

# Algoritmi avansați

## C8 - Acoperiri convexe

Mihai-Sorin Stupariu

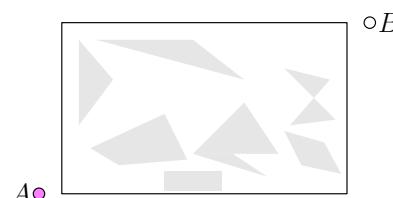
Sem. al II-lea, 2021 - 2022

## Introducere

Criterii numerice. Raport și test de orientare

## Introducere

- ▶ Algoritmii geometrici sunt legați de Geometria Computațională: *complexitatea computațională a problemelor geometrice în contextul analizei algoritmilor* [Lee & Preparata, 1984]
- ▶ Problemă (exemplu):



Cum poate fi deplasat discul din A în B fără a atinge obstacolele?

- ▶ Complexitatea: memorie, timp, calcule
- ▶ Probleme abordate: acoperiri convexe, proximitate, intersecții, căutare, etc.
- ▶ Tehnici utilizate: construcții incrementale, divide et impera, plane-sweep, transformări geometrice, etc.

## Acoperiri convexe

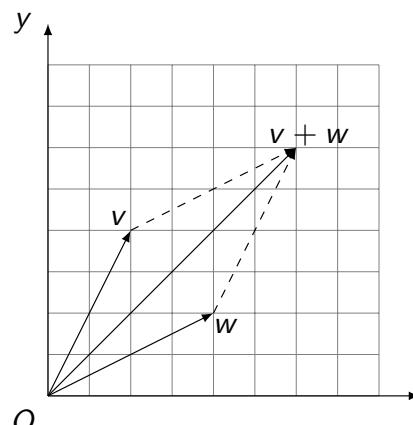
## Introducere

- ▶ Note istorice:
  - ▶ Ideea de construcție geometrică realizată într-un număr finit de pași: **Elementele lui Euclid**
  - ▶ Abordare numerică - sistem de coordonate în plan: Descartes (sec. XVII)
  - ▶ **"Simplicitatea" construcțiilor geometrice:** Lemoine (~ 1900)
  - ▶ a doua jumătate a sec. XX: formularea / rezolvarea unor probleme de GC; în 1975 este folosit prima dată termenul de *Computational Geometry* (M.I. Shamos, "Geometric Complexity", Proc. 7th ACM Annual Symposium on Theory of Computing)
- ▶ Domenii de aplicabilitate: grafică pe calculator, pattern recognition, robotică, statistică, gestionarea bazelor de date numerice, cercetări operaționale

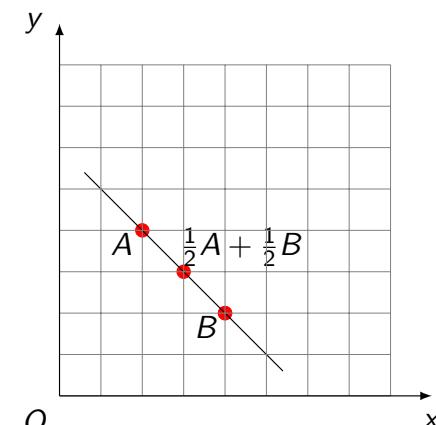
## Bibliografie

- ▶ M. de Berg, M. van Kreveld, M. Overmars si O. Schwarzkopf, *Computational Geometry, Algorithms and Applications*, Springer, 2008.  
(Site: <http://www.cs.uu.nl/geobook/>)
- ▶ F. Preparata si M. Shamos, *Computational Geometry: An Introduction*, Springer, 1985.
- ▶ S. Devadoss, J. O'Rourke, *Discrete and Computational Geometry*, Princeton University Press, 2011.
- ▶ (Site: <http://cs.smith.edu/~orourke/DCG/>)
- ▶ D. Lee, F. Preparata, *Computational Geometry - A Survey*, IEEE Transactions on Computers, **33** (1984), 1072-1101.
- ▶ B. Gärtner, M. Hoffmann, *Computational Geometry. Lecture Notes*, ETH Zürich, 2013.

## Vectori și puncte



Combinări liniare  
 $\alpha v + \beta w$  ( $\alpha, \beta \in \mathbb{R}$ )



Combinări affine  
 $\lambda A + \mu B$  ( $\lambda, \mu \in \mathbb{R}$  și  $\lambda + \mu = 1$ )

## Criterii numerice – poziția relativă a unor puncte

- ▶ Stabilirea unor relații între puncte / ordonare
- ▶ **Context 1D**
  - ▶ ordonare (**relativ la un sistem de coordonate**)
  - ▶ raport (**independent de alegerea unui sistem de coordonate**)
- ▶ **Context 2D**
  - ▶ ordonare (**relativ la un sistem de coordonate** – posibile alegeri: coordonate carteziene, coordonate polare)
  - ▶ testul de orientare (**independent de alegerea unui sistem cartezian de coordonate**)

## Conceptul de raport

- ▶ **Lemă** Fie  $A$  și  $B$  două puncte distințe în  $\mathbb{R}^n$ . Pentru orice punct  $P \in AB$ ,  $P \neq B$  există un unic scalar  $r \in \mathbb{R} \setminus \{-1\}$  astfel ca  $\vec{AP} = r \vec{PB}$ . Reciproc, fiecărui scalar  $r \in \mathbb{R} \setminus \{-1\}$ , îi corespunde un unic punct  $P \in AB$ .
- ▶ **Definiție** Scalarul  $r$  definit în lema anterioară se numește **raportul** punctelor  $A, B, P$  (sau **raportul în care punctul  $P$  împarte segmentul  $[AB]$** ) și este notat cu  $r(A, P, B)$ .

$$\text{Cas 1}$$

A horizontal line segment with points A, P, and B in order. The distance AP is labeled  $n \cdot \vec{PB}$  where  $n > 0$ .

$$\vec{AP} = n \cdot \vec{PB}, n > 0$$

$$\text{Cas 2}$$

A horizontal line segment with points A, B, and P in order. The distance AP is labeled  $n \cdot \vec{PB}$  where  $n < 0$ .

$$\vec{AP} = n \cdot \vec{PB}, n < 0$$

- ▶ **Observație importantă.** În calcularea raportului, ordinea punctelor este esențială. Modul în care este definită această noțiune (mai precis ordinea în care sunt considerate punctele) diferă de la autor la autor.

## Raport- exemple

- (i) În  $\mathbb{R}^2$  considerăm punctele  $A = (1, 1)$ ,  $B = (2, 2)$ ,  $C = (7, 7)$ . Determinăm raportul  $r(A, B, C)$ .

$$r = ? \text{ a. } \hat{u}. \quad \overrightarrow{AB} = r \overrightarrow{BC}$$

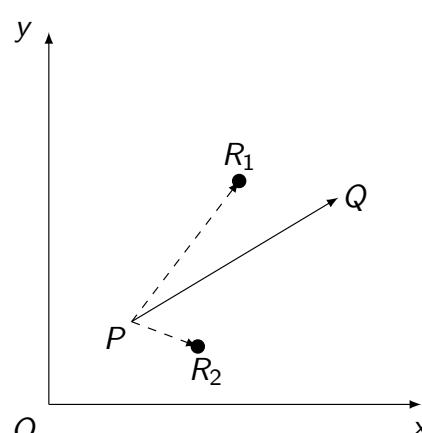
$$\overrightarrow{AB} = B - A = (x_B - x_A, y_B - y_A) = (1, 1)$$

$$\overrightarrow{BC} = C - B = (x_C - x_B, y_C - y_B) = (5, 5)$$

$$\overrightarrow{AB} = \frac{1}{5} \cdot \overrightarrow{BC}$$

r(A, B, C)

## Testul de orientare - motivație



Poziția relativă a două puncte față de un vector / o muchie orientată

## Raport- exemple

- (ii) În  $\mathbb{R}^3$  considerăm punctele  $A = (1, 2, 3)$ ,  $B = (2, 1, -1)$ ,  $C = (0, 3, 7)$ . Atunci punctele  $A, B, C$  sunt coliniare și avem  $r(A, C, B) = -\frac{1}{2}$ ,  $r(B, C, A) = -2$ ,  $r(C, A, B) = 1$ ,  $r(C, B, A) = -2$ .

- (iii) Fie  $A, B$  două puncte din  $\mathbb{R}^n$  și  $M = \frac{1}{2}A + \frac{1}{2}B$ . Atunci  $r(A, M, B) = 1$ ,  $r(M, A, B) = -\frac{1}{2}$ .

## Enunț principal

- **Propoziție.** Fie  $P = (p_1, p_2)$ ,  $Q = (q_1, q_2)$  două puncte distincte din planul  $\mathbb{R}^2$ , fie  $R = (r_1, r_2)$  un punct arbitrar și

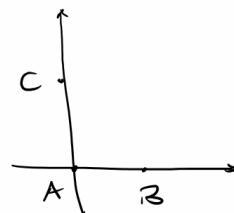
$$\Delta(P, Q, R) = \begin{vmatrix} 1 & 1 & 1 \\ p_1 & q_1 & r_1 \\ p_2 & q_2 & r_2 \end{vmatrix}.$$

Atunci  $R$  este situat:

- (i) pe dreapta  $PQ \Leftrightarrow \Delta(P, Q, R) = 0$  ("ecuația dreptei");
- (ii) "în dreapta" segmentului orientat  $\overrightarrow{PQ} \Leftrightarrow \Delta(P, Q, R) < 0$ ;
- (iii) "în stânga" segmentului orientat  $\overrightarrow{PQ} \Leftrightarrow \Delta(P, Q, R) > 0$ .

- **Obs.** Testul de orientare se bazează pe calculul unui polinom de gradul II ( $\Delta(P, Q, R)$ ).

## Testul de orientare - exemplu



$$A = (0, 0)$$

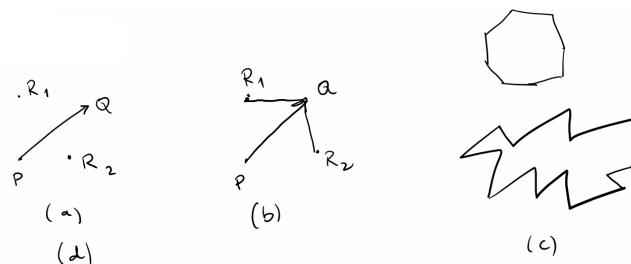
$$B = (1, 0)$$

$$C = (0, 1)$$

$$\Delta(A, B, C) = \begin{vmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} = 1 > 0$$

deci  $C$  este în stânga muchiei orientate  $\overrightarrow{AB}$ .

## Aplicații



- (a) dacă un punct este în dreapta / stânga unei muchii orientate;
- (b) natura unui viraj în parcurgerea unei linii poligonale (la dreapta / la stânga);
- (c) natura unui poligon (convex / concav);
- (d) dacă două puncte sunt de o parte și de alta a unui segment / a unei drepte.

## Limitări - robustețe și erori de rotunjire

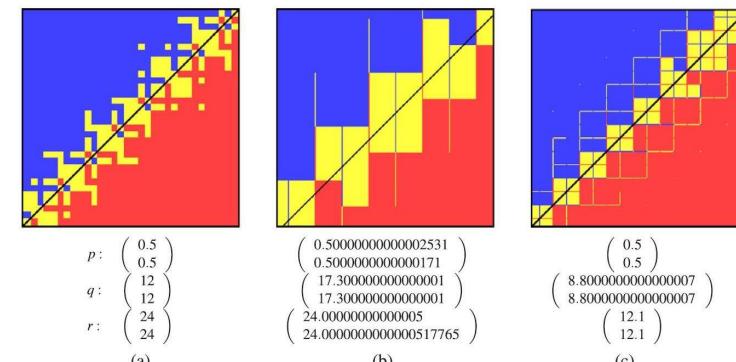
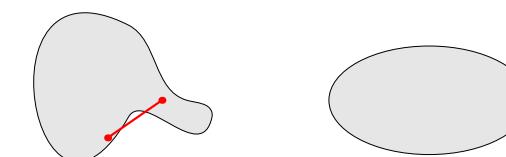


Fig. 2. The weird geometry of the float-orientation predicate: The figure shows the results of  $\text{float\_orient}(p_x + Xu_x, p_y + Yu_y, q, r)$  for  $0 \leq X, Y \leq 255$ , where  $u_x = u_y = 2^{-53}$  is the increment between adjacent floating-point numbers in the considered range. The result is color coded: Yellow (red, blue, resp.) pixels represent collinear (negative, positive, resp.) orientation. The line through  $q$  and  $r$  is shown in black.

## Mulțimi convexe: generalități

### ► Conceptul de mulțime convexă:

O mulțime  $M \subset \mathbb{R}^m$  este convexă dacă oricare ar fi  $p, q \in M$ , segmentul  $[pq]$  este inclus în  $M$ .

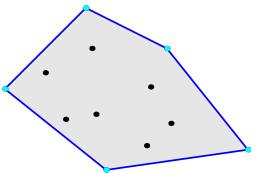


Mulțimea din stânga nu este convexă, întrucât există două puncte, pentru care segmentul determinat nu este inclus în mulțime (punctele cu această proprietate nu sunt unice!).

### ► Problematizare:

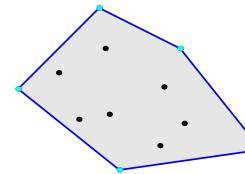
Mulțimile finite cu cel puțin două elemente nu sunt convexe → necesară **acoperirea convexă**.

## Acoperire convexă a unei mulțimi finite $\mathcal{P}$ : concept



- ▶ Caracterizări echivalente:

## Acoperire convexă a unei mulțimi finite $\mathcal{P}$ : concept



- ▶ Caracterizări echivalente:

- ▶ Cea mai "mică" (în sensul incluziunii) mulțime convexă care conține  $\mathcal{P}$ .
- ▶ Intersecția tuturor mulțimilor convexe care conțin  $\mathcal{P}$ .
- ▶ Mulțimea tuturor combinațiilor convexe ale punctelor din  $\mathcal{P}$ . O **combinatie convexă** a punctelor  $P_1, P_2, \dots, P_n$  este un punct  $P$  de forma

$$P = \alpha_1 P_1 + \dots + \alpha_n P_n, \quad \alpha_1, \dots, \alpha_n \in [0, 1], \quad \alpha_1 + \dots + \alpha_n = 1.$$

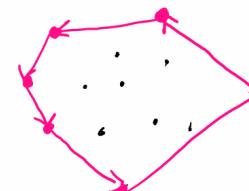
- ▶ **Problematizare:** Aceste caracterizări echivalente nu conduc la un algoritm de determinare a acoperirii convexe.

## Acoperire convexă a unei mulțimi finite $\mathcal{P}$ : problematizare

- ▶ Dacă  $\mathcal{P} \subset \mathbb{R}^d$  este finită, acoperirea sa convexă,  $\text{Conv}(\mathcal{P})$  este un **politop convex**.
- ▶ Cazuri particulare:  $d = 1$  (segment);  $d = 2$  (poligon);  $d = 3$  (poliedru).
- ▶ Cazul  $d = 1$ : acoperirea convexă este un segment; algoritmic: parcurgere a punctelor (complexitate  $O(n)$ ).
- ▶ În continuare:  $d = 2$ .

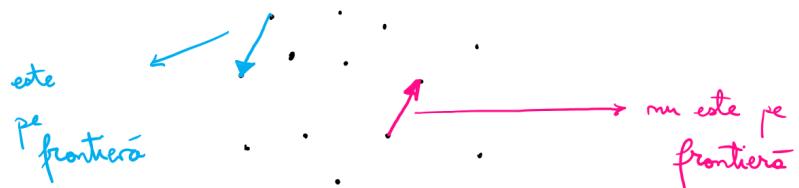
## Acoperire convexă a unei mulțimi finite $\mathcal{P}$ (practic)

- ▶ De fapt, dacă  $\mathcal{P}$  este finită, acoperirea sa convexă,  $\text{Conv}(\mathcal{P})$  este un poligon convex.



- ▶ **Problemă:**  
Cum determinăm, algoritmic, vârfurile acestui poligon?
- ▶ **Convenție:** Sensul de parcurgere a frontierei este cel trigonometric.

## Un algoritm "lent": idee de lucru



- ▶ Sunt considerate **muchiile orientate**.
- ▶ **Q:** Cum se decide dacă o muchie orientată fixată este pe **frontiera**?
- ▶ **A:** Toate celelalte puncte sunt "în stanga" ei (v. "testul de orientare").

## Algoritmul "lent": comentarii

- ▶ Complexitatea:  $O(n^3)$
- ▶ Complexitate algebrică: polinoame de gradul II
- ▶ Tratarea cazurilor degenerate: poate fi adaptat.
- ▶ Robustetea: datorită erorilor de rotunjire este posibil ca algoritmul să nu returneze o listă coerentă de muchii.

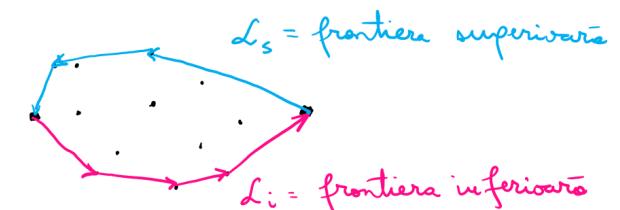
## Un algoritm lent

**Input:** O mulțime de puncte  $\mathcal{P}$  din  $\mathbb{R}^2$ .

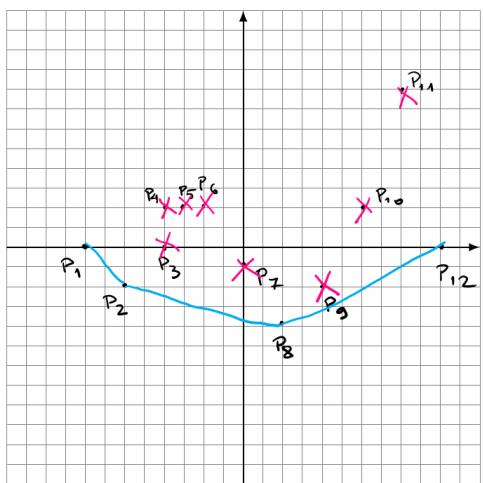
**Output:** O listă  $\mathcal{L}$  care conține vârfurile ce determină frontieră acoperirii convexe, parcursă în sensul trigonometric.

1.  $E \leftarrow \emptyset, \mathcal{L} \leftarrow \emptyset$  /\* $E$  este lista muchiilor orientate\*/
2. **for**  $(P, Q) \in \mathcal{P} \times \mathcal{P}$  cu  $P \neq Q$
3.     **do**  $valid \leftarrow \text{true}$
4.     **for**  $R \in \mathcal{P} \setminus \{P, Q\}$
5.         **do if**  $R$  "în dreapta" lui  $\overrightarrow{PQ}$
6.             **then**  $valid \leftarrow \text{false}$
7.         **if**  $valid = \text{true}$  **then**  $E = E \cup \{\overrightarrow{PQ}\}$
8. din  $E$  se construiește lista  $\mathcal{L}$  a vârfurilor acoperirii convexe /\*este necesar ca  $E$  să fie **coerentă**\*/

## Graham's scan, varianta Andrew: idee de lucru



- ▶ Punctele sunt mai întâi sortate și renumerate **lexicografic**.
- ▶ Algoritmul este de tip **incremental**, punctele fiind adăugate unul câte unul, fiind apoi eliminate anumite puncte.
- ▶ **Q:** Cum se decide dacă trei puncte sunt vârfuri consecutive ale acoperirii convexe?
- ▶ **A:** Se efectuează un "viraj la stânga" în punctul din mijloc.



$$P_1 = (-8, 0); P_2 = (-6, -2); P_3 = (-4, 0)$$

$$P_4 = (-4, 2); P_5 = (-3, 2); P_6 = (-2, 2)$$

$$P_7 = (0, -1); P_8 = (2, -4); P_9 = (4, -2)$$

$$P_{10} = (6, 2); P_{11} = (8, 8); P_{12} = (10, 0)$$

## Graham's scan, varianta Andrew: comentarii

- Complexitatea:  $O(n \log n)$ .
- Tratarea cazurilor degenerate: corect.
- Robustetea: datorită erorilor de rotunjire este posibil ca algoritmul să returneze o listă eronată (dar coerentă) de muchii.

## Graham's scan, varianta Andrew (algoritm)

**Input:** O mulțime de puncte  $\mathcal{P}$  din  $\mathbb{R}^2$ .

**Output:** O listă  $\mathcal{L}$  care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Sortare lexicografică, renumerotare  $P_1, P_2, \dots, P_n$  conform ordonării
2.  $\mathcal{L} \leftarrow (P_1, P_2)$
3. **for**  $i \leftarrow 3$  **to**  $n$
4.     **do** adaugă  $P_i$  la sfârșitul lui  $\mathcal{L}$
5.     **while**  $\mathcal{L}$  are mai mult de două puncte  
          **and** ultimele trei nu determină un viraj la stânga
6.     **do** șterge penultimul punct
7.     **return**  $\mathcal{L}$ ;
8. Parcurge pași analogi pentru a determina  $\mathcal{L}_s$
9. Concatenează  $\mathcal{L}_i$  și  $\mathcal{L}_s$

## Jarvis' march / Jarvis' wrap [1973]

- Algoritm de tip **incremental**. Nu necesită sortare prealabilă.
- Inițializare: un punct care este sigur un vârf al acoperirii convexe (e.g. punctul cel mai de jos / din stânga / stânga jos).
- Lista se actualizează prin determinarea succesorului: "cel mai la dreapta" punct.
- Implementare: două abordări (i) ordonare; (ii) testul de orientare.
- Complexitate:  $O(hn)$ , unde  $h$  este numărul punctelor de pe frontieră acoperirii convexe.
- **Algoritmul lui Chan** "combină" ideile celor doi algoritmi, ajungând la complexitatea-timp  $O(n \log h)$ .

## Jarvis' march (algoritm)

**Input:** O mulțime de puncte necoliniare  $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$  din  $\mathbb{R}^2$  ( $n \geq 3$ ).

**Output:** O listă  $\mathcal{L}$  care conține vârfurile ce determină frontieră acoperirii convexe, parcursă în sens trigonometric.

1. Determinarea unui punct din  $\mathcal{P}$  care aparține frontierei (de exemplu cel mai mic, folosind ordinea lexicografică); acest punct este notat cu  $A_1$ .
2.  $k \leftarrow 1; \mathcal{L} \leftarrow (A_1); valid \leftarrow \text{true}$
3. **while**  $valid = \text{true}$
4.     **do** alege un pivot arbitrar  $S \in \mathcal{P}$ , diferit de  $A_k$
5.         **for**  $i \leftarrow 1$  **to**  $n$
6.             **do if**  $P_i$  este la dreapta muchiei orientate  $A_k S$
7.                 **then**  $S \leftarrow P_i$
8.             **if**  $S \neq A_1$
9.                 **then**  $k \leftarrow k + 1;$   
 $A_k = S$   
adăugă  $A_k$  la  $\mathcal{L}$
10.          **else**  $valid \leftarrow \text{false}$
11. **return**  $\mathcal{L}$

## alte direcții de lucru

- ▶ **Aplicații:** grafică pe calculator, robotică, GIS, recunoașterea formelor, gestionarea bazelor de date multi-dimensionale, etc.
- ▶ Pot fi stabilite legături cu algoritmi studiați în alt context. De exemplu: *Traveling Salesman Problem*, [în context euclidian](#) (costurile sunt date de distanțele dintre puncte). În acest caz, ordinea în care nodurile de pe frontieră apar în traseul optim coincide cu ordinea în care acestea apar în parcurgerea frontierei acoperirii convexe - exemplu.
- ▶ Algoritmi pentru spații euclidiene de dimensiune  $m \geq 3$ .
- ▶ Algoritmi eficienți pentru determinarea acoperirii convexe pentru vârfurile unui poligon arbitrar.
- ▶ Algoritmi dinamici (on-line, real-time, convex hull maintenance).

# Algoritmi avansați

## C9 - Triangularea poligoanelor

Mihai-Sorin Stupariu

Sem. al II-lea, 2021 - 2022

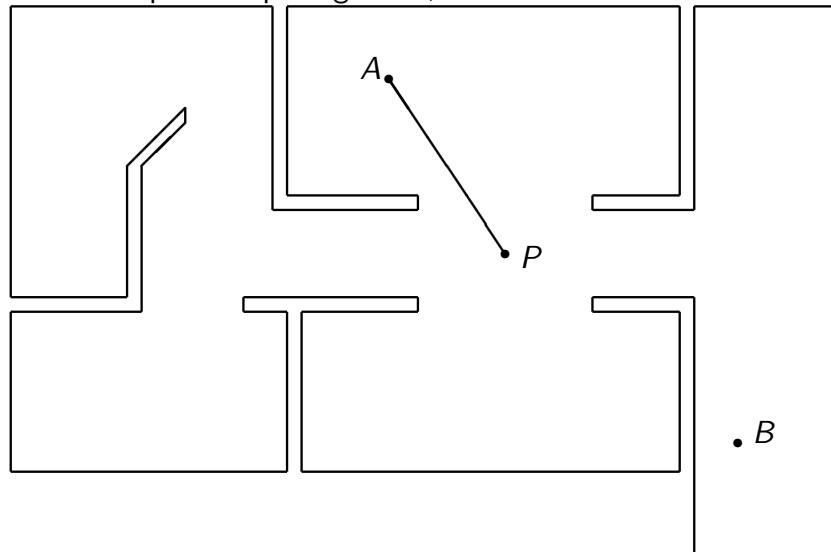
## Problema galeriei de artă

Algoritmi avansați - C9. Triangularea poligoanelor  
Problema galeriei de artă

1 / 37

### Supravegherea unei galerii de artă

Camera din  $P$  poate supraveghea  $A$ , dar nu  $B$ .



## Algoritmi de triangulare- "Ear clipping"

### Triangularea poligoanelor - un algoritm eficient

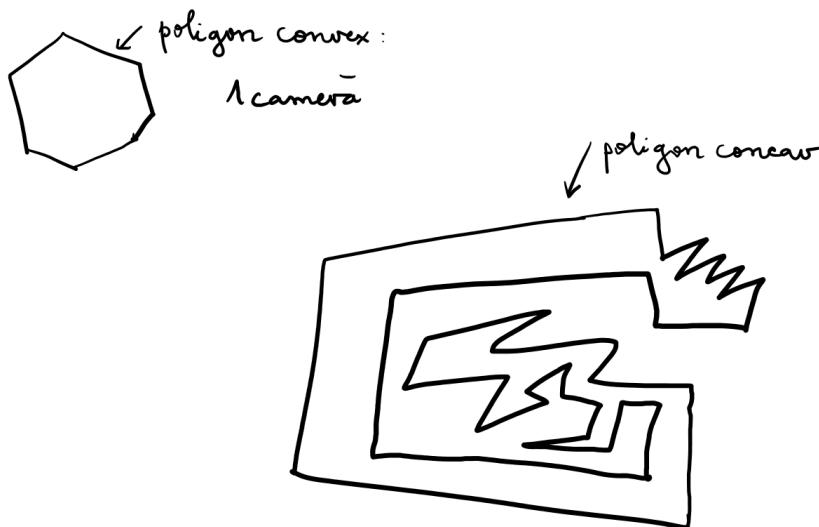
Algoritmi avansați - C9. Triangularea poligoanelor  
Problema galeriei de artă

2 / 37

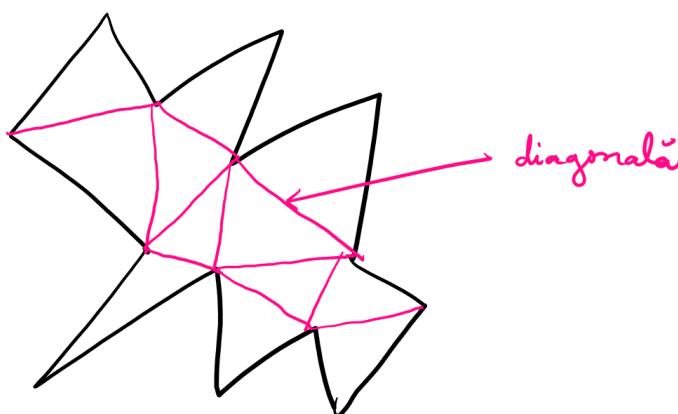
### Formalizare

- ▶ O galerie de artă poate fi interpretată (în contextul acestei probleme) ca un poligon  $\mathcal{P}$  (adică o linie poligonală fără autointersecții) având  $n$  vârfuri.
- ▶ O cameră video (vizibilitate  $360^0$ ) poate fi identificată cu un punct din interiorul lui  $\mathcal{P}$ ; ea poate supraveghea acele puncte cu care poate fi unită printr-un segment inclus în interiorul poligonului.
- ▶ **Problema galeriei de artă:** *câte camere video sunt necesare pentru a supraveghea o galerie de artă și unde trebuie amplasate acestea?*

## Comentarii



## Despre triangulări



## Numărul de camere vs. forma poligonului

- ▶ Se dorește exprimarea numărului de camere necesare pentru supraveghere în funcție de  $n$  (sau controlarea acestuia de către  $n$ ).
- ▶ Pentru a supraveghea un spațiu având forma unui poligon convex, este suficientă o singură cameră.
- ▶ Numărul de camere depinde și de forma poligonului: cu cât forma este mai "complexă", cu atât numărul de camere va fi mai mare.
- ▶ **Principiu:** Poligonul considerat: descompus în triunghiuri (triangulare).

## Definiție formală

- ▶ Fie  $\mathcal{P}$  un poligon plan.
- ▶ (i) O **diagonală** a lui  $\mathcal{P}$  este un segment ce unește două vârfuri ale acestuia și care este situat în interiorul lui  $\mathcal{P}$ .
- ▶ (ii) O **triangulară**  $T_{\mathcal{P}}$  a lui  $\mathcal{P}$  este o descompunere a lui  $\mathcal{P}$  în triunghiuri, dată de o mulțime maximală de diagonale ce nu se intersectează.

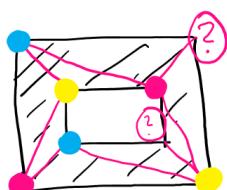
## Rezultate

- ▶ **Lemă.** Orice poligon admite o diagonală.
- ▶ **Teoremă.** Orice poligon admite o triangulare. Orice triangulare a unui poligon cu  $n$  vârfuri conține exact  $n - 2$  triunghiuri.

## Rezolvarea problemei galeriei de artă

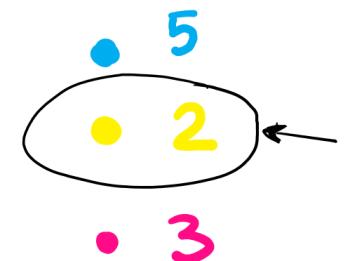
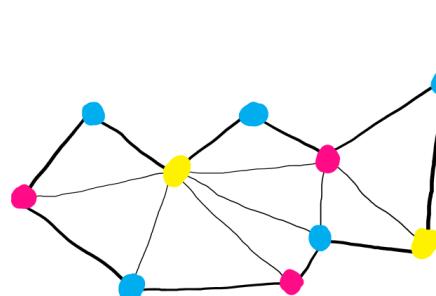
- ▶ Amplasarea camerelor se poate face în vîrfurile poligonului.
- ▶ Dată o pereche  $(\mathcal{P}, \mathcal{T}_{\mathcal{P}})$  se consideră o 3-colorare a acestieia: fiecărui vîrf îi corespunde o culoare dintr-un set de 3 culori și pentru fiecare triunghi, cele 3 vârfuri au culori distincte.
- ▶ **Observație.** Dacă  $\mathcal{P}$  este linie poligonală fără autointersecții o astfel de colorare există, deoarece graful dual asociat perechii  $(\mathcal{P}, \mathcal{T}_{\mathcal{P}})$  este arbore.

Contraexemplu - 3 colorare



## Rezolvarea problemei galeriei de artă

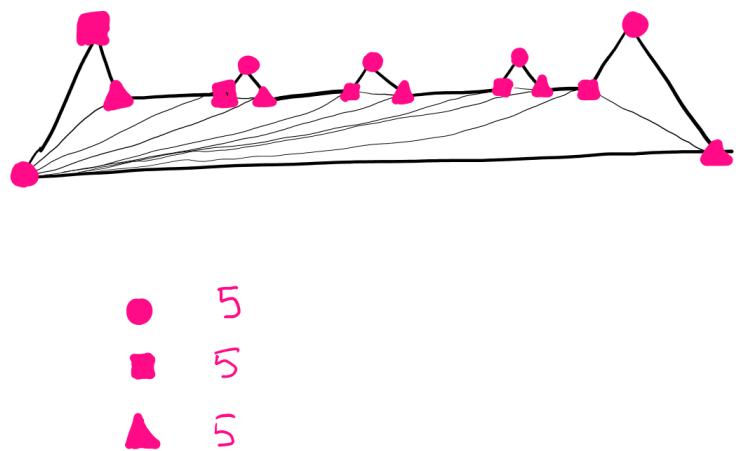
- ▶ Amplasarea camerelor se poate face în vîrfurile poligonului.
- ▶ Dată o pereche  $(\mathcal{P}, \mathcal{T}_{\mathcal{P}})$  se consideră o 3-colorare a acestieia: fiecărui vîrf îi corespunde o culoare dintr-un set de 3 culori și pentru fiecare triunghi, cele 3 vârfuri au culori distincte.
- ▶ **Observație.** Dacă  $\mathcal{P}$  este linie poligonală fără autointersecții o astfel de colorare există, deoarece graful dual asociat perechii  $(\mathcal{P}, \mathcal{T}_{\mathcal{P}})$  este arbore.



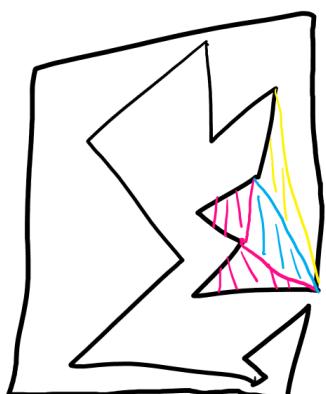
## Teorema galeriei de artă

- ▶ **Teoremă.** [Chvátal, 1975; Fisk, 1978] Pentru un poligon cu  $n$  vârfuri,  $\lceil \frac{n}{3} \rceil$  camere sunt **uneori necesare și întotdeauna suficiente** pentru ca fiecare punct al poligonului să fie vizibil din cel puțin una din camere.
- ▶ Despre Teorema Galeriei de Artă: J. O'Rourke, *Art Gallery Theorems and Algorithms*
- ▶ Despre numărul de culori utilizat: L. Erickson, S. LaValle, *A chromatic art gallery problem*

## Teorema galeriei de artă - justificare, exemplu

• uneori necesare

## Triangularea unui poligon - intuiție



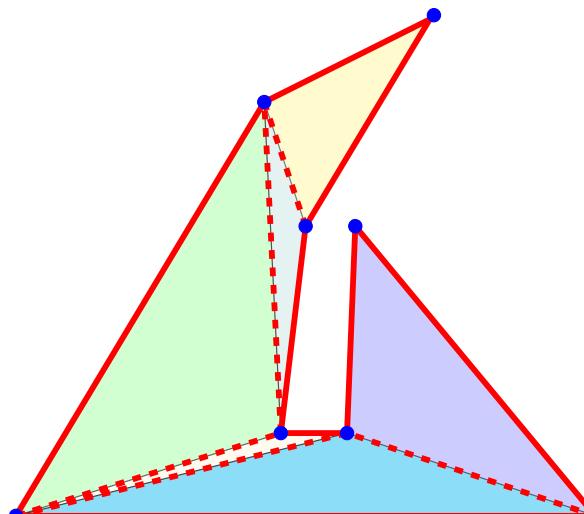
## Teorema galeriei de artă - justificare, exemplu

• în totdeauna suficiente

notăm cu  $n_1, n_2, n_3$  numărul de vârfuri colorate cu cele 3 culori :  $n_1 + n_2 + n_3 = n$

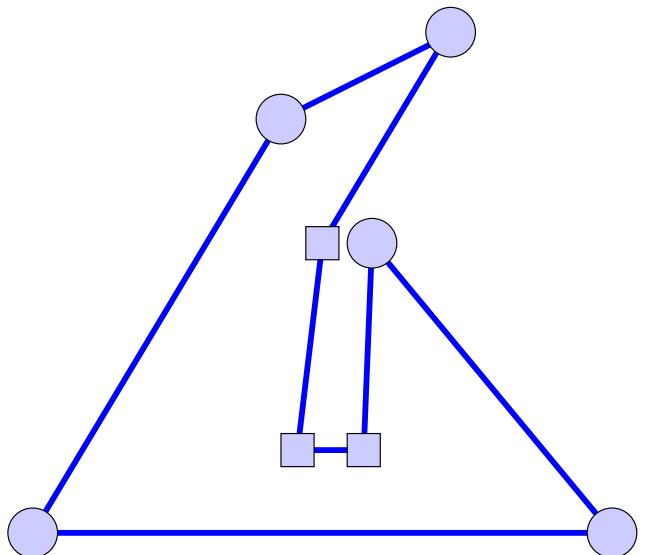
$$\begin{aligned} \text{P.p. abs. } n_1 > \left\lceil \frac{n}{3} \right\rceil &\Rightarrow n_1 > \frac{n}{3} \quad \left( \begin{array}{l} \text{def.} \\ \text{partii} \\ \text{în trei} \end{array} \right) \\ n_2 > \left\lceil \frac{n}{3} \right\rceil &\Rightarrow n_2 > \frac{n}{3} \\ n_3 > \left\lceil \frac{n}{3} \right\rceil &\Rightarrow n_3 > \frac{n}{3} \\ \Rightarrow \exists i \text{ a.i. } n_i &\leq \left\lceil \frac{n}{3} \right\rceil \end{aligned} \quad \left. \begin{array}{l} \Rightarrow n_1 + n_2 + n_3 \\ > n \\ \text{contradicție} \end{array} \right.$$

## Triangularea unui poligon - algoritmul "Ear clipping"



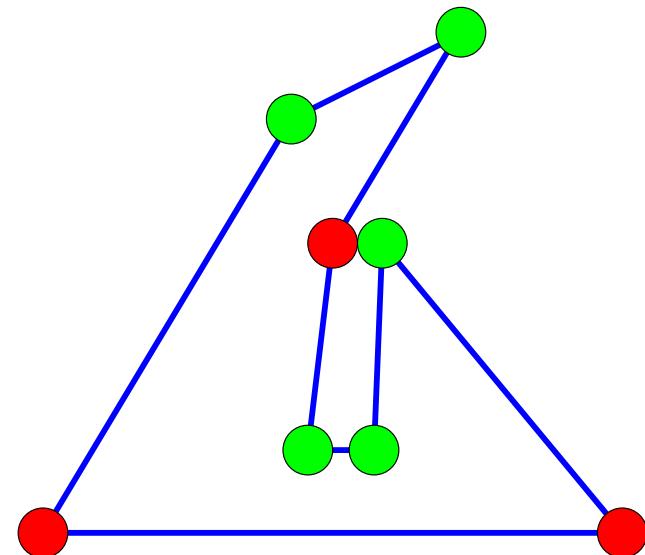
... se obține o triangulare a poligonului.

Clasificarea vârfurilor unui poligon - convexe/concave



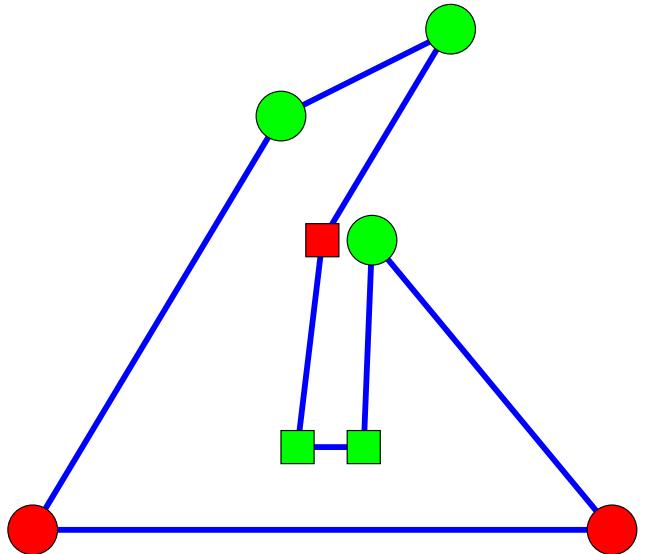
Vârfurile convexe (cerc) / concave (pătrat).

Clasificarea vârfurilor unui poligon - principale/neprincipale



Vârfurile principale (verde) / neprincipale (roșu).

## Clasificarea vârfurilor unui poligon



Patru tipuri de vârfuri.

Metode de triangulare: ear cutting / clipping / trimming

- ▶ Concepție pentru un poligon  $P = (P_1, P_2, \dots, P_n)$ :
    - **Vârf convex/concav ("reflex")**: se stabilește cu testul de orientare. Un vârf este convex  $\Leftrightarrow$  are același tip de viraj ca vârful "cel mai din stânga".
    - **Vârf principal**:  $P_i$  este principal dacă  $[P_{i-1}P_{i+1}]$  este diagonală (echivalent: nu există un alt vârf în interiorul sau pe laturile  $\Delta P_{i-1}P_iP_{i+1}$ ).
    - **Ear (vârf / componentă de tip E)**: este un vârf principal convex [Meisters, 1975]. Dacă  $P_i$  este componentă de tip E, atunci segmentul  $[P_{i-1}P_{i+1}]$  nu intersectează laturile poligonului și este situat în interiorul acestuia, adică este "diagonală veritabilă", iar  $\Delta P_{i-1}P_iP_{i+1}$  poate fi "eliminat".
    - **Mouth (vârf / componentă de tip M)**: este un vârf principal concav [Toussaint, 1991].
  - ▶ **Criterii de clasificare a vârfurilor:** (i) vârf convex/concav; (ii) vârf principal/nu.
  - ▶ **Teoremă.** (Two Ears Theorem [Meisters, 1975]) *Orice poligon cu cel puțin 4 vârfuri admite cel puțin două componente de tip E care nu se suprapun.*
  - ▶ **Corolar.** *Orice poligon admite (cel puțin) o diagonală.*
  - ▶ Găsirea unei componente de tip E: complexitate  $O(n)$  [ElGindy, Everett, Toussaint, 1993]. Se bazează pe Two Ears Theorem!
  - ▶ Algoritmul de triangulare bazat de metoda ear cutting: complexitate  $O(n^2)$ .
  - ▶ **Link despre triangulații.** [Link pentru algoritmul Ear cutting](#)

## Metode de triangulare: descompunerea în poligoane monotone

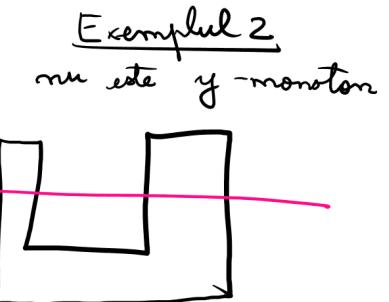
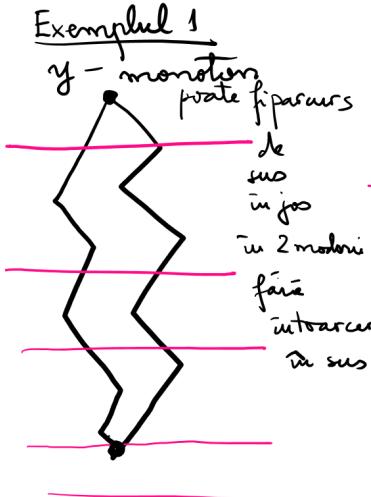
- ▶ Algoritmi de triangulare eficienți: complexitate  $O(n)$  pentru poligoane  $y$ -monotone [Garey et al., 1978] (algoritmul este descris pe slide-urile următoare).
- ▶ Descompunerea unui poligon oarecare în componente  $y$ -monotone poate fi realizată cu un algoritm de complexitate  $O(n \log n)$  [Lee, Preparata, 1977]. În concluzie, avem următoarea **Teoremă**. *Un poligon poate fi triangulat folosind un algoritm de complexitate  $O(n \log n)$ .*
- ▶ Există și alte clase de algoritmi mai rapizi; [Chazelle, 1990]: algoritm liniar.
- ▶ Găsirea unui algoritm liniar "simplu" Problemă în *The Open Problems Project*

## Metoda - paradigma dreptei de baleiere (*line sweep*)

- ▶ Se consideră o dreaptă de baleiere orizontală. Algoritmul reține o serie de informații legate de structura geometrică analizată.
- ▶ **Statut** al dreptei de baleiere: stivă a vârfurilor deja întâlnite, dar care "mai au nevoie de diagonale" / "mai pot să apară în triunghiuri". (Clarificare. **Q:** Când este eliminat un vârf? **A:** Când a fost trasată o diagonală situată "mai jos de acesta").
- ▶ **Evenimente:** modificarea statutului. Sunt vârfurile poligonului, în prealabil ordonate după  $y$ ; pentru fiecare vârf știm dacă este pe lanțul din stânga sau pe cel din dreapta.
- ▶ **Invariant:** "pâlnie" (*funnel*) în care (i) vârful de sus este convex; (ii) pe o parte: o muchie; (iii) pe cealaltă parte: muchie / succesiune de vârfuri concave.

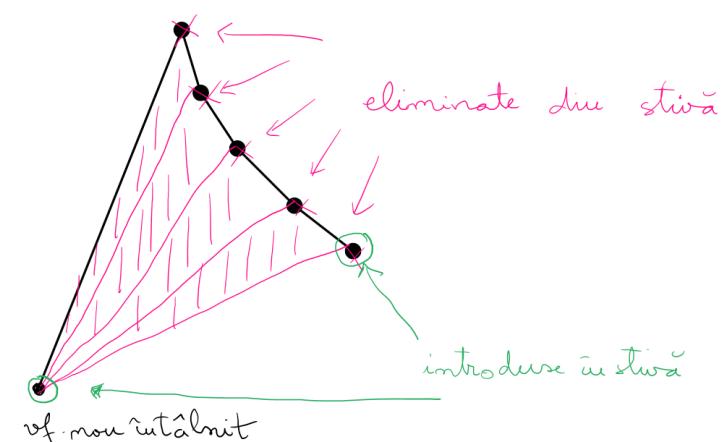
## Metode de triangulare: descompunerea în poligoane monotone

- ▶ Concept: **poligon  $y$ -monoton**



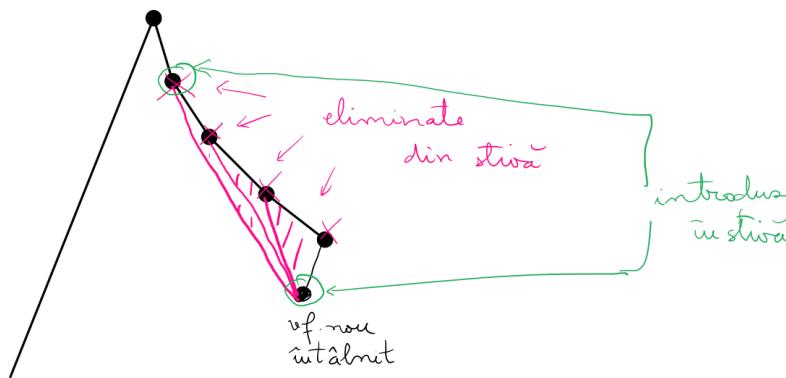
## Evenimente - cazul 1

1. Vârful nou întâlnit este pe lanțul opus ultimului vârf din stivă.



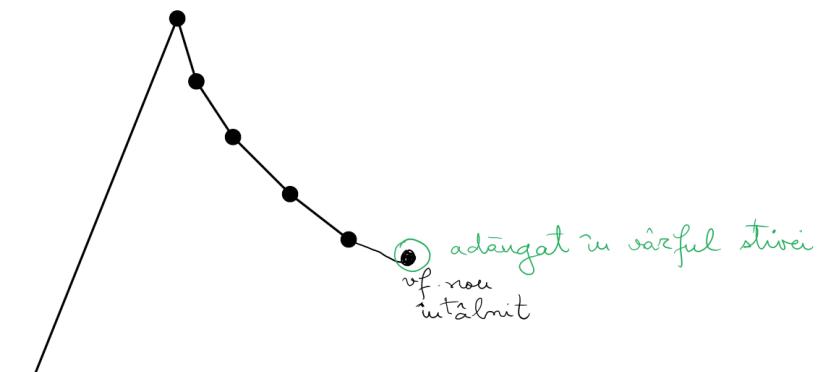
## Evenimente - cazul 2a

2a. Vârful nou întâlnit este pe același lanț cu ultimul vârf din stivă.

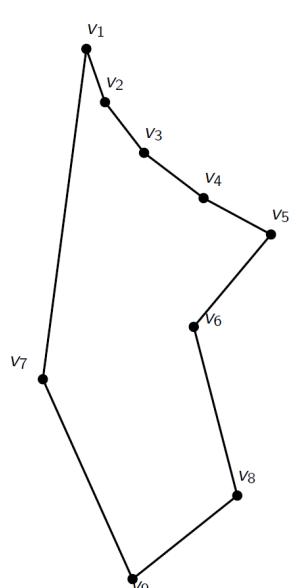


## Evenimente - cazul 2b

2b. Vârful nou întâlnit este pe același lanț cu ultimul vârf din stivă.

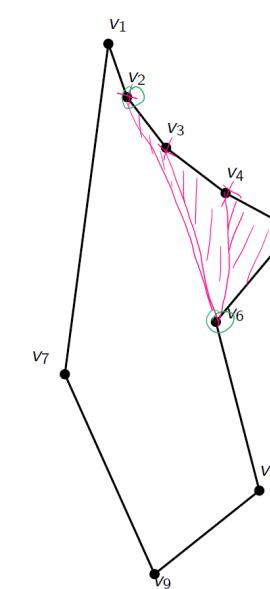


## Exemplu



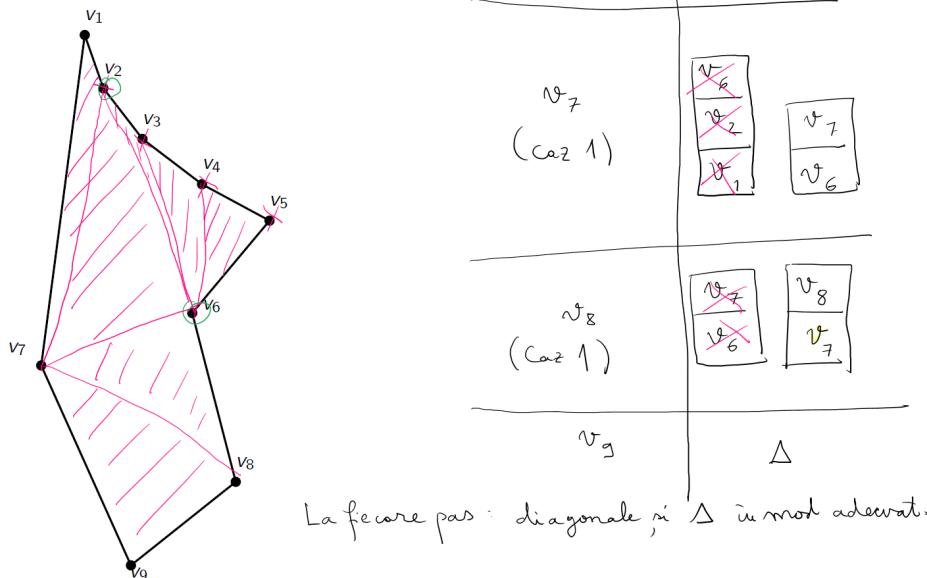
Eveniment	Stațiu
	$v_2$ $v_1$
$v_3$ (caz 2b)	$v_3$ $v_2$ $v_1$
$v_4$ (caz 2b)	$v_4$ $v_3$ $v_2$ $v_1$

## Exemplu



Eveniment	Stațiu
$v_5$ (caz 2b)	$v_5$ $v_4$ $v_3$ $v_2$ $v_1$
$v_6$ (caz 2a)	<del><math>v_5</math></del> <del><math>v_4</math></del> <del><math>v_3</math></del> <del><math>v_2</math></del> $v_6$ $v_2$ $v_1$

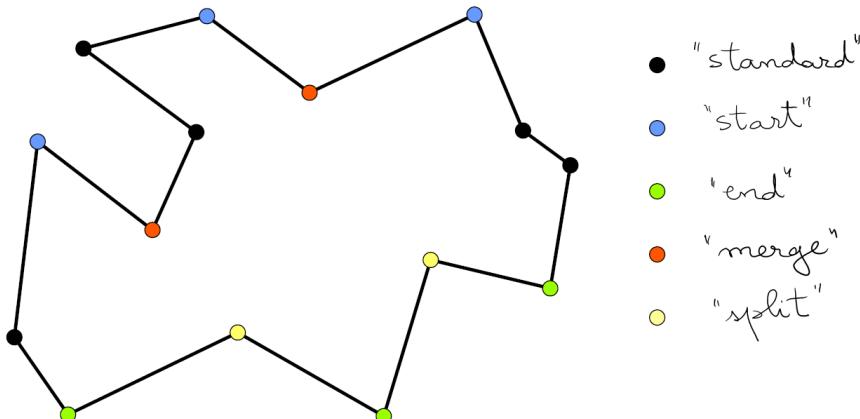
## Exemplu



Algoritmi avansati - C9. Triangularea poligoanelor

Triangularea poligoanelor - un algoritm eficient

## Tipuri de vârfuri



## Triangularea poligoanelor monotone

**Input:** Un poligon  $y$ -monoton  $\mathcal{P}$

**Output:** O triangulare a lui  $\mathcal{P}$

1. Lanțul vârfurilor din partea stângă și al celor din partea dreaptă sunt unite într-un singur sir, ordonat descrescător, după y (dacă ordonata este egală, se folosește abscisa). Fie  $v_1, v_2, \dots, v_n$  sirul ordonat.
  2. Initializează o stivă vidă  $\mathcal{S}$  și inserează  $v_1, v_2$ .
  3. **for**  $j = 3$  **to**  $n - 1$ 
    4.     **do if**  $v_j$  și vârful din top al lui  $\mathcal{S}$  sunt în lanțuri diferite
    5.         **then** extrage toate vârfurile din  $\mathcal{S}$
    6.         inserează diagonale de la  $v_j$  la vf. extrase, exceptând ultimul
    7.         inserează  $v_{j-1}$  și  $v_j$  în  $\mathcal{S}$
    8.     **else** extrage un vârf din  $\mathcal{S}$
    9.         extrage celelalte vârfuri din  $\mathcal{S}$  dacă diagonalele formate cu  $v_j$  sunt în interiorul lui  $\mathcal{P}$ ; inserează aceste diagonale; inserează înapoi ultimul vârf extras
    10.         inserează  $v_j$  în  $\mathcal{S}$
  11. adaugă diagonale de la  $v_n$  la vf. stivei (exceptând primul și ultimul)

## Rezultate

- ▶ Folosind un algoritm bazat pe paradigma dreptei de baleiere și clasificarea vârfurilor indicată, un poligon cu  $n$  vârfuri poate fi descompus în poligoane  $y$ -monotone cu un algoritm având complexitatea-timp  $O(n \log n)$ .
  - ▶ Conform algoritmului descris, un poligon  $y$ -monoton poate fi triangulat în timp liniar.
  - ▶ **Teoremă (rezultatul principal)** *Un poligon cu  $n$  vârfuri poate fi triangulat cu complexitatea-timp  $O(n \log n)$  și complexitatea-spațiu  $O(n)$ .*

# Algoritmi avansați

## C10 - Triangularea mulțimilor de puncte

Mihai-Sorin Stupariu

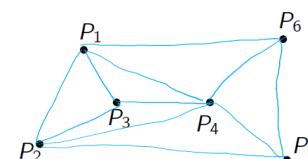
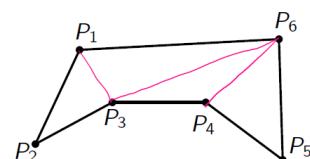
Sem. al II-lea, 2021 - 2022

Algoritmi avansați - C10. Triangularea mulțimilor de puncte  
Triangularea unei mulțimi arbitrară de puncte

## Triangularea unei mulțimi arbitrară de puncte

### Problematizare

- ▶ Tema anterioară: triangularea unui poligon (listă ordonată de puncte  $(P_1, P_2, \dots, P_n)$ ).
- ▶ Are sens să vorbim de triangulare pentru mulțimea  $\{P_1, P_2, \dots, P_n\}$ ?
- ▶ **Exemplu:**



- ▶ În cele ce urmează vom considera doar mulțimi de puncte din planul  $\mathbb{R}^2$ .

Algoritmi avansați - C10. Triangularea mulțimilor de puncte  
Triangularea unei mulțimi arbitrară de puncte

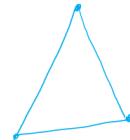
Algoritmi avansați - C10. Triangularea mulțimilor de puncte  
Triangularea unei mulțimi arbitrară de puncte

### Problematizare

- ▶ **Definiție.** O **triangulare** a unei mulțimi  $\mathcal{P}$  este o subdivizare maximală a acoperirii convexe  $\text{Conv}(\mathcal{P})$  a lui  $\mathcal{P}$  cu triunghiuri ale căror vârfuri sunt elemente ale lui  $\mathcal{P}$  (fără autointersecții!)
- ▶ Trebuie făcută distincție între triangulare a unui poligon  $(P_1, P_2, \dots, P_n)$  și triangulare a mulțimii subdiacente  $\{P_1, P_2, \dots, P_n\}$  (coincid dacă poligonul este convex!)
- ▶ **Comentariu:** Triangulările mulțimilor de puncte sunt esențiale în grafica pe calculator.

## Exemple

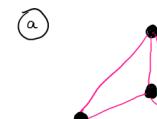
(i) 3 puncte necoliniare



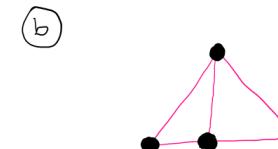
3 vârfuri  
3 muchii  
1 față

## Exemple

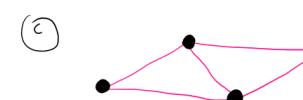
(ii) 4 puncte necoliniare, nesituate toate pe o aceeași dreaptă



4 vârfuri  
6 muchii  
3 fete (3 Δ)



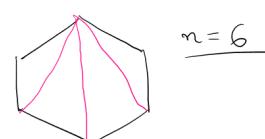
4 vârfuri  
5 muchii  
2 fete (2 Δ)



4 vârfuri  
5 muchii  
2 fete (2 Δ)

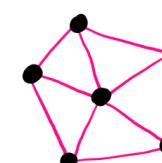
## Elemente ale unei triangulații

- Dată o mulțime de puncte  $\mathcal{P}$  și o triangulare  $\mathcal{T}_P$  a sa:  
**vârfuri, muchii, triunghiuri.**
- Legătură cantitativă între aceste elemente?
- **Propoziție.** Fie  $\mathcal{P}$  o mulțime de  $n$  puncte din plan nesituate toate pe o aceeași dreaptă. Notăm cu  $k$  numărul de puncte de pe frontieră acoperirii convexe  $\text{Conv}(\mathcal{P})$ . Orice triangulare a lui  $\mathcal{P}$  are  $(2n - k - 2)$  triunghiuri și  $(3n - k - 3)$  muchii.
- **Exemplu:** Cazul unui poligon convex: un poligon convex cu  $n$  vârfuri poate fi triangulat cu  $(n - 2)$  triunghiuri, având  $(2n - 3)$  muchii.



$4 \Delta$   
9 muchii

## Demonstrație



Graf:

noduri le: punctele inițiale ( $n$ )  
muchile: laturile  $\Delta$  ( $m_m = ?$ )  
fetele: fetele  $\Delta$  + față exterioară ( $m_t + 1$ )  
||?  
??

• Relația lui Euler:  $n - m_m + (m_t + 1) = 2$

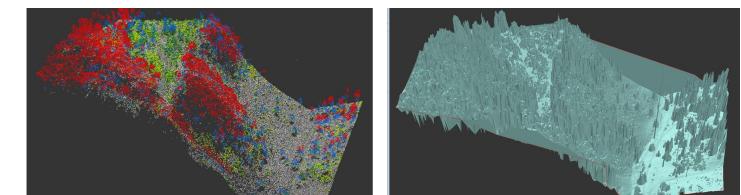
• Incidență dintre muchii și fete

$2 \cdot m_m$  = "perspectiva muchiilor"

$$\begin{aligned} & \frac{2 \cdot m_m}{3 \cdot m_t + k} = \frac{\text{ptr. } \Delta}{\text{ptr. față exterioară}} \\ & \Rightarrow \dots \\ & \Rightarrow \dots \\ & m_m = \dots \\ & m_t = \dots \end{aligned}$$

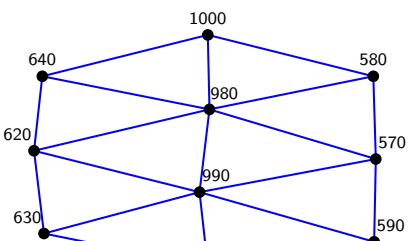
## Problematizare

- ▶ **Problemă.** Se fac măsurători ale altitudinii pentru un teren. Se dorește **reprazentarea tridimensională** (cât mai sugestivă) . Alternativ: se dorește **generarea unui teren** pentru o aplicație.

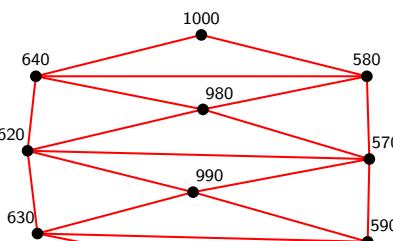


## Problematizare - continuare

- ▶ **Problemă (reformulată).** Cum "comparăm triangulările" unei mulțimi de puncte fixate?
- ▶ **Exemplu.** Măsurători ale altitudinii.



Triangulare 1

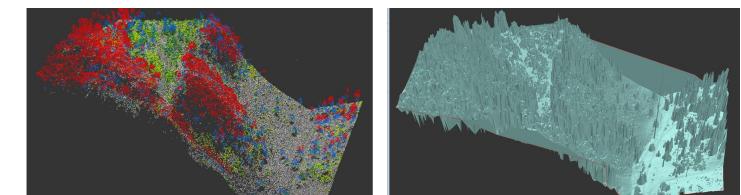


Triangulare 2

- ▶ **Întrebări naturale:** (i) Există o triangulare "convenabilă" a unei mulțimi de puncte? (ii) Cum poate fi determinată eficient o astfel de triangulare?

## Problematizare

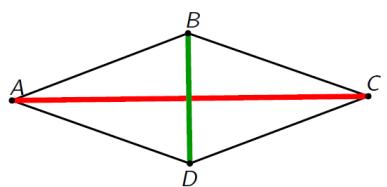
- ▶ **Problemă.** Se fac măsurători ale altitudinii pentru un teren. Se dorește **reprazentarea tridimensională** (cât mai sugestivă) . Alternativ: se dorește **generarea unui teren** pentru o aplicație.



## Terminologie, triangulări legale

- ▶ Fixată: o mulțime de puncte  $\mathcal{P}$ . **În cele ce urmează vom presupune că  $\mathcal{P}$  este o mulțime de puncte din planul  $\mathbb{R}^2$ .**
- ▶ Fie  $\mathcal{T}$  o triangulare a lui  $\mathcal{P}$  cu  $m$  triunghiuri. Fie  $\alpha_1, \alpha_2, \dots, \alpha_{3m}$  unghiuurile lui  $\mathcal{T}$ , ordonate crescător. **Vectorul unghiuurilor lui  $\mathcal{T}$  este  $A(\mathcal{T}) = (\alpha_1, \alpha_2, \dots, \alpha_{3m})$ .**
- ▶ **Relație de ordine pe mulțimea triangulărilor lui  $\mathcal{P}$ :** ordinea lexicografică pentru vectorii unghiuurilor. Fie  $\mathcal{T}$  și  $\mathcal{T}'$  două triangulări ale lui  $\mathcal{P}$ . Atunci  $A(\mathcal{T}) > A(\mathcal{T}')$  dacă  $\exists i$  astfel ca  $\alpha_j = \alpha'_j, \forall 1 \leq j < i$  și  $\alpha_i > \alpha'_i$ .
- ▶ **Triangulare unghiular optimă:**  $\mathcal{T}$  astfel ca  $A(\mathcal{T}) \geq A(\mathcal{T}')$ , pentru orice triangulare  $\mathcal{T}'$ .

## Exemplu - cazul unui patrulater convex



diagonala  $AC \Rightarrow \Delta BAC \cong \Delta DAC$

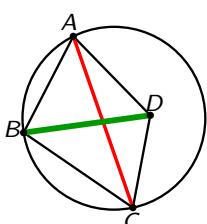
diagonala  $BD \Rightarrow \Delta ABD \cong \Delta CBD$

diagonala  $AC \Rightarrow$  vectorul unghiurilor  $(\alpha)$   
 $BD \Rightarrow$   $\_\_\_\_\_ (\beta)$

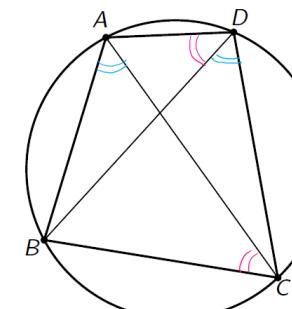
"cel mai mic unghi care apare în triunghiul datează de  $AC$  este mai mic decât cel mai mic unghi care apare în triunghiul datează de  $BD$ ".

## Muchii ilegale

- ▶ **Conceptul de muchie ilegală.** Fie  $A, B, C, D \in \mathbb{R}^2$  fixate astfel ca  $ABCD$  să fie un patrulater convex; fie  $\mathcal{T}_{AC}, \mathcal{T}_{BD}$  triunghiurile date de diagonalele  $AC$ , respectiv  $BD$ . Muchia  $AC$  este **ilegală** dacă  $\min A(\mathcal{T}_{AC}) < \min A(\mathcal{T}_{BD})$ .
- ▶ **Criteriu geometric** pentru a testa dacă o muchie este legală: muchia  $AC$ , adiacentă cu triunghiurile  $\Delta ACB$  și  $\Delta ACD$  este ilegală dacă și numai dacă punctul  $D$  este situat în interiorul cercului circumscris  $\Delta ABC$ .



## Exemplu - cazul unui patrulater inscriptibil

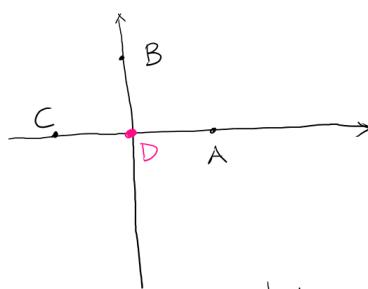


În acest caz triunghiurile formate de diagonale au "cele mai mici unghiuri" congruente  $\Rightarrow$  nu putem "distinge" între cele două diagonale.

## Muchii ilegale

- ▶ **Criteriu numeric / analitic** pentru a testa dacă o muchie este ilegală.
    - ▶ Pentru puncte  $A = (x_A, y_A), B = (x_B, y_B), C = (x_C, y_C), D = (x_D, y_D)$ :
- $$\Theta(A, B, C, D) = \begin{vmatrix} x_A & y_A & x_A^2 + y_A^2 & 1 \\ x_B & y_B & x_B^2 + y_B^2 & 1 \\ x_C & y_C & x_C^2 + y_C^2 & 1 \\ x_D & y_D & x_D^2 + y_D^2 & 1 \end{vmatrix}$$
- ▶ (i) Punctele  $A, B, C, D$  sunt conciclice  $\Leftrightarrow \Theta(A, B, C, D) = 0$ .
  - ▶ (ii) Fie  $A, B, C$  astfel ca  $ABC$  să fie un viraj la stânga. Un punct  $D$  este situat în interiorul cercului circumscris  $\Delta ABC \Leftrightarrow \Theta(A, B, C, D) > 0$ .

## Exemplu



$$A = (1, 0)$$

$$B = (0, 1)$$

$$C = (-1, 0)$$

•  $\Delta ABC$  este viraj la stanga

$$D = (0, 0)$$

$$\bullet \text{ (H)}(A, B, C, D) = \begin{vmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ -1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{vmatrix} = (-1)^{4+4} \begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ -1 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 1 & 1 \\ -1 & 1 \end{vmatrix} \geq 0$$

D este în interiorul cercului circumscris  $\Delta ABC$ .

Exercițiu: calculați  $\text{ptr. } E = (0, -1)$  și  $F = (0, -2)$

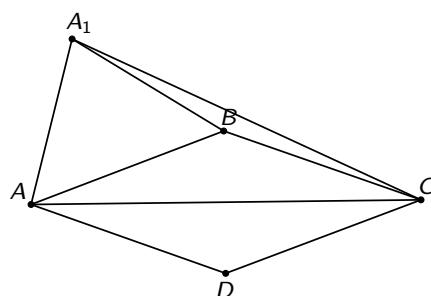
Algoritmi avansați - C10. Triangularea mulțimilor de puncte

15 / 21

Triangulări legale și triangulări unghiular optime

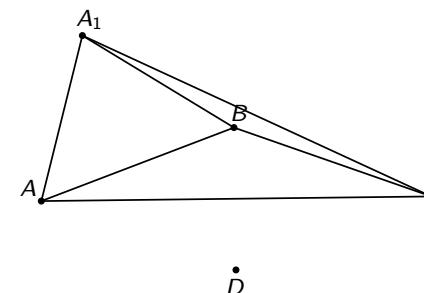
## Muchii ilegale, triangulări legale

► **Concluzie:** Dacă muchia  $AC$  este ilegală, printr-un *flip* (înlocuirea ei cu  $BD$ ), cel mai mic unghi poate fi mărit (local).



## Muchii ilegale, triangulări legale

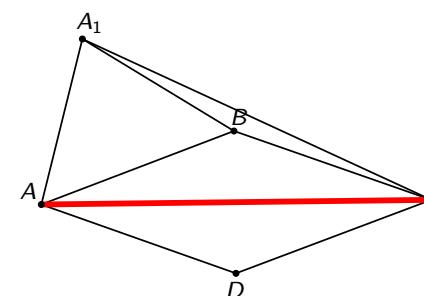
► **Concluzie:** Dacă muchia  $AC$  este ilegală, printr-un *flip* (înlocuirea ei cu  $BD$ ), cel mai mic unghi poate fi mărit (local).



Triangulări legale și triangulări unghiular optime

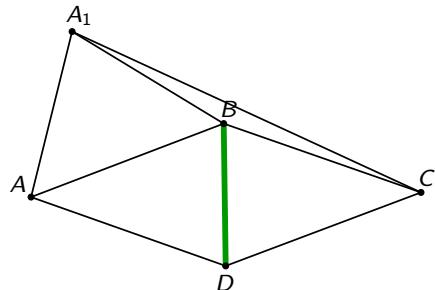
## Muchii ilegale, triangulări legale

► **Concluzie:** Dacă muchia  $AC$  este ilegală, printr-un *flip* (înlocuirea ei cu  $BD$ ), cel mai mic unghi poate fi mărit (local).



## Muchii ilegale, triangulări legale

- ▶ **Concluzie:** Dacă muchia  $AC$  este ilegală, printr-un *flip* (înlocuirea ei cu  $BD$ ), cel mai mic unghi poate fi mărit (local).



## Triangulări unghiular optime vs. triangulări legale

- ▶ **Propoziție.** Fie  $\mathcal{P}$  o mulțime de puncte din plan.

- (i) Orice triangulare unghiular optimă este legală.
- (ii) Dacă  $\mathcal{P}$  este în poziție generală (oricare patru puncte nu sunt conciclice), atunci există o unică triangulare legală, iar aceasta este unghiular optimă.

- ▶ **Teoremă.** Fie  $\mathcal{P}$  o mulțime de  $n$  puncte din plan, în poziție generală. *Triangularea unghiular optimă poate fi construită, folosind un algoritm incremental randomizat, în timp mediu  $O(n \log n)$ , folosind  $O(n)$  memorie medie.*

# Algoritmi avansați

## C11 - Diagrame Voronoi

Mihai-Sorin Stupariu

Sem. al II-lea, 2021 - 2022

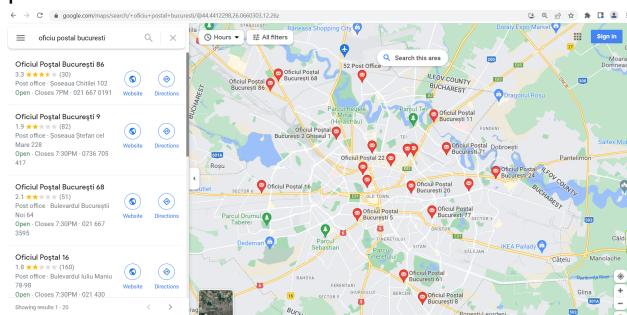
Algoritmi avansați - C11. Diagrame Voronoi  
Definiție, proprietăți elementare

## Definiție, proprietăți elementare

## Diagrame Voronoi și triangulări Delaunay

## Motivație

- **Problema oficiilor poștale:** Se consideră o mulțime de puncte (oficile poștale) din plan. Ce zone vor deservi aceste oficii?



Sursa: Google Maps

- Ideea de a delimita "zone de influență" a apărut cu multă vreme în urmă (de exemplu în lucrările lui Descartes, dar și în legătură cu alte probleme; este utilizată în mod curent în variii domenii. În plus, astfel de "împărțiri" apar în natură. Conceptul aferent este cel de **diagramă Voronoi** - există o multitudine de aplicații.

Algoritmi avansați - C11. Diagrame Voronoi

## Un algoritm eficient

Algoritmi avansați - C11. Diagrame Voronoi  
Definiție, proprietăți elementare

## Aplicații - rețele de transport

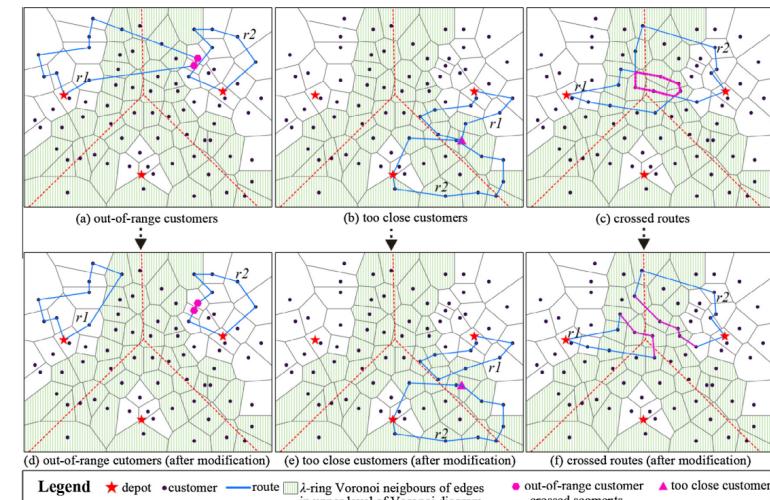


Fig. 3. Inappropriate routes between depots and their modifications.

Sursa: [Tu et al., 2014]

## Aplicații - rețele de transport

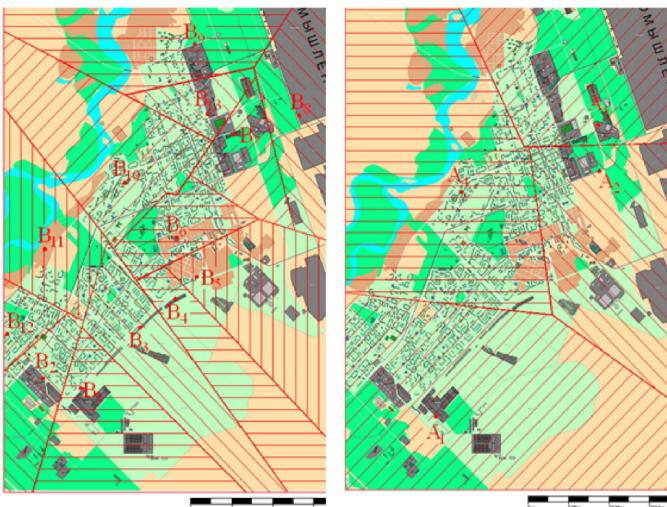


Fig. 4. Voronoi diagram to optimize: (a) the zones serviced by transport terminals of large city districts;(b) freight delivery zones from the logistic centers of a large city.

Sursa: [Lebedeva et al., 2018]

Algoritmi avansați - C11. Diagrame Voronoi  
Definiție, proprietăți elementare

## Aplicații - medicină

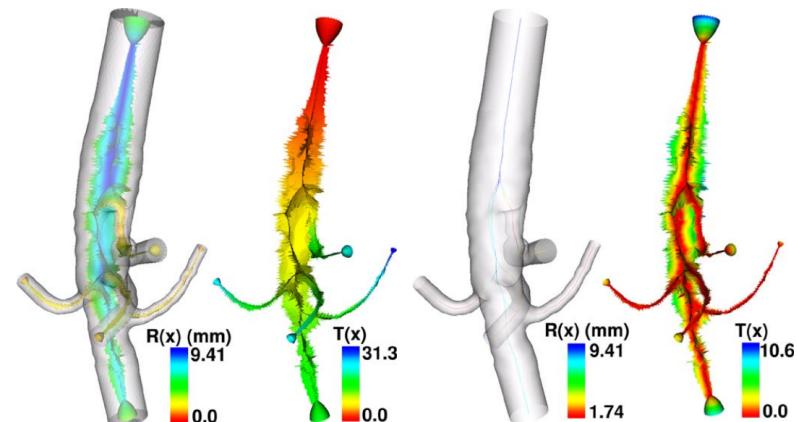


Fig. 8. Voronoi diagram colored according to maximal inscribed sphere radius values  $R(x)$  (left), and solution  $T(x)$  of Eikonal equation from central path points (right) for the abdominal aorta model (Surface: 8096 points, 16188 triangles. Voronoi diagram: 26463 points, 18370 polygons. Voronoi diagram computation time: 10 s. Solution of Eikonal equation time: 8 s).

Fig. 9. Calculated central paths (left) and solution of Eikonal equation from central path points (right) for subsequent surface characterization (total five central paths backtracking time: 1 s. Solution of Eikonal equation time: 8 s).  $R(x)$  is maximal inscribed ball radius values defined on central paths, and  $T(x)$  is radial traveltine (since in this case  $F(x) = 1$  everywhere,  $T(x)$  also represents geodesic distance).

Sursa: [Antiga et al., 2003]

Algoritmi avansați - C11. Diagrame Voronoi  
Definiție, proprietăți elementare

## Aplicații - medicină

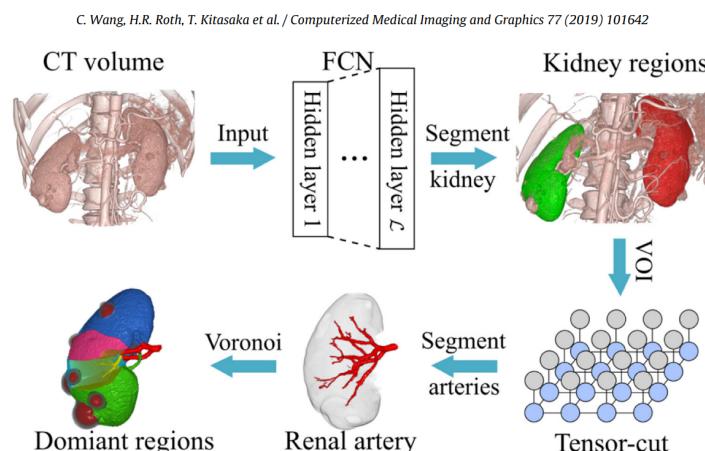
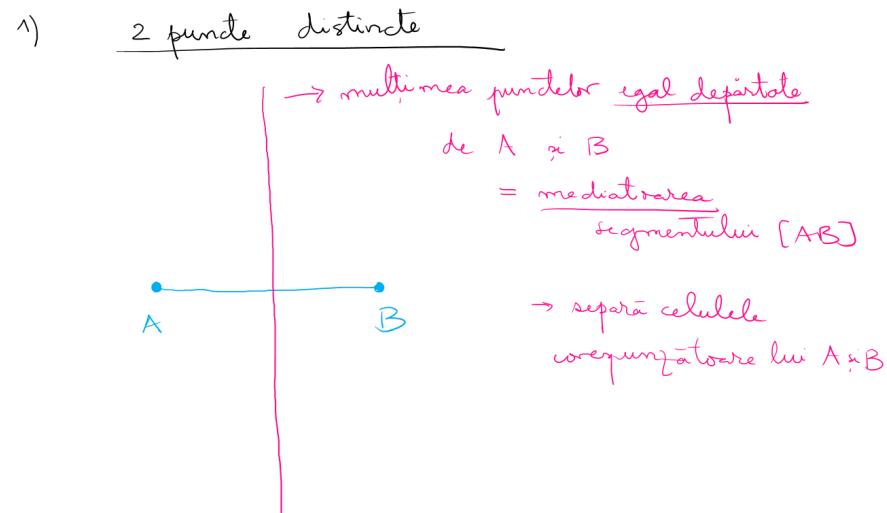


Fig. 1. Workflow: Our precise estimation approach can be divided into three parts: kidney segmentation, renal artery segmentation and estimation of vascular dominant regions.

Sursa: [Wang et al., 2019]

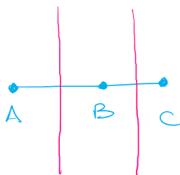
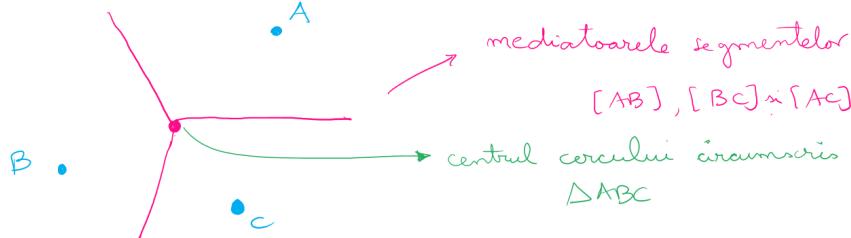
Algoritmi avansați - C11. Diagrame Voronoi

## Exemple - diagramme Voronoi în context 2D



Algoritmi avansați - C11. Diagrame Voronoi

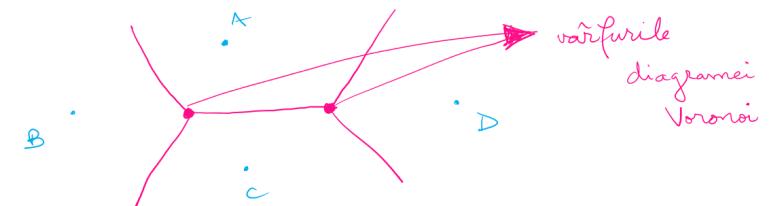
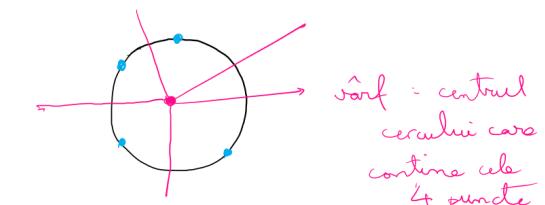
## Exemple - diagrame Voronoi în context 2D

2) 3 puncte distincte, coliniare3) 3 puncte necoliniare

## Formalizare

- Fie  $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$  o mulțime de puncte din planul  $\mathbb{R}^2$ , numite **situri**.
- **Diagrama Voronoi** a lui  $\mathcal{P}$  (notată  $\text{Vor}(\mathcal{P})$ ) este o divizare a planului  $\mathbb{R}^2$  în  $n$  celule  $\mathcal{V}(P_1), \dots, \mathcal{V}(P_n)$  cu proprietatea că
$$P \in \mathcal{V}(P_i) \Leftrightarrow d(P, P_i) \leq d(P, P_j), \forall j = 1, \dots, n.$$
- Două celule adiacente au în comun o *muchie* sau un *vârf* (punct de intersecție a muchiilor).
- **Atenție!** Vârfurile lui  $\text{Vor}(\mathcal{P})$  sunt diferențe de punctele din  $\mathcal{P}$ .
- Uneori, prin abuz de limbaj, este precizată doar împărțirea în muchii / vârfuri.
- Diagrame Voronoi pot fi construite pentru **diverse funcții distanță** (e.g. **distanța Manhattan**); forma celulelor depinde de forma "cercului" în raport cu funcția distanță respectivă.

## Exemple - diagrame Voronoi în context 2D

4) 4 puncte necoliniare, neconciacise4') 4 puncte coincidice

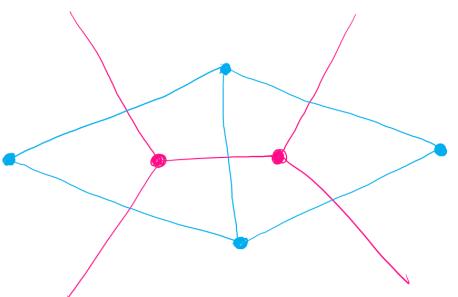
## Structura unei diagrame Voronoi

- Fie  $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$  o mulțime de situri (puncte) din planul  $\mathbb{R}^2$ .
- Celula asociată unui punct este o intersecție de semiplane:
$$\mathcal{V}(P_i) = \bigcap_{j \neq i} h(P_i, P_j),$$
unde  $h(P_i, P_j)$  este semiplanul determinat de mediatoarea segmentului  $[P_i P_j]$  care conține punctul  $P_i$ . În particular: fiecare celulă este o mulțime convexă.  
Aplicabilitate: algoritm (lent) de determinare a diagramei Voronoi.
- Dacă toate punctele sunt coliniare, atunci diagrama Voronoi asociată  $\text{Vor}(\mathcal{P})$  conține  $n - 1$  drepte paralele între ele (în particular, pentru  $n \geq 3$ , ea nu este conexă).
- În caz contrar, diagrama este conexă, iar muchiile sale sunt fie *segmente*, fie *semidrepte* (cui corespund acestea?).
- **Propoziție.** Fie o mulțime cu  $n$  situri. Atunci, pentru diagrama Voronoi asociată au loc inegalitățile
$$n_v \leq 2n - 5, \quad n_m \leq 3n - 6,$$
unde  $n_v$  este numărul de vârfuri ale diagramei și  $n_m$  este numărul de muchii al acesteia.

$$n_v \leq 2n - 5, \quad n_m \leq 3n - 6,$$

unde  $n_v$  este numărul de vârfuri ale diagramei și  $n_m$  este numărul de muchii al acesteia.

## Legătura cu triangulările legale / unghiular optime

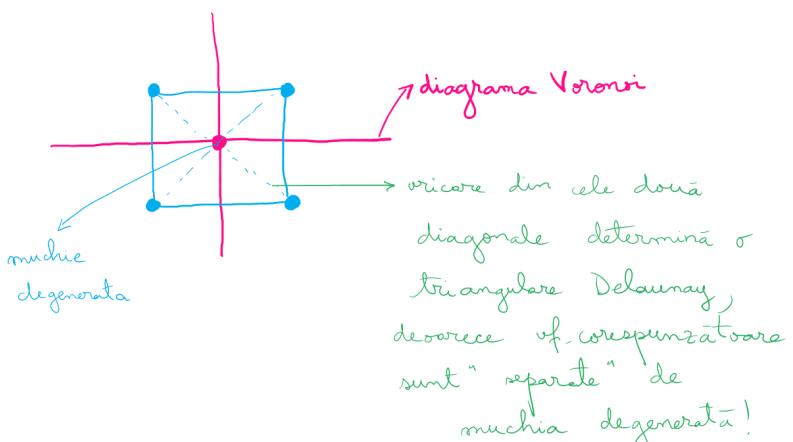


► Construcție:

- Mulțime de puncte  $\mathcal{P}$  în planul  $\mathbb{R}^2 \implies$
- Diagrama Voronoi  $\text{Vor}(\mathcal{P}) \implies$
- Graful dual  $\mathcal{G}(\mathcal{P})$ . **Noduri:** fețele diagramei Voronoi (siturile). **Arce:** dacă celulele (fețele diagramei Voronoi corespunzătoare) au o muchie comună  $\implies$
- Triangulare  $\mathcal{T}_{\mathcal{P}}$  (numită **triangulare Delaunay**)

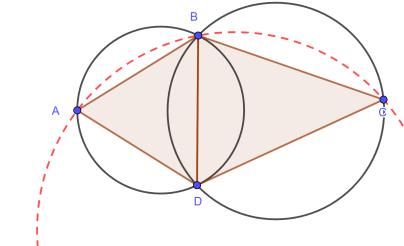
## Legătura cu triangulările legale / unghiular optime - cazul unui pătrat

**Întrebare:** Cum “funcționează” această construcție când punctele din  $\mathcal{P}$  sunt (de exemplu) vârfurile unui pătrat?



## Legătura cu triangulările legale / unghiular optime

- **Propoziție.** Fie  $\mathcal{T}$  o triangulare a lui  $\mathcal{P}$ . Atunci  $\mathcal{T}$  este o triangulare Delaunay dacă și numai dacă pentru orice triunghi din  $\mathcal{T}$  cercul circumscris nu conține în interiorul său niciun punct al lui  $\mathcal{P}$ .



Pentru punctele  $A, B, C, D$  din figură, triangularea construită este o triangulare Delaunay. În schimb, triunghiul  $\Delta ABC$  nu poate participa la realizarea unei triangulări Delaunay, deoarece  $D$  este în interiorul cercului circumscris acestuia.

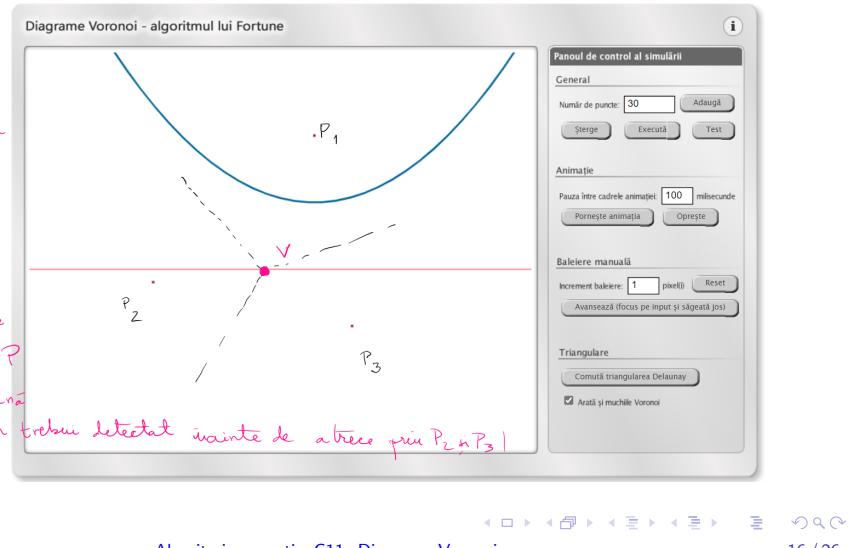
- **Teoremă.** O triangulare este legală dacă și numai dacă este o triangulare Delaunay.
- **Teoremă.** Orice triangulare unghiular optimă este o triangulare Delaunay. Orice triangulare Delaunay maximizează cel mai mic unghi, comparativ cu toate triangulările lui  $\mathcal{P}$ .
- Alte link-uri: <http://www.cs.cornell.edu/info/people/chew/Delaunay.html>  
<http://cgm.cs.mcgill.ca/~godfried/teaching/projects.pr.98/tesson/taxi/taxivoro.html>

## Algoritmul lui Fortune [1987]

- Metodă clasică (și eficientă) de determinare a diagramei Voronoi pentru o mulțime de puncte din planul  $\mathbb{R}^2$ . Complexitate:  $O(n \log n)$ . Detalii: <http://www.ams.org/samplings/feature-column/fcarc-voronoi>. Suport vizual disponibil pe grupul MTeams.
- **Principiu (paradigmă):** sweep line / dreaptă de baleiere.

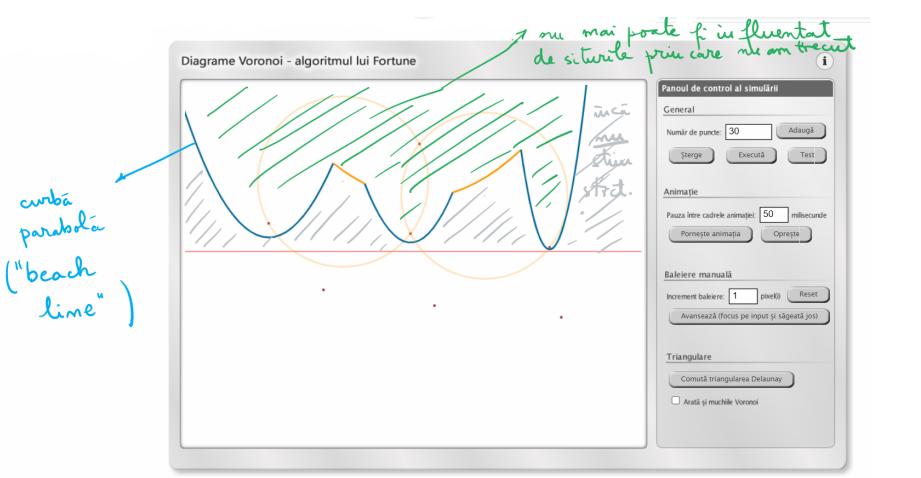
## Adaptarea paradigmiei dreptei de baleiere - problematizare

**Inconvenient:** la întâlnirea unui vîrf al diagramei, dreapta de baleiere nu a întâlnit încă toate siturile (punkte din  $\mathcal{P}$ ) care determină acest vîrf!



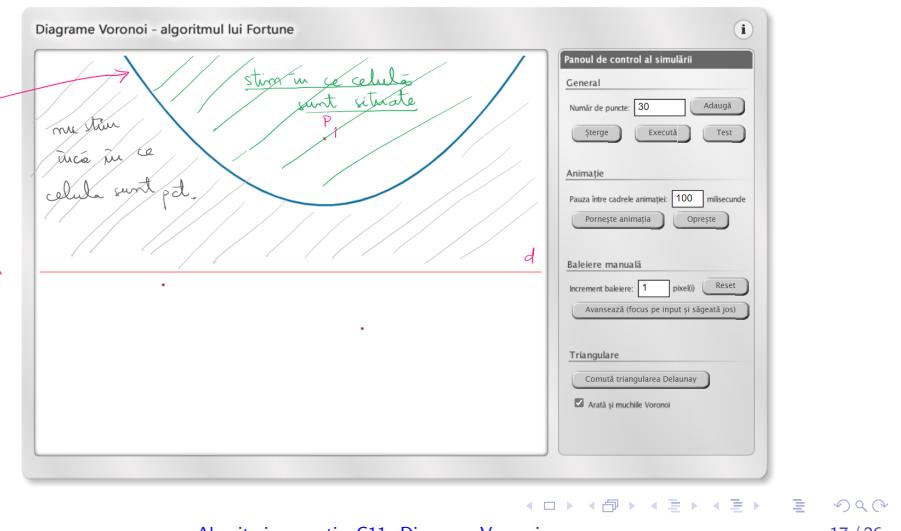
## Adaptarea paradigmiei dreptei de baleiere - curba parabolică

Din punct de vedere practic, apare o reuniune de arce de parabolă (**curbă parabolică**), ceea ce este situat deasupra acestei curbe nu mai poate fi influențat de evenimentele nedetectate.



## Adaptarea paradigmiei dreptei de baleiere

**Adaptare:** nu reținem informația legată de intersecția dintre dreapta de baleiere și diagramă, ci doar informația legată de partea diagramei care nu mai poate fi influențată de punctele situate de dincolo de dreapta de baleiere.



## Despre curba parabolică (I)

### ► Curba parabolică (beach line):

- Este o reuniune de arce de parabolă.
- Un punct de pe curba parabolică este egal depărtat de situl care determină arcul de parabolă și dreapta de baleiere. Presupunem că dreapta de baleiere  $d$  are ecuația  $y = 0$ , iar situl  $P_i$ , situat deasupra lui  $d$  (adică  $y_i > 0$ ) are coordonatele  $(x_i, y_i)$ . Locul geometric al punctelor egal depărtate de  $d$  și  $P_i$  are ecuația

$$y = \frac{1}{2y_i}x^2 - \frac{x_i}{y_i}x + \frac{x_i^2 + y_i^2}{2y_i}.$$

- Punctele de racord ale arcelor de parabolă aparțin multor diagramei Voronoi;
- Curba parabolică este  $x$ -monotonă, adică orice dreaptă verticală o intersecează exact într-un punct (la ce folosește această proprietate?).

## Despre curba parabolică (II)

### ► Modificarea curbei parabolice:

- **Site event / eveniment de tip locație.** (i) La trecerea printr-un sit apare un arc de parabolă (care "la început" este degenerat) și, reciproc, apariția unui nou arc este posibilă doar la trecerea printr-un sit. (ii) În consecință, la un moment fixat, curba parabolică are maxim  $(2n - 1)$  arce.
- **Circle event / eveniment de tip cerc.** (i) La întâlnirea punctului inferior al unui cerc care trece prin cel puțin trei situri și este tangent la dreapta de baleiere dispare un arc de parabolă și, reciproc, arcele de parabolă dispar doar la acest tip de evenimente. (ii) Un eveniment de tip cerc este dat de trei arce de parabolă consecutive de pe curba parabolică, deci trebuie testate toate tripletele consecutive de arce, pe măsură ce ele apar. (iii) Un astfel de eveniment este asociat unui vîrf al diagramei Voronoi. (iv) Există triplete de arce consecutive (muchii ale diagramei Voronoi) pentru care muchiile nu se întâlnesc. (v) Unele evenimente de tip cerc detectate nu au loc.

## Structuri de date utilizate

- **Diagrama Voronoi:** listă dublu înlănită DCEL  $\mathcal{D}$
- **Evenimente:** coadă de priorități / evenimente  $\mathcal{Q}$  și arbore de căutare binar echilibrat. Sunt reținute evenimentele de tip sit, precum și evenimentele de tip cerc - detectate pe parcursul algoritmului.
- **Statut:** Structura curbei parabolice - arbore de căutare binar echilibrat  $\mathcal{T}$ .
  - **pe frunze:** siturile care au arce active, la ajungerea într-un sit se inserează un arc, iar la un eveniment de tip cerc se șterge un arc;
  - **în nodurile interne:** punctele de racord ale arcelor de parabolă (memorate simbolic) - corespund muchiilor diagramei Voronoi
  - pointeri (eventual nuli) de la frunze către evenimentele de tip cerc; de la nodurile interne către muchiile diagramei Voronoi
- **Analiză preliminară a complexității:** (i) Câte evenimente sunt în total? (ii) Care este complexitatea-timp pentru a actualiza coada de evenimente la un eveniment de tip cerc? (iii) Care este complexitatea-timp (totală) pentru a actualiza coada de evenimente? (iv) Întrebări similare pentru statut.

## Adaptarea paradigmelor dreptei de baleiere - formalizare

- **Statut:** structura curbei parabolice (succesiunea de arce de parabolă); este modificat de două tipuri de evenimente

### ► Evenimente:

- site event / eveniment de tip locație: întâlnirea unui sit, adică a unui punct din mulțimea  $\mathcal{P}$  (apare un arc de parabolă)
- circle event / eveniment de tip cerc: întâlnirea unui "punct inferior" al unui cerc care trece prin cel puțin trei situri, tangent la dreapta de baleiere (dispare un arc de parabolă) - **vîrf al diagramei Voronoi**

## Algoritmul

**Input.** O mulțime de situri  $\mathcal{P} = \{p_1, \dots, p_n\}$  de situri în plan.

**Output.** Diagrama Voronoi  $\text{Vor}(\mathcal{P})$  în interiorul unui *bounding box*, descrisă printr-o listă dublu înlănită  $\mathcal{D}$ .

1. Inițializări: coada de evenimente  $\mathcal{Q} \leftarrow \mathcal{P}$  (preprocesare: ordonare după  $y$ ), statut (arbore balansat)  $\mathcal{T} \leftarrow \emptyset$ ; listă dublu înlănită  $\mathcal{D} \leftarrow \emptyset$ .
2. **while**  $\mathcal{Q} \neq \emptyset$ 
  3.     **do** elimină evenimentul cu cel mai mare  $y$  din  $\mathcal{Q}$
  4.     **if** evenimentul **ev** este un eveniment de tip sit
    5.         **then** **PROCESSEvSIT**( $p_i$ ), cu  $p_i = \text{ev}$
    6.         **else** **PROCESSEvCERC**( $\gamma$ ), cu  $\gamma = \text{arc(ev)} \in \mathcal{T}$
  7. Nodurile interne încă prezente în  $\mathcal{T}$  corespund semidreptelor diagramei Voronoi. Consideră un *bounding box* care conține toate vîfurile diagramei Voronoi în interiorul său și leagă semidreptele de acest *bounding box*, prin actualizarea corespunzătoare a lui  $\mathcal{D}$ .
  8. Traversează muchiile pentru a adăuga celulele diagramei și pointeri corespunzători.

## Procedura PROCESSEVSIT ( $p_i$ )

1. Dacă  $\mathcal{T}$  este vidă, inserează  $p_i$  și revine, dacă nu continuă cu 2.–5.
2. Caută în  $\mathcal{T}$  arcul  $\alpha$  situat deasupra lui  $p_i$ . Dacă frunza reprezentând  $\alpha$  are un pointer către un eveniment de tip cerc **ev** din  $\mathcal{Q}$ , atunci **ev** este o alarmă falsă și trebuie șters.
3. Înlocuește frunza lui  $\mathcal{T}$  care reprezintă  $\alpha$  cu un subarbore cu trei frunze: cea din mijloc reține situl  $p_i$  și celelalte două situl  $p_j$  asociat lui  $\alpha$ . Memorează perechile reprezentând punctele de racord în două noduri interne. Efectuează rebalansări în  $\mathcal{T}$ , dacă este necesar.
4. Generează noi înregistrări de tip semi-muchie în structura diagramei Voronoi ( $\mathcal{D}$ ), pentru muchiile care separă celulele  $V(p_i)$  și  $V(p_j)$ , corespunzând celor două noi puncte de racord.
5. Verifică tripletele de arce consecutive nou create, pentru a verifica dacă muchiile corespunzătoare punctelor de racord se întâlnesc. Dacă da, inserează evenimente de tip cerc în  $\mathcal{Q}$  și adaugă pointeri de la nodurile lui  $\mathcal{T}$  la evenimentele corespunzătoare din  $\mathcal{Q}$ .

## Rezultate principale

- ▶ **Teoremă.** *Diagrama Voronoi a unei mulțimi de  $n$  situri poate fi determinată cu un algoritm bazat pe paradigma dreptei de baleiere având complexitate-timp  $O(n \log n)$  și complexitate-spațiu  $O(n)$ .*
- ▶ **Teoremă.** *Triangularea Delaunay a unei mulțimi de  $n$  situri poate fi determinată cu un algoritm bazat pe paradigma dreptei de baleiere având complexitate-timp  $O(n \log n)$  și complexitate-spațiu  $O(n)$ .*

## Procedura PROCESSEVERC ( $\gamma$ )

1. Șterge frunza  $\gamma \in \mathcal{T}$  care corespunde arcului de cerc  $\alpha$  care dispare. Actualizează în nodurile interne perechile care corespund punctelor de racord. Efectuează rebalansări în  $\mathcal{T}$ , dacă este necesar. Șterge toate evenimentele de tip cerc care îi corespund lui  $\alpha$  (cu ajutorul pointerilor de la predecesorul și succesorul lui  $\gamma$  în  $\mathcal{T}$ ).
2. Adaugă centrul cercului care determină evenimentul ca înregistrare de tip vârf în  $\mathcal{D}$ . Creează înregistrări de tip semi-muchie corespunzând noului punct de racord de pe linia parabolică și asignează pointeri corespunzători.
3. Verifică tripletele de arce consecutive nou create (care au foștii vecini ai lui  $\alpha$  în centru), pentru a verifica dacă muchiile corespunzătoare punctelor de racord se întâlnesc. Dacă da, inserează evenimente de tip cerc în  $\mathcal{Q}$  și adaugă pointeri de la nodurile lui  $\mathcal{T}$  la evenimentele corespunzătoare din  $\mathcal{Q}$ .

# Algoritmi avansați

C12 - Elemente de programare liniară

Mihai-Sorin Stupariu

Sem. al II-lea, 2021-2022

Motivație: turnarea pieselor în matrițe

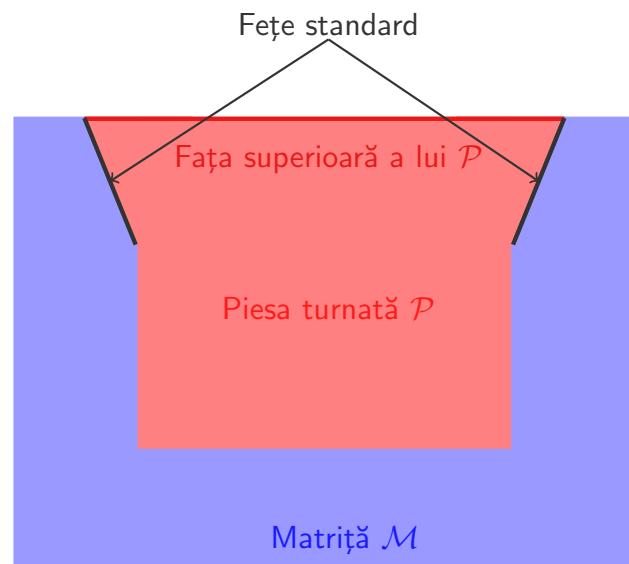
Intersecții de semiplane - abordare cantitativă

Dualitate

Intersecții de semiplane - abordare calitativă. Programare liniară

Algoritmi avansați - C12. Elemente de programare liniară  
Motivație: turnarea pieselor în matrițe

## Turnarea pieselor în matrițe



1 / 46

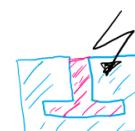
Algoritmi avansați - C12. Elemente de programare liniară

2 / 46

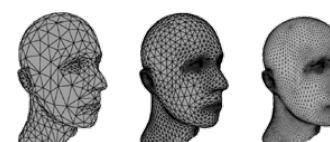
Motivație: turnarea pieselor în matrițe

## Problematizare

- ▶ Turnarea pieselor în matrițe și extragerea lor fără distrugerea matriței.
- ▶ Neajunsuri: unele obiecte pot rămâne blocați; există obiecte pentru care nu există o matriță adecvată.



- ▶ **Problema studiată.** Dat un obiect, există o matriță din care să poată fi extras (și dacă da, cu un algoritm eficient?)



Sursa: <https://www.graphics.rwth-aachen.de/publication/03149/>

Algoritmi avansați - C12. Elemente de programare liniară

6 / 46

Algoritmi avansați - C12. Elemente de programare liniară

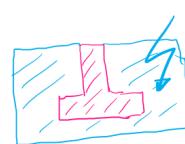
7 / 46

## Convenții

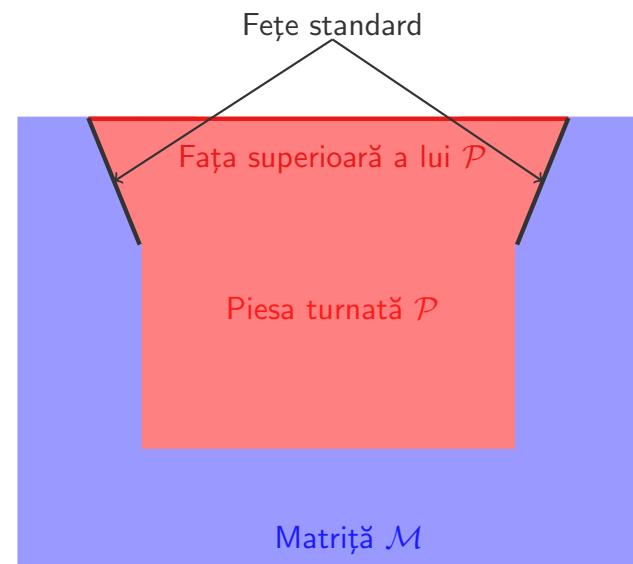
- Obiectele: **poliedrale**.
- Matrițele: formate dintr-o singură piesă; fiecărui obiect  $\mathcal{P}$  îi este asociată o matriță  $M_{\mathcal{P}}$
- Obiectul: extras printr-o singură translație (sau o succesiune de translații)
- **Alegerea orientării:** diverse orientări ale obiectului pot genera diverse matrițe...



- ... astfel încât doar în unele configurații este posibilă extragerea obiectului



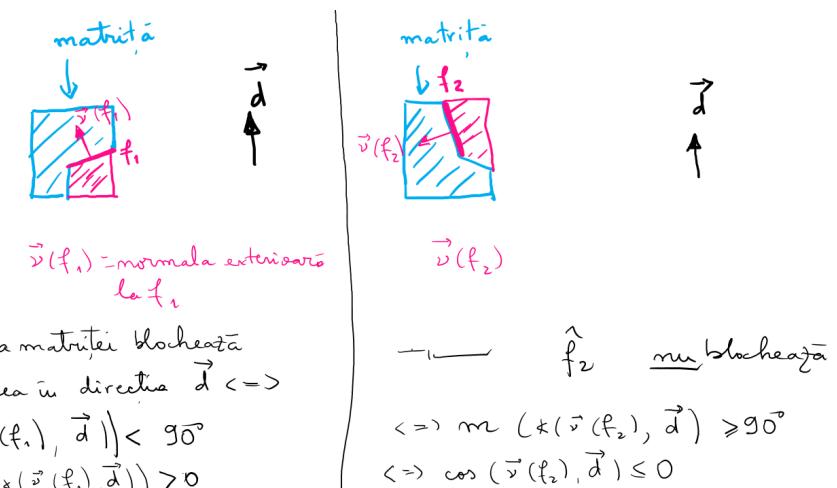
## Turnarea pieselor în matrițe



## Terminologie și convenții

- **Față superioară:** prin convenție, obiectele au (cel puțin) o față superioară (este orizontală, este singura care nu este adiacentă cu matriță). Celelalte fețe: **standard**; orice față standard  $f$  a obiectului corespunde unei fețe standard  $\hat{f}$  a matriței.
- **Obiect care poate fi turnat (castable):** există o orientare pentru care acesta poate fi turnat și apoi extras printr-o translație (succesiune de translații): *direcție admisibilă*.
- **Convenții:** Matriță este paralelipipedică și are o cavitate corespunzătoare obiectului; față superioară a obiectului (și a matriței) este perpendiculară cu planul  $Oxy$ .

## Descrierea proprietății de a putea extrage o piesă într-o direcție dată



Această condiție trebuie verificată pînă la următoarele fețe!

## Detaliere (scriere în coordonate)

$$\text{Dacă } v, w \in \mathbb{R}^3 : \cos(\varphi(v, w)) = \frac{\langle v, w \rangle}{\|v\| \|w\|} \begin{pmatrix} \langle v, w \rangle = \\ v_1 w_1 + v_2 w_2 + v_3 w_3 \end{pmatrix}$$

Cum vrem să extragem obiectul "în sus", f.r.g. putem pp. că  $\vec{d} = (d_x, d_y, 1)$  (de ce ?)

În  $f$  o față fixată a obiectului;  $\vec{v}(f) = (v_x, v_y, v_z)$

Iaptul că față  $\hat{f}$  a matriței nu blochează extragerea în direcția  $\vec{d}$   $\Leftrightarrow$

$$\langle \vec{v}(f), \vec{d} \rangle \leq 0 \Leftrightarrow$$

$$v_x \cdot d_x + v_y \cdot d_y + v_z \leq 0 \quad (\times_f)$$

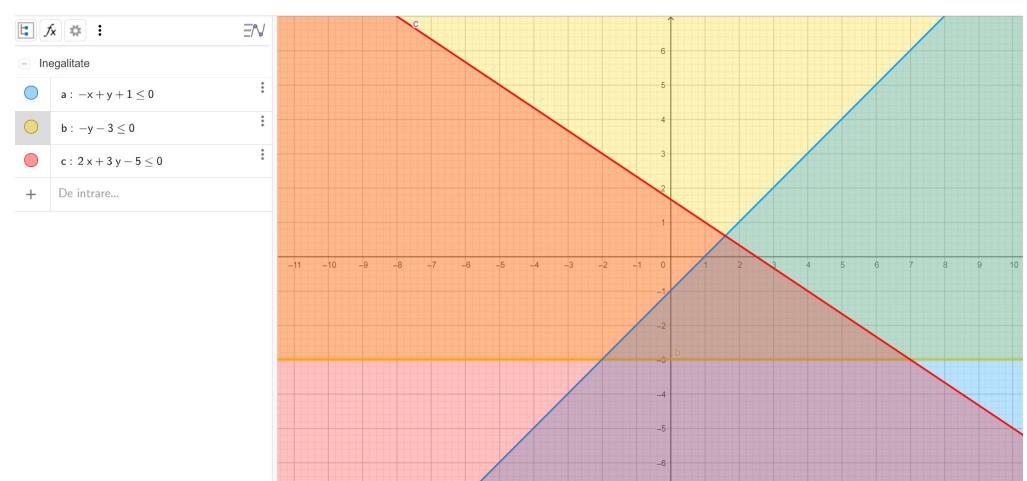
Fixată  $f$  ( $\rightarrow (v_x, v_y, v_z)$ ) cointan  $\vec{d}$  ( $d_x, d_y$ ) și să se verifice  $(\times_f)$

$(\times_f)$ : inecuație care descrie un semiplan

## Exemple

### 1. Intersecția semiplanelor

$$-x + y + 1 \leq 0; \quad -y - 3 \leq 0; \quad 2x + 3y - 5 \leq 0.$$



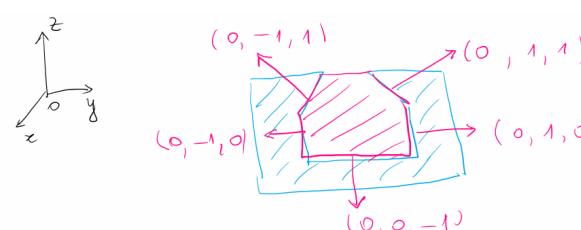
## Fundamente geometrice

- ▶ **Condiție necesară:** direcția de extragere  $\vec{d}$  trebuie să aibă componenta  $z$  pozitivă
- ▶ **În general:** o față standard  $\hat{f}$  a matriței (corespunzătoare unei fețe  $f$  a piesei) pentru care unghiul dintre normala exteroioră  $\vec{v}(f)$  la față  $f$  și  $\vec{d}$  este mai mic de  $90^\circ$  împiedică translația în direcția  $\vec{d}$
- ▶ **Propoziție.** Un poliedru  $P$  poate fi extras din matriță sa  $M_P$  prin translație în direcția  $\vec{d}$  dacă și numai dacă  $\vec{d}$  face un unghi de cel puțin  $90^\circ$  cu normala exteroioră a fiecărei fețe standard a lui  $P$ .
- ▶ **Reformulare.** Dat  $P$ , trebuie găsită o direcție  $\vec{d}$  astfel încât, pentru fiecare față standard  $f$ , unghiul dintre  $\vec{d}$  și  $\vec{v}(f)$  să fie cel puțin  $90^\circ$ .
- ▶ **Analitic - pentru o față:** fiecare față definește un semiplan, i.e. dată o față standard  $f$  a poliedrului / matriței, a găsi o direcție admisibilă revine la a rezolva o inecuație  $(\times_f)$ , care corespunde unui semiplan.
- ▶ **Analitic - toate fețele:** Fie  $P$  un poliedru; față superioară fixată, paralelă cu planul  $Oxy$ . Considerăm matrița asociată și toate fețele matriței (i.e. toate fețele standard ale poliedrului). A determina o direcție admisibilă revine la a determina o direcție care verifică toate inegalitățile de tip  $(*)$ , deci un sistem de inecuații.
- ▶ **Concluzie:** Pentru a stabili dacă există o direcție admisibilă, trebuie stabilit dacă o intersecție de semiplane este nevidă.

## Exemple

### 2 (a). Normalele exterioare ale fețelor standard sunt coliniare cu vectorii

$$(0, -1, 1), (0, 1, 1), (0, 1, 0), (0, 0, -1), (0, -1, 0).$$



sistem incompatibil,  
obiectul nu poate  
fi extras

$$\left. \begin{array}{l} (0, -1, 1) \rightsquigarrow 0 \cdot x + (-1) \cdot y + 1 \leq 0 \\ (0, 1, 1) \rightsquigarrow 0 \cdot x + 1 \cdot y + 1 \leq 0 \\ (0, 1, 0) \rightsquigarrow 0 \cdot x + 1 \cdot y + 0 \leq 0 \\ (0, 0, -1) \rightsquigarrow 0 \cdot x + 0 \cdot y + (-1) \leq 0 \\ (0, -1, 0) \rightsquigarrow 0 \cdot x + (-1) \cdot y + 0 \leq 0 \end{array} \right\} \begin{array}{l} y \geq 1 \\ y \leq -1 \\ y \leq 0 \\ -1 \leq 0 \\ y \geq 0 \end{array}$$

## Temă

- 2 (b).** Normalele exterioare ale fețelor standard sunt coliniare cu vectorii  $(0, 1, 0), (0, 1, -1), (0, 0, -1), (0, -1, -1), (0, -1, 0)$ .

### (i) Caracterizare explicită - Formularea problemei

- Fie  $\mathcal{H} = \{H_1, H_2, \dots, H_n\}$  o mulțime de semiplane din  $\mathbb{R}^2$ ; semiplanul  $H_i$  dat de o relație de forma

$$a_i x + b_i y + c_i \leq 0$$

- Intersecția  $H_1 \cap H_2 \cap \dots \cap H_n$  este dată de un sistem de inecuații; este o mulțime poligonală convexă, mărginită de cel mult  $n$  muchii (poate fi vidă, mărginită, nemărginită,...)

## Intersecții de semiplane - probleme studiate, rezultate

### ► Probleme studiate:

- (i) **Caracterizare explicită:** Să se determine care sunt elementele (vârfuri, muchii, etc.) care determină o intersecție de semiplane.
- (ii) **Calitativ:** Să se stabilească dacă o intersecție de semiplane este nevidă.

### ► Rezultate: (descrise în detaliu ulterior)

- (i) *Intersecția unei mulțimi de  $n$  semiplane poate fi determinată cu complexitate-timp  $O(n \log n)$  și folosind  $O(n)$  memorie.*
- (ii) *Se poate stabili cu complexitate-timp medie  $O(n)$  dacă o intersecție de semiplane este nevidă.*
- (ii)' *Fie  $\mathcal{P}$  un poliedru cu  $n$  fețe. Se poate decide dacă  $\mathcal{P}$  reprezintă un obiect care poate fi turnat cu complexitate-timp medie  $O(n^2)$  și folosind  $O(n)$  spațiu. În caz afirmativ, o mătriță și o direcție admisibilă în care poate fi extras  $\mathcal{P}$  este determinată cu aceeași complexitate-timp.*

### Dualitate – motivație euristică

- De câte informații (numerice) este nevoie pentru a indica **un punct în plan?**
- 2
- De câte informații (numerice) este nevoie pentru a indica **o dreaptă în plan?**
- 2
- Există o modalitate naturală de a stabili o corespondență între puncte și drepte?
- DA: dualitate**
- Cum se reflectă / respectă diferite proprietăți geometrice (de exemplu incidența) prin dualitate?

## Dualitate – definiții

- unei punct  $p = (p_x, p_y)$  din planul  $\mathbb{R}^2$  (plan primal) i se asociază o dreaptă notată  $p^*$  (în planul dual)

$$p^*: (y = p_x \cdot x - p_y)$$

duala lui  $p$

- unei drepte nevertecale  $d: (y = m_d \cdot x + n_d)$  din planul primal i se asociază un punct din planul dual, notat  $d^*$ :

$$d^* = (m_d, -n_d)$$

dualul lui  $d$

Obs. Această transformare este polaritatea față de parabola  $y = \frac{x^2}{2}$ .

## Dualitate – proprietăți elementare

### 1) Păstrează incidenta

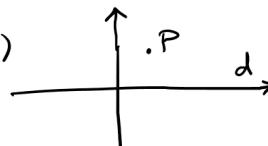
$p$  este situat deasupra dreptei  $d$  (neverticală)  $\Leftrightarrow$

$$d^* \text{ --- --- } p^*$$

Exemplu

Pl. primal  
 $p = (1, 1)$

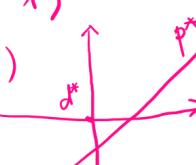
$$d: (y = 0)$$



Pl. dual

$$p^*: (y = x - 1)$$

$$d^* = (0, 0)$$



## Dualitate – proprietăți elementare

### 1) Păstrează incidenta

$$p \in d \Leftrightarrow d^* \in p^*$$

Exemplu

Pl. primal

$$d: (y = 2x + 1)$$

$$p = (1, 3)$$

Pl. dual

$$d^* = (2, -1)$$

$$p^*: (y = x - 3)$$

## Dualitate – “dicționar” concepte și configurații

### Plan primal

Punct  $p$

Dreaptă neverticală  $d$

Dreaptă determinată de două puncte

Punctul  $p$  deasupra dreptei  $d$

Segment

### Plan dual

Dreaptă neverticală  $p^*$

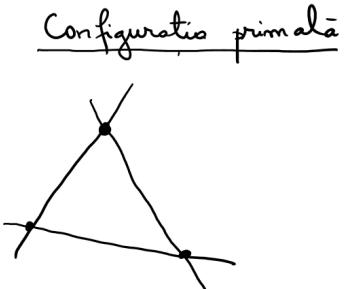
Punct  $d^*$

Punct de intersecție a două drepte

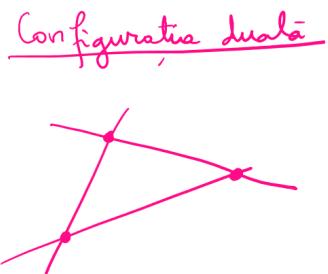
Punctul  $d^*$  deasupra dreptei  $p^*$

Fascicul de drepte (wedge)

## Exemplu



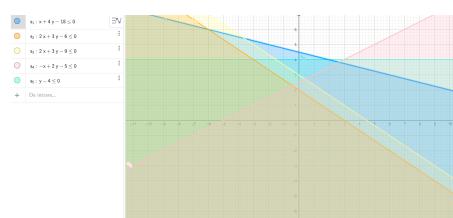
3 puncte necoliniare  
și dreptele determinate  
de ele



3 drepte care nu  
trece prin același punct  
și punctele determinate  
de ele

## Semiplane inferioare și semiplane superioare

Când determinăm o intersecție de semiplane inferioare / superioare, nu sunt neapărat relevante toate semiplanele. În figura de mai jos sunt considerate cinci semiplane inferioare  $s_1, s_2, s_3, s_4, s_5$  dintre care relevante pentru intersecție sunt doar  $s_2$  și  $s_4$ .



## Semiplane inferioare și semiplane superioare

### ► Exemple.



semiplan inferior



semiplan superior

### ► Dat un semiplan delimitat de o dreaptă neverticală

$$ax + by + c \leq 0$$

cum se decide dacă este semiplan inferior sau semiplan superior?

Exemple:

$$-x + y + 3 \leq 0 \quad \text{semiplan inferior}$$

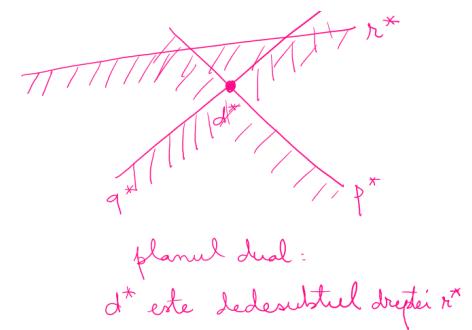
$$x - y - 3 \leq 0 \quad \text{semiplan superior}$$

## Observația fundamentală

Fie  $p, q$  cu  $p \neq q$  și dreapta  $d = pq$  neverticală. Fie  $r$  un punct situat dedesubtul dreptei  $d = pq$ . Care este configurația duală?



planul primal:  
 $r$  este dedesubtul dreptei  $pq = d$

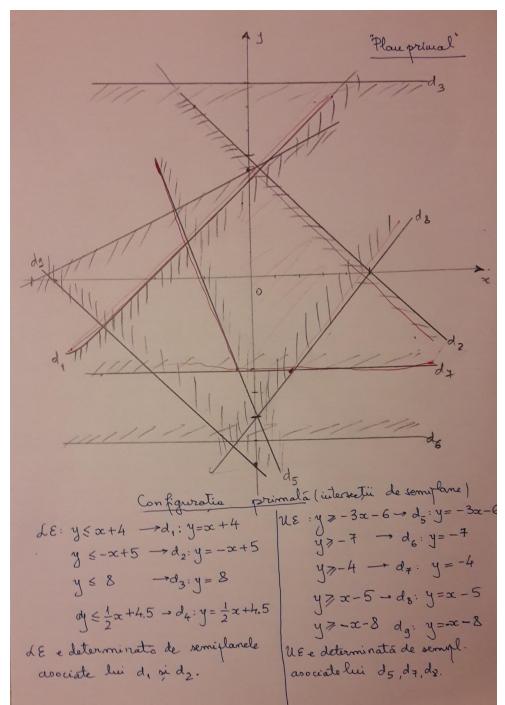


planul dual:  
 $d^*$  este dedesubtul dreptei  $r^*$

## Observația fundamentală

- ▶ Fie  $\mathcal{P}$  o mulțime de puncte.
- ▶ **Q:** Ce înseamnă că un segment  $[pq]$  ( $p, q \in \mathcal{P}$ ) participă la frontieră superioară a acoperirii convexe a lui  $\mathcal{P}$ ?
- ▶ **A:** Toate celelalte puncte sunt dedesubtul dreptei  $d = pq$ .
- ▶ Configurația duală: Punctul  $d^*$  este situat dedesubtul dreptelor corespunzătoare celorlalte puncte și, prin trecere la semiplane inferioare, "contează" semiplanele inferioare determinate de  $p^*$  și  $q^*$ .

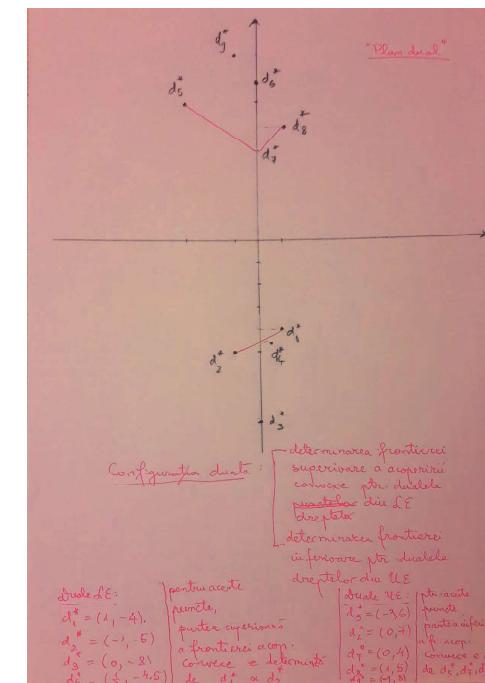
## Exemplu



## Concluzie pentru (i) - abordarea cantitativă

- ▶ Pentru a determina o intersecție de **semiplane inferioare** se consideră mulțimea de puncte din planul dual și se determină **frontiera superioară** a acoperirii convexe a mulțimii respective. Un rezultat analog are loc pentru intersecții de **semiplane superioare** și **frontiera inferioară** a acoperirii convexe a mulțimii de puncte duale. În consecință:
- ▶ **Teoremă** *Intersecția a n semiplane poate fi descrisă cu un algoritm de complexitate  $O(n \log n)$ .*

## Exemplu



## (ii) Abordarea calitativă. Motivație

- Sunt realizate 3 produse (notate 1, 2 și 3) pe 2 aparate (notate X și Y).
- Ciclul de producție este săptămânal (40h de lucru). Timpul de producție (în minute) pentru produs este indicat în tabel.

	X	Y	Obs.	Nr. prod.	Spațiu	Profit
1	10	27	pe ambele	$x_1$	$0.1m^2$	10
2	12	19	în paralel, simultan	$x_2$ , respectiv $y_2$	$0.2m^2$	13
3	8	24	în paralel, simultan	$x_3$ , respectiv $y_3$	$0.05m^2$	9

- Aparatele X și Y au un interval de menenanță de 5%, respectiv 7% din timpul de lucru. Spațiul total de depozitare este de  $50m^2$ .

► Modelul matematic:

**Constrângerile:**

$$\begin{aligned} 0.1x_1 + 0.2(x_2 + y_2) + 0.05(x_3 + y_3) &\leq 50 && \text{Spațiu de depozitare} \\ 10x_1 + 12x_2 + 8x_3 &\leq 0.95 \cdot 40 \cdot 60 && \text{Timp aparatul } X \\ 27x_1 + 19y_2 + 24y_3 &\leq 0.93 \cdot 40 \cdot 60 && \text{Timp aparatul } Y \end{aligned}$$

**Cerință:**

$$\text{maximizează } (10x_1 + 13(x_2 + y_2) + 9(x_3 + y_3))$$

## Exemplu - cazul 1D ( $d = 1$ )

Coordonata  $x$ :

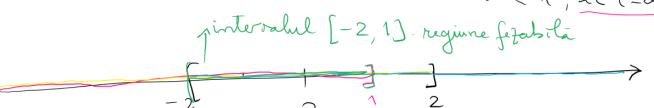
funție obiectiv

maximizează  $(cx)$

$$\left\{ \begin{array}{l} a_1x \leq b_1 \text{ - interval} \\ a_2x \leq b_2 \\ \vdots \\ a_nx \leq b_n \end{array} \right. \quad \text{constrângerile}$$

Exemplu concret: maximizează  $(2x)$

$$\left\{ \begin{array}{l} 3x \leq 6 \\ -2x \leq 4 \\ 6x \leq 6 \end{array} \right. \quad \left\{ \begin{array}{l} x \leq 2, \quad x \in (-\infty, 2] \\ x \geq -2, \quad x \in [-2, \infty) \\ x \leq 1, \quad x \in (-\infty, 1] \end{array} \right.$$



Maximul funcției obiectiv este egal cu 2 și se obține printr-o  $x = 1$

**Lemă.** (Pentru  $d = 1$ ) Un program liniar 1-dimensional poate fi rezolvat în timp liniar.

## Problematizare, terminologie

- Formulare generală (în spațiul  $d$ -dimensional):

$$\text{maximizează } (c_1x_1 + c_2x_2 + \dots + c_dx_d)$$

date constrângerile liniare (inegalități)

$$\left\{ \begin{array}{l} a_{11}x_1 + a_{12}x_2 + \dots + a_{1d}x_d \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2d}x_d \leq b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nd}x_d \leq b_n \end{array} \right. \quad (1)$$

- Denumiri:

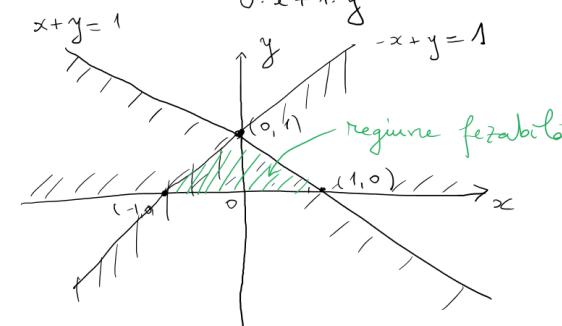
- date de intrare:  $(a_{ij})_{i=1,n, j=1,d}, (b_i)_{i=1,n}, (c_j)_{j=1,d}$
- funcție obiectiv:  $(c_1x_1 + c_2x_2 + \dots + c_dx_d)$
- constrângerile: inegalitățile (1)
- regiune realizabilă (fezabilă): intersecția semispațiilor care definesc constrângerile problemei

- Obs. Interpretare a cerinței de maximizare:** Maximizarea funcției obiectiv revine la a determina un punct al cărui vector de poziție are proiecția maximă de direcția dată de vectorul  $\vec{c} = (c_1, c_2, \dots, c_d)$ .

## Exemplu - cazul 2D ( $d = 2$ )

Notăm coordonatele cu  $x$  și  $y$ .

maximizează  $(y)$ ;  $\vec{c} = (0, 1)$ , date  $\left\{ \begin{array}{l} x + y \leq 1 \\ -x + y \leq 0 \\ -x + y \leq 1 \end{array} \right.$



Funcția are valoarea maximă 1, atinsă în punctul  $(0, 1)$ .

## Probleme de programare liniară în plan ( $d = 2$ )

### ► Convenții și terminologie:

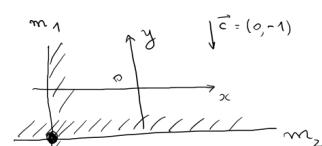
- Coordonatele:  $x$  și  $y$
- Funcția obiectiv:  $f_{\vec{c}}(p) = c_x x + c_y y$ , unde  $\vec{c} = (c_x, c_y)$ .
- Constrângerile:  $h_1, h_2, \dots, h_n$  (semiplane); se notează  $H = \{h_1, h_2, \dots, h_n\}$
- Regiunea fezabilă este  $C = h_1 \cap h_2 \cap \dots \cap h_n$ .
- **Program liniar:**  $(H, \vec{c})$ .
- **Scop:** Se caută  $p \in C$  astfel ca  $f_{\vec{c}}(p)$  să fie maximă.

► Pentru o problemă de programare liniară în plan pot fi distinse patru situații: (i) o soluție unică; (ii) toate punctele de pe o muchie sunt soluții; (iii) regiunea fezabilă este nemărginită și pot fi găsite soluții de-a lungul unei semidrepte; (iv) regiunea fezabilă este vidă.

## Algoritm incremental pentru rezolvarea unei probleme de programare liniară 2D

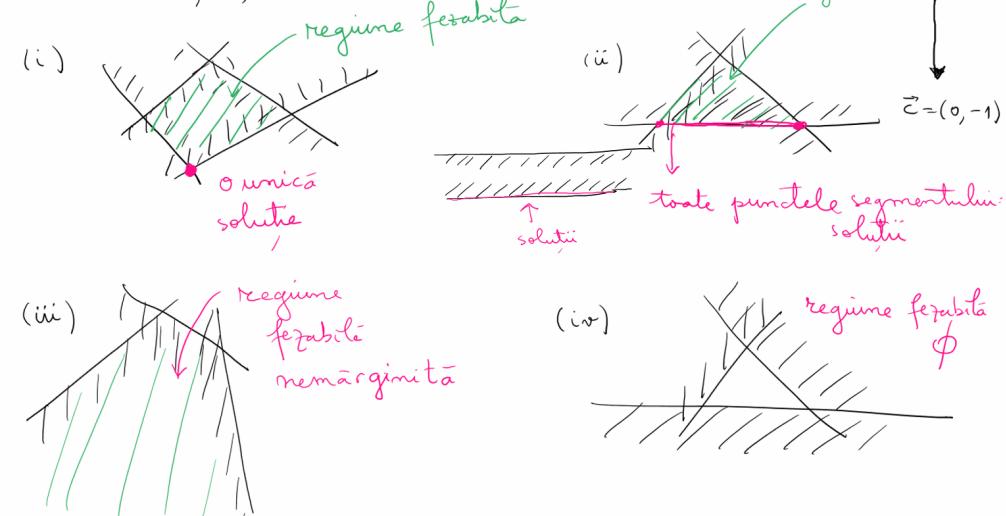
- **Principii:**
  - constrângerile sunt adăugate una câte una;
  - presupunem că la fiecare pas soluția (punctul de maxim) există, apoi actualizează;
  - sunt adăugate la început constrângerile care garantează mărginirea programului liniar, definite astfel: se alege  $M >> 0$  și se definesc noi constrângerile convenabile;
  - se lucrează cu convenția de ordonare lexicografică, astfel încât există o **unică** soluție optimă.
- Vom considera în continuare  $\vec{c} = (0, -1)$ , iar noile constrângerile vor fi:

$$m_1 : x \geq -M, \quad m_2 : y \geq -M.$$



## Cazul 2D ( $d = 2$ ) - exemple de regiuni fezabile

Fie  $\vec{c} = (0, -1)$



## Notății

- Fie  $(H, \vec{c})$  un program liniar cu constrângerile  $h_1, h_2, \dots, h_n$ . Se notează:

$$H_i = \{m_1, m_2, h_1, h_2, \dots, h_i\}, \text{ mulțime de semiplane}$$

$$C_i = m_1 \cap m_2 \cap h_1 \cap h_2 \cap \dots \cap h_i, \text{ regiune fezabilă.}$$

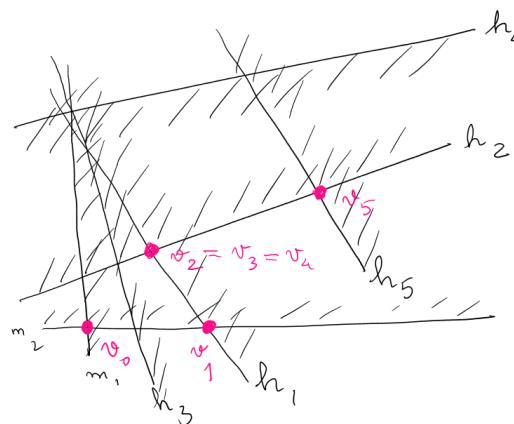
Notația este pentru  $i = 0, \dots, n$ , în particular

$$H_0 = \{m_1, m_2\} \quad C_0 = m_1 \cap m_2.$$

- **Observații:**

- $C_0 \supseteq C_1 \supseteq C_2 \supseteq \dots \supseteq C_n = C$ .
- Pentru fiecare  $i$ , regiunea fezabilă  $C_i$ , dacă este nevidă, are un vîrf care reprezintă o soluție optimă a problemei  $(H_i, \vec{c})$ . Punctul este notat cu  $v_i$  (deinde de alegerea lui  $m_1$  și  $m_2$ ).

## Exemplu



Algoritmi avansați - C12. Elemente de programare liniară  
Intersecții de semiplane - abordare calitativă. Programare liniară

Algoritm LPMARG2D ( $H, \vec{c}, m_1, m_2$ )

- **Input.** Un program liniar  $(H \cup \{m_1, m_2\}, \vec{c})$  din  $\mathbb{R}^2$
  - **Output.** Dacă  $(H \cup \{m_1, m_2\}, \vec{c})$  nu e realizabil (fezabil), raportează. În caz contrar, indică punctul cel mai mic lexicografic  $p$  care maximizează  $f_{\vec{c}}(p)$ .
1.  $v_0 \leftarrow$  "colțul" lui  $c_0$
  2. fie  $h_1, h_2, \dots, h_n$  semiplanele din  $H$
  3. **for**  $i \leftarrow 1$  **to**  $n$
  4.   **do if**  $v_{i-1} \in h_i$
  5.     **then**  $v_i \leftarrow v_{i-1}$
  6.     **else**  $v_i \leftarrow$  punctul  $p$  de pe  $d_i$  care maximizează  $f_{\vec{c}}(p)$  date constrângerile din  $H_i$
  7.     **if**  $p$  nu există
  8.       **then** raportează "nefezabil" **end**
  9. **return**  $v_n$

## Observații

- Fie  $1 \leq i \leq n$ , presupunem că  $C_{i-1}$  și  $v_{i-1}$  sunt determinate. Considerăm  $h_i$ . Sunt două situații:

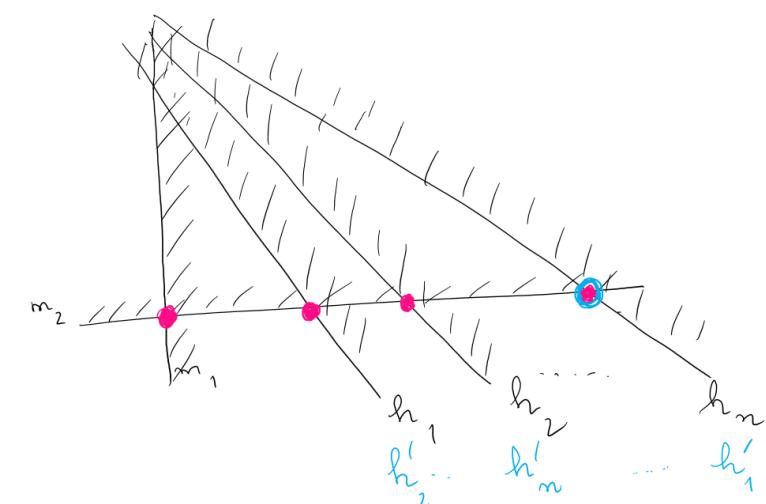
- (i) dacă  $v_{i-1} \in h_i$ , atunci  $v_i = v_{i-1}$ ,
- (ii) dacă  $v_{i-1} \notin h_i$ , atunci

fie  $C_i = \emptyset$

fie  $v_i \in d_i$ , unde  $d_i$  este dreapta care mărginește  $h_i$ .

În acest caz, găsirea lui  $v_i$  revine la găsirea lui  $p \in d_i$  care maximizează  $f_{\vec{c}}(p)$ , date constrângerile deja existente ( $p \in h, \forall h \in H_i$ ). De fapt, aceasta este o problemă pe programare liniară 1-dimensională, care are complexitatea-timp liniară, adică  $O(i)$ .

## Comentariu - ordinea contează

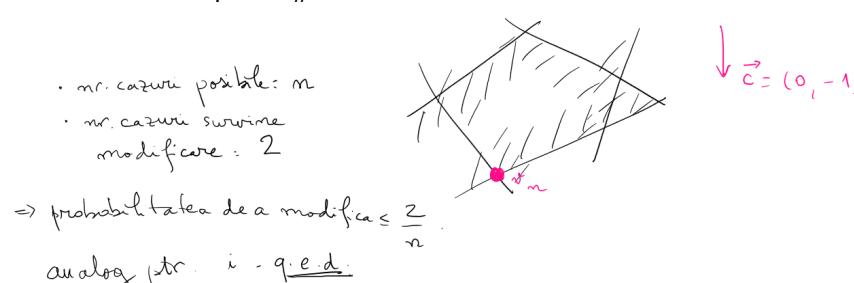


## Algoritm aleatoriu

- ▶ Pasul 2. este înlocuit cu:  
2'. Calculează o permutare arbitrară a semiplanelor, folosind o procedură adecvată.
- ▶ Algoritmul incremental LPMARG2D are complexitate-timp  $O(n^2)$ , iar varianta bazată pe alegerea aleatorie a semiplanelor are complexitate-timp medie  $O(n)$  ( $n$  este numărul semiplanelor).

## Analiza complexității-timp - varianta algoritmului probabilist (II)

- ▶ Demonstrăm că  $\mu(X_i) \leq \frac{2}{i}$ , pentru orice  $i = 1, \dots, n$ , adică probabilitatea ca  $v_{i-1} \notin h_i$  este  $\leq \frac{2}{i}$ .
- ▶ Arătăm inegalitatea pentru  $i = n$  (cazul general, analog).  
Presupunem algoritmul terminat,  $v_n$  vârful optim.
  - Care este probabilitatea ca  $v_{n-1} \notin h_n$ , adică la adăugarea lui  $h_n$ , vârful  $v_{n-1}$  să fie modificat în  $v_n$ ?  $\Leftrightarrow$
  - Care este probabilitatea ca eliminând unul dintre semiplane să fie modificat vârful optim  $v_n$ ?



## Analiza complexității-timp - varianta algoritmului probabilist (I)

Jie  $(X_i)_{i=1..n}$  variabilă aleatoare definită astfel:

$$X_i = \begin{cases} 0, & \text{dacă } v_{i-1} \in h_i \text{ (adică este ales pasul 5)} \\ 1, & \dots \quad v_{i-1} \notin h_i \text{ (pasul 6)} \end{cases}$$

$\Rightarrow$  timpul total  $\sum_{i=1}^n X_i O(i)$

Valoarea așteptată (timpul mediu):

$$\mathbb{E} \left[ \sum_{i=1}^n X_i O(i) \right] \stackrel{\text{altă notatie}}{=} \mu \left( \sum_{i=1}^n X_i O(i) \right) = \sum_{i=1}^n O(i) \cdot \underline{\mu(X_i)} \leq \\ \leq \sum_{i=1}^n O(i) \cdot \frac{2}{i} = O(n)$$

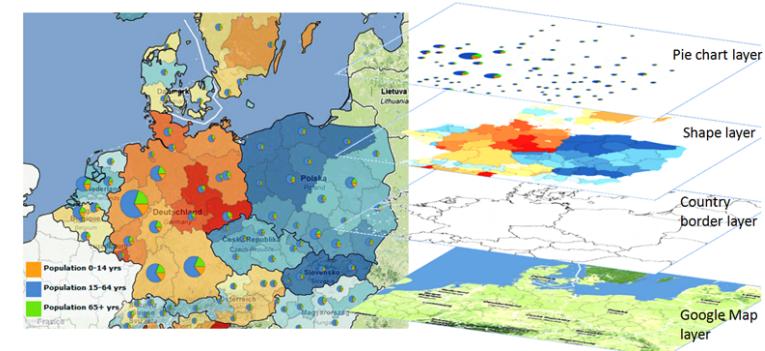
↑ afirmație

Motivație - reprezentarea datelor geo-spațiale

## Probleme de localizare

Mihai-Sorin Stupariu

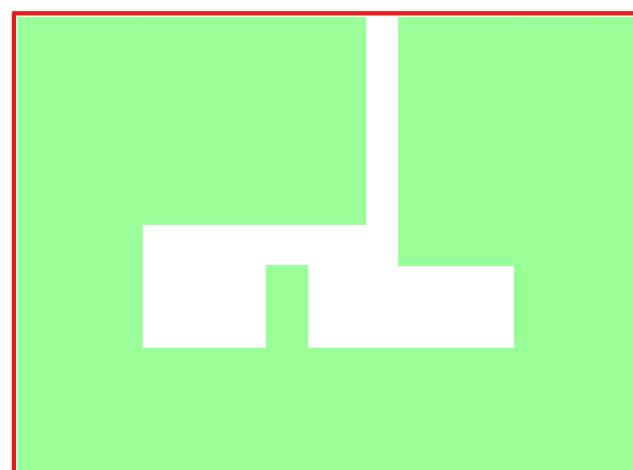
Sem. I, 2021-2022



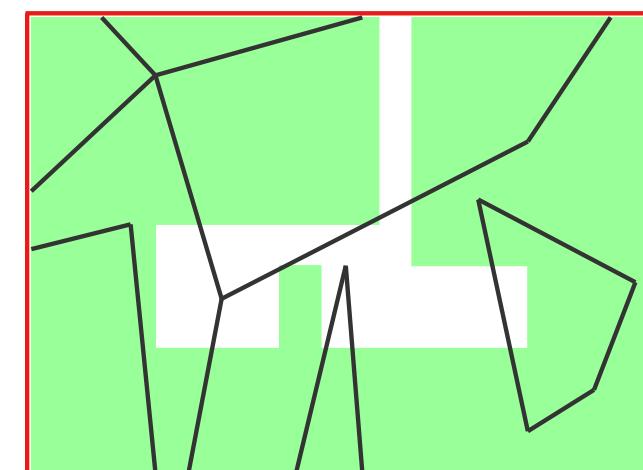
Sursa: <https://s-media-cache-ak0.pinimg.com/originals/37/90/86/37908600ab7db99c424c3bc6e1ddb740.jpg>

Ce structură de date este adecvată pentru a memora astfel de informații?

## Problematizare



## Problematizzare



## Conceptul cheie

Concept: DCEL (doubly connected edge list)  
 bazat pe: semi-muchie (muchie orientată)  
"half-edge"



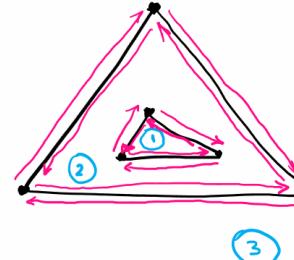
- Conceptul cheie este cel de semi-muchie (muchie orientată), cf. "half-edge".
- Dat un poligon (eventual cu goluri):
  - frontieră exterioară**, care poate fi parcursă cu ajutorul semi-muchiilor astfel încât poligonul să fie la stânga frontierei, iar virajele convexe să fie la stânga,
  - frontieră interioară** (dacă există goluri), caz în care poligonul este tot la stânga, dar virajele în vîrfurile convexe sunt la dreapta.

## Subdiviziuni planare

- Conceptul de subdiviziune planară: vîrfuri, muchii, fețe.
- Listă de muchii dublu înlăntuite / DCEL - Doubly Connected Edge List** [Müller și Preparata, 1978] (trei înregistrări: vîrfuri, fețe, muchii orientate (semi-muchii)).
  - Vîrf**  $v$ : coordonatele lui  $v$  în  $Coordinates(v)$ , pointer  $IncidentEdge(v)$  spre o muchie orientată care are  $v$  ca origine
  - Față**  $f$ : pointer  $OuterComponent(f)$  spre o muchie orientată corespunzătoare frontierei externe (pentru față nemărginită este **nil**); listă  $InnerComponents(f)$ , care conține, pentru fiecare gol, un pointer către una dintre muchiile orientate de pe frontieră
  - Muchie orientată**  $\vec{e}$ : pointer  $Origin(\vec{e})$ , pointer  $Twin(\vec{e})$  pointer  $IncidentFace(\vec{e})$ , pointer  $Next(\vec{e})$ , pointer  $Prev(\vec{e})$ .
- Oricarei subdiviziuni planare  $\mathcal{S}$  i se asociază o listă de muchii dublu înlăntuite  $\mathcal{D}_{\mathcal{S}}$ .
- Obs.** Explicați cum, folosind pointerii de mai sus: (i) poate fi parcursă frontieră exterioară / interioară a unei fețe (a unui poligon); (ii) pot fi găsite toate semi-muchiile incidente cu un vîrf.

## Exemplu

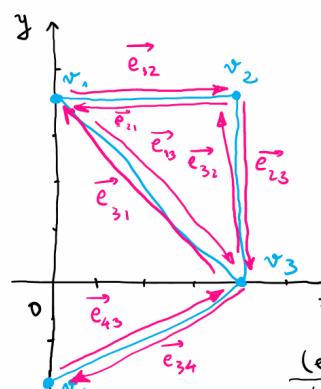
### Exemplu



6 vîrfuri  
12 semi-muchii  
3 fețe

- ! de returnat
- semi-muchiile generează frontiere ale fețelor, fata delimitată fiind la stânga
- frontiere
  - exterioare (viraje la stânga în vf. convexe)
  - interioare (viraje la dreapta în vf. convexe)

## Exemplu



Vîrf	Coordinate	Incident Edge
$v_1$	(0, 4)	
$v_2$	(4, 4)	
$v_3$	(4, 0)	
$v_4$	(0, -2)	

Față	Outer Component	Inner Component
(exterior) $f_1$	nil	$\vec{e}_{12}$
(D) $f_2$	$\vec{e}_{23}$	nil

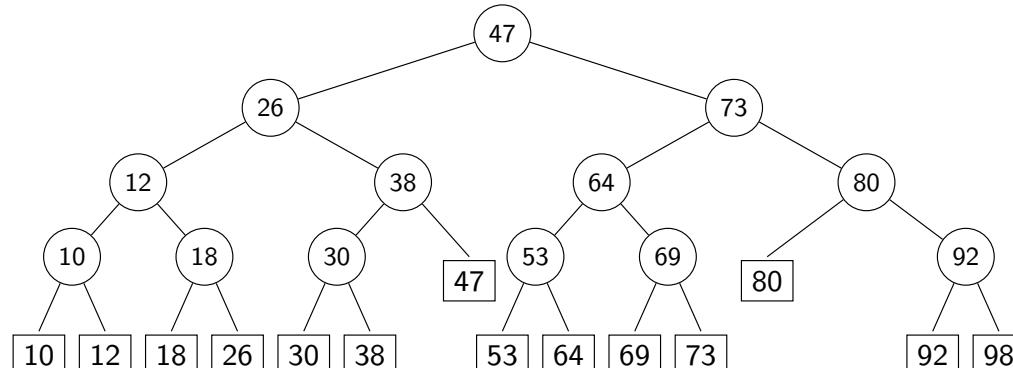
Semi-muchie	Origin	Twin	Incident Face	Next	Prev
$\vec{e}_{12}$	$v_1$	$\vec{e}_{21}$	$f_1$	$\vec{e}_{33}$	$\vec{e}_{31}$
$\vec{e}_{23}$	$v_2$	$\vec{e}_{32}$	$f_1$	$\vec{e}_{34}$	$\vec{e}_{12}$
$\vec{e}_{34}$	$v_3$	$\vec{e}_{43}$	$f_2$	$\vec{e}_{43}$	$\vec{e}_{23}$

## Căutare ortogonală – motivăție

## Exemplu.

Baza de date a unei bănci: informații numerice referitoare la clienți: data nașterii, număr de copii, venitul lunar, valoarea depozitelor, valoarea ratelor de plată, valoarea comisioanelor plătite anual, etc. → stocarea se realizează folosind puncte dintr-un spațiu numeric  $d$ -dimensional  $\mathbb{R}^d$ .

A identifica un "grup-țintă" de clienți (de exemplu pentru lansarea unui produs), având anumite caracteristici – e.g. vîrstă între 30-40 ani, 2-4 copii, un venit lunar între 3000-5000 lei, etc. revine la efectuarea căutării prin care să fie determinate punctele situate într-un "paralelipiped"  $d$ -dimensional.



## Căutare 1-dimensională: formularea problemei

**Cadru.** Fie  $M = \{a_1, a_2, \dots, a_n\}$  o mulțime de numere reale. Fie  $I = [x, x'] \subset \mathbb{R}$  un interval real. Se dorește determinarea elementelor lui  $M$  situate în intervalul  $I$ .

**Structura de date utilizată:** Arbore binar de căutare echilibrat.

## Rezultatul principal - căutare 1D

**Teorema.** Fie  $M$  o mulțime de  $n$  puncte din  $\mathbb{R}$ . Mulțimea  $M$  poate fi memorată într-un arbore binar de căutare echilibrat, folosind  $O(n)$  memorie și cu timp de construcție  $O(n \log n)$ . Determinarea unor puncte dintr-un interval  $I$  poate fi realizată cu complexitate-timp  $O(k + \log n)$ , unde  $k$  este numărul de puncte din  $M \cap I$ .

## Rezultatul principal - căutare 2D

**Teoremă.** Fie  $M$  o mulțime de  $n$  puncte din planul  $\mathbb{R}^2$ . Un arbore de intervale (range tree) pentru  $M$  necesită  $O(n \log n)$  memorie și poate fi construit în timp  $O(n \log n)$ . Determinarea unor puncte dintr-un dreptunghi  $D$  poate fi realizată cu complexitate-timp  $O(k + \log^2 n)$ , unde  $k$  este numărul de puncte din  $M \cap D$ .

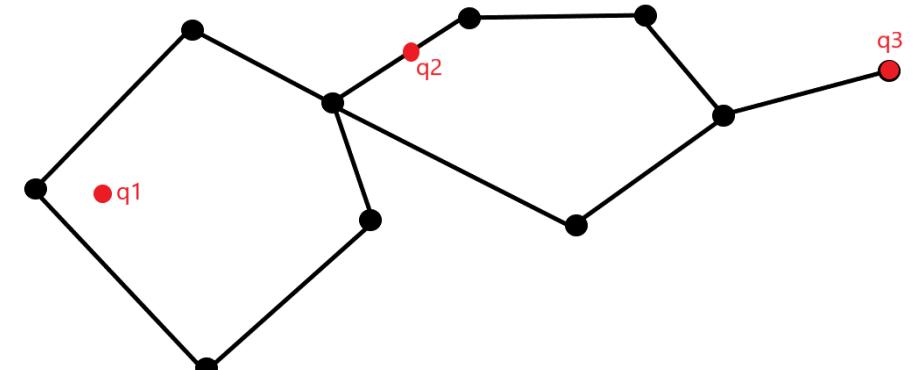
## Formalizare

- ▶ Fie  $\mathcal{S}$  o subdiviziune planară cu  $n$  muchii. Problema localizării unui punct revine la
  - ▶ a reține informațiile referitoare la  $\mathcal{S}$  pentru a putea răspunde la interogări de tipul:
  - ▶ dat un punct  $p$ , se raportează fața  $f$  care îl conține pe  $p$ ; în cazul în care  $p$  este situat pe un segment sau coincide cu un vârf, este precizat acest lucru.
- ▶ Lucrul cu coordonate: folosirea relației de ordine!

## Localizarea punctelor – problematizare

- ▶ Căutare cu Google Maps
- ▶ Interrogare pentru localizarea unui punct: dată o hartă și un punct  $p$ , indicat prin cordonatele sale, să se determine regiunea hărții în care este situat  $p$ .
- ▶ Harta: subdiviziune planară, formată din vârfuri, (semi)muchii, fețe.
- ▶ Necesități: pre-procesare a informației; interogare rapidă.

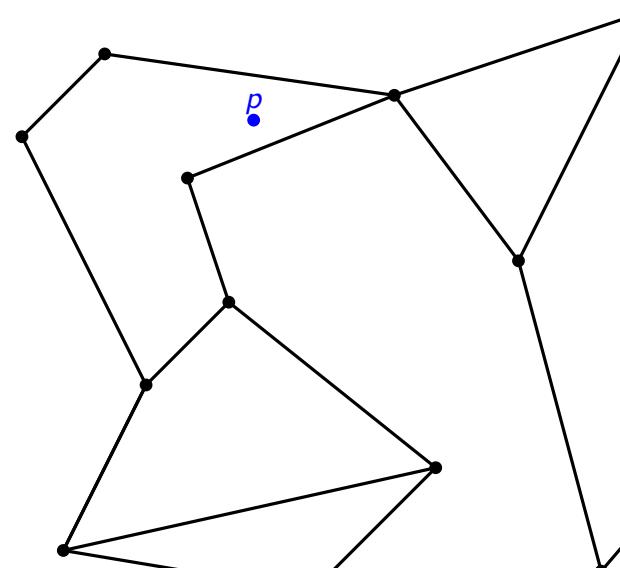
## Intuiție



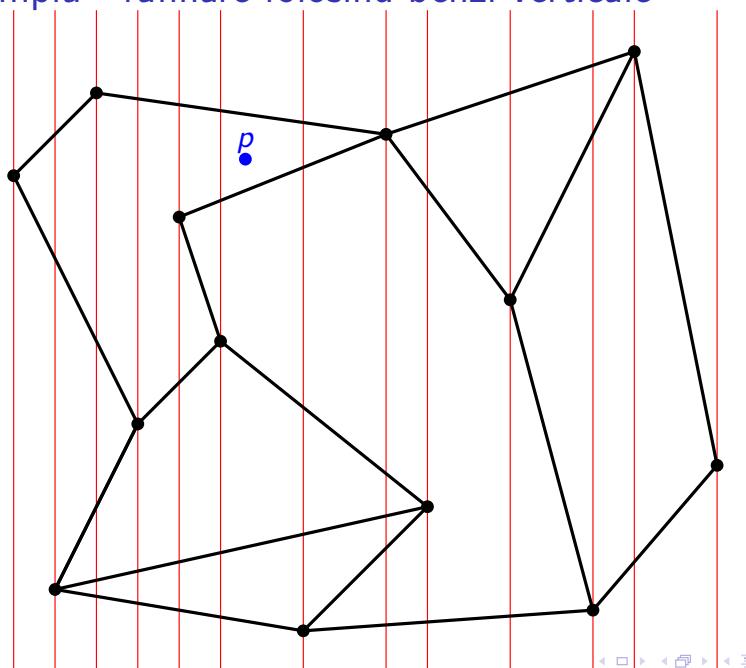
## Intuiție - concluzie

- ▶ Subdivizare a planului în fâșii (benzi) verticale (cf. "slabs")
    - ▶ **căutare după abscisă** - pentru identificarea fâșiei verticale (timp de căutare  $O(\log n)$ );
    - ▶ **căutare în cadrul unei fâșii** - pentru localizare în cadrul fâșiei verticale, realizată în raport cu segmente.

## Exemplu

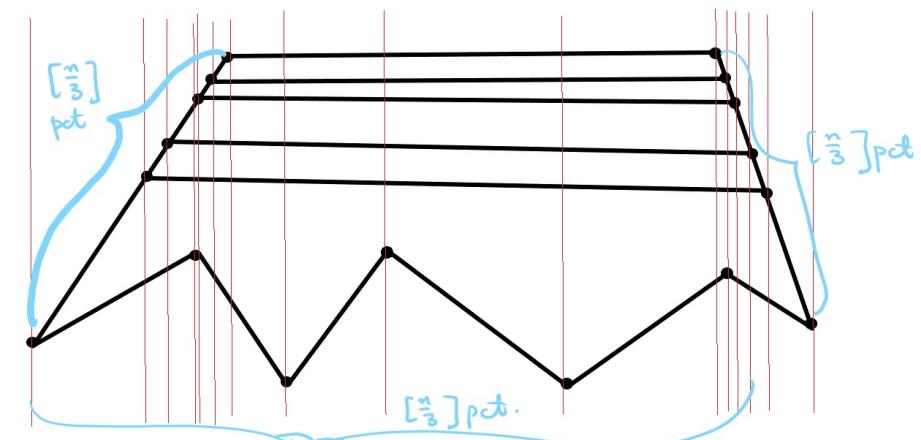


Exemplu - rafinare folosind benzi verticale

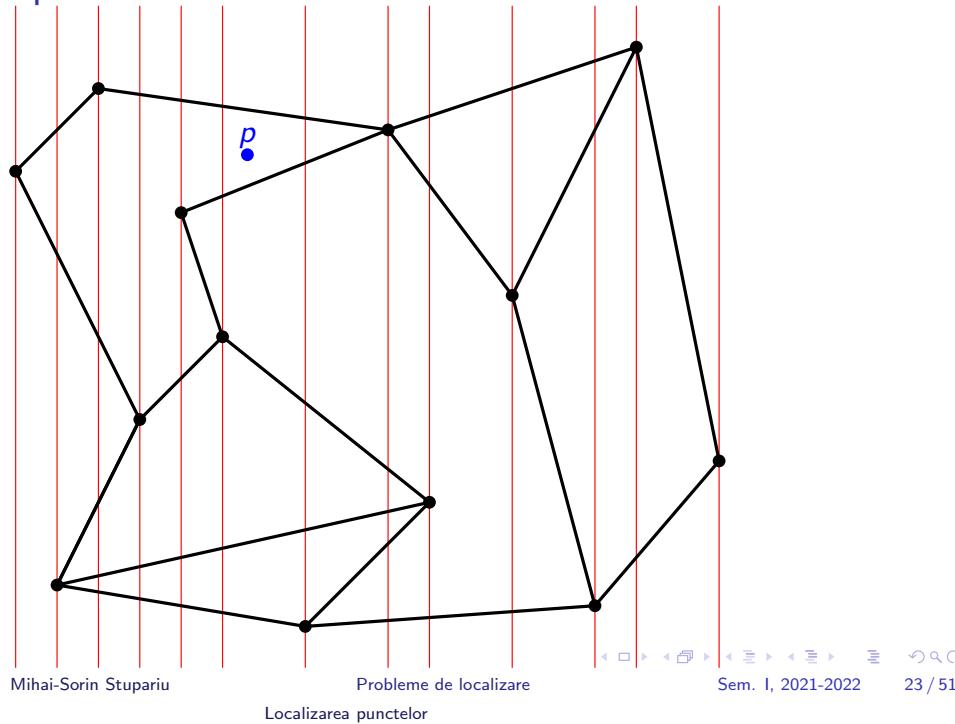


O astfel de rafinare nu este eficientă

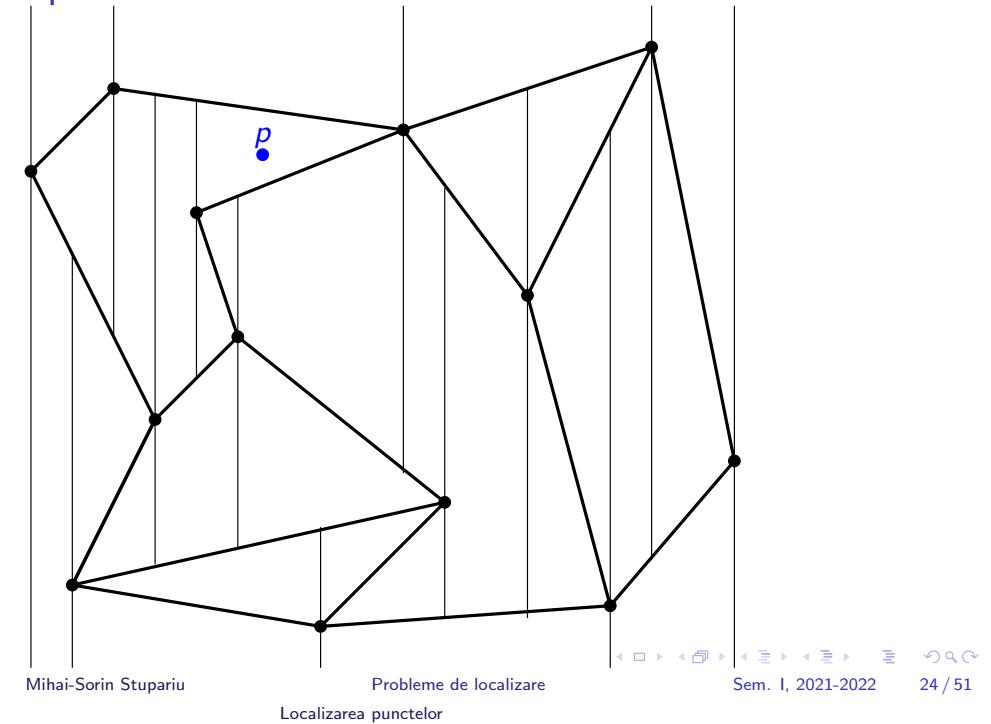
Memoria necesară poate fi uneori  $O(n^2)$



## Exemplu - rafinare folosind benzi verticale

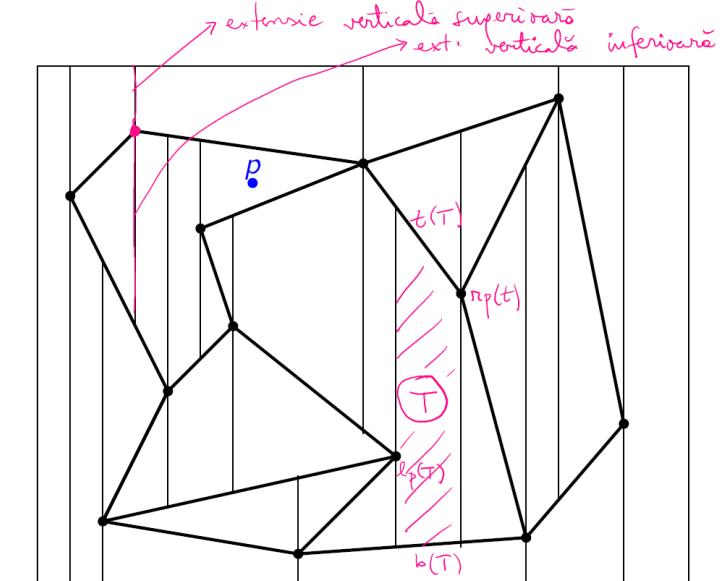


## Exemplu - rafinare eficientă



## Simplificări și ipoteze

- Se consideră o mulțime  $S$  de  $n$  segmente astfel ca oricare două dintre ele (i) fie nu au niciun punct comun; (ii) fie au un vârf comun.
- Simplificare 1:* Se consideră un dreptunghi  $D$  cu laturile paralele cu axele de coordonate care include toată subdiviziunea inițială.
- Simplificare 2:* Se presupune că nu există două vârfuri (extremități ale segmentelor din  $S$ ) distincte care au aceeași coordonată  $x$  (în particular nu există segmente verticale).
- Concluzie:* Se consideră o mulțime de  $n$  segmente  $S$  care verifică ipotezele de mai sus: *mulțime de segmente în poziție generală*. **Harta trapezoidală / descompunere verticală / descompunere cu trapeze (trapezoidal map)**  $\mathcal{T}(S)$  a lui  $S$  este subdiviziunea indușă de  $S$ , dreptunghiul  $D$  și de extensiile verticale inferioare și superioare (concept introdus de Seidel, 1991).

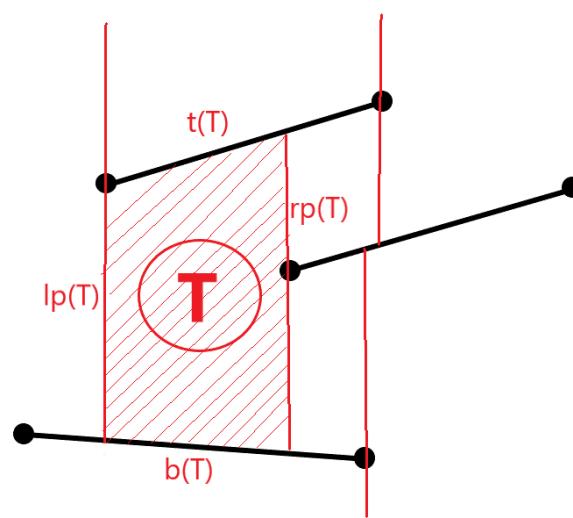


## Hărți trapezoidale – probleme studiate

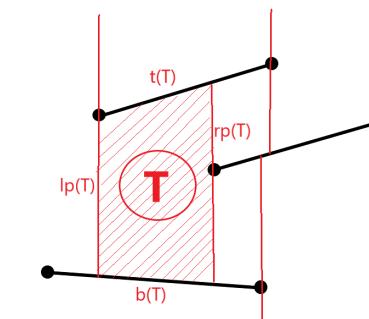
- ▶ Descrierea obiectelor geometrice din care sunt formate – ce informații se rețin?
- ▶ Aspecte legate de complexitate?
- ▶ Structuri de date adecvate?
- ▶ Un algoritm eficient?

## Descrierea obiectelor

Informații geometrice sunt reținute pentru un trapez

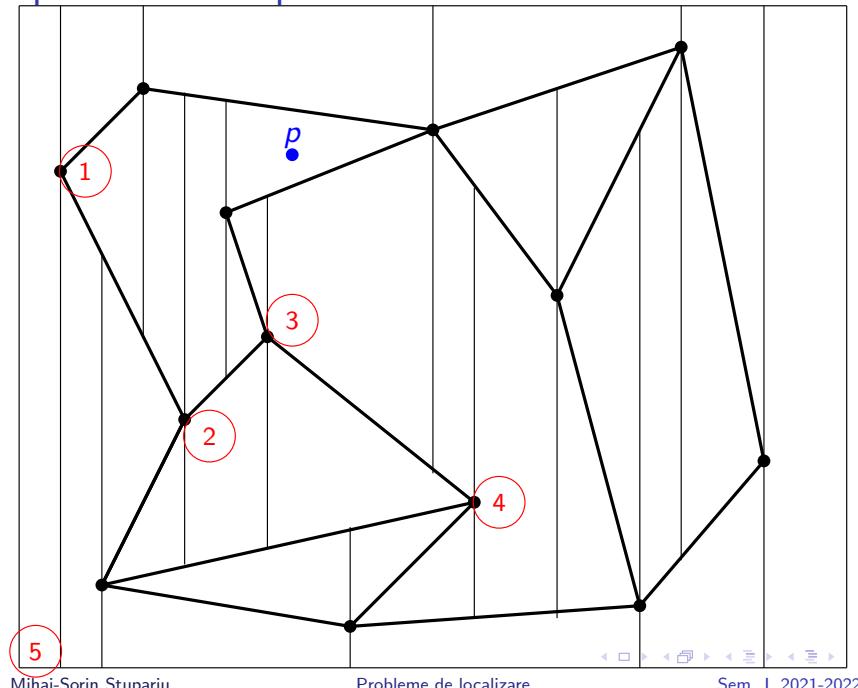


Informații geometrice sunt reținute pentru un trapez



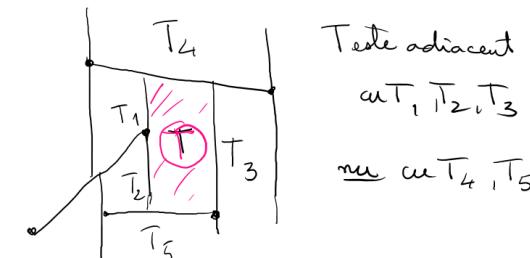
- ▶  $t(T)$ ,  $b(T)$ ,  $lp(T)$ ,  $rp(T)$  determină în mod unic un trapez fixat  $T$ .  $t(T)$ ,  $b(T)$  sunt **segmente**, iar  $lp(T)$ ,  $rp(T)$  sunt **vârfuri** (extremități ale segmentelor)
- ▶ Există cinci cazuri posibile pentru marginea stângă  $lp$  (analog pentru marginea dreaptă  $lp$ ).

## Exemplu - hartă trapezoidală



## Complexitate și alte aspecte cantitative

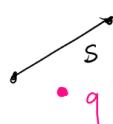
- ▶ **Lema 2.** Fie  $S$  o mulțime de  $n$  segmente în poziție generală. Harta trapezoidală  $\mathcal{T}(S)$  conține cel mult  $6n + 4$  vârfuri și cel mult  $3n + 1$  trapeze.
- ▶ **Lema 3.** Fie  $S$  o mulțime de  $n$  segmente în poziție generală. Fiecare trapez  $T$  este adiacent cu cel mult patru trapeze (cel mult un vecin stânga superior, cel mult un vecin stânga inferior, cel mult un vecin dreapta superior, cel mult un vecin dreapta inferior).



## Căutarea într-o hartă trapezoidală

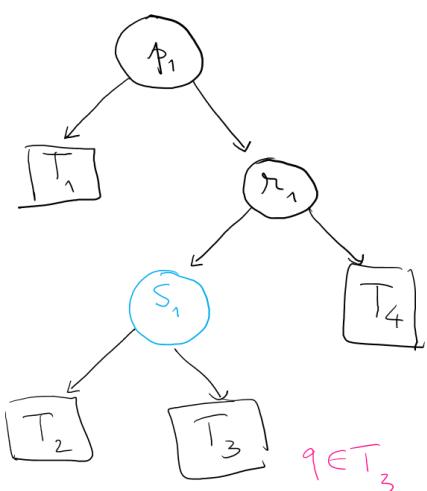
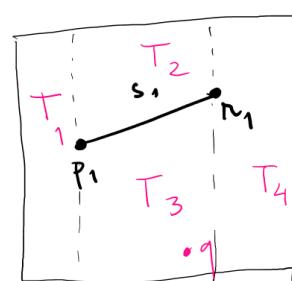
 $x$ -nod ( $p$ )

$q$  este la stg/ dreapta  
dr. verticale care  
trece prin  $p$   
(comparații de abscise)

 $y$ -nod ( $s$ )

$q$  este deasupra/  
de dedesubtul lui  $s$   
(testul de orientare)

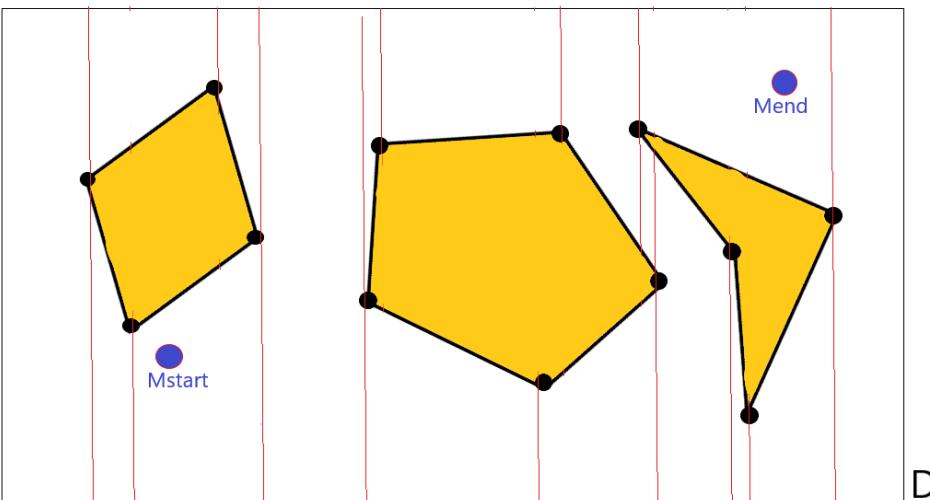
## Exemplul 1 - structură de căutare asociată





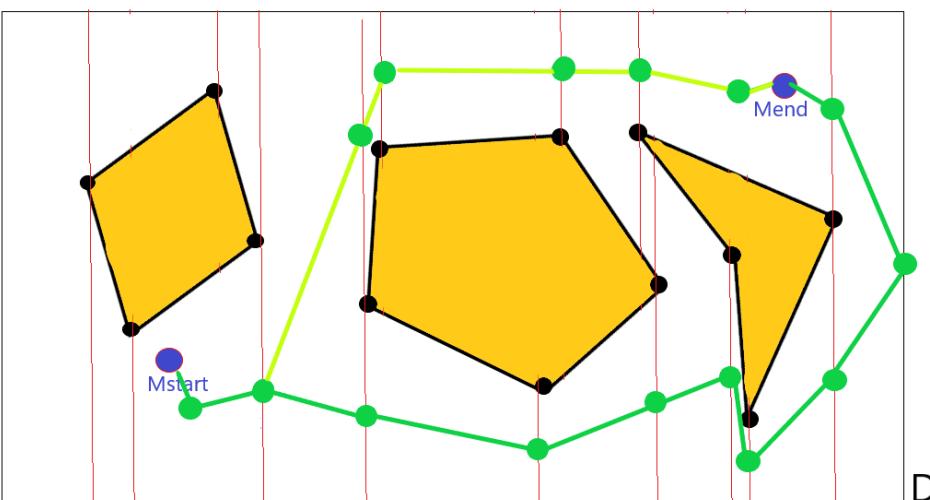


## Illustrare



Date  $M_{\text{start}}$ ,  $M_{\text{end}}$  se caută un drum de la  $M_{\text{start}}$  la  $M_{\text{end}}$  în interiorul spațiului liber.

- ▶ Dacă  $M_{\text{start}}$  și  $M_{\text{end}}$  sunt în interiorul aceluiași trapez: segmentul  $[M_{\text{start}} M_{\text{end}}]$ .
  - ▶ Dacă sunt în trapeze diferite: se folosesc centrele de greutate (mijloacele liniilor mijlocii) ale trapezelor în care sunt situate punctele și mijloacele laturilor adiacente (muchii verticale!) Se utilizează un graf asociat spațiului liber, care poate fi construit în timp liniar din  $\mathcal{T}(\mathcal{C}_I)$ .



## Algoritm DETERMINA SPATIU LIBER (S)

- ▶ **Input.** O mulțime  $\mathcal{P}$  de poligoane disjuncte.
  - ▶ **Output.** O hartă trapezoidală  $\mathcal{C}_I$  a spațiului liber (pentru un robot-punct).
  1. Fie  $S$  mulțimea muchiilor poligoanelor din  $\mathcal{P}$ .
  2. Determină harta trapezoidală  $\mathcal{T}(S)$ , folosind algoritmul HARTA TRAPEZOIDALA.
  3. Elimină trapezele situate în interiorul poligoanelor și returnează subdiviziunea obținuta.

## Algoritm DETERMINADRUM ( $\mathcal{T}(\mathcal{C}_l)$ , $\mathcal{G}_d$ , $M_{\text{start}}$ , $M_{\text{end}}$ )

► **Input.** Harta trapezoidală  $\mathcal{T}(\mathcal{C}_l)$  a spațiului liber, graful drumurilor  $\mathcal{G}_d$ , punctul de start  $M_{\text{start}}$ , punctul final  $M_{\text{end}}$ .

► **Output.** Un drum de la  $M_{\text{start}}$  la  $M_{\text{end}}$ , dacă există. În caz contrar, algoritmul precizează că nu există un drum.

1. caută trapeze în  $\mathcal{T}(\mathcal{C}_l)$  conținând  $M_{\text{start}}$ , respectiv  $M_{\text{end}}$ .
2. **if** există  $\Delta_{\text{start}}$ , respectiv  $\Delta_{\text{end}}$  conținând  $M_{\text{start}}$ , respectiv  $M_{\text{end}}$
3.   **then** fie  $v_{\text{start}}$  și  $v_{\text{end}}$  centrele  $\Delta_{\text{start}}$ ,  $\Delta_{\text{end}}$  (noduri din  $\mathcal{G}_d$ )
4.   caută un drum în  $\mathcal{G}_d$  de la  $v_{\text{start}}$  la  $v_{\text{end}}$  folosind BFS
5.   **if** există drum  $\delta$
6.     **then** indică drumul  $[M_{\text{start}} v_{\text{start}}] \cup \delta \cup [v_{\text{end}} M_{\text{end}}]$
7.     **else** raportează că nu există drum de la  $M_{\text{start}}$  la  $M_{\text{end}}$
8. **else** raportează că nu există drum de la  $M_{\text{start}}$  la  $M_{\text{end}}$

## Rezultatul principal

► **Teoremă.** Fie  $\mathcal{R}$  un robot-punct care se deplasează într-o mulțime  $S$  de obstacole poligonale, având în total  $n$  muchii. Utilizând timp mediu de preprocesare  $O(n \log n)$  pentru mulțimea  $S$ , un drum liber de coliziuni între două puncte fixate poate fi calculat (dacă există!) în timp mediu  $O(n)$ .