PROBLEME DE GEOMETRIE COMPUTAȚIONALĂ

Acest material își propune să introducă anumite noțiuni de geometrie computațională și să prezinte mai detaliat o clasă largă de probleme de programare: acoperirile convexe.

Noțiuni introductive

Întreaga discuție din acest material se învârte în jurul noțiunii de punct. Prin urmare, vrem să îl definim mathematic.

Definiție: Numim punct un element $P \in \mathbb{R}^d$

În funcție de dimensiunea d în care lucrăm, este necesar să introducem diverse metrici pentru a caracteriza punctele. Pentru simplitate, vom discuta în cadrul acestui material numai cazurile $d = 1$ și $d = 2$.

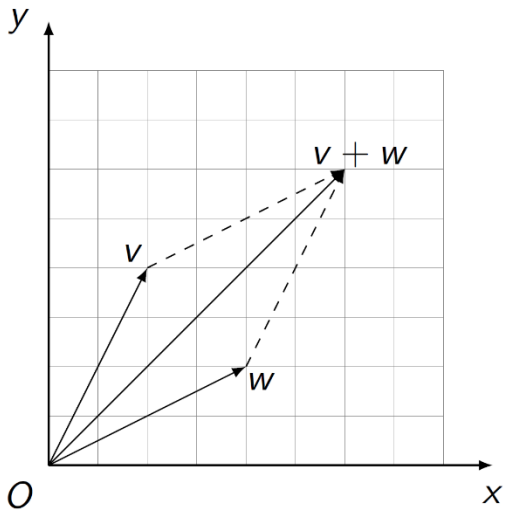
Pentru început, ne interesează să găsim un mod în care putem să ordonăm punctele, relativ la un sistem de coordonate.

Reprezentarea în $\mathbb{R}^d$

În ceea ce privește reprezentarea punctelor, putem realiza acest lucru în două moduri: prin **combinații lineare** de vectori, sau prin **combinații afine**.

Putem descompune un vector într-o combinație liniară a 2 vectori support v și w, de forma

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



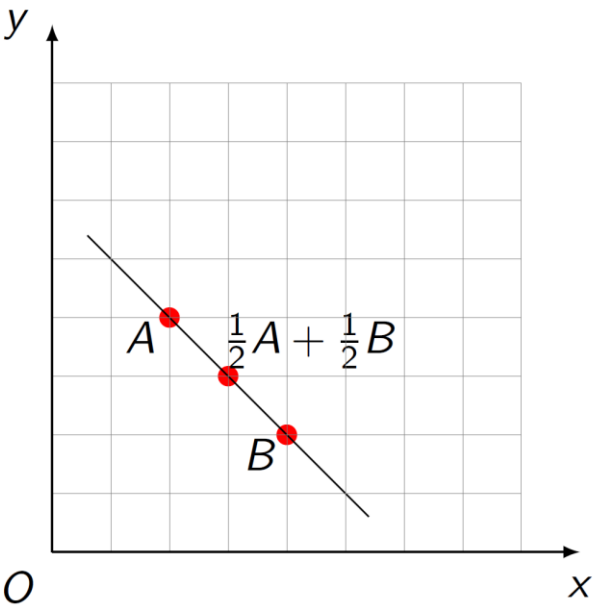
Ilustrarea cazului $\alpha = \beta = 1$

Pe baza acestei observații, s-a introdus notația combinațiilor afine. În loc de a exprima explicit vectorii suport, descompunerea se face în funcție de 2 puncte din plan și de vectorii de poziție ai acestora.

Definiție: Într-un sistem de coordonate de origine O numim vector de poziție al punctului A vectorul $\overline{OA}$.

Din simplitate, s-a renunțat la O în scrierea curentă. Prin urmare, dacă avem un anumit punct, putem găsi 2 puncte A și B ca să îl exprimăm sub forma unei combinații afine de forma

$$\lambda A + \mu B$$



Ilustrarea cazului $\alpha = \beta = 1$

Lemă: Fie A și B două puncte în $\mathbb{R}^d$. Pentru orice punct $P \in AB, P \neq B$, există un unic scalar $r \in \mathbb{R} \setminus \{-1\}$ astfel ca $\overline{AP} = r \cdot \overline{PB}$. Reciproc, fiecărui scalar $r \in \mathbb{R} \setminus \{-1\}$, îi corespunde un unic punct $P \in AB$.

Definiție: Scalarul definit mai sus se numește raportul punctelor A, P, B sau raportul în care punctul P împarte segmental $[AB]$. Acesta se notează cu $r(A, P, B)$.

Teoremă: Fie A, P, B trei puncte coliniare cu $P \neq B$. Atunci

$$P = \frac{1}{r+1}A + \frac{r}{r+1}B, \text{ unde } r = r(A, P, B);$$

Dacă renotăm $r(A,P,B)$ putem obține forme echivalente:

$$P = (1-\alpha)A + \alpha B \qquad \text{dacă } r(A,P,B) = \frac{\alpha}{1-\alpha}$$

$$P = \frac{\alpha}{\alpha+\beta}A + \frac{\beta}{\alpha+\beta}B \qquad \text{dacă } r(A,P,B) = \frac{\beta}{\alpha}$$

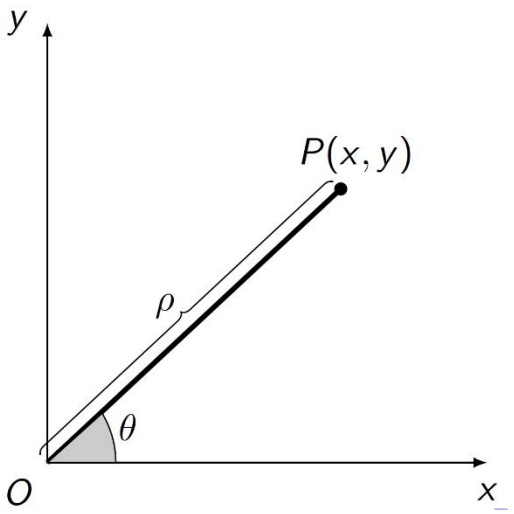
Observații: Fie $P \in AB \setminus \{A,B\}$. Avem:

- $r(A,P,B) > 0 \leftrightarrow P \in (AB)$
- $r(B,P,A) = \frac{1}{r(A,P,B)}$

Definiție: Numim coordonatele polare ale punctului P de coordonate carteziene (x,y) sistemul (ρ,θ) asociat pentru care au loc relațiile:

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

Reprezentarea grafică în funcție de coordonatele polare



Toate aceste noțiuni ne sunt necesare să vedem poziția relativă a două puncte față de un vector/muchie orientată.

Teoremă: Fie $P = (p_1, p_2),\ Q = (q_1, q_2)$ două puncte distincte din $\mathbb{R}^2$ și $R = (r_1, r_2)$ un punct arbitrar. Notăm

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

Avem că:

- $R \in PQ \leftrightarrow \Delta(P, Q, R) = 0$
- R se află "în dreapta" segmentului $\overline{PQ} \leftrightarrow \Delta(P, Q, R) < 0$
- R se află "în stânga" segmentului $\overline{PQ} \leftrightarrow \Delta(P, Q, R) > 0$

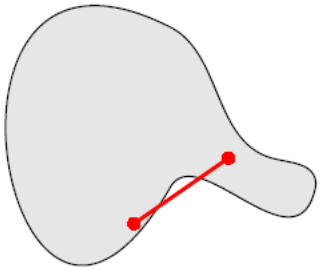
Acoperiri convexe

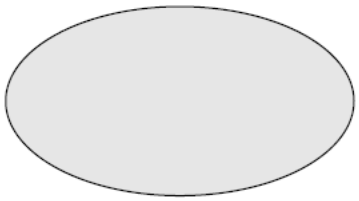
Este necesar să introducem conceptul de mulțime convexă.

Definiție: Pentru $P, Q \in \mathbb{R}^d$ segmentul $[PQ]$ este mulțimea combinațiilor convexe dintre P și Q:

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

Definiție: Spunem că o mulțime M din $\mathbb{R}^d$ este convexă dacă $\forall\ P, Q \in M$, segmentul $[PQ]$ este inclus în M.





Mulțimea din stânga nu este convexă deoarce segmentul evidențiat nu se află în mulțime.

Mulțimea din dreapta este convexă.

În practică, dacă trebuie să lucrăm cu diverse date, acestea fiind sub forma unei mulțimi care nu este convexă, este util ca în locul datelor inițiale să folosim acoperirea convexă a mulțimii date. În acest fel avem parte de toate avantajele convexității, fără a altera datele inițiale.

Definiție: Pentru o mulțime $\mathcal{P}$, notăm cu $Conv(\mathcal{P})$acoperirea sa convexă și o putem defini în mai multe moduri:

- Cea mai mică mulțime convexă care o conține pe $\mathcal{P}$
- Intersecția tuturor mulțimilor convexe care o conțin pe $\mathcal{P}$
- Mulțimea tuturor combinațiilor convexe din $\mathcal{P}$

$$\{\alpha_1 P_1 + \alpha_2 P_2 + \cdots + \alpha_n P_n \mid \alpha_1, \ldots, \alpha_n \in [0,1] \text{ și } \alpha_1 + \cdots + \alpha_n = 1\}$$

Pentru cazul $d = 1$, $Conv(\mathcal{P})$ este un segment, iar acesta poate fi determinat în $O(n)$, parcurgând toate punctele și calculând extremele.

Studiem mai departe cum putem determina algoritmic acoperirea convexă pentru cazul $d = 2$

Algoritmi de calculare a acoperirii convexe

Algoritmul naiv 1

Introducem câteva noțiuni auxiliare.

Definiție: Fie o mulțime $\mathcal{P}$. Spunem că un punct M din $\mathcal{P}$ este **punct extrem** dacă nu există 3 puncte în $\mathcal{P}$ care să determine un triunghi astfel încât M să se afle pe frontiera (cu excepția vârfurilor) sau în interiorul acestuia.

Acest algoritm pleacă de la observația că dacă alegem toate punctele extreme, obținem acoperirea convexă a mulțimii. După determinarea acestora, calculăm centrul de greutate pentru acoperirea convexă și ordonăm punctele obținute în funcție de unghiul polar raportat la centrul de greutate.

Implementare

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

Output: O listă L care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

M ← ∅ // M este mulțimea punctelor extreme

for P ∈ $\mathcal{P}$

do valid ← true

for (A, B, C) ∈ P × P × P distincte 2 × 2, distincte de P

do if P în interiorul ΔABC sau pe laturile sale

then valid ← false

if valid == true **then** M = M ∪ {P}

do calculează centrul de greutate al lui M

do sortează punctele din M după unghiul polar, obținând lista L

Complexitate: $O(n^4 + n + n\log n)$

Algoritmul naiv 2

Acest algoritm se bazează pe o idee oarecum similară. În loc să încercăm să găsim vârfurile din acoperirea convexă, încercăm să găsim segmentele ce alcătuiesc frontiera acesteia. Vom itera prin toate segmentele din mulțime și le reținem doar pe acelea pentru care toate celelalte puncte se găsesc "în stânga".

Implementare

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

Output: O listă L care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

E ← ∅, L ← ∅ //E este lista muchiilor orientate

for (P, Q) ∈ P × P cu P ≠ Q

do valid ← true

for R ∈ P \ {P, Q}

do if R "ˆın dreapta" lui $\overline{PQ}$ or (P, Q, R coliniare and r(P, R, Q) < 0)

then valid ← false

if valid == true **then** E = E ∪ $\{\overline{PQ}\}$

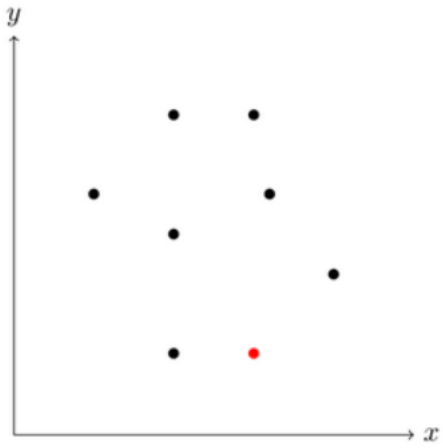
din E se construiește lista L a vârfurilor acoperirii //este necesar ca E să fie corentă

Complexitate: $O(n^3)$

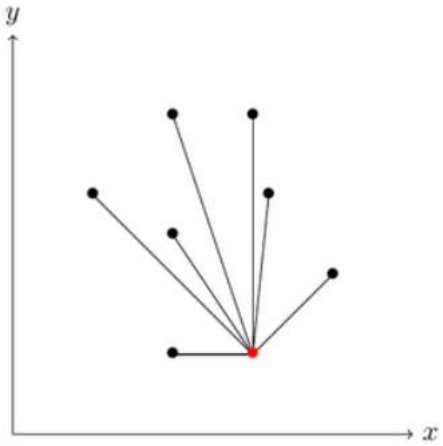
Graham's Scan

Acest algoritm este considerat a fi unul dintre primele din geometria computațională. În urma executării, el ne generează punctele care constituie vârfurile acoperirii convexe, în ordinea în care acestea sunt întâlnite.

Punctul de începere al algoritmului îl constituie alegerea unui punct de care știm sigur că se va afla pe acoperirea convexă. Un astfel de punct îl constituie acela cu cea mai mică coordonată y. Dacă există mai multe astfel de puncte, îl alegem pe acela cu cea mai mare coordonată x.



În continuare, sortăm toate punctele rămase în funcție de unghiul polar, raportat la punctul ales mai sus. Dacă 2 puncte formează același unghi, atunci vom alege mai întâi punctul aflat la o distanță mai mica de cel roșu.



Mai departe, algoritmul iterează peste aceste puncte și verifică dacă ultimele 3 adăugate nu formează un colț concav, raportat la întregul poligon.

Implementare

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

Output: O listă L care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

Determinarea unui punct interior din Conv(P) // un exemplu este detaliat mai sus

Efectuarea unei translații, punctul ales anterior devenind noua origine O

Sortare și ordonare în raport cu unghiul polar și distanța polară, renumerotare $P_1, P_2, \dots, P_n$ conform ordonării

L $\leftarrow (P_1, P_2)$

for i $\leftarrow$ 3 to n

do adaugă P_i la sfârșitul lui L

while L are mai mult de două puncte **and** ultimele trei nu determină un colț concav

do șterge penultimul punct

return L

Complexitate: $O(n \log n)$

O explicație amănunțită a algoritmului Graham Scan, precum și o variantă de implementare puteți studia aici: https://medium.com/@pascal.sommer.ch/a-gentle-introduction-to-the-convex-hull-problem-62dfcabee90c sau aici: https://www.geeksforgeeks.org/convex-hull-set-2-graham-scan/

Mai mulți algoritimi asupra acoperirilor sunt:

- Acoperirea convexă folosind Divide et Impera : https://www.geeksforgeeks.org/convex-hull-using-divide-and-conquer-algorithm/
- Jarvis' Algorithm: https://www.geeksforgeeks.org/convex-hull-set-1-jarviss-algorithm-or-wrapping/
- The Quickhull Algorithm: https://www.geeksforgeeks.org/quickhull-algorithm-convex-hull/

Bibliografie

- https://www.geeksforgeeks.org/
- A gentle introduction to the convex hull problem, Pascal Sommer, 10.12.2016
- Geometrie computațională – suport de curs, Mihai-Sorin Stupariu, UNIBUC, 2019-2020