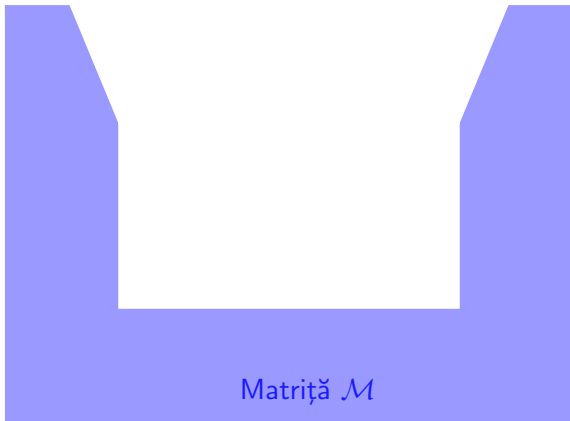


Elemente de programare liniară

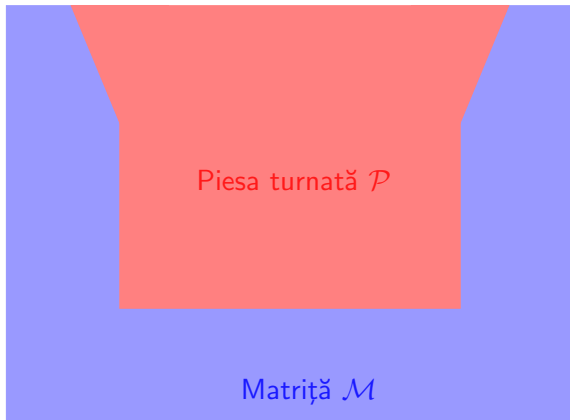
Mihai-Sorin Stupariu

Sem. I, 2018-2019

Turnarea pieselor în matrițe



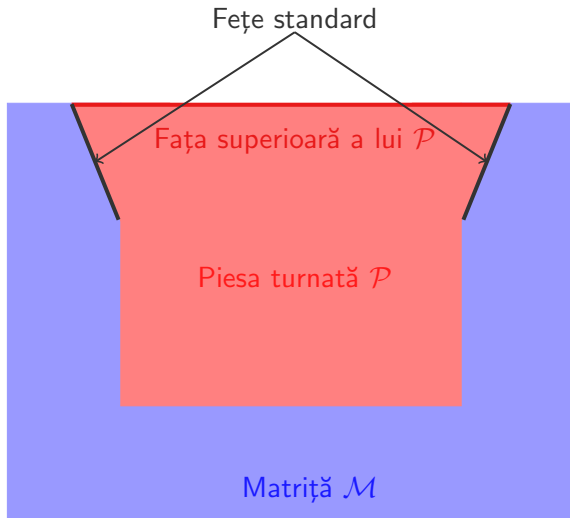
Turnarea pieselor în matrițe



Turnarea pieselor în matrițe



Turnarea pieselor în matrițe



Problematizare

- ▶ Turnarea pieselor în matrițe și extragerea lor fără distrugerea matriței.

Problematizare

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

Problematizare

- ▶ Turnarea pieselor în matrițe și extragerea lor fără distrugerea matriței.
- ▶ Neajunsuri: unele obiecte pot rămâne blocate în matrițe; există obiecte pentru care nu există o matriță adecvată; extragerea obiectului depinde de poziția matriței.
- ▶ **Problema studiată.** Dat un obiect, există o matriță din care să poată fi extras?

Problematizare

- ▶ Turnarea pieselor în matrițe și extragerea lor fără distrugerea matriței.
- ▶ Neajunsuri: unele obiecte pot rămâne blocate în matrițe; există obiecte pentru care nu există o matriță adecvată; extragerea obiectului depinde de poziția matriței.
- ▶ **Problema studiată.** Dat un obiect, există o matriță din care să poată fi extras?
- ▶ **Convenții.**

Problematizare

- ▶ Turnarea pieselor în matrițe și extragerea lor fără distrugerea matriței.
- ▶ Neajunsuri: unele obiecte pot rămâne blocate în matrițe; există obiecte pentru care nu există o matriță adecvată; extragerea obiectului depinde de poziția matriței.
- ▶ **Problema studiată.** Dat un obiect, există o matriță din care să poată fi extras?
- ▶ **Convenții.**
 - ▶ Obiectele: **poliedrale**.

Problematizare

- ▶ Turnarea pieselor în matrițe și extragerea lor fără distrugerea matriței.
- ▶ Neajunsuri: unele obiecte pot rămâne blocate în matrițe; există obiecte pentru care nu există o matriță adecvată; extragerea obiectului depinde de poziția matriței.
- ▶ **Problema studiată.** Dat un obiect, există o matriță din care să poată fi extras?
- ▶ **Convenții.**
 - ▶ Obiectele: **poliedrale**.
 - ▶ Matrițele: formate dintr-o singură piesă; fiecărui obiect \mathcal{P} îi este asociată o matriță $\mathcal{M}_{\mathcal{P}}$

Problematizare

- ▶ Turnarea pieselor în matrițe și extragerea lor fără distrugerea matriței.
- ▶ Neajunsuri: unele obiecte pot rămâne blocate în matrițe; există obiecte pentru care nu există o matriță adecvată; extragerea obiectului depinde de poziția matriței.
- ▶ **Problema studiată.** Dat un obiect, există o matriță din care să poată fi extras?
- ▶ **Convenții.**
 - ▶ Obiectele: **poliedrale**.
 - ▶ Matrițele: formate dintr-o singură piesă; fiecărui obiect \mathcal{P} îi este asociată o matriță $\mathcal{M}_{\mathcal{P}}$
 - ▶ Obiectul: extras printr-o singură translație (sau o succesiune de translații)

Terminologie și convenții

- ▶ **Alegerea orientării:** diverse orientări ale obiectului pot genera diverse matrițe.

Terminologie și convenții

- ▶ **Alegerea orientării:** diverse orientări ale obiectului pot genera diverse matrițe.
- ▶ **Fața superioară:** prin convenție, obiectele au (cel puțin) o față superioară (este orizontală, este singura care nu este adiacentă cu matrița). Celelalte fețe: **standard**; orice față standard f a obiectului corespunde unei fețe standard \hat{f} a matriței.

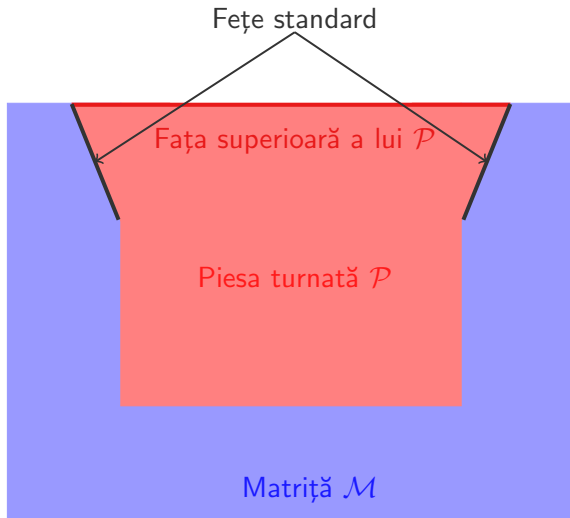
Terminologie și convenții

- ▶ **Alegerea orientării:** diverse orientări ale obiectului pot genera diverse matrițe.
- ▶ **Fața superioară:** prin convenție, obiectele au (cel puțin) o față superioară (este orizontală, este singura care nu este adiacentă cu matrița). 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*.

Terminologie și convenții

- ▶ **Alegerea orientării:** diverse orientări ale obiectului pot genera diverse matrițe.
- ▶ **Fața superioară:** prin convenție, obiectele au (cel puțin) o față superioară (este orizontală, este singura care nu este adiacentă cu matrița). 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:** Matrița este paralelipipedică și are o cavitate corespunzătoare obiectului; fața superioară a obiectului (și a matriței) este perpendiculară cu planul Oxy .

Turnarea pieselor în matrițe



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 exterioară $\vec{\nu}(f)$ la față f și \vec{d} este mai mic de 90° împiedică translația în direcția \vec{d}

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 exterioară $\vec{\nu}(f)$ la față f și \vec{d} este mai mic de 90° împiedică translația în direcția \vec{d}
- ▶ **Propoziție.** *Un poliedru \mathcal{P} poate fi extras din matrița sa $\mathcal{M}_{\mathcal{P}}$ prin translație în direcția \vec{d} dacă și numai dacă \vec{d} face un unghi de cel puțin 90° cu normala exterioară a fiecărei fețe standard a lui \mathcal{P} .*
- ▶ **Reformulare.** Dat \mathcal{P} , trebuie găsită o direcție \vec{d} astfel încât, pentru fiecare față standard f , unghiul dintre \vec{d} și $\vec{\nu}(f)$ să fie cel puțin 90° .

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 exterioară $\vec{\nu}(f)$ la față f și \vec{d} este mai mic de 90° împiedică translația în direcția \vec{d}
- ▶ **Propoziție.** *Un poliedru \mathcal{P} poate fi extras din matrița sa $\mathcal{M}_{\mathcal{P}}$ prin translație în direcția \vec{d} dacă și numai dacă \vec{d} face un unghi de cel puțin 90° cu normala exterioară a fiecărei fețe standard a lui \mathcal{P} .*
- ▶ **Reformulare.** Dat \mathcal{P} , trebuie găsită o direcție \vec{d} astfel încât, pentru fiecare față standard f , unghiul dintre \vec{d} și $\vec{\nu}(f)$ să fie cel puțin 90° .
- ▶ **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 $(*_f)$, care corespunde unui semiplan.

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 exterioară $\vec{\nu}(f)$ la față f și \vec{d} este mai mic de 90° împiedică translația în direcția \vec{d}
- ▶ **Propoziție.** *Un poliedru \mathcal{P} poate fi extras din matrița sa $\mathcal{M}_{\mathcal{P}}$ prin translație în direcția \vec{d} dacă și numai dacă \vec{d} face un unghi de cel puțin 90° cu normala exterioară a fiecărei fețe standard a lui \mathcal{P} .*
- ▶ **Reformulare.** Dat \mathcal{P} , trebuie găsită o direcție \vec{d} astfel încât, pentru fiecare față standard f , unghiul dintre \vec{d} și $\vec{\nu}(f)$ să fie cel puțin 90° .
- ▶ **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 $(*_f)$, care corespunde unui semiplan.
- ▶ **Analitic - toate fețele:** Fie \mathcal{P} un poliedru; fața 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.

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 exterioară $\vec{\nu}(f)$ la față f și \vec{d} este mai mic de 90° împiedică translația în direcția \vec{d}
- ▶ **Propoziție.** *Un poliedru \mathcal{P} poate fi extras din matrița sa $\mathcal{M}_{\mathcal{P}}$ prin translație în direcția \vec{d} dacă și numai dacă \vec{d} face un unghi de cel puțin 90° cu normala exterioară a fiecărei fețe standard a lui \mathcal{P} .*
- ▶ **Reformulare.** Dat \mathcal{P} , trebuie găsită o direcție \vec{d} astfel încât, pentru fiecare față standard f , unghiul dintre \vec{d} și $\vec{\nu}(f)$ să fie cel puțin 90° .
- ▶ **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 $(*_f)$, care corespunde unui semiplan.
- ▶ **Analitic - toate fețele:** Fie \mathcal{P} un poliedru; fața 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

1. Intersecția semiplanelor

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

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

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

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

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

Intersecții de semiplane - probleme studiate, rezultate

► Probleme studiate:

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.

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.

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: (descrie în detaliu ulterior)

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: (descrie 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.*

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 matrice și o direcție admisibilă în care poate fi extras \mathcal{P} este determinată cu aceeași complexitate-timp.*

(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

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

(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

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

- 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,...)

Algoritm INTERSECTIISEMIPLANE (\mathcal{H})

- **Input.** O mulțime \mathcal{H} de semiplane din planul \mathbb{R}^2

Algoritm INTERSECTIISEMIPLANE (\mathcal{H})

- ▶ **Input.** O mulțime \mathcal{H} de semiplane din planul \mathbb{R}^2
- ▶ **Output.** Regiunea poligonală convexă $\mathcal{C} = \cap_{H \in \mathcal{H}} H$

Algoritm INTERSECTIISEMIPLANE (\mathcal{H})

- ▶ **Input.** O mulțime \mathcal{H} de semiplane din planul \mathbb{R}^2
- ▶ **Output.** Regiunea poligonală convexă $\mathcal{C} = \cap_{H \in \mathcal{H}} H$

1. 1. **if** $\text{card}(\mathcal{H}) = 1$

Algorithm INTERSECTIISEMIPLANE (\mathcal{H})

- ▶ **Input.** O mulțime \mathcal{H} de semiplane din planul \mathbb{R}^2
 - ▶ **Output.** Regiunea poligonală convexă $\mathcal{C} = \cap_{H \in \mathcal{H}} H$
1. **if** $\text{card}(\mathcal{H}) = 1$
 2. **then** $\mathcal{C} \leftarrow H \in \mathcal{H}$

Algoritm INTERSECTIISEMIPLANE (\mathcal{H})

- ▶ **Input.** O mulțime \mathcal{H} de semiplane din planul \mathbb{R}^2
 - ▶ **Output.** Regiunea poligonală convexă $\mathcal{C} = \cap_{H \in \mathcal{H}} H$
1. **if** $\text{card}(\mathcal{H}) = 1$
 2. **then** $\mathcal{C} \leftarrow H \in \mathcal{H}$
 3. **else** descompune \mathcal{H} în două mulțimi $\mathcal{H}_1, \mathcal{H}_2$ având fiecare $\lfloor n/2 \rfloor$ elemente

Algoritm INTERSECTIISEMIPLANE (\mathcal{H})

- ▶ **Input.** O mulțime \mathcal{H} de semiplane din planul \mathbb{R}^2
- ▶ **Output.** Regiunea poligonală convexă $\mathcal{C} = \cap_{H \in \mathcal{H}} H$

1. **if** $\text{card}(\mathcal{H}) = 1$
2. **then** $\mathcal{C} \leftarrow H \in \mathcal{H}$
3. **else** descompune \mathcal{H} în două mulțimi $\mathcal{H}_1, \mathcal{H}_2$ având fiecare $\lfloor n/2 \rfloor$ elemente
4. $\mathcal{C}_1 \leftarrow \text{INTERSECTIISEMIPLANE}(\mathcal{H}_1)$

Algoritm INTERSECTIISEMIPLANE (\mathcal{H})

- **Input.** O mulțime \mathcal{H} de semiplane din planul \mathbb{R}^2
- **Output.** Regiunea poligonală convexă $\mathcal{C} = \cap_{H \in \mathcal{H}} H$

1. **if** $\text{card}(\mathcal{H}) = 1$
2. **then** $\mathcal{C} \leftarrow H \in \mathcal{H}$
3. **else** descompune \mathcal{H} în două mulțimi $\mathcal{H}_1, \mathcal{H}_2$ având fiecare $\lfloor n/2 \rfloor$ elemente
4. $\mathcal{C}_1 \leftarrow \text{INTERSECTIISEMIPLANE}(\mathcal{H}_1)$
5. $\mathcal{C}_2 \leftarrow \text{INTERSECTIISEMIPLANE}(\mathcal{H}_2)$

Algoritm INTERSECTIISEMIPLANE (\mathcal{H})

- ▶ **Input.** O mulțime \mathcal{H} de semiplane din planul \mathbb{R}^2
 - ▶ **Output.** Regiunea poligonală convexă $\mathcal{C} = \cap_{H \in \mathcal{H}} H$
1. **if** $\text{card}(\mathcal{H}) = 1$
 2. **then** $\mathcal{C} \leftarrow H \in \mathcal{H}$
 3. **else** descompune \mathcal{H} în două mulțimi $\mathcal{H}_1, \mathcal{H}_2$ având fiecare $\lceil n/2 \rceil$ elemente
 4. $\mathcal{C}_1 \leftarrow \text{INTERSECTIISEMIPLANE}(\mathcal{H}_1)$
 5. $\mathcal{C}_2 \leftarrow \text{INTERSECTIISEMIPLANE}(\mathcal{H}_2)$
 6. $\mathcal{C} \leftarrow \text{INTERSECTEAZAREGIUNICONVEXE}(\mathcal{C}_1, \mathcal{C}_2)$

Rezultate principale

- ▶ În algoritm, \mathcal{C}_1 și \mathcal{C}_2 sunt regiuni poligonale convexe cu cel mult n vârfuri; numărul lor de muchii este cel mult n , întrucât fiecare are cel mult $\left\lfloor \frac{n}{2} \right\rfloor + 1$ muchii, iar $\mathcal{C}_1 \cap \mathcal{C}_2$ are cel mult n fețe, deci $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_1 \cap \mathcal{C}_2$ au complexitate $O(n)$.

Rezultate principale

- ▶ În algoritm, \mathcal{C}_1 și \mathcal{C}_2 sunt regiuni poligonale convexe cu cel mult n vârfuri; numărul lor de muchii este cel mult n , întrucât fiecare are cel mult $\left\lceil \frac{n}{2} \right\rceil + 1$ muchii, iar $\mathcal{C}_1 \cap \mathcal{C}_2$ are cel mult n fețe, deci $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_1 \cap \mathcal{C}_2$ au complexitate $O(n)$.
- ▶ **Propoziție.** *Adaptând algoritmii de overlay, intersecția dintre două regiuni convexe (INTERSECTEAZAREGIUNICONVEXE) poate fi calculată cu complexitate-timp $O(n \log n)$.*

Rezultate principale

- ▶ În algoritm, \mathcal{C}_1 și \mathcal{C}_2 sunt regiuni poligonale convexe cu cel mult n vârfuri; numărul lor de muchii este cel mult n , întrucât fiecare are cel mult $\left\lceil \frac{n}{2} \right\rceil + 1$ muchii, iar $\mathcal{C}_1 \cap \mathcal{C}_2$ are cel mult n fețe, deci $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_1 \cap \mathcal{C}_2$ au complexitate $O(n)$.
- ▶ **Propoziție.** *Adaptând algoritmii de overlay, intersecția dintre două regiuni convexe (INTERSECTEAZAREGIUNICONVEXE) poate fi calculată cu complexitate-timp $O(n \log n)$.*
- ▶ Notând cu $T(n)$ complexitatea-timp pentru a determina intersecția dintre n semiplane, relația de recurență este

$$T(n) = \begin{cases} O(1), & n = 1 \\ O(n \log n) + 2T(\frac{n}{2}), & n > 1. \end{cases}$$

Rezultate principale

- ▶ În algoritmul, \mathcal{C}_1 și \mathcal{C}_2 sunt regiuni poligonale convexe cu cel mult n vârfuri; numărul lor de muchii este cel mult n , întrucât fiecare are cel mult $\lfloor \frac{n}{2} \rfloor + 1$ muchii, iar $\mathcal{C}_1 \cap \mathcal{C}_2$ are cel mult n fețe, deci $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_1 \cap \mathcal{C}_2$ au complexitate $O(n)$.
- ▶ **Propoziție.** Adaptând algoritmii de overlay, intersecția dintre două regiuni convexe (`INTERSECTEAZAREGIUNICONVEXE`) poate fi calculată cu complexitate-timp $O(n \log n)$.
- ▶ Notând cu $T(n)$ complexitatea-timp pentru a determina intersecția dintre n semiplane, relația de recurență este

$$T(n) = \begin{cases} O(1), & n = 1 \\ O(n \log n) + 2T(\frac{n}{2}), & n > 1. \end{cases}$$

- ▶ **Teoremă.** Algoritmul `INTERSECȚIISEMIPLANE` are complexitate $O(n \log^2 n)$.

Rezultate principale

- ▶ În algoritmul \mathcal{C}_1 și \mathcal{C}_2 sunt regiuni poligonale convexe cu cel mult n vârfuri; numărul lor de muchii este cel mult n , întrucât fiecare are cel mult $\lfloor \frac{n}{2} \rfloor + 1$ muchii, iar $\mathcal{C}_1 \cap \mathcal{C}_2$ are cel mult n fețe, deci $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_1 \cap \mathcal{C}_2$ au complexitate $O(n)$.
- ▶ **Propoziție.** Adaptând algoritmiile de overlay, intersecția dintre două regiuni convexe (`INTERSECTEAZAREGIUNICONVEXE`) poate fi calculată cu complexitate-timp $O(n \log n)$.
- ▶ Notând cu $T(n)$ complexitatea-timp pentru a determina intersecția dintre n semiplane, relația de recurență este

$$T(n) = \begin{cases} O(1), & n = 1 \\ O(n \log n) + 2T(\frac{n}{2}), & n > 1. \end{cases}$$

- ▶ **Teoremă.** Algoritmul `INTERSECTIISEMIPLANE` are complexitate $O(n \log^2 n)$.
- ▶ **Teoremă.** Algoritmul `INTERSECTEAZAREGIUNICONVEXE` poate fi îmbunătățit, astfel încât complexitatea-timp să fie liniară.
- ▶ **Teoremă.** Intersecția unei mulțimi de n semiplane poate fi determinată cu complexitate-timp $O(n \log n)$ și folosind $O(n)$ memorie.

Dualitate — motivație euristică

- ▶ De câte informații (numerice) este nevoie pentru a indica **un punct în plan**?

Dualitate — motivație euristică

- ▶ De câte informații (numerice) este nevoie pentru a indica **un punct în plan**?
- ▶ 2

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**?

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

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?

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**

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 — “dicționar” concepte și configurații

Plan primal	Plan dual
Punct p	Dreaptă neverticală p^*
Dreaptă neverticală d	Punct d^*
Dreaptă determinată de două puncte	Punct de intersecție a două drepte
Punctul p deasupra dreptei d	Punctul d^* deasupra dreptei p^*
Segment	Fascicul de drepte (<i>wedge</i>)

(ii) Abordarea calitativă. Motivație

- Sunt realizate 3 produse (notate 1, 2 și 3) pe 2 aparate (notate X și Y).

(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

(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 mentenanță de 5%, respectiv 7% din timpul de lucru. Spațiul total de depozitare este de 50m^2 .

(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 mentenanță de 5%, respectiv 7% din timpul de lucru. Spațiul total de depozitare este de 50m^2 .
- ▶ Modelul matematic:

Constrângeri:

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

Cerința:

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

Problematizare, terminologie

► **Formulare generală (în spațiul d -dimensional):**

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

date constrângerile liniare (inegalități)

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

Problematizare, terminologie

► Formulare generală (în spațiul d -dimensional):

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

date constrângerile liniare (inegalități)

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

► Denumiri:

Problematizare, terminologie

► Formulare generală (în spațiul d -dimensional):

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

date constrângerile liniare (inegalități)

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

► Denumiri:

- date de intrare: $(a_{ij})_{i=\overline{1,n}, j=\overline{1,d}}, (b_i)_{i=\overline{1,n}}, (c_j)_{j=\overline{1,d}}$

Problematizare, terminologie

► Formulare generală (în spațiul d -dimensional):

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

date constrângerile liniare (inegalități)

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

► Denumiri:

- date de intrare: $(a_{ij})_{i=\overline{1,n}, j=\overline{1,d}}, (b_i)_{i=\overline{1,n}}, (c_j)_{j=\overline{1,d}}$
- funcție obiectiv: $(c_1x_1 + c_2x_2 + \dots + c_dx_d)$

Problematizare, terminologie

► Formulare generală (în spațiul d -dimensional):

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

date constrângerile liniare (inegalități)

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

► Denumiri:

- date de intrare: $(a_{ij})_{i=\overline{1,n}, j=\overline{1,d}}, (b_i)_{i=\overline{1,n}}, (c_j)_{j=\overline{1,d}}$
- funcție obiectiv: $(c_1x_1 + c_2x_2 + \dots + c_dx_d)$
- constrângeri: inegalitățile (1)

Problematizare, terminologie

► Formulare generală (în spațiul d -dimensional):

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

date constrângerile liniare (inegalități)

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

► Denumiri:

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

Problematizare, terminologie

► Formulare generală (în spațiul d -dimensional):

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

date constrângerile liniare (inegalități)

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

► Denumiri:

- date de intrare: $(a_{ij})_{i=\overline{1,n}, j=\overline{1,d}}, (b_i)_{i=\overline{1,n}}, (c_j)_{j=\overline{1,d}}$
 - funcție obiectiv: $(c_1x_1 + c_2x_2 + \dots + c_dx_d)$
 - constrângeri: inegalitățile (1)
 - regiune realizabilă (fezabilă): intersecția semispațiilor care definesc constrângerile problemei
- **Exemple:** probleme de programare liniară 1-dimensională, 2-dimensională.

Rezultate

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

Rezultate

- ▶ **Lemă.** (Pentru $d = 1$) *Un program liniar 1-dimensional poate fi rezolvat în timp liniar.*
- ▶ **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 $\vec{c} = (c_1, c_2, \dots, c_d)$.

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

- ▶ **Convenții și terminologie:**

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

► Convenții și terminologie:

- Coordonatele: x și y

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 $\vec{c} = (c_x, c_y)$.

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 $C = h_1 \cap h_2 \cap \dots \cap h_n$.

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.

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

- **Input.** Un program liniar ($H \cup \{m_1, m_2\}, \vec{c}$) din \mathbb{R}^2

Algorithm 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 $f_{\vec{c}}(p)$.

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

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

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

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** $v_{i-1} \in h_i$

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** $v_i \leftarrow v_{i-1}$

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

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

Algorithm 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**

Algorithm 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

Algoritm aleatoriu

- ▶ Pasul **2.** este înlocuit cu:
 - 2'. Calculează o permutare arbitrară a semiplanelor, folosind o procedură adecvată.

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).