UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

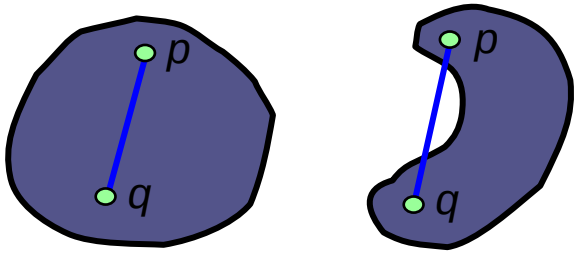
Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Geometrie computationala

3. Acoperiri convexe in plan: Notiuni de baza. Gift Wrapping. Algoritmul lui Graham

Convexitate

- O multime S este *convexa* daca pentru orice pereche de puncte $p,q \in S$ avem segmentul $pq \subseteq S$.



convexa nonconvexa

- Formal, daca S este o multime intr-un spatiu vectorial real sau complex:

$$\forall\, p, q \in S, atunci\; pt + q(1-t) \in S, \forall t \in [0,1]$$

Infasuratori convexe (*Convex Hull*)

Fie o multime $S = \{p_1, p_2, \ldots, p_N\}$. Infasuratoarea convexa *CH(S)* este:

- Cel mai mic poligon convex care contine toate punctele din S
- Intersectia tuturor multimilor convexe ce contin S
- Intersectia tuturor semispatiilor ce contin S
- Reuniunea tuturor triunghiurilor determinate de puncte in S
- Multimea tuturor combinatiile convexe de puncte din S

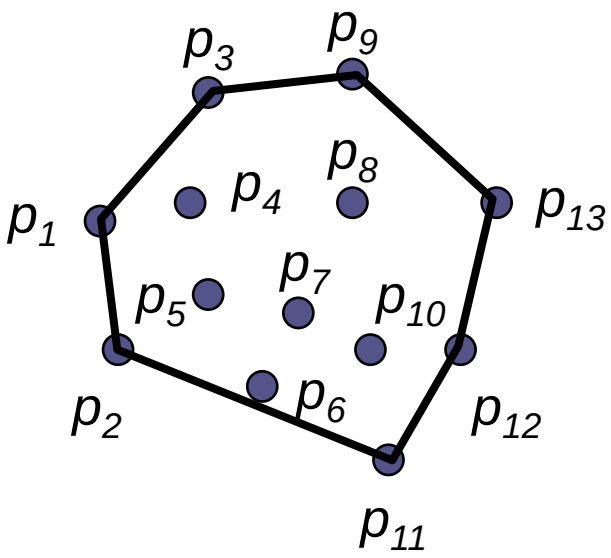
Aplicatii

- Detectia coliziunilor
  - jocuri video: inlocuitor mai bun pentru *bounding-box*
- Aproximarea si compararea formelor
  - *pattern matching*
- Pas de preprocesare pentru multi algoritmi in geometria computationala
- Diametrului unui set de puncte este distanta maxima dintre doua puncte din CH
- Infasuratoarea convexa este cea mai raspandita structura in geometria computationala

Notiuni de baza

Problema: Fiind data o multime de n puncte P in plan, sa se calculeze infasuratoarea sa convexa $CH(P)$.

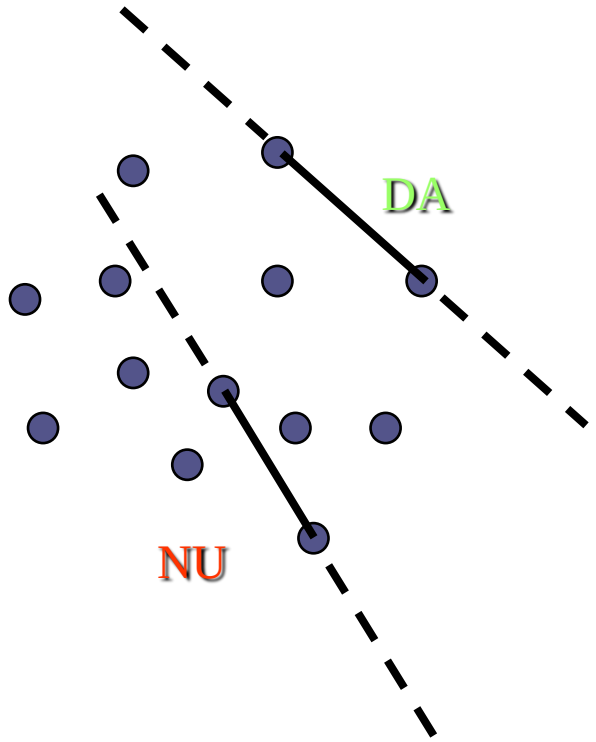
- $CH(P)$ este un poligon convex
- $CH(P)$ este o submultime a lui P
- Complexitate similara cu algoritmii de sortare
- Complexitate *teoretica* pentru poligoane cunoscute: $O(n)$



Intrare: $p_1, \ldots, p_{13}$

Iesire: $p_1, p_2, p_{11}, p_{12}, p_{13}, p_9, p_3$

Algoritmul naiv



Algoritmul naiv

Algoritm

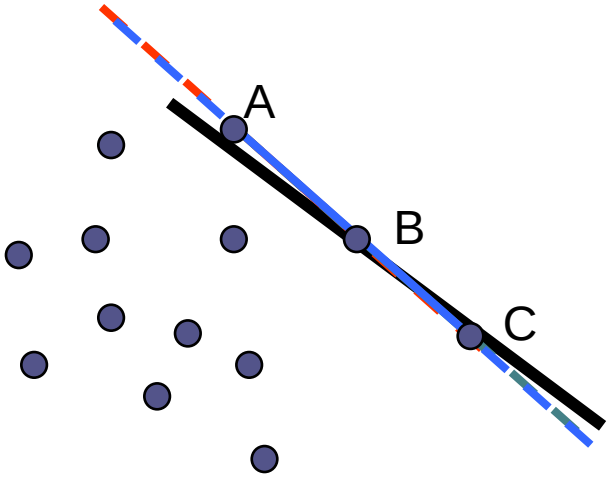
- Pentru fiecare pereche de puncte se construiesc segmentul dintre ele si *dreapta suport*
- Se gasesc toate segmentele ale caror drepte suport impart planul in doua jumatati, astfel incat un semiplan contine *toate* celelalte puncte.
- Se construieste infasuratoarea convexa din aceste segmente.

Complexitate

- Toate perechile: $O(\binom{n}{2}) = O(\frac{n(n-1)}{2}) = O(n^2)$
- Se verifica toate punctele pentru fiecare pereche: $O(n)$ fiecare, $O(n^3)$ in total.

Posibile probleme

- Corectitudinea algoritmului poate fi influentata in cazul in care exista 3 puncte coliniare. Segmentele *AB*, *BC* si AC vor fi *toate* incluse in infasuratoarea convexa.

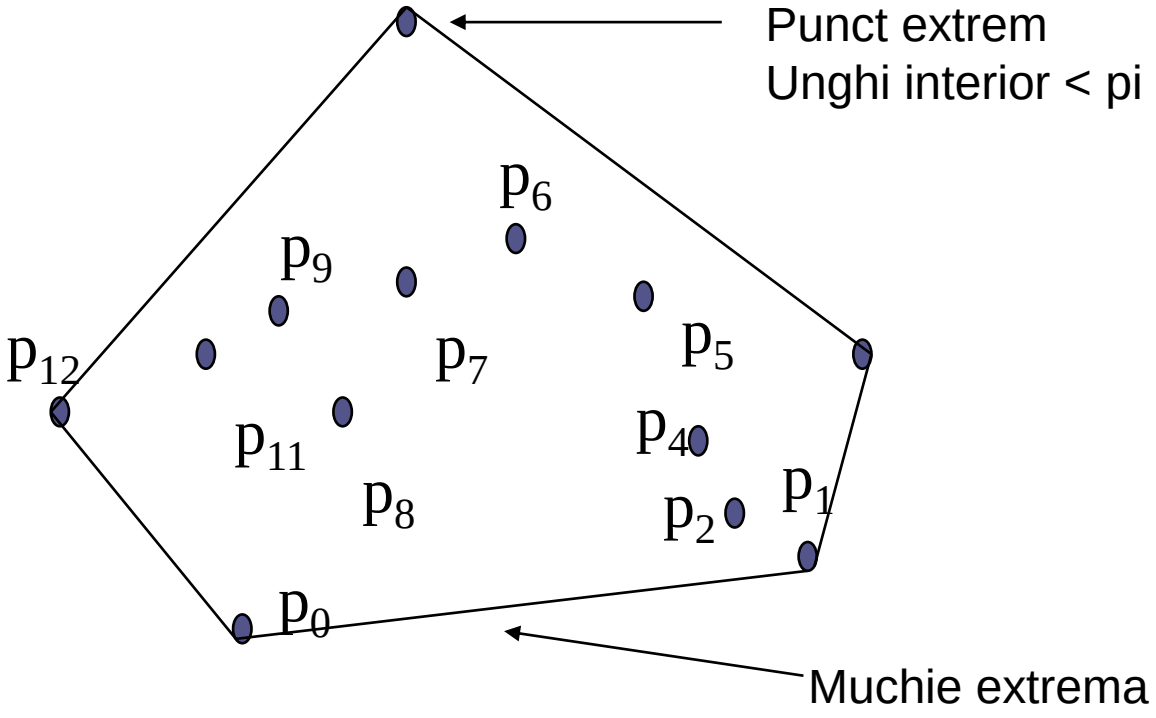


- Probleme numerice – se poate concluziona ca *nici unul* din cele 3 segmente (sau o pereche eronata a lor) apartine infasuratorii convexe.

Presupuneri legate de pozitia punctelor

- Cand se modeleaza un algoritm geometric, mai intai facem unele presupuneri cu scop de simplificare, ex:
  - Nu exista 3 puncte coliniare
  - Nu exista 2 puncte cu aceeasi coordonata x sau y
  - Altele: nu exista 3 puncte pe acelasi cerc, etc.
- Mai tarziu se ia in considerare cazul general:
  - Comportamentul algoritmului la cazuri speciale
  - Va ramane algoritmul corect?
  - Va ramane timpul de rulare neschimbat?
  - Se va modifica/extinde algoritmul pentru a trata aceste situatii.

Extreme



Un punct *nu* este **extrem** pentru o multime S daca este continut intr-un triunghi ale carui varfuri sunt puncte din S, dar nu este unul din varfurile sale.

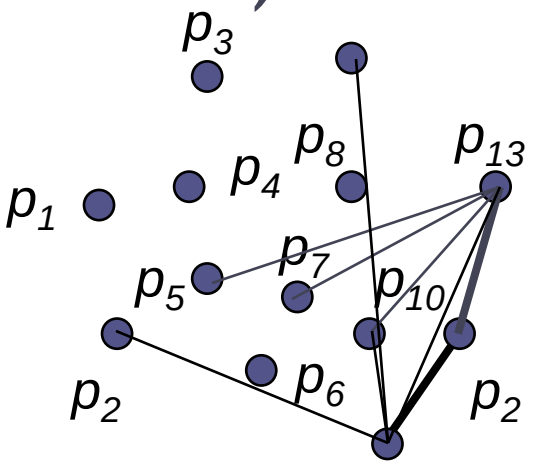
Gift Wrapping (Jarvis' march)

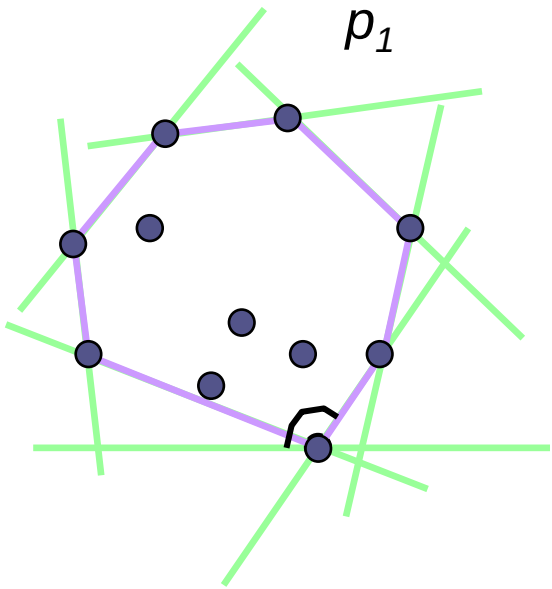
Algoritm:

1. Prima muchie $p_1 p_2$ din CH:
   - p_1 punctul extrem cel mai jos
   - p_2 punctul pentru care $p_1 p_2$ face unghiul cel mai mic cu orizontala
2. Pentru punctele ramase p_i ($i > 2$) :

- Se calculeaza unghiul α_i antiorar fata de muchia precedenta
- Fie p_j punctul cu cel mai mic α_i
- Muchia ($p_i p_j$) devine o noua muchie a infasuratorii CH

Figurativ: Se roteste antiorar o linie prin p_i pana atinge un alt punct.





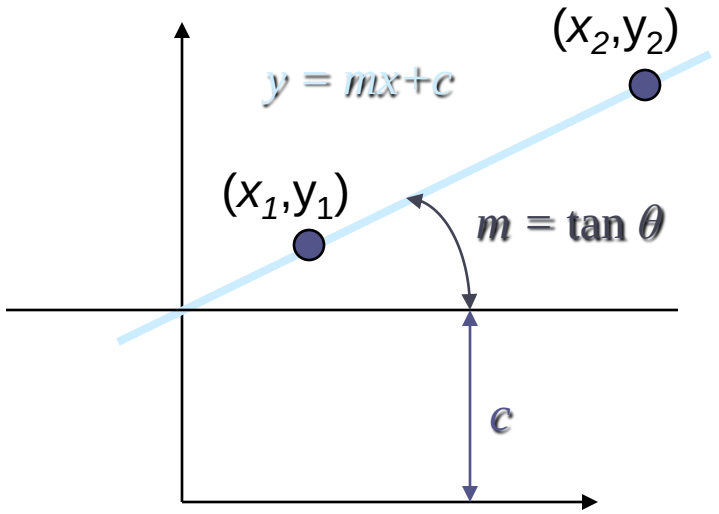
Equatia dreptei

- (x_1,y_1) si (x_2,y_2) doua puncte.
- Ecuatia explicita a dreptei:

$$y = mx + c$$

$$y = \frac{y_2 - y_1}{x_2 - x_1} x + \frac{y_1 x_2 - y_2 x_1}{x_2 - x_1}$$

$$\tan\theta = \frac{y_2 - y_1}{x_2 - x_1}$$



- Caz particular: $x_1 = x_2$ (dreapta verticala)

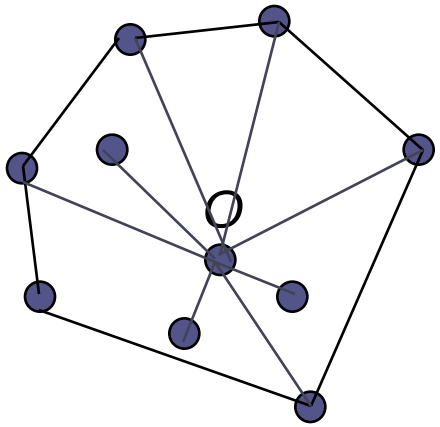
Complexitate

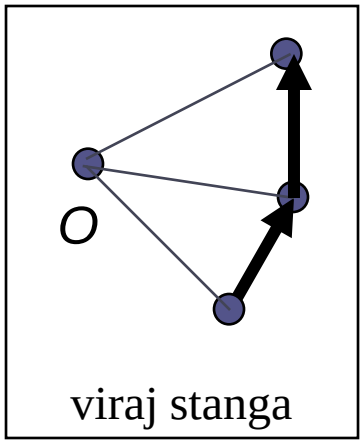
- n puncte, la fiecare pas n comparatii: $O(n^2)$
- De fapt complexitatea este $O(nh)$, $h = |CH(S)|$
- De obicei $h << n$ si atunci este comparabil cu algoritmul lui Graham $O(n \log n)$

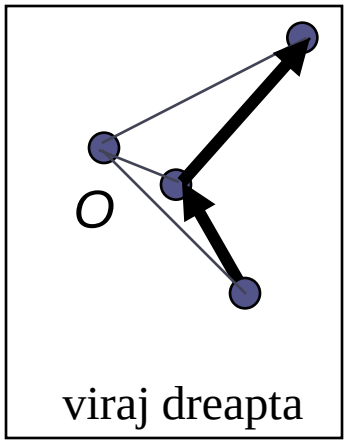
Algoritmul lui Graham

Algoritm:

- Se gaseste un punct *O* in interiorul infasuratorii (ex: *centroidul* – media aritmetica a punctelor pe coordonate)
- Se calculeaza unghiul antiorar α_i de la *O* la celelalte puncte fata de orizontala.
- Se sorteaza punctele dupa unghiul α_i si dupa distanta fata de *O* in caz de egalitate
- Se construieste infasuratoarea prin verificarea tripletelor de puncte in ordinea sortata si prin adaugarea celor ce reprezinta “viraje la stanga” (se elimina “virajele la dreapta”).

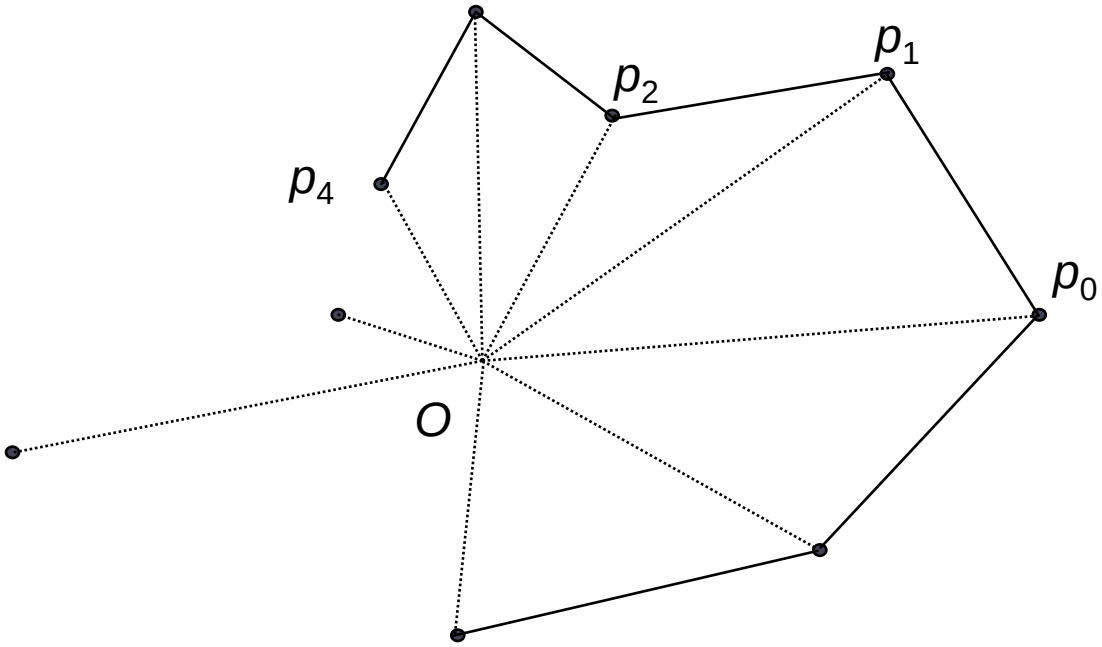






Exemplu

- Daca $p_1p_2p_3$ vireaza dreapta sau sunt coliniare, se elimina p_2 din $CH(S)$ si se continua cu $p_0p_1p_3$
- Daca $p_1p_2p_3$ vireaza stanga, avanseaza la $p_2p_3p_4$



Parcurgerea punctelor din CH(S)

```
1.  begin
2.      v = START
3.      w = PRED[v]  /* w salveaza punctul dinainte de START */
4.      f = FALSE  /* f indica daca scanarea a ajuns din nou la START */
5.      while  (NEXT[v] ≠ START or f = FALSE)
6.          if  (NEXT[v] = w) then
7.              f = TRUE
8.          endif
9.          if  (Left(v, NEXT[v], NEXT[NEXT[v]])) then
10.             v = NEXT[v]  /* avanseaza */
11.         else
12.             delete NEXT[v]  /* eliminare, operatie cu liste in O(1) */
13.             v = PRED[v]  /* backtrack */
14.         endif
15.     endwhile
16. end
```

Graham: analiza complexitatii

- Se foloseste o stiva pentru procesarea punctelor sortate.
- Complexitate: $O(n \log n)$ determinata de pasul de sortare
- D_i = numarul de puncte scoase din stiva la procesarea p_i,

$$\text{timp} = \sum_{i=1}^{n} (D_i + 1) = n + \sum_{i=1}^{n} D_i$$

- Fiecare punct este adaugat la stiva o singura data.
- Odata ce un punct este scos din stiva nu poate fi adugat inca o data.
- Asadar

$$\sum_{i=1}^{n} D_i \leq n$$