1)}$.*

Theorem. *If G , a vertex-weighted graph of order n , has a small tree decomposition $(T, \{V_t : t \in T\})$ of width w , a maximum weight stable set in G can be computed in time $4^{w+1} \cdot w \cdot n$.*

① Evaluare: [~croitoru/ag/week01.pdf](#) pagina 13

② Programare test final: 17,18 ianuarie; atenție la anunțurile de la orar

(sunt precizate orele pe grupe și sălile de examen; în situații excepționale, studenții pot veni la alte grupe decât cele programate, dar numai cu acordul meu prealabil).

③ Seminarul special de vineri seara se suspendă.



Succes la examene!