Acoperiri convexe

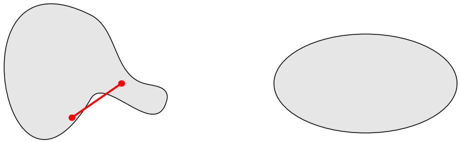
Mihai-Sorin Stupariu

Sem. I, 2019 - 2020

Mulțimi convexe

- **Conceptul de mulțime convexă:**
O mulțime $\mathcal{M} \subset \mathbf{R}^d$ este convexă dacă oricare ar fi $P, Q \in \mathcal{M}$, segmentul $[PQ]$ este inclus în $\mathcal{M}$; pentru $P, Q \in \mathbf{R}^d$, segmentul $[PQ]$ este mulțimea combinațiilor convexe dintre P și Q:

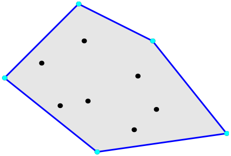
$$[PQ] = \{(1-t)P + tQ | t \in [0,1]\} = \{\alpha P + \beta Q | \alpha, \beta \in [0,1], \alpha + \beta = 1\}.$$



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 $\longrightarrow$ necesară **acoperirea convexă**.

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 **combinație convexă** a punctelor $P_1, P_2, \ldots, P_n$ este un punct P de forma

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

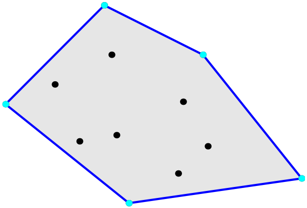
- **Problematizare:** Aceste caracterizări echivalente nu conduc la un algoritm de determinare a acoperirii convexe.

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

- Dacă $\mathcal{P}$ este finită, acoperirea sa convexă, $\mathrm{Conv}(\mathcal{P})$ 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 $O(n)$).

Cazul d=2: abordări posibile

- **În continuare:** $d = 2$.
- **Problemă:** Cum determinăm, algoritmic, vârfurile acoperirii convexe pentru o mulțime finită $\mathcal{P}$ din $\mathbf{R}^2$ (ca mulțime, ca listă)?



- **Două abordări posibile:**
  - Determinarea punctelor extreme.
  - Determinarea muchiilor frontierei acoperirii convexe.

Determinarea punctelor extreme și ordonarea lor

- Un punct M al unei mulțimi convexe $\mathcal{S}$ este **punct extrem** al lui $\mathcal{S}$ dacă **nu** există $A, B \in \mathcal{S}$ cu $A \neq B$ astfel ca $M \in [AB]$.
- **Teoremă (caracterizarea punctelor extreme).** Fie $\mathcal{P}$ o mulțime finită și $\mathrm{Conv}(\mathcal{P})$ acoperirea sa convexă. Un punct $M \in \mathcal{P}$ nu este punct extrem al lui $\mathrm{Conv}(\mathcal{P})$ $\Leftrightarrow$ este situat pe laturile sau în interiorul unui triunghi având vârfurile în $\mathcal{P}$, dar nu este, el însuși, vârf al triunghiului.
- **Teoremă (ordonarea punctelor extreme).** Fie $\mathcal{P}$ o mulțime finită și $\mathrm{Conv}(\mathcal{P})$ acoperirea sa convexă. Ordonând punctele extreme ale lui $\mathrm{Conv}(\mathcal{P})$ după unghiul polar (format într-un sistem de coordonate polare având originea într-un punct interior al lui $\mathrm{Conv}(\mathcal{P})$), se obțin vârfurile consecutive ale lui $\mathrm{Conv}(\mathcal{P})$.

Comentarii

- Cum se stabilește dacă un punct P aparține unui triunghi ABC sau interiorului acestuia?
  - Folosind arii.
  - Verificând dacă P situat pe laturi sau situat de aceeași parte a fiecărei laturi ca și vârful opus (v. "Testul de orientare"), etc.
- Coordonate carteziene (x, y) și coordonate polare (ρ, θ):

$$\begin{cases} x = \rho \cos \theta \\ y = \rho \sin \theta \end{cases} \qquad \begin{cases} \rho = \sqrt{x^2 + y^2} \\ \theta = \operatorname{arctg} \frac{y}{x} \end{cases}$$

- Pentru a **ordona / sorta** punctele nu este nevoie ca unghiurile polare să fie calculate explicit! Are loc relația $\theta(Q) > \theta(P) \Leftrightarrow Q$ este "în stânga" muchiei orientate $\overrightarrow{OP}$ (v. "Testul de orientare").
- $M_1, \ldots, M_q$ puncte extreme ale lui $\mathrm{Conv}(\mathcal{P}) \Rightarrow$ centrul de greutate $\frac{1}{q}M_1 + \ldots + \frac{1}{q}M_q$ este situat în interiorul $\mathrm{Conv}(\mathcal{P})$.

Algoritmul lent 1

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

$\mathcal{M} \leftarrow \emptyset$ /*$\mathcal{M}$ este mulțimea punctelor extreme*/
for $P \in \mathcal{P}$
    do *valid* $\leftarrow$ true
    for $(A, B, C) \in \mathcal{P} \times \mathcal{P} \times \mathcal{P}$ distincte 2×2, distincte de P
        do if P în interiorul ΔABC sau pe laturile sale
            then *valid* $\leftarrow$ false
    if *valid*=true **then** $\mathcal{M} = \mathcal{M} \cup \{P\}$
do calculează centrul de greutate al lui $\mathcal{M}$
do sortează punctele din $\mathcal{M}$ după unghiul polar, obținând lista $\mathcal{L}$

Comentarii

- Complexitatea: $O(n^4)$ (pașii 1-7: $O(n^4)$; pasul 8: $O(n)$; pasul 9: $O(n \log n)$).
- Complexitatea algebrică: necesare polinoame de gradul II
- Tratează corect cazurile degenerate (dacă A, B, C sunt coliniare pe frontieră, cu $C \in [AB]$, doar A și B sunt detectate ca fiind puncte extreme!)

Determinarea muchiilor frontierei

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

Algoritmul lent 2

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera 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* $\leftarrow$ true
4. **for** $R \in \mathcal{P} \setminus \{P, Q\}$
5. **do if** R "în dreapta" lui $\overrightarrow{PQ}$
6. **then** *valid* $\leftarrow$ false
7. **if** *valid*=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***/

Comentarii

- Complexitatea: $O(n^3)$.
- Complexitatea algebrică: necesare polinoame de gradul II
- Tratarea cazurilor degenerate: poate fi adaptat. Pasul 5 trebuie rafinat:

  5. **do if** R "în dreapta" lui $\overrightarrow{PQ}$ **or** $(P, Q, R$ coliniare **and** $r(P, R, Q) < 0)$
  6. **then** *valid* $\leftarrow$ false

- Robustețea: datorită erorilor de rotunjire este posibil ca algoritmul să nu returneze o listă coerentă de muchii.

Graham’s scan [1972]

- Punctele sunt mai întâi **sortate** (lexicografic, după unghiul polar și distanța polară) și renumerotate.
- Algoritm de tip **incremental**, punctele fiind adăugate unul câte unul la lista $\mathcal{L}$ a frontierei acoperirii convexe. Pe parcurs, anumite puncte sunt eliminate - actualizare locală a listei vârfurilor acoperirii convexe.
- **Q:** Cum se decide dacă trei puncte sunt vârfuri consecutive ale acoperirii convexe (parcursă în sens trigonometric)?
- **A:** Se efectuează un ”viraj la stânga” în punctul din mijloc.

Graham's scan (algoritm)

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Determinarea unui punct interior din $\mathrm{Conv}(\mathcal{P})$ /* de ex. baricentrul
2. Efectuarea unei translații, punctul interior de la 1 devine originea O
3. Sortare și ordonare în raport cu unghiul polar și distanța polară, renumerotare $P_1, P_2, \ldots, P_n$ conform ordonării
4. $\mathcal{L} \leftarrow (P_1, P_2)$
5. **for** $i \leftarrow 3$ **to** n
6.   **do** adaugă P_i la sfârșitul lui $\mathcal{L}$
7.   **while** $\mathcal{L}$ are mai mult de două puncte
        and ultimele trei <u>nu</u> determină un viraj la stânga
8.   **do** șterge penultimul punct
9. **return** $\mathcal{L}$

Graham's scan, varianta lui Andrew [1979]

- Punctele sunt mai întâi **sortate** (lexicografic, după coordonatele carteziene) și renumerotate.
- Algoritmul determină două liste, reprezentând marginea **inferioară** și cea **superioară** a frontierei, pentru a le determina sunt folosite la inițializare punctele P_1, P_2, respectiv P_n, P_{n-1}. În final, aceste liste sunt concatenate.
- Principiul: asemănător celui de la Graham's scan: punctele sunt adăugate unul câte unul la listă. Se efectuează testul de orientare pentru ultimele trei puncte și este eliminat penultimul punct, în cazul în care ultimele trei puncte nu generează un viraj la stânga.

Comentarii - Graham's scan

- Algoritm specific pentru context 2D. Nu este on-line, fiind nevoie de toate punctele.
- Complexitatea: $O(n \log n)$; spațiu: $O(n)$; complexitate algebrică: polinoame de gradul II.
- Tratarea cazurilor degenerate: corect.
- Robustețea: datorită erorilor de rotunjire este posibil ca algoritmul să returneze o listă eronată (dar coerentă) de muchii.
- Graham's scan este optim pentru "cazul cel mai nefavorabil".
- **Teoremă.** *Problema sortării este transformabilă în timp liniar în problema acoperirii convexe.*

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 frontiera acoperirii convexe.

Jarvis' march (algoritm)

Input: O mulțime de puncte necoliniare $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ din $\mathbf{R}^2$ ($n \geq 3$).
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera 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* $\leftarrow$ true
3. **while** *valid*$=$ 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** $k \leftarrow k + 1$;
               $A_k = S$
               adaugă A_k la $\mathcal{L}$
10.   **else** *valid* $\leftarrow$ false
11. **return** $\mathcal{L}$

Algoritm pentru determinarea punctelor antipodale

Input: Vârfurile unui poligon convex $\mathcal{P}$ din $\mathbf{R}^2$ $(n \geq 3)$, organizate ca o listă $=(P_0, P_1, \ldots, P_n)$, gestionată cu un pointer $\textsc{next}$.

Output: Afișează perechile de vârfuri antipodale ale lui $\mathcal{P}$.

1. $P \leftarrow P_n$; $Q \leftarrow P_0 (= \textsc{next}[P])$
2. **while** $\mathscr{A}(\Delta(P, \textsc{next}[P], \textsc{next}[Q])) > \mathscr{A}(\Delta(P, \textsc{next}[P], Q))$
3.   **do** $Q \leftarrow \textsc{next}[Q]$
4. $Q_0 \leftarrow Q$
5. **while** $(Q \neq P_0)$
6.   **do** $P \leftarrow \textsc{next}[P]$
7. $\textsc{print}(P, Q)$
8.   **while** $\mathscr{A}(\Delta(P, \textsc{next}[P], \textsc{next}[Q])) > \mathscr{A}(\Delta(P, \textsc{next}[P], Q))$
9.   **do** $Q \leftarrow \textsc{next}[Q]$
10.   **if** $((P, Q) \neq (Q_0, P_0))$ **then** $\textsc{print}(P, Q)$
11.   **if** $\mathscr{A}(\Delta(P, \textsc{next}[P], \textsc{next}[Q])) = \mathscr{A}(\Delta(P, \textsc{next}[P], Q))$ **then**
12.   **if** $((P, Q) \neq (Q_0, P_n))$ **then** $\textsc{print}(P, \textsc{next}[Q])$

Alte direcții de lucru

- **Aplicații:** grafică pe calculator, robotică, GIS, recunoașterea formelor, gestionarea bazelor de date multi-dimensionale, etc.
- 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).