Optimisation mathématique : introduction et application à la gestion de l'eau

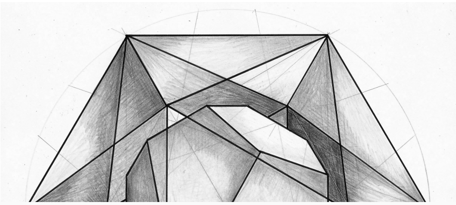
UCA – Master Hydroprotech

Sophie Demassey (CMA, Mines Paris – PSL)
sophie.demassey@minesparis.psl.eu http://sofdem.github.io/hydroprotech/

Programme

décision, optimisation

optimisation combinatoire



décision, optimisation

décision = optimisation

sélectionner la **meilleure** des alternatives **possibles** – les **solutions** –
au regard d'un critère quantitatif – l'**objectif**.

problème décisionnel

Comment **concevoir et piloter** un **système/processus** pour **répondre à un besoin** dans les **limites du possible** tout en **maximisant** un score ?

Optimiser

- identifier les alternatives possibles (**solutions**)
- attribuer un score quantifiant la valeur de chaque alternative
- trouver une alternative de plus haut score

optimisation

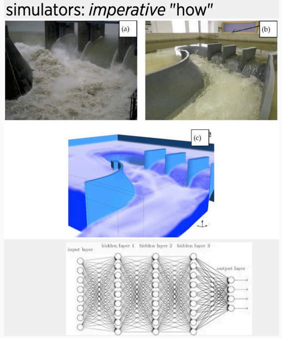
modèle représentation *précise* du système et du score pour **évaluer**

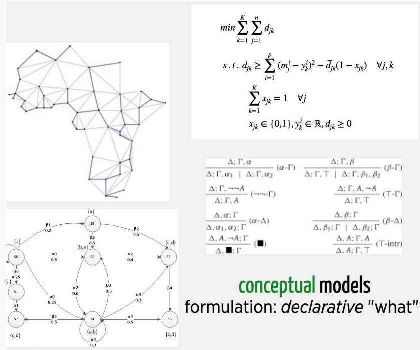
algorithme parcours *rapide* des alternatives à la recherche du meilleur score

MODÈLES

La faisabilité et la valeur d'une décision sont observées sur un modèle du système/processus (non sur le système/processus même)

physical and virtual/numerical models
simulators: *imperative* "how"





MODÈLE D'OPTIMISATION MATHÉMATIQUE

définition : programme mathématique

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } g(x) \le 0 \\ &x \in \mathbb{R}^n \end{aligned}$$

minimiser ou maximiser sur des contraintes $\le, \ge$ ou $=$, mais jamais $>$ ou $<$

programme = **planification** (des opérations militaires ou logistiques)

- $x \in \mathbb{R}^n$: les n variables de décision
- $f : \mathbb{R}^n \to \mathbb{R}$: la fonction objectif — $\max f(x) \equiv -\min(-f)(x)$
- $g : \mathbb{R}^n \to \mathbb{R}^m$ définit m contraintes — $g(x) \le 0 \equiv -g(x) \ge 0 \equiv g(x) + s = 0, s \ge 0$

solutions : $\mathbb{R}^n$

solutions réalisables : $\{x \in \mathbb{R}^n : g(x) \ge 0\}$

solution optimales : $\arg\min_{x \in \mathbb{R}^n} \{f(x) : g(x) \ge 0\}$

MODÈLE D'OPTIMISATION LINÉAIRE

un programme mathématique $\min_{x \in \mathbb{R}^n} \{f(x) | g(x) \ge 0\}$ avec f et g linéaires/affines : $f(x) = c^\top x, g(x) = Ax - b$ avec $c \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$.

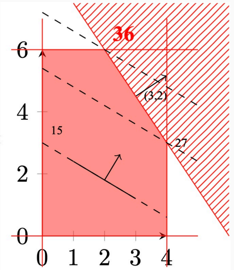
définition : programme linéaire (PL)

$$\begin{aligned} &\min c^\top x \\ &\text{s.t. } Ax \ge b \\ &x \in \mathbb{R}^n \end{aligned}$$

$$\begin{aligned} &\min \sum_{j=1}^{n} c_j x_j \\ &\text{s.t. } \sum_{j=1}^{n} a_{ij} x_j \ge b_i, && \forall i = 1, \dots, m \\ &x_j \in \mathbb{R} && \forall j = 1, \dots, n \end{aligned}$$

EXEMPLE 1

$$\begin{aligned}
\max\ & 3x_1 + 5x_2 \\
\text{s.t. } & x_1 \leq 40 \\
& x_2 \leq 60 \\
& 3x_1 + 2x_2 \leq 180 \\
& x_1, x_2 \geq 0
\end{aligned}$$



- contrainte linéaire = demi-plan de $\mathbb{R}^n$, solutions réalisables = polyèdre
- les solutions (x_1, x_2) de score p = le plan (ligne pointillée) $3x_1 + 5x_2 = p$
- solution optimale = sommet du polyèdre sur la ligne pointillée la plus haute : $(x_1 = 20, x_2 = 60)$ avec $p = 360$
- algorithme du simplexe : parcourt les sommets du polyèdre en suivant les arêtes *améliorantes*. Complexité théorique exponentielle, mais rapide en pratique.

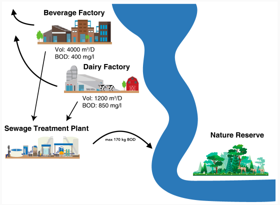
EXERCICE 2 : MODÉLISER

la Demande Biochimique en O_2 mesure la pollution de l'eau en masse d'O_2 requise pour biodégrader la matière organique présente dans l'eau

traitement de l'eau [Zhou, Sustainability 2019]

Par jour, deux usines produisent resp. $1200m^3$ (DBO=$850mg/L$) et $4000m^3$ ($DBO = 400mg/L$) d'eaux usées. Les systèmes de traitement respectifs ramènent 1 tonne DBO à 100kg et 50kg pour un coût de 400 et 500 euros. La part traitée est rejetée dans la rivière dans la limite autorisée de $DBO = 170kg$. La part non traitée a un coût d'évacuation de 0.56 et 0.25 euro par m^3. Est-il possible de respecter la limite environnemental dans un budget journalier de 1250 euros ?

GESTION DE LA QUALITÉ DE L'EAU



- dairy : traitée : 1 tonne $\rightarrow$ 100kg DBO = 400 euros, évacuée : 0.56 euros$/m^3$
- beverage : traitée : 1 tonne $\rightarrow$ 50kg DBO = 500 euros, évacuée : 0.25 euros$/m^3$
- quel volume d'eau traiter pour minimiser DBO dans un budget de 1250 euros ?

- x_1, x_2 : volumes traités (en m^3)
- eau usée évacués : volumes (en m^3) ? coût (en euros) ?
- volumes : $y_1 = (1200 - x_1)$, $y_2 = (4000 - x_2)$, coût : $0.56 * y_1 + 0.25 * y_2$
- eau traitée : DBO avant (en kg) ? après (en kg) ? coût (en euros) ?
- avant : $r_1 = 850 * x_1 * 10^{-3}$, $r_2 = 400 * x_2 * 10^{-3}$, après : $10\% r_1 + 5\% r_2$
- coût : $0.4 * r_1 + 0.5 * r_2$

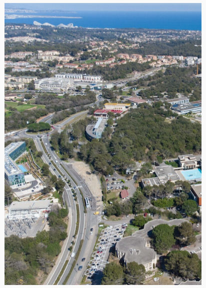
EXERCICE 2 : MODÈLE PL

- Quel volume d'eau traiter pour minimiser DBO dans un budget de 1250 euros ? La valeur DBO est-elle $\leq 170kg$?
- x_1, x_2 : volumes traités (m^3)

$$\begin{aligned}
\min\ & 0.1r_1 + 0.05r_2 \\
\text{s.t. } & 0.4r_1 + 0.5r_2 + 0.56(1200 - x_1) + 0.25(4000 - x_2) \leq 1250 \\
& r_1 = 0.85 * x_1 \\
& r_2 = 0.4 * x_2 \\
& 0 \leq x_1 \leq 1200 \\
& 0 \leq x_2 \leq 4000
\end{aligned}$$

PRÉCISION & APPROXIMATION

prendre une décision



résoudre un modèle

$$min\ z = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} d_{ij} x_{ij}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = 1, \quad \forall i \in N$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1, \quad \forall j \in N$$

problème concret $\longrightarrow$ modèle abstrait $\xrightarrow{\text{solve}}$ solution théorique $\longrightarrow$ décision pratique

résoudre un modèle $\neq$ résoudre un problème

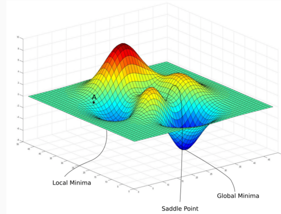
RÉSOUDRE : THÉORIE VS PRATIQUE

modèle $\neq$ problème

- données incertaines (prédiction) et imprécises (tronquées)
- dynamiques/fonctions simplifiées
- objectif conceptuel

résoudre $\min f(x) : g(x) \leq 0$**?**

- tolérance de faisabilité : $g(x) \leq \epsilon$
- tolérance d'optimalité : $f(x) \leq \min f + \epsilon$
- optimum local vs global
- garantie théorique vs pratique : forte complexité, convergence lente, temps limité



algorithmes variés pour des besoins variés

ALGORITHMES D'OPTIMISATION

simulation : **évalue** la faisabilité et le score d'une solution
heuristique : **parcourt et évalue progressivement** quelques solutions
optimisation : parcourt et évalue **(implicitement) toutes** les solutions

2 principes

- generate & test
- divide & conquer

GENERATE & TEST

méthodes numériques et black-box

1. évalue un candidat, 2. choix du prochain candidat

simulate

guess $\xrightarrow[k=0]{}$ x_k $\longrightarrow$ $f(x_k)$ $\longrightarrow$ stop ?

$k++$

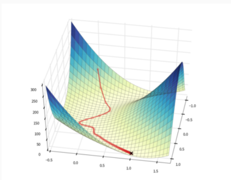
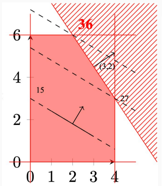
direction/step ?

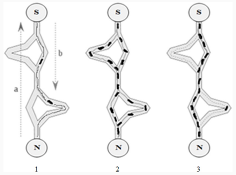
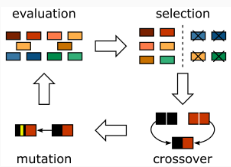
stratégies de choix : aléatoire, énumération, proximité, direction de descente,...

GENERATE & TEST : EXEMPLES

stratégies de choix : aléatoire, énumération, proximité, direction de descente,...

- **méthodes du 1er ordre** (*gradient, simplex*) : direction de descente $f'(x) < 0$
- **recherche locale** : choix du meilleur voisin
- **algorithmes particulaires** (*fourmis*) : mémoire collective
- **algorithmes évolutionnaires** (*génétique*) : reproduction d'une population

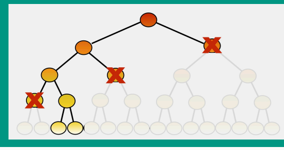



DIVIDE & CONQUER

principe

- séparer l'espace de recherche
- résoudre une relaxation / estimer borne
- backtracker ou continuer

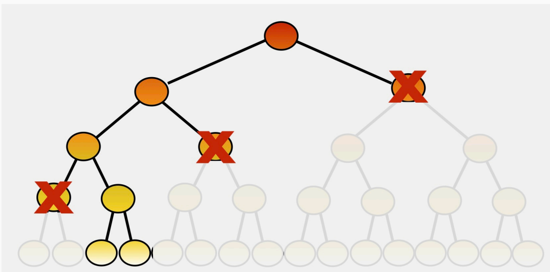


Pour un problème de minimisation :

- **relaxation :** modèle simplifié, rapide à optimiser, donne une borne inférieure
- une solution réalisable donne une borne supérieure
- certificat d'optimalité si les bornes coincident

DIVIDE & CONQUER : EXEMPLES

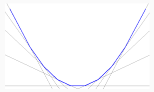
- **heuristique gloutonne** : pas de backtrack
- **algorithmes de graphe, programmation dynamique**
- **méthodes arborescentes**
- **branch-and-bound** en optimisation combinatoire

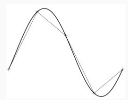


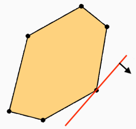
OPTIMISATION COMBINATOIRE

INTÉRÊT DE LA PROGRAMMATION LINÉAIRE

- **nombreuses applications :**
  modèle direct de problèmes pratiques,
  approximation de problèmes convexes,
  base de problèmes non convexes ou logiques
  (associés à des variables à valeur dans $\mathbb{Z}$)
- **facile à résoudre :**
  complexité polynomiale,
  propriétés fortes (dualité),
  algorithmes exacts efficaces,
  implémentations disponibles







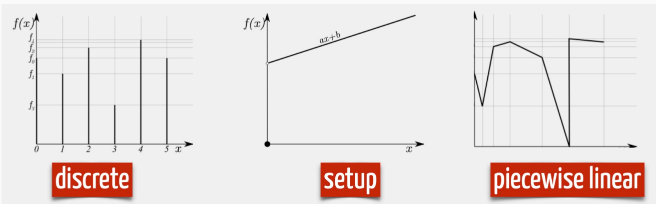
MODÈLE D'OPTIMISATION LINÉAIRE EN NOMBRES ENTIERS

définition : programme linéaire en nombres entiers (PLNE)

$$
\begin{aligned}
\min \; & \sum_{j=1}^{n} c_j x_j & \\
\text{s.t.} \; & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, & \forall i = 1, \ldots, m \\
& x_j \in \mathbb{R} & \forall j = 1, \ldots, p \\
& x_j \in \mathbb{Z} & \forall j = p+1, \ldots, n
\end{aligned}
$$

MODÉLISER AVEC DES VARIABLES DISCRÈTES

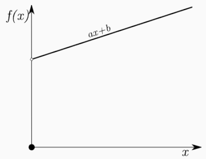
- décisions discrètes : on/off $x \in \{0,1\}$, niveau $z \in \{0, 1, \ldots, N\}$
- conditions logiques : $z \leq N(1-x)$... *si* $x = 1$ *alors* $z = 0$
- fonctions non-linéaires :



MODÉLISER DES CONDITIONS LOGIQUES

condition	example	linéarisation
exclusion	c *faux ou vrai*	$y \in \{0,1\}$
exclusion	*soit* c_1 *soit* c_2	$y_1 + y_2 = 1$
disjonction	c_1 *ou* c_2	$y_1 + y_2 \geq 1$
implication	*si* c_1 *alors* c_2	$y_2 \geq y_1$
alternative	1 *parmi* n	$\sum_{i=1}^{n} y_i = 1$
compteur	k *parmi* n	$\sum_{i=1}^{n} y_i = k$
borne	*au moins* k *parmi* n	$\sum_{i=1}^{n} y_i \geq k$
borne	*au plus* k *parmi* n	$\sum_{i=1}^{n} y_i \leq k$

MODÉLISER DES FONCTIONS NON-LINÉAIRES

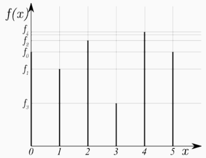


set-up :

$f(x) = ax + by$

$\epsilon y \leq x \leq Uy$

$y \in \{0, 1\}$



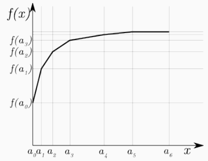
discret :

$f(x) = \sum_i y_i f_i$

$\sum_i i y_i = x$

$\sum_i y_i = 1$

$y_i \in \{0, 1\}\ i = 0..n$



morceaux :

$f(x) = \sum_i \lambda_i f(a_i)$

$\sum_i a_i \lambda_i = x$

$\sum_i \lambda_i = 1$

$\lambda_i \in [0, 1]\ i = 0..n$

$SOS2(\lambda_i)$

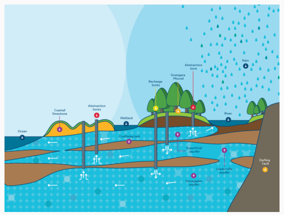
EXERCICE 3 : MODÉLISER

extraction d'eau

décider de l'emplacement de pompes (toutes identiques) parmi un ensemble fini J de candidats pour minimiser le coût total, étant donnés :

- le coût d'installation c_j et le débit moyen q_j d'une pompe à l'emplacement $j \in J$
- les limites minimale $Qmin$ et maximale $Qmax$ du débit moyen total d'extraction
- la limite maximale de 3 pompes installées
- l'interdiction de placer 2 pompes simultanément aux emplacement j_1 et j_2.

EXERCICE 3 : EXTRACTION DE L'EAU



- coût c_j, débit q_j à l'emplacement $j \in J$
- débit min $Qmin$ et max $Qmax$
- nombre max 3 pompes
- exclusion j_1 et j_2

- $x_j \in \{0, 1\}$: pompe installée en $j \in J$?
- minimisation du coût : $\sum_{j \in J} c_j x_j$
- limites de débit : $Qmin \leq \sum_{j \in J} q_j x_j \leq Qmax$
- nombre max : $\sum_{j \in J} x_j \leq 3$
- exclusion : $x_1 + x_2 \leq 1$

EXERCICE 3 : MODÈLE PLNE

$$
\begin{aligned}
\text{min } & \sum_{j \in J} c_j x_j \\
\text{s.t. } & \sum_{j \in J} q_j x_j \geq Qmin \\
& \sum_{j \in J} q_j x_j \leq Qmax \\
& \sum_{j \in J} x_j \leq 3 \\
& x_1 + x_2 \leq 1 \\
& x_j \in \{0, 1\}, \quad \forall j \in J
\end{aligned}
$$

EXERCICE 4 : MODÉLISER

extraction d'eau (variante avec pompes individuelles)

les pompes sont maintenant choisies parmi un ensemble fini K et :

- le coût d'investissement c_k dépend uniquement de la pompe $k \in K$
- le débit moyen q_{jk} dépend de la pompe $k \in K$ et de l'emplacement $j \in J$

$x_{jk} \in \{0,1\}$: pompe $k \in K$ installée en $j \in J$?

EXERCICE 4 : MODÈLE PLNE

$x_{jk} \in \{0,1\}$: pompe $k \in K$ installée en $j \in J$?

$$
\begin{aligned}
\min \; & \sum_{j \in J} \sum_{k \in K} c_k x_{jk} \\
\text{s.t. } & Qmin \leq \sum_{j \in J} \sum_{k \in K} q_{jk} x_{jk} \leq Qmax \\
& \sum_{j \in J} x_{jk} \leq 1, \quad \forall k \in K \\
& \sum_{k \in K} x_{jk} \leq 1, \quad \forall j \in J \\
& \sum_{k \in K} (x_{1k} + x_{2k}) \leq 1 \\
& x_{jk} \in \{0,1\}, \quad \forall j \in J, k \in K.
\end{aligned}
$$

EXERCICE NOTÉ

En solo ou en binôme, implémentez :

- soit le programme linéaire de l'exercice 2 (traitement) en le résolvant plusieurs fois pour différentes valeurs de budget
- soit le programme linéaire en nombres entiers de l'exercice 3 (extraction) en générant les données de manière aléatoire comme par exemple ci-dessous :

```
import random
import matplotlib.pyplot as plt
```

Données du problème

```
[7] random.seed(1)

    nplaces = 10                                     # nombre d'emplacements potentiels
    cost = [random.randint(5,25) for _ in range(nplaces)]    # cout d'investissement
    flow = [random.randint(100,300) for _ in range(nplaces)] # debit moyen
    flowmin = 500
    flowmax = 600
    nmax = 3
```