Cours n°1: Introduction à l'optimisation discrète et à la complexité de problèmes d'optimisation, premiers exemples de programmation linéaire

Nicolas DUPIN

https://github.com/ndupin/ORteaching
http://nicolasdupin2000.wixsite.com/research

version du 19 mars 2022

Issu et étendu d'enseignements donnés à l'ENSTA Paris, Polytech' Paris-Saclay, et à l'université Paris-Saclay

Plan

Plan

Des questions pour vous

- Pouvez vous citer des algorithmes d'optimisation ?
- Avez vous déjà entendu parler de :
  - Recherche Opérationnelle (RO) ?
  - Programmation Linéaire (PL) ?
  - Programmation Linéaire en Nombres Entiers (PLNE) ?
  - Optimisation combinatoire ?
- Savez vous dans quelles entreprises la RO et la PLNE sont utilisées ?

Vous êtes pardonnés !

- Ce cours aurait très bien pu s'appeler "Introduction à la Recherche Opérationnelle", légèrement plus précis qu' "Optimisation et Applications", mais moins parlant !

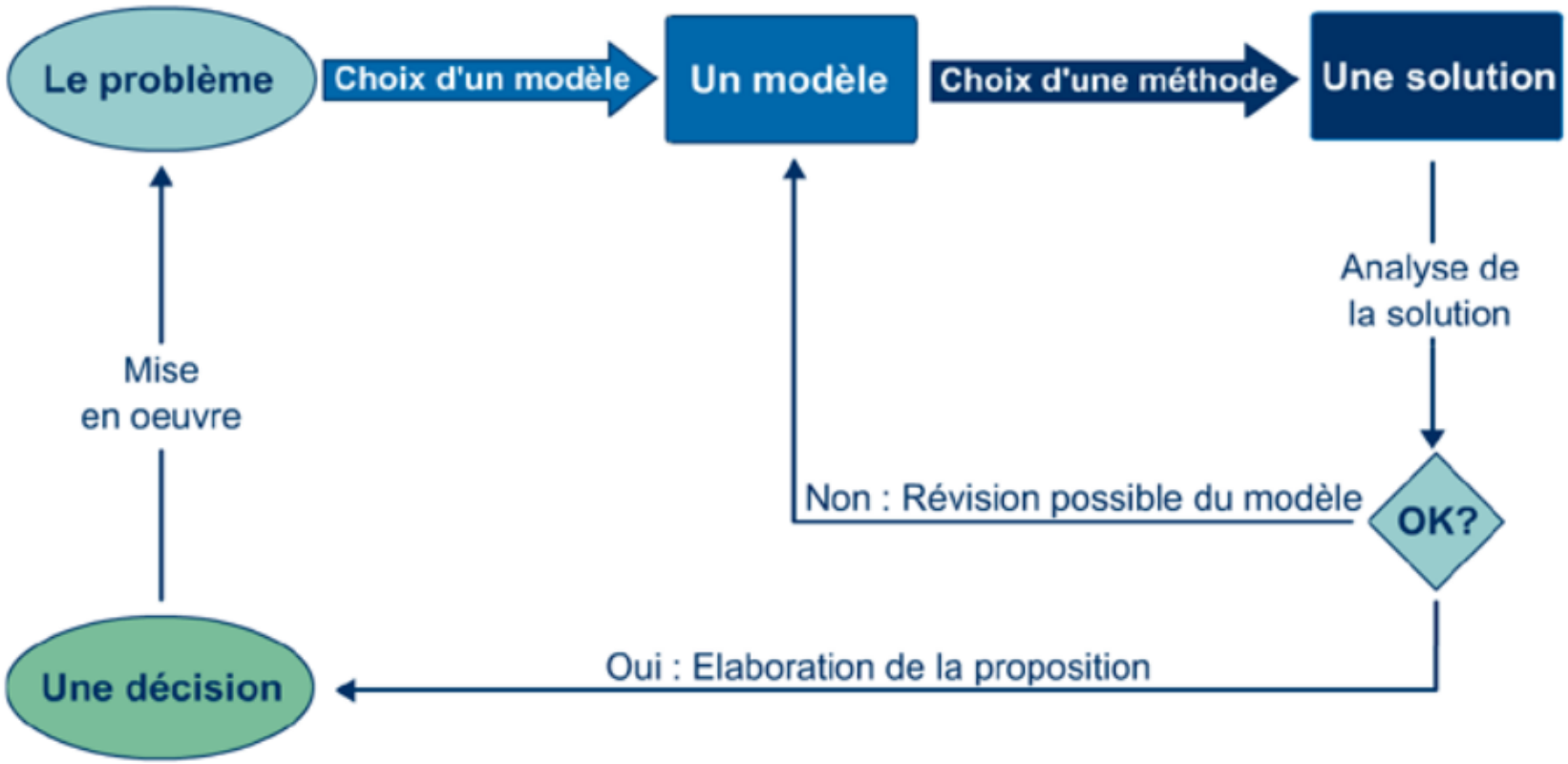
- Les dénominations qui définissent le mieux la majeure partie de ce cours sont souvent assez peu parlantes pour les non-initiés :
  - Recherche Opérationnelle (RO) ?
  - Programmation Linéaire (PL) ?
  - Programmation Linéaire en Nombres Entiers" (PLNE) ?
  - Optimisation combinatoire ?

- ex : recherche opérationnelle, est une traduction de "Operational Research", recherche des opérations, pour recherche sur les problèmes opérationnels dont font face des entités comme les armées (à l'origine de la RO), des organisations ou des entreprises

Un aperçu de la définition de la recherche opérationnelle ?

- Recherche Opérationnelle (RO, aussi appelée aide à la décision) : ensemble des méthodes analytiques avancées d'aide à une meilleure prise de décisions.
- En utilisant la modélisation mathématique pour analyser des situations complexes, la RO donne aux décideurs la capacité à prendre rapidement des décisions plus efficaces.
- Modélisation mathématique d'un problème réel et techniques de résolution appropriées.
- Résolution : utilisation de puissance de calcul informatique, permet de traiter une combinatoire inaccessible à une expertise ou décision purement humaine. Résultat d'un modèle (donc imparfait par essence), se doit d'être confronté et validé par une expertise humaine.
- Optimisation sous contraintes : cadre de modélisation courant en RO, mais la RO ne se restreint pas à l'optimisation sous contraintes.

Démarche de recherche opérationnelle



Grands acteurs de recherche opérationnelle en France

- Laboratoires académiques : sections CNU 26 (maths appliquées), 27 (informatique) et 61 (génie industriel).
- R&D industrielles et praticiens : Renault, Air France, Orange, EDF, RTE, ENGIE, SNCF, Saint Gobain, Thales, Dassault ...
- Prestataires spécialisés en RO : Eurodécision, Artelys, LocalSolver ...
- ROADEF : Société Française de Recherche Opérationnelle et d'Aide à la Décision. Association regroupant à la fois les équipes académiques de RO (expertise méthodologique), les développeurs de logiciels spécifiques et des départements de RO appliquées (expertise opérationnelle)

http ://www.roadef.org/societe-francaise-recherche-operationnelle-aide-decision

$\Longrightarrow$ La recherche opérationnelle, une discipline de recherche appliquée avec des problématiques issues d'applications industrielles.

P.S : informations utiles pour chercher un stage dans le domaine

Méthodes/Outils de la recherche opérationnelle ?

- Algorithmes dédiés à des problèmes spécifiques (ex : programmation dynamique, au prochain cours).
- Optimisation continue sous contraintes, pour des problèmes continus ou comme brique d'une approche de résolution d'optimisation discrète. Un aperçu minimal pour ce cours, approfondi en M1 de maths.
- PLNE : méthodes exactes, assez "lourdes", très répandues dans l'industrie. Le coeur de ce cours. Extension non-linéaire en M1.
- Les (Méta)-Heuristiques : approches locales plus "légères", permettant un meilleur passage à l'échelle que les méthodes exactes telles que la PLNE. Un aperçu dans ce cours, approfondi en M1 info.
- Programmation Par Contraintes : méthodes arborescente, avec des similarités mentionnées avec la PLNE dans ce cours, cours spécifique "IA et contraintes"
- Pans plus théoriques de théorie de la décision. (application en économie)

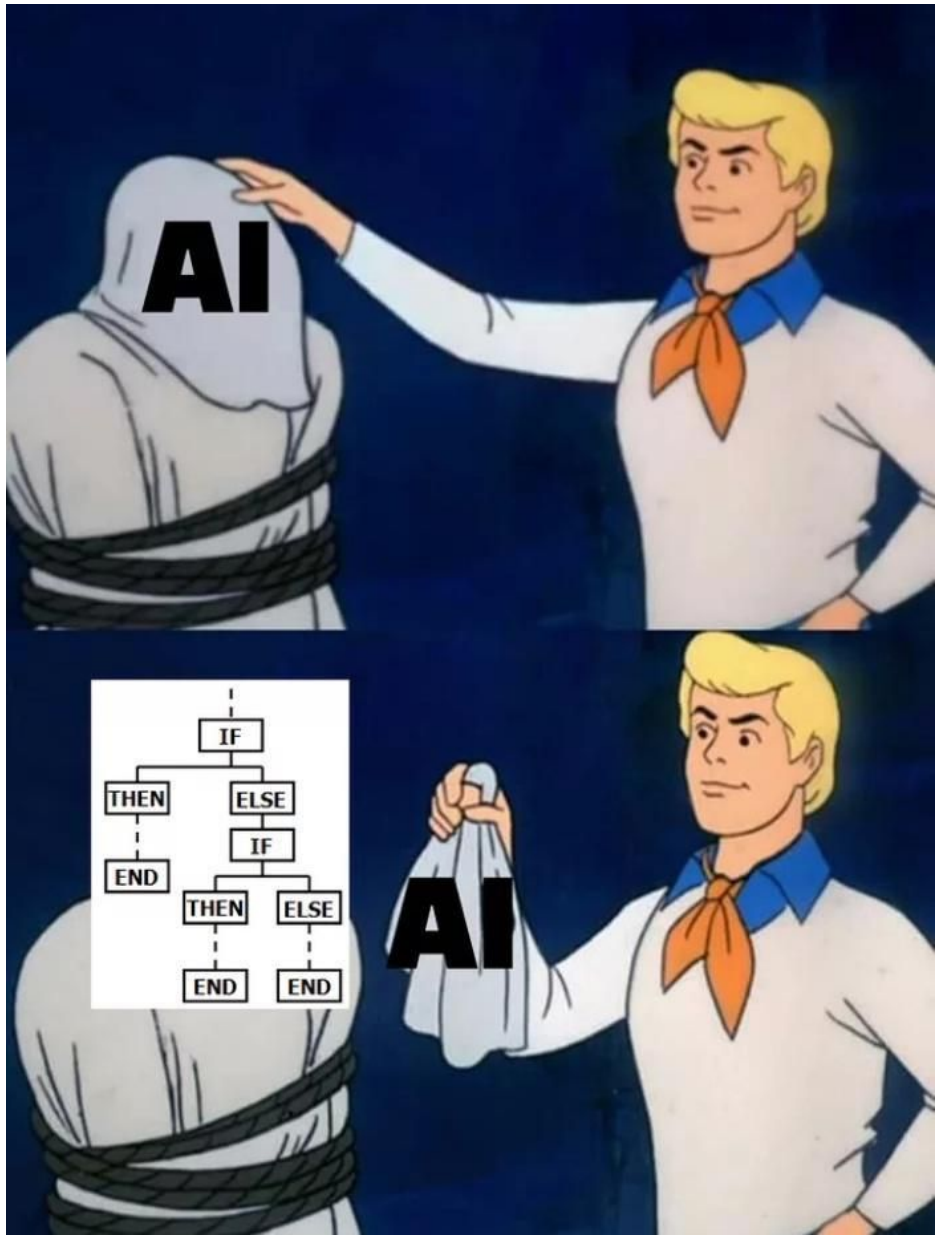
Quel lien entre RO et intelligence artificielle ?

- Question piège ! Les délimitations et définitions ne sont pas forcément très nette, ou universelles.
- Certains voient la RO comme une discipline de l'IA, ça en fait bondir d'autres ...
- Avis personnel : les définitions littérales d'IA et RO ont des similitudes, on comprend mieux les points de vue des uns et des autres et regardant l'aspect historique et applicatif pour cerner des délimitations.
- ex : En IA, on demande souvent des temps de réponses d'algorithmes très courts, ou des contraintes de calculs embarqués nécessitent d'avoir une implémentation légère. En PLNE, on pourra avoir des temps de calculs longs (une nuit) et nécessitant une bonne puissance de calcul.
- Plus de consensus : la Programmation Par Contraintes et les (Méta)-Heuristiques sont des sous domaines communs à la RO et l'IA.
- dans les sous sections CNU 27, les méta-heuristiques apparaissent juste dans Intelligence Artificielle, pas de RO. Des papiers de RO se trouvent redirigés dans la section Informatique.IA d'ArXiv ...

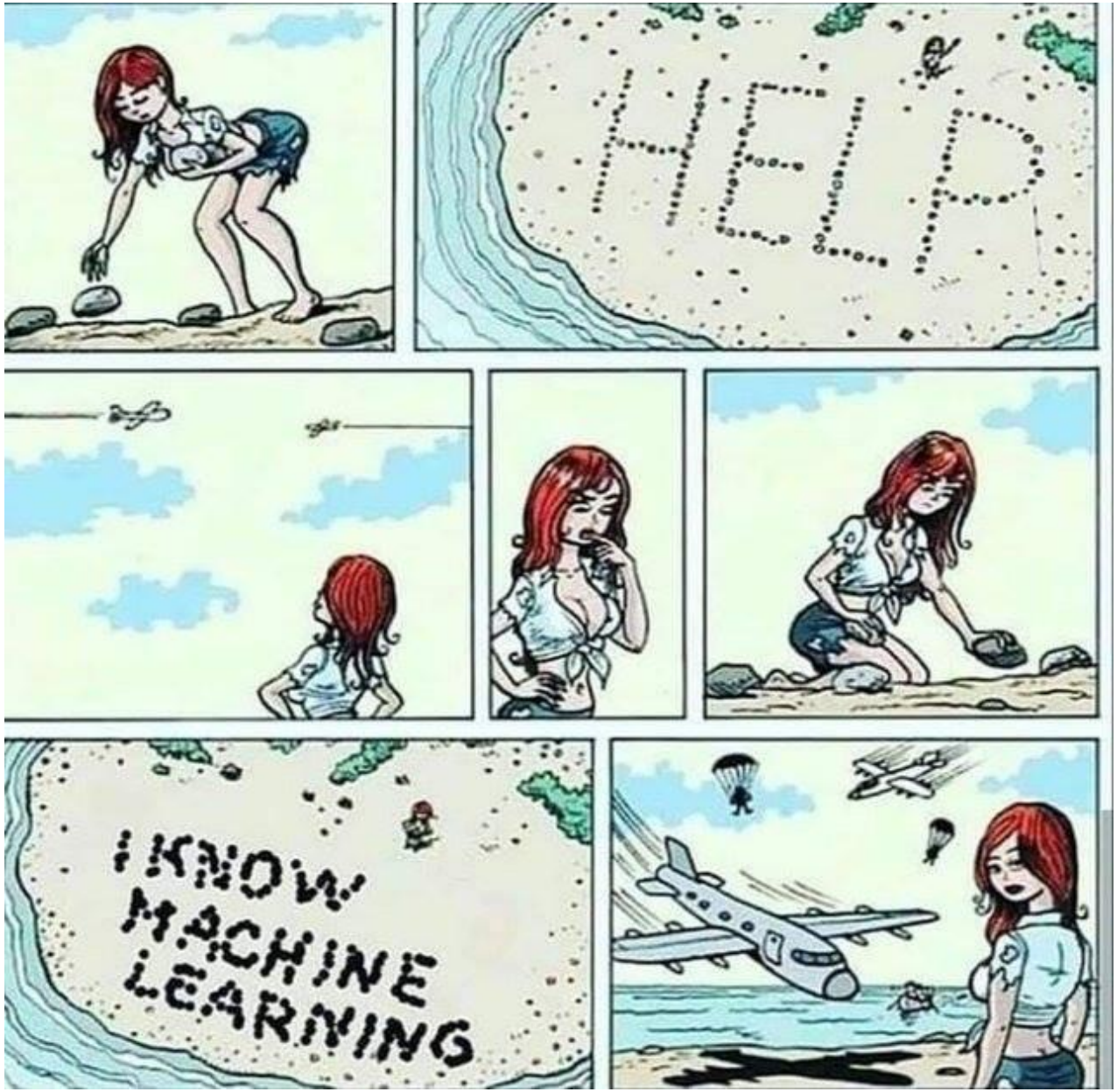
Pourquoi la RO est-elle moins connue que le Machine Learning ?

- RO était un domaine plus "hype" dans les années 50-60, alors que les capacités de calculs n'étaient pas celles d'aujourd'hui. Des déceptions par rapport aux promesses du domaine face aux capacités pratiques ?
- L'IA était déjà développée dans les années 50, puis des phases d'engouement et de déceptions ("hivers" de l'IA) ...
- Courbe du "hype", fort engouement pour le Deep Learning après les résultats spectaculaires d'AlphaGO, révolution brutale qui a marqué les esprits généralement.
- La PLNE (et aussi l'IA plus traditionnelle), les progrès ont été plus progressifs, tant sur les algorithmes que sur les conséquences des améliorations des capacités de calculs, les progrès touchent plus les connaisseurs du domaine.
- En pratique des experts/prestataires de RO peuvent être catalogués dans l'IA, ou des problèmes de RO donnés à des data scientists.

Pour rire de certains excès actuels



Pour rire de certains excès actuels



Quelles différences entre PLNE et Machine Learning ? (1)

- ML : besoin de bcp de données représentatives pour utiliser des algorithmes assez simples (légers et rapides à l'utilisation) utilisant la masse des données, pour y extraire son "intelligence" (éventuellement avec une longue phases dédiée à l'apprentissage pour calibrer la méthode). Méthodes prédictives.
- PLNE et PPC ; réponse à un problème donné à un instant donné, besoin des paramètres du moment présent, et algorithmes plus lourds utilisant fortement les spécificités d'un problème . . .
- PPC : méthode de décision, répond à une question oui/non dans sa conception initiale.
- PLNE, méta-heuristiques : optimise une fonction de score.
- ML : essaie de reproduire pour prévoir.

$\Longrightarrow$ Cadres d'hypothèses différents, parfois il est possible de combiner les avantages de ces différentes techniques en les hybridant

Quel lien entre RO et Machine Learning ?

- Bases communes : algèbre linéaire dans $\mathbb{R}^n$, structures discrètes combinatoires, et surtout APPROCHE EXPERIMENTALE VALIDEE PAR RESULTATS EXPERIMENTAUX SUR DES JEUX DE DONNEES.
- De l'apprentissage est utilisé pour accélérer les solveurs de PLNE, quand on regarde dans la boîte noire. (ex : Cplex), ou dans les approches méta-heuristiques.
- Le ML utilise des résolutions de problèmes d'optimisation et des techniques d'optimisation : optimisation des paramètres d'un réseau de neurones, clustering k-means, régression linéaire, moindre carrés ...
- Reinforced Learning : "Q-learning" est un cas de programmation dynamique, et ε-greedy très très proche de l'algorithme du recuit simulé (heuristique d'optim combinatoire)
- Un pan de recherche très actif actuellement à hybrider RO et ML.

$\Longrightarrow$ Des cours de RO permet de compléter la palette d'outils de l'informaticien et du data scientist, ce cours vise à sensibiliser à l'existence de méthodes et à l'utilisation simple d'outils, sans pour autant poursuivre un cursus de RO.

Approche pédagogique

- Les fondements théoriques sont limités au nécessaire pour appréhender les hypothèses et l'utilisation d'un outil de PLNE. Peu de démonstrations, uniquement quand cela présente un intérêt pédagogique.
- Illustrations par des exemples/contre-exemples pour comprendre des hypothèses, ou la généralité et les limites de certaines méthodes.
- Cours calibré sur le potentiel applicatif : les algorithmes à utiliser tels quels dans les solveurs et les pans théoriques associés sont présentés plus brièvement. On présentera plus précisément les techniques ayant un fort impact sur l'utilisation pratique de solveurs.
- Cours peut illustrer ou donner des applications d'autres UE (algorithmique, algorithmes d'IA)
- Support de cours fournis : présentations très détaillées avec illustrations, une prise de notes personnelles en complément est plus que conseillée.

Plan

Optimisation, définition de base

Soit X l'ensemble des décisions possibles d'un problème d'optimisation. On dit aussi que X est l'ensemble des *solutions* (ou *solutions réalisables*) du problème d'optimisation.

On définit une fonction de score f, dite *fonction objectif*, pour pouvoir évaluer les décisions possibles, que l'on cherchera à maximiser ou minimiser.

Si on cherche à minimiser f , x_1 est préféré à x_2, si $f(x_1) < f(x_2)$. L'ordre total sur $\mathbb{R}$ permet de toujours comparer des solutions deux à deux

Quitte à changer f en $-f$, on peut supposer que l'on minimise une fonction objectif pour la suite.

Exemple : pour le meilleur itinéraire entre 2 points, on peut minimiser le temps, la distance, le coût du trajet (essence+péages) ou l'impact carbone, ou maximiser le profit (avec du car-sharing).

Dans ce cours, un objectif unique à choisir, ou alors composer une unique fonction de score à partir de ces trois critères (extension optimisation multi-objectif en M2)

Optimisation, cadre très général

Problème d'optimisation :

$$\min_{x \in X} f(x)$$

où X est l'ensemble des solutions réalisables

pour tout $x \in X$, on a une fonction qui permet d'évaluer la solution réalisable $f : X \longmapsto \mathbb{R}$.

Question : comment calculer $\min_{x \in X} f(x)$ et déterminer la meilleure solution x ?

Pré-requis : a t'on l'existence d'un min, plus petit élément, et pas une borne inférieure non atteinte ou $-\infty$?

Bullshit Optimization

L'optimisation, comme l'IA a aussi ses vendeurs de soupe. L'optimisation a été dévoyée par certains éléments de langage, ou des pratiques qui n'ont rien à voir avec de l'optimisation dans le cadre du cours :

- "Optimisation fiscale"
- "Optimisation comptable/budgétaire"
- "Optimisation d'effectifs de personnels/de ressources humaines", "variables d'ajustement"
- "Optimisation de process"
- "Do more with less"

A t'on vraiment la définition d'un tel ensemble X et une fonction objectif unique qui permet d'évaluer toute alternative ?

Optimisation souvent employé pour "amélioration de l'existant", sans critères quantitatifs et rationnels.

Attention à la mise en pratique de modèles simplistes (un modèle est toujours imparfait).

Optimisation de code

L'optimisation de code : travailler son code pour le rendre plus efficace.

Optimisation pour "amélioration", l'amélioration de l'existant, c'est le quotidien d'un ingénieur.

Ici, pour avoir l'évaluation du gain (temps et/ou mémoire), il faut avoir implémenté la variante.

En optimisation, le plus souvent, on a une évaluation de la fonction objectif sans avoir à mettre en pratique.

Si l'évaluation de chaque alternative est directe, la difficulté réside dans le nombre d'alternatives pour l'optimisation dans le cadre du cours.

L'optimisation de code a son intérêt dans la bonne implémentation d'algorithmes d'optimisation.

Dans la théorie de la RO, on regarde principalement le volet algorithmique.

Optimisation discrète, cadre général

Problème d'optimisation discrète :

$$\min_{x \in X} f(x)$$

où X est un ensemble FINI

$\min_{x \in X} f(x)$ est bien défini mathématiquement comme un plus petit élément d'un ensemble fini.

ex1 : plus court chemin (temps ou distance) dans un graphe routier

ex2 : arbre couvrant de poids minimum dans un graphe

Optimisation discrète, tout énumérer ?

Mathématiquement : sur un ensemble fini, on peut tout énumérer et considérer la meilleure solution

$\Longrightarrow$ algorithme "brute force"

Dans un graphe complet à n sommets :

- 2^{n-2} chemins élémentaires possibles entre deux sommets

- $n!$ arbres couvrants

$\Longrightarrow$ l'algorithme "brute force" n'est pas envisageable hors de petites tailles de n

Algorithmes d'optimisation discrète que vous connaissez déjà

- Plus court chemin dans un graphe avec poids positifs : algorithme de Dijkstra.
- Arbre couvrant de poids minimal (ou maximal) dans un graphe : algorithmes de Prim ou de Kruskal

Une solution particulière est construite et on prouve qu'elle est optimale, pas besoin de tout énumérer

énumérer toutes les solutions réalisables d'un problème (pour les compter par exemple), est un problème plus difficile que de trouver la meilleure solution

Optimisation continue en dimension 1

Problème d'optimisation de la variable réelle :

$$\min_{x \in X} f(x)$$

où $X = \mathbb{R}$ ou X est un intervalle de $\mathbb{R}$

Bien connu avant votre bac, étude de fonction (si suffisamment régulière), étude de la monotonie

Optimisation en dimension 1 : une unique variable à optimiser, des problèmes très simples

Pas toujours des solutions analytiques, la résolution de $f'(x) = 0$ peut se faire par un algorithme numérique (ex : dichotomique)

Optimisation continue sans contrainte en dimension 2 ou plus ?

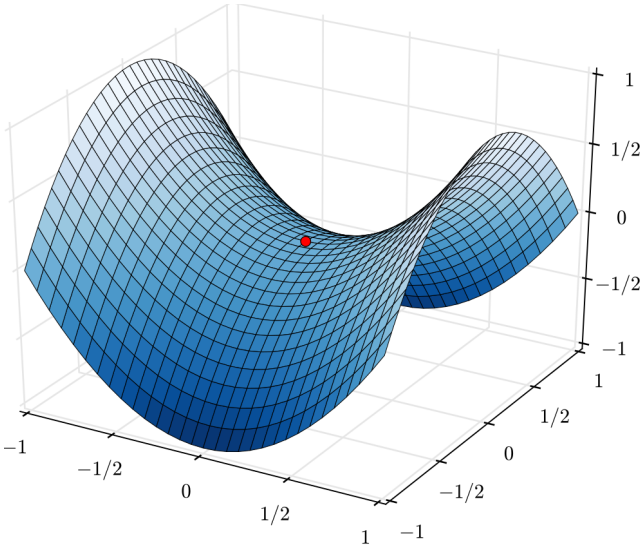
Problème d'optimisation en dimension 2 :

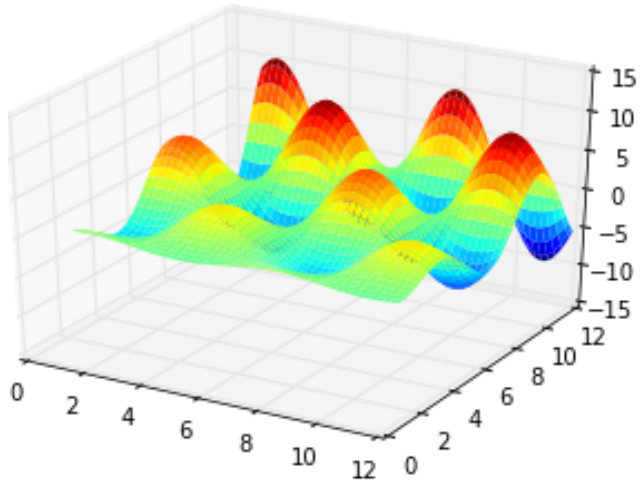
$$\min_{(x,y)\in X} f(x,y)$$

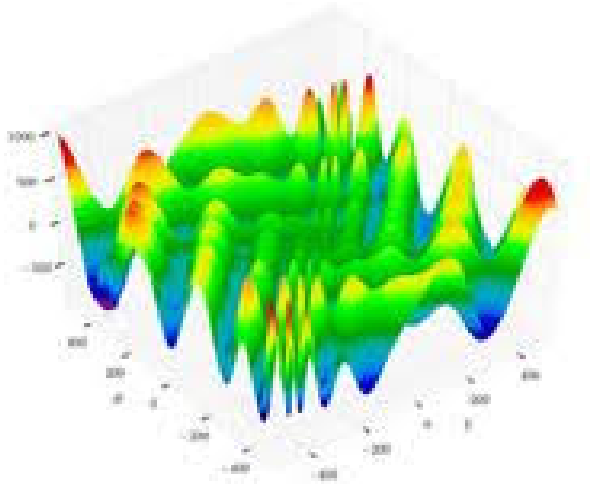
où $X = \mathbb{R}^2$

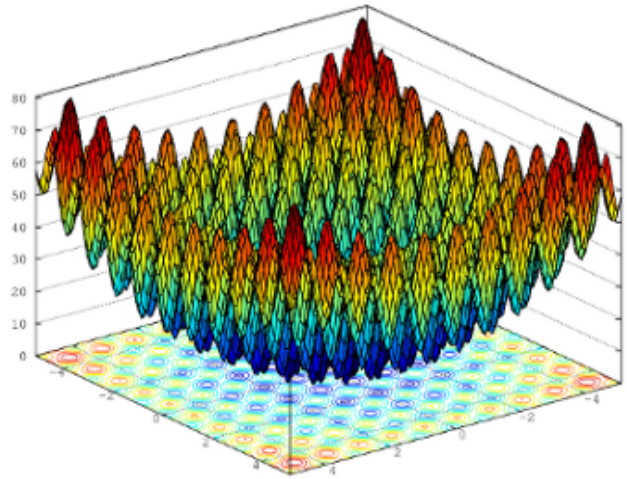
Vu en L2 math, des propriétés de la dimension 1 s'étendent : minimums/maximums locaux, gradients et convexité (locale ou globale)

Pas de monotonie, il peut être compliqué d'énumérer tous les minimums ou maximums locaux









$\Longrightarrow$ Même en dimension 2, on peut oublier la notion de monotonie et les tableaux de variation ! !

Optimisation continue en dimension 2 et plus ?

$$\min_{x \in X} f(x)$$

où $X = \mathbb{R}^n$ ou $X = \mathbb{R}^n_+$ avec $n \geqslant 2$

La théorie de la dimension 2 est quasi identique à celle de la dimension n (pour le M1 maths)

Une différence majeure : en dimension 2, on peut calculer facilement le déterminant d'une matrice et factoriser un polynôme de degré 2 (donc factoriser un polynôme caractéristique)

Algorithmes numériques de résolution : algorithmes de gradients (ordre 1), de Newton (ordre 2), utilisés dans la phase de back-propagation d'un réseau de neurones.

Aussi de l'optimisation "boîte noire" et "sans gradient", sans connaître la dérivée de f. (M2 info IA)

Optimisation sous contraintes, dans le cadre de ce cours

- Optimisation sous contrainte : $\min_{x \in X} f(x)$ où $X = \{x \in \mathbb{Z}^n \times \mathbb{R}^p, g(x) \leqslant 0\}$
- Cela se note aussi $\begin{array}{ll} \min\limits_{x} & f(x) \\ \text{s.c} : & g(x) \leqslant 0 \\ & x \in \mathbb{Z}^n \times \mathbb{R}^p \end{array}$ (s.c ou s.t : pour "sous contrainte")
- X est l'ensemble des solutions réalisables, $X \subset \mathbb{Z}^n \times \mathbb{R}^p$, décisions discrètes et continues. Pour ce cours, X est borné (cas majeur applicatif, qui garantit l'existence d'un minimum si f et g sont continues).
- $g : x \in \mathbb{Z}^n \times \mathbb{R}^p \longmapsto \mathbb{R}^m$, définit m **contraintes** d'inégalités (peut modéliser des égalités) : ce que la solution DOIT satisfaire,
- $f : X \longmapsto \mathbb{R}$: fonction **objectif** unique, définit une fonction de score pour classer les solutions, l'optimisation oriente vers des solutions que l'on SOUHAITE selon le choix de la fonction f.

Optimisation continue/discrète/combinatoire

- Optimisation sous contraintes : $$\begin{array}{ll} \min\limits_{x} & f(x) \\ \text{s.c} : & g(x) \leqslant 0 \\ & x \in \mathbb{Z}^n \times \mathbb{R}^p \end{array}$$

- Si $n = 0$, uniquement des variables continues, cadre de l'*optimisation continue*.

- Si $p = 0$, uniquement des variables entières, cadre de l'*optimisation discrète*. En particulier, des décisions binaires OUI ou NON opérer une action. Une variable binaire $x_i \in \{0, 1\}$ s'écrit comme $x_i \in \mathbb{Z}$ avec les contraintes $-x_i \leqslant 0$ $x_i - 1 \leqslant 0$

- Si les variables peuvent être indicées comme des permutations/partitions d'un ensemble, cadre de l'*optimisation combinatoire*.

$\Longrightarrow$ Cadre commun pour l'optimisation continue et discrète que vous connaissez déjà

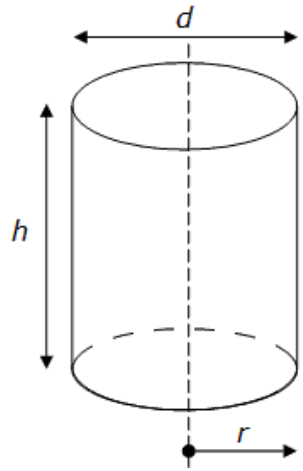
Optimisation sous contraintes, cadre plus général

- Quid $\min_{x \in X} f(x)$ où $X = \{x \in E, g(x) \leqslant 0\}$ où E est un espace de fonctions, ex $L^2([0,1])$, $L^\infty([0,1])$
- Un premier exemple : problème de commande optimale, optimiser la trajectoire d'une fusée pour rentrer dans l'atmosphère
- Un second exemple :optimisation de forme, optimiser la forme d'une aile d'avions pour optimiser l'aéro-dynamisme.
- Dans ces cas là, ce n'est pas en général une bonne idée de discrétiser bestialement l'espace de solutions, méthodes spécifiques partageant des notions de base d'optimisation.

$\Longrightarrow$ Master de maths appliquées pour de tels développements

Application boîte de conserve (1)

Une boîte de conserve est un cylindre, de hauteur h et de rayon r. On veut optimiser les dimensions (ie en fonction de h et r), de sorte à minimiser la surface $S(r,h) = 2\pi r^2 + 2\pi rh$, à un volume donné $V_0 = \pi r^2 h$ de la boîte de conserve

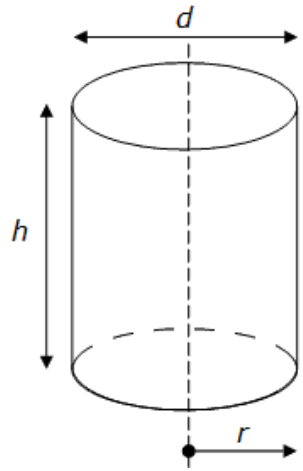


Cela s'écrit $\min_{(r,h)\in\{(r,h)\in\mathbb{R}^2 | g(r,h)\leqslant 0} S(r,h)$ avec $g : \mathbb{R}^2 \longrightarrow \mathbb{R}^4$ définie par :

$$g(r,h) = \begin{pmatrix} \pi r^2 h - V_0 \\ -\pi r^2 h + V_0 \\ -r \\ -h \end{pmatrix} \leqslant \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Application boîte de conserve (2)

Une boîte de conserve est un cylindre, de hauteur h et de rayon r. On veut optimiser les dimensions , de sorte à maximiser le volume $V(r,h) = \pi r^2 h$ à surface donnée $S_0 = 2\pi r^2 + 2\pi rh$.



Cela s'écrit :

$$\min_{(r,h)\in\{(r,h)\in\mathbb{R}^2 | g(r,h)\leqslant 0} V(r,h) \text{ avec } g : \mathbb{R}^2 \longrightarrow \mathbb{R}^4 \text{ définie par :}$$

$$g(r,h) = \begin{pmatrix} 2\pi r^2 + 2\pi rh - S_0 \\ S_0 - 2\pi r^2 - 2\pi rh \\ -r \\ -h \end{pmatrix} \leqslant \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Variables et paramètres

- Optimisation sous contraintes : $\min_{x \in X} f(x)$ où $X = \{x \in \mathbb{N}^n \times \mathbb{R}_+^p, g(x) \leqslant 0\}$
- $x \in X$ désignent les variables, décisions ou leviers d'optimisation.
- A ne pas confondre avec des PARAMETRES, qui peuvent apparaître dans la fonction objectif ou dans les contraintes.
- Exemple : V_0, S_0, pour les boîtes de conserves .
- A un jeu de paramètres donné est associé une solution optimale.
- Dans un premier temps, on supposera les paramètres connus et déterministes. La programmation stochastique traitera de données d'entrées soumises à des incertitudes, et cherchera à minimiser l'impact des aléas sur la solution optimisée.

Objectif ou contrainte ?

- En pratique, les notions de contraintes et d'objectif ne sont pas toujours naturelles et peuvent causer des ambiguïtés.
- La fonction objectif définit une fonction de score pour classer les solutions. Exemples :
  - Temps : réaliser au plus tôt un ensemble de tâches.
  - Minimiser un coût financier/maximiser un profit.
- Pas forcément d'objectif unique et GLOBAL peuvent se dégager d'une solution, des notions locales de préférences peuvent apparaître. Cela induit des difficultés de modélisation. Cas de plusieurs objectifs potentiellement antagonistes : optimisation multi-objectif.
- Si on vous dit : "mon objectif numéro 1, c'est satisfaire mes contraintes ?"
  - S'il n'est pas possible de satisfaire toutes les contraintes, quel est le compromis à trouver ?
  - S'il est possible de satisfaire toutes les contraintes, quelle sont les solutions à favoriser ?

Contraintes d'égalités ou d'inégalités ?

On peut écrire un pb d'optimisation contrainte sous deux formes équivalentes :

$$\begin{array}{ll} \min\limits_{x} & f(x) \\ \text{s.c} : & g(x) \leqslant 0 \\ & x \in \mathbb{Z}^n \times \mathbb{R}^p \end{array} \quad \text{ou} \quad \begin{array}{ll} \min\limits_{x} & f(x) \\ \text{s.c} : & g(x) = 0 \\ & x \in \mathbb{N}^n \times \mathbb{R}_+^p \end{array}$$

Sous la forme d'égalités, on se ramène à des inégalités en changeant une égalité en deux inégalités, et on met la positivité des variables comme une contrainte.

Sous la forme d'inégalités, on rajoute une variables par contrainte d'inégalité, soit un vecteur $\eta \geqslant 0$, et on transforme chaque variable x_i et $x_i = z_i - y_i$ avec $z_i, y_i \geqslant 0$ (y, z représentent les parties positives et négatives)

$$\begin{array}{ll} \min\limits_{x} & f(x) \\ \text{s.c} : & g(x) \leqslant 0 \\ & x \in \mathbb{Z}^n \times \mathbb{R}^p \end{array} = \begin{array}{ll} \min\limits_{y,z,\eta} & f(z-y) \\ \text{s.c} : & g(z-y) = -\eta \\ & z, y \in \mathbb{N}^n \times \mathbb{R}_+^p, \eta \geqslant 0 \end{array}$$

Résultat d'existence

Theorem

Soit $g : \mathbb{R}^{n+p} \to \mathbb{R}$ *une fonction continue. On suppose que* $X = \{x \in \mathbb{N}^n \times \mathbb{R}_+^p, g(x) \leqslant 0\}$ *est borné. Alors* $X = \{x \in \mathbb{N}^n \times \mathbb{R}_+^p, g(x) \leqslant 0\}$ *est un compact et* $\min_{x \in X} f(x)$ *et* $\max_{x \in X} f(x)$ *sont bien définis.*

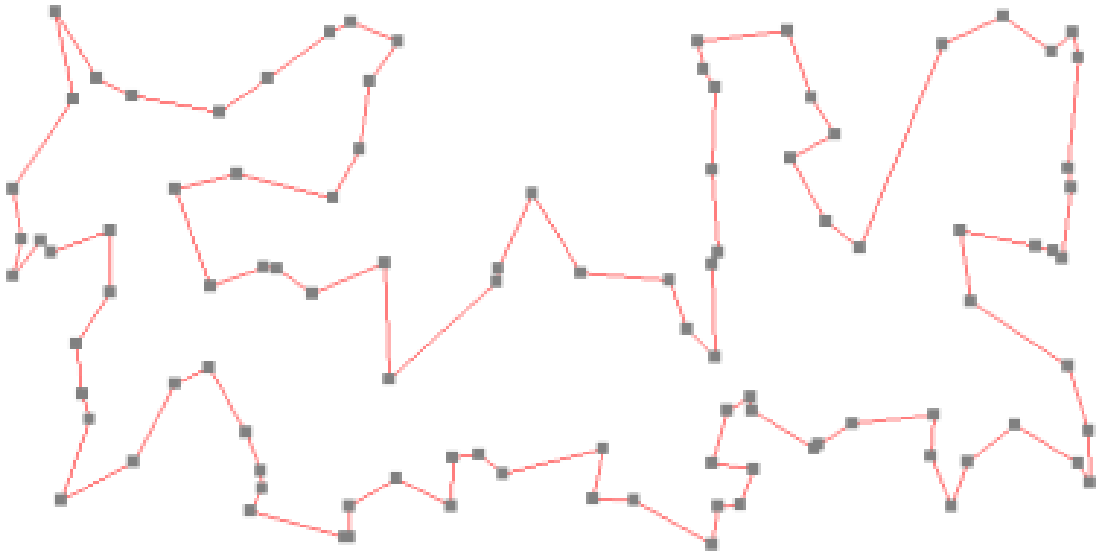
Preuve : Si g est continue, $A = \{x \in \mathbb{R}^{n+p}, g(x) \leqslant 0\} = g^{<-1>}(\mathbb{R}_-)$ est fermé, image réciproque d'un fermé. $X = A \cap \mathbb{N}^n \times \mathbb{R}_+^p$ est alors fermé par intersection de fermés. X est donc bien un compact.

Exemple d'optimisation peu structurée

- On a n points (x_n) de $\mathbb{R}^d$ sans aucune structure donnée. n très grand (contexte big data)
- On a un point donné y de $\mathbb{R}^d$. On veut déterminer quel est le point x_n le plus proche de y.
- Algorithme ; énumérer toutes les n possibilités, et garder en mémoire le point le plus proche trouvé.
- On n'aura pas d'autre alternative pour trouver l'optimum que de tout énumérer

Le problème de voyageur de commerce

- Un voyageur de commerce doit transiter par n villes données.
- Minimiser la longueur de son trajet.



Exemples d'optimisation très structurée : le voyageur de commerce

- Structure du problème : trouver une liste ordonnée des villes minimisant la somme des distances entre deux points successifs.
- Cadre optimisation combinatoire : trouver une permutation f de $[\![1, n]\!]$ dans $[\![1, n]\!]$ avec $f(1) = 1$ (choix arbitraire de première ville) minimisant la somme des distances entre deux points successifs.
- $(n-1)!$ possibilités, impossible d'énumérer toutes les possibilités même avec des petites valeurs de n.
- On pourra formuler le problème en optimisation sous contraintes avec un nombre de variables et de contraintes polynomial en n.
- On pourra prouver l'optimalité de solutions sans tout énumérer.

Les algorithmes de résolution de ce cours rentrent dans un cadre structuré.

Optimisation linéaire

Optimisation linéaire : l'objectif et les contraintes sont des fonctions linéaires

Programme Linéaire (PL, Linear Programming LP) : les variables sont continues :

$$\begin{aligned} & \min_{x \in \mathbb{R}_+^p} \sum_{i=1}^{p} c_i x_i \\ s.c : \quad & \sum_{i=1}^{p} A_{i,c} x_i \geqslant b_c \quad \forall c \in [\![1; C]\!] \end{aligned} \tag{1}$$

Programme Linéaire en Nombres Entiers (PLNE, Integer Linear Programming ILP) : les variables sont discrètes :

$$\begin{aligned} & \min_{x \in \mathbb{N}^m} \sum_{i=1}^{m} c_i x_i \\ s.c : \quad & \sum_{i=1}^{m} A_{i,c} x_i \geqslant b_c \quad \forall c \in [\![1; C]\!] \end{aligned} \tag{2}$$

Cas général : Programme Linéaire Mixte en Nombres Entiers (Mixed Integer Linear Programming)

Forme canonique, forme standard d'un PLNE

Soit un PLNE sous sa forme canonique :

$$\begin{aligned} & \max_{x_i} \sum_{i=1}^{N} c_i x_i \\ s.c : \quad & \sum_{i=1}^{m} A_{i,c} x_i \leqslant b_c, \ \forall c \in [\![1; C]\!] \\ & \forall i \in [\![1, N]\!], \ x_i \in \{0, 1\} \end{aligned} \tag{3}$$

Les contraintes sous la forme d'inégalités peuvent être transformées en égalités, en introduisant une variable d'écart à chaque contrainte $\forall c \in [\![1; C]\!], s_c \geqslant 0$:

$$\begin{aligned} & \max_{x_i} \sum_{i=1}^{N} c_i x_i \\ s.c : \quad & \sum_{i=1}^{m} A_{i,c} x_i + s_c = b_c, \ \forall c \in [\![1; C]\!] \\ & \forall i \in [\![1, N]\!], \ x_i \in \{0, 1\} \\ & \forall c \in [\![1; C]\!], \ s_c \geqslant 0 \end{aligned} \tag{4}$$

Pourquoi l'optimisation linéaire ?

- Optimisation continue sans contrainte : déjà qqch de difficile. Algorithmes de gradient s'appliquent pour des fonctions convexes, sinon donnent un optimum local qui peut être très différent de l'optimum global.
- Optimisation continue avec contrainte : algorithmes de gradients s'étendent, mais induisent encore plus de difficultés numériques
- Optimisation linéaire : de bons algorithmes (complexité polynomiale). Le célèbre algorithme du simplexe peut être vu comme une spécialisation d'algorithme de gradient projeté.
- PLNE : classe de problèmes NP-difficiles, mais modélisent de nombreux problèmes industriels (même non linéaires à première vue), solveurs de résolution générique qui font l'objet de travaux et d'une concurrence acharnée entre solveurs, grands impacts financiers.

Bilan partiel

- Optimisation discrète/continue/mixte.
- Optimisation sans/sous contrainte
- Optimisation linéaire/non-linéaire.
- Optimisation déterministe/stochastique.
- Optimisation structurée vs non-structurée

Plan

Motivations

- La théorie de la complexité des algorithmes est un pan fondamental de l'informatique.
- Vous avez déjà entendu parler de problèmes des classes P et $\mathcal{NP}$ et du prix du Millénaire visant à prouver que $\mathcal{P} = \mathcal{NP}$ ou $\mathcal{P} \neq \mathcal{NP}$?
- Des cours complets définissent le concept de Machine de Turing pour définir ces notions de manière rigoureuse (une UE entière)
- Ici, on donne juste l'intuition de ces définitions, et des éléments pour s'y repérer.
- Complexité définie pour des problèmes de décision, mais aussi étendue pour des problèmes d'optimisation, donne des repères dans le cadre de ce cours.

Complexité d'un algorithme, comparaisons asymptotiques

- Question : si je donne à un programme une entrée de taille N, quel est l'ordre de grandeur, en fonction de N, du temps de terminaison/du nombre d'opérations qu'il va effectuer ?
- Si on note T_N le temps de terminaison pour une donnée d'entrée de taille N, quel comportement asymptotique pour T_N lorsque $N \to +\infty$?
- Classification sur des fonctions de références simples.
- Utilisation des notations de Landau de comparaison asymptotiques
- N.B : la complexité d'un algorithme mesure l'évolution asymptotique de ressources limitées, souvent le temps et l'espace mémoire.

Exemple "big data" du plus proche voisin

- On a n points (x_n) de $\mathbb{R}^d$ sans aucune structure donnée.
- On a un point donné y de $\mathbb{R}^d$. On veut déterminer quel est le point x_n le plus proche de y ?
- Algorithme : énumérer toutes les n possibilités, et garder en mémoire le point le plus proche trouvé.
- Quelle est la complexité de cet algorithme ?

Exemple "big data" du plus proche voisin

- On a n points (x_n) de $\mathbb{R}^d$ sans aucune structure donnée.
- On a un point donné y de $\mathbb{R}^d$. On veut déterminer quel est le point x_n le plus proche de y ?
- Algorithme : énumérer toutes les n possibilités, et garder en mémoire le point le plus proche trouvé.
- A chaque étape, on fait un calcul de distance, une comparaison, et au plus une affectation pour modifier la solution courante.
- Si A est le temps nécessaire à un calcul de distance et une comparaison et B le temps d'une affectation :

$$An \leqslant T_n \leqslant (A+B)n$$

Notations de Landau, grand O, petit o et équivalents

$f(n) \in o(g(n))$ ou $f(n) = o(g(n))$, "f est négligeable devant g asymptotiquement" :

$$\forall \varepsilon > 0 \; \exists n_0 \; \forall n > n_0 \; |f(n)| \leqslant |g(n)| \cdot \varepsilon$$

$f(n) \sim g(n)$, "f et g sont équivalents asymptotiquement" :

$$\forall \varepsilon > 0 \; \exists n_0 \; \forall n > n_0 \; |f(n) - g(n)| \leqslant |g(n)| \cdot \varepsilon$$

$f(n) \in O(g(n))$ ou $f(n) = O(g(n))$, "f est bornée par g asymptotiquement" : si

$$\exists k > 0, \exists n_0 \; \forall n > n_0 \; |f(n)| \leqslant |g(n)| \cdot k$$

Notations de Landau, grand Omega et grand Theta

$f(n) \in \Omega(g(n))$, "f est minorée par g asymptotiquement" :

$$\exists k > 0, \exists n_0 \; \forall n > n_0 \; |g(n)| \cdot k \leqslant |f(n)|$$

$f(n) \in \Theta(g(n))$, "f est de l'ordre de g asymptotiquement" :

$$\exists k_1, k_2 > 0, \exists n_0 \; \forall n > n_0 \; |g(n)| \cdot k_1 < |f(n)| < |g(n)| \cdot k_2$$

$\Longrightarrow$ Pour classifier des algorithmes selon leur comportement asymptotique, on cherche une fonction usuelle simple a_n, avec $T_n \in \Theta(a_n)$.

Relations

$f(n) \sim g(n) \Longrightarrow f(n) \in \Theta(g(n)) \Longrightarrow f(n) \in O(g(n))$

$f(n) \in o(g(n)) \Longrightarrow f(n) \in O(g(n))$

$f(n) \sim g(n) \Longleftrightarrow |f(n) - g(n)| \in o(g(n)) \Longleftrightarrow |f(n) - g(n)| \in o(f(n))$

$f(n) \in \Omega(g(n)) \Longleftrightarrow g(n) \in O(f(n))$

$f(n) \in \Theta(g(n)) \Longleftrightarrow f(n) \in O(g(n))$ et $f(n) = \Omega(g(n))$

$f(n) \in \Theta(h(n))$ et $g(n) \in \Theta(h(n)) \Longrightarrow f(n) + g(n) \in \Theta(h(n))$ $f(n) \in O(h(n))$ et $g(n) \in O(h(n)) \Longrightarrow f(n) + g(n) \in O(h(n))$

$\Longrightarrow$ Permet de simplifier les calculs de complexité sans avoir à expliciter les temps de calculs en fonctions des temps requis par les opérations élémentaires

Remarque : comportement asymptotique vs constantes

L'étude de comportement asymptotique donne les phases clés de l'algorithme sur les temps de calculs

L'analyse peut mettre en avant que des étapes ont des temps de calculs négligeables devant d'autres

Regarder le facteur multiplicatif peut avoir une influence en pratique (un parcours vs deux parcours d'un conteneur, même comportement asymptotique, mais facteur multiplicatif a son importance en pratique)

Ex1 : accès en mémoire RAM ou dans le cache induit une grosse différence de facteur multiplicatif.

Ex2 : parallélisation (massive) peut diminuer les facteurs multiplicatifs pour un gain pratique appréciable.

Algorithme polynomial et notations de Landau

Si on s'intéresse juste au caractère au plus polynomial d'un algorithme :

- $T_n \in O(a_n)$ avec $\frac{a_n}{n^k} \to 0$ pour $n \to +\infty$ et un entier k donné, suffit pour garantir l'appartenance à P.
- $T_n \in \Omega(a_n)$ avec $\frac{a_n}{n^k} \to +\infty$ pour $n \to +\infty$ pour tout entier k suffit pour garantir l'algorithme considéré est non polynomial.

Exemples de complexité

- Pour trier un tableau de taille N, le tri bulle tourne en temps $O(N^2)$, en utilisant un espace mémoire additionnel en $O(1)$.
- Pour trier un tableau de taille N, le tri fusion tourne en temps $O(N \log N)$, en utilisant un espace mémoire additionnel en $O(N)$.
- Plus court chemin dans un graphe $G = (V, E)$ avec poids (distances) positifs, algorithme de Dijkstra en temps $O((|E| + |V|) \log |V|)$.
- Arbre couvrant de poids minimal dans un graphe valué $G = (V, E)$, algorithme de Prim-Kruskal en temps $O(|V| \log |V|)$.

Complexité d'un problème de décision

- Problème de décision : question mathématique dont la réponse est "oui" ou "non".
- Théorie de la complexité, on classifie sur la distinction polynomial/non polynomial.
- Machine de Turing déterministe : "modèle théorique d'ordinateur"
- Problème $\mathcal{P}$: Problème où la décision est calculée (et prouvée) en temps polynomial de la taille des données sur une machine de Turing déterministe.
- Machine de Turing non déterministe : "machine capable d'exécuter en parallèle un nombre fini d'alternatives" (fini mais grand potentiellement)
- Problème $\mathcal{NP}$: Problème où la décision est calculée en temps polynomial de la taille des données sur une machine de Turing non-déterministe.

Complexité d'un problème de décision, "intuitivement"

- Problème de décision : question mathématique dont la réponse est "oui" ou "non".
- Si on a un algorithme polynomial permettant de répondre à toute instance du problème de décision, le problème est dans $\mathcal{P}$.
- Si en ayant un "certificat", ie la solution, on a un algorithme polynomial permettant de vérifier que le certificat est valide, est dans $\mathcal{NP}$.
- $\mathcal{P} \subset \mathcal{NP}$ mais on ne sait pas démonter à ce jour si $\mathcal{P} = \mathcal{NP}$ (peu probable) ou s'il existe des problèmes de $\mathcal{NP}$ qui ne sont pas polynomiaux.

Test de primalités (1)

- Problème de décision : un entier $n > 1$ donné est t'il premier ?
- Algorithme naïf : tester tous les entiers entre 2 et $\sqrt{n}$ si ils divisent n.
- Opération de division euclidienne en $\log n$, répétées au plus $O(\sqrt{n})$ fois, cela fait un temps total en temps total en $O(\sqrt{n}\log n)$, c'est même un $o(n)$.

$\Longrightarrow$ Le problème de la primalité est-il polynomial ?

Test de primalités (2)

- Problème de décision : un entier $n > 1$ donné est t'il premier ?
- Taille des données : $\log n$, l'entier n est codé sur $\log n$ bits

  Temps total précédent en $O(\sqrt{n} \log n)$, en $O(n)$ mais pas en $O(\log^k n)$.
- L'algorithme naïf n'est pas polynomial, parce qu'il n'est pas polynomial en la taille en mémoire des données d'entrée, qui est ici $\log n$.
- Ce problème est dans $\mathcal{NP}$, si on a un certificat, un produit de deux entiers $n_1, n_2 < n$ avec $n_1 \times n_2 = n$, vérifier la validité du certificat demande des comparaisons en $O(1)$ et le calcul du produit en $O(\log n^2)$.

Test de primalités (3)

La fin de l'histoire, le test de primalité est bien dans la classe $\mathcal{P}$:

- Pour un entier donné n, l'algorithme AKS indique la primalité ou non de n en $O(\log n^{7,5})$:

  Agrawal, M., Kayal, N., Saxena, N. (2004). PRIMES is in P. Annals of mathematics, 781-793.

- Algorithme inefficace en pratique ! Complexité asymptotique et efficacité pratique sur des tailles de données finies sont deux choses bien différentes !

- En pratique, le test de primalité de Miller Rabin serait déterministe, avec une complexité en $O(\log^6 n)$, si la conjecture de Riemann généralisée est prouvée. Il existe des versions probabilistes plus rapides, avec une borne probabiliste sur le taux de faux-positifs.

$\mathcal{P} \neq \mathcal{NP}$, hypothèse de Riemann généralisée, on voit surgir deux problèmes "mythiques" des maths modernes

A bien retenir

Complexité : on regarde, la complexité par rapport à la taille des données d'entrée du problème.

On commence par chercher un représentant simple tel que la taille d'entrée du problème soit en $\Theta(a_n)$.

La taille d'entrée peut dépendre de la représentation. Ex : pour un graphe $G = (V, E)$ avec un encodage binaire, $|V|^2$ booléens pour tout graphe, espace en $\Theta(N^2)$, mais on peut avoir un encodage en $\Theta(|E|)$, utile pour des graphes peu denses.

Alors si le temps de calcul est en $O(a_n^k)$ pour un certain $k \in \mathbb{N}$, le problème est bien polynomial

idem pour caractérisation linéaire ou quadratique d'un algorithme.

Notations de Landau, algorithmes vs problèmes

A un algorithme donné, on peut étudier assez facilement la complexité et élaborer des O et Θ pour le temps de terminaison en général.

Un problème se résout en $O(a_n)$, s'il existe un algorithme valide pour résoudre toute instance du problème avec une terminaison en temps $O(a_n)$.

Un problème peut avoir une borne en $\Omega(a_n)$ si tout un algorithme valide termine en $\Omega(a_n)$ en considérant toutes les instances du problème (plus dur à prouver).

Un problème se résout en $\Theta(a_n)$ si a_n est la meilleure complexité asymptotique prouvée pour résoudre le problème.

Exemple pour le tri d'un tableau de taille N :
- le tri bulle trie un tableau en $O(N^2)$
- le tri fusion trie un tableau en $O(N \log N)$
- vérifier qu'un tableau est trié se vérifie en $\Theta(N)$
- le tri d'un tableau quelconque avec certaines hypothèses est en $\Omega(N \log N)$

Remarque : Complexité vs efficacité pratique

- Pendant longtemps, la classification polynomial/exponentiel était considérée comme une barrière pour l'application pratique.
- Des algorithmes polynomiaux peuvent être inutilisables en pratique sur de grandes tailles : espace mémoire ou degré trop grand. (ex : AKS)
- En pratique, algorithmes de complexité linéaire ou mieux logarithmique peuvent considérés comme utilisables en pratique.
- De nombreux progrès pratiques d'algorithmes d'optimisation exponentiels engendrent un domaine de plus en plus large où ils sont utilisables.

Ordre de grandeur du temps nécessaire à l'exécution d'un algorithme d'un type de complexité

Temps	Type de complexité	Temps pour n = 5	Temps pour n = 10	Temps pour n = 20	Temps pour n = 50	Temps pour n = 250	Temps pour n = 1 000	Temps pour n = 10 000	Temps pour n = 1 000 000	Problème exemple
$O(1)$	complexité constante	10 ns	10 ns	10 ns	10 ns	10 ns	10 ns	10 ns	10 ns	Accès tableaux
$O(\log(n))$	complexité logarithmique	10 ns	10 ns	10 ns	20 ns	30 ns	30 ns	40 ns	60 ns	Recherche dichotomique
$O(\sqrt{n})$	complexité racinaire	22 ns	32 ns	45 ns	71 ns	158 ns	316 ns	1 µs	10 µs	
$O(n)$	complexité linéaire	50 ns	100 ns	200 ns	500 ns	2.5 µs	10 µs	100 µs	10 ms	Parcours de liste
$O(n\log(n))$	complexité linéarithmique	40 ns	100 ns	260 ns	850 ns	6 µs	30 µs	400 µs	60 ms	Tris dont le Tri fusion ou le Tri par tas
$O(n^2)$	complexité quadratique (polynomiale)	250 ns	1 µs	4 µs	25 µs	625 µs	10 ms	1 s	2.8 heures	Parcours de tableaux 2D
$O(n^3)$	complexité cubique (polynomiale)	1.25 µs	10 µs	80 µs	1.25 ms	156 ms	10 s	2.7 heures	316 ans	Multiplication matricielle non-optimisée.
$2^{\text{poly}(\log(n))}$	complexité sous-exponentielle	30 ns	100 ns	492 ns	7 µs	5 ms	10 s	3.2 ans	10^{20} ans	
$2^{\text{poly}(n)}$	complexité exponentielle	320 ns	10 µs	10 ms	130 jours	10^{59} ans	...	...	...	Problème du sac à dos par force brute.
$O(n!)$	complexité factorielle	1.2 µs	36 ms	770 ans	10^{48} ans	...	...	...	...	Problème du voyageur de commerce (avec une approche naïve).

$\mathcal{P} = \mathcal{NP}$? Problèmes $\mathcal{NP}$-complets et $\mathcal{NP}$-difficiles

- $\mathcal{P} \subset \mathcal{NP}$ mais l'inclusion ou la non-inclusion réciproque font toujours l'objet de conjectures.
- Problèmes $\mathcal{NP}$-complets : Sous ensemble de problèmes de $\mathcal{NP}$, tel que si on démontre qu'un de ces problèmes est dans $\mathcal{P}$, cela implique $\mathcal{P} = \mathcal{NP}$. (problèmes se ramenant entre eux par transformation polynomiale)
- Problèmes NP-difficiles : Problèmes pas forcément dans $\mathcal{NP}$, mais "aussi difficiles" que les problèmes $\mathcal{NP}$-complets : si on démontre qu'un des problèmes $\mathcal{NP}$-difficiles est dans $\mathcal{P}$, cela implique $\mathcal{P} = \mathcal{NP}$.
- Il existe des problèmes de références prouvés $\mathcal{NP}$ complets : SAT, 3-SAT, les premiers a voir été prouvés $\mathcal{NP}$-complets. (Théorème de Cook)

Problèmes SAT et 3-SAT

Problèmes SAT : problème de satisfaction booléennes, à partir d'une expression faisant intervenir N booléens $x_1, \ldots, x_N$ et leur négation $\overline{x}_1, \ldots, \overline{x}_N$, écrite sous forme normale conjonctive (FNC), existe t'il une affectation des variables telle que l'expression est vraie ?

Table de vérité : 2^N cas pour tout énumérer. Question, a t'on un algorithme polynomial ? SAT est le premier problème $\mathcal{NP}$-complet

3-SAT, problème de satisfaction booléennes, avec uniquement des termes de degré 3 dans la FNC (exactement ou au plus, cela revient au même, transformation polynomiale) 3-SAT est $\mathcal{NP}$-complet

2-SAT, avec une FNC de degré 2, définit un problème polynomial

Comment prouver qu'un problème est $\mathcal{NP}$-complet ?

- Il existe des problèmes de références prouvés $\mathcal{NP}$-complets.
- Pour prouver qu'un problème $\mathcal{P}_{new}$ est $\mathcal{NP}$-complet, on se ramène à un problème de références $\mathcal{NP}$-complets $\mathcal{P}_{ref}$:
  - on exhibe une transformation polynomiale f entre toute instance du problème $\mathcal{P}_{new}$ en une instance de $\mathcal{P}_{ref}$, $f : i \in \mathcal{P}_{new} \mapsto f(i) \in \mathcal{P}_{ref}$, $f(i)$ de taille polynomiale en i
  - On prouve que i vrai dans $\mathcal{P}_{new} \iff f(i)$ vrai $f(c)$ dans $\mathcal{P}_{ref}$
  - On prouve que la transformation d'une solution de $f(\mathcal{P}_{new})$ dans $\mathcal{P}_{ref}$ à une solution de $\mathcal{P}_{new}$ est polynomiale
- Preuve : si on a un algorithme polynomial pour résoudre le nouveau problème, cela impliquerait que le problème $\mathcal{NP}$-complet de référence se résout de manière polynomiale.

$\Longrightarrow$ On parle alors de réduction à un problème de référence

Des problèmes $\mathcal{NP}$ qui ne sont pas $\mathcal{NP}$-complets ?

$\mathcal{P} \neq \mathcal{NP}$ est une conjecture fondamentale en théorie de la complexité

Autre conjecture : existe-t'il des problèmes qui soient dans $\mathcal{NP}$ sans être $\mathcal{NP}$-complets ?

Pour plusieurs problèmes $\mathcal{NP}$, on ne sait pas prouver à l'heure actuelle s'ils sont dans $\mathcal{P}$ ou $\mathcal{NP}$-complets :

- Problème de factorisation : donner la factorisation d'un entier en produit de facteurs premiers
- Problème de l'isomorphisme de graphes : deux graphes donnés sont ils identiques à une ré-indexation près des sommets ?

Complexité d'un problème d'optimisation

- Pour un problème de minimisation, les caractérisations $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NP}$-difficiles sont associées au problème de décision :

  Pour une instance donnée du problème et une valeur k donnée, existe-t-il une solution du problème de coût $\leqslant k$?

- Vérification $\mathcal{NP}$: ayant la solution et le coût, un calcul polynomial du coût permet de vérifier qu'on est dans $\mathcal{NP}$.

- Si l'optimum se calcule constructivement en temps polynomial, le problème est dans $\mathcal{P}$. (hypothèse plus forte)

- En pratique, on regarde si un problème d'optimisation donné a une résolution polynomiale

- Le fait qu'il soit $\mathcal{NP}$-difficile de déterminer s'il existe une solution réalisable à un problème d'optimisation sous contrainte est plus fort (implique) que le problème d'optimisation est $\mathcal{NP}$-difficile.

Complexité polynomiale de problèmes d'optimisation

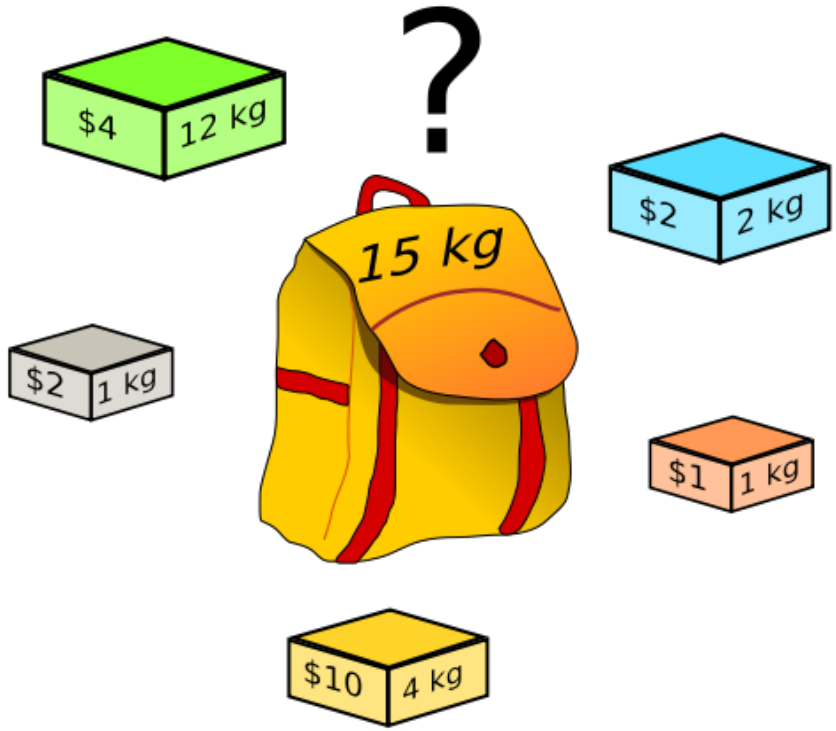
- Si une solution optimale d'un problème d'optimisation se construit en temps polynomial, le problème est dans $\mathcal{P}$.
- Face à un problème d'optimisation, la première chose à regarder est la complexité $\mathcal{P}$ ou $\mathcal{NP}$-complet, l'algorithme de Branch&Bound n'est pas à utiliser pour des problèmes polynomiaux, un algorithme dédié polynomial sera plus efficace. Ecrire un problème en un PLNE de taille polynomiale permet juste de dire que le problème est $\mathcal{NP}$.
- La programmation dynamique est un outil puissant pour démontrer l'existence d'un algorithme de résolution polynomiale.
- Un algorithme glouton peut être optimal, cf théorie des matroïdes, pour montrer que des classes de problèmes sont dans $\mathcal{P}$:

https://www.gerad.ca/~alainh/Matroides.pdf

https://www.lirmm.fr/~gioan/telecharger/2013%20Gioan%20Ramirez-Alfonsin%20-%20Cours%20sur%20les%20matroides%20(orientes)%20-%20GDRIM2013.pdf

Plan

Problème de sac à dos



- On dispose de N objets, de masses en kg (m_i) et de valeur financière (c_i).
- On dispose d'un sac à dos pouvant porter jusqu'à M kg.
- Problème : Remplir le sac à dos en maximisant la valeur financière contenue dans le sac à dos.

Problème de sac à dos

Objet	A	B	C	D	E	F	G	H
Masse	2	3	5	2	4	6	3	1
Prix	5	8	14	6	13	17	10	4

Contenance du sac à dos : 12kg

Comment formuler le problème de sac à dos ?

- Variables : $x_A, \ldots, x_H \in \{0, 1\}$, $x_A = 1$ si on choisit l'objet A, 0 sinon.
- Comment peut on écrire le prix du contenu du sac à dos ?
- Comment peut on écrire la masse du sac à dos ?

Objet	A	B	C	D	E	F	G	H
Masse	2	3	5	2	4	6	3	1
Prix	5	8	14	6	13	17	10	4

Contenance du sac à dos : 12kg

1. Formulation de la partie "masse".
   - Si on prend uniquement l'objet A, le sac pèsera 2 kg (et $x_A = 1$) sinon 0, $\rightarrow$ dans ces deux cas, le sac pèse $2x_A$ kg.

Objet	A	B	C	D	E	F	G	H
Masse	2	3	5	2	4	6	3	1
Prix	5	8	14	6	13	17	10	4

Contenance du sac à dos : 12kg

1. Formulation de la partie "masse".
   - Si on prend uniquement l'objet A, le sac pèsera 2 kg (et $x_A = 1$) sinon 0, $\rightarrow$ dans ces deux cas, le sac pèse $2x_A$ kg.
   - Idem avec l'objet B, la contribution de B dans le poids du sac est : $3x_B$ kg $\rightarrow$ La contribution des objets A et B dans le poids du sac à dos est additive : $2x_A + 3x_B$ kg .

     ...

Objet	A	B	C	D	E	F	G	H
Masse	2	3	5	2	4	6	3	1
Prix	5	8	14	6	13	17	10	4

Contenance du sac à dos : 12kg

1. Formulation de la partie "masse".
   - Si on prend uniquement l'objet A, le sac pèsera 2 kg (et $x_A = 1$) sinon 0, $\rightarrow$ dans ces deux cas, le sac pèse $2x_A$ kg.
   - Idem avec l'objet B, la contribution de B dans le poids du sac est : $3x_B$ kg $\rightarrow$ La contribution des objets A et B dans le poids du sac à dos est additive : $2x_A + 3x_B$ kg .

     ...
   - Au final, la masse du sac à dos vaut :
     $2x_A + 3x_B + 5x_C + 2x_D + 4x_E + 6x_F + 3x_G + x_H$

     $\Longrightarrow$ On a la contrainte :
     $2x_A + 3x_B + 5x_C + 2x_D + 4x_E + 6x_F + 3x_G + x_H \leqslant 12$

Formulation du sac à dos en optimisation linéaire

Objet	A	B	C	D	E	F	G	H
Masse	2	3	5	2	4	6	3	1
Prix	5	8	14	6	13	17	10	4

Contenance du sac à dos : 12kg

2. Formulation de la partie "prix".
   - Si on prend uniquement l'objet A, le sac vaudra 5 €(et $x_A = 1$) sinon 0 $\rightarrow$ dans ces deux cas, le sac vaut $5x_A$ €.

Formulation du sac à dos en optimisation linéaire

Objet	A	B	C	D	E	F	G	H
Masse	2	3	5	2	4	6	3	1
Prix	5	8	14	6	13	17	10	4

Contenance du sac à dos : 12kg

2. Formulation de la partie "prix".
   - Si on prend uniquement l'objet A, le sac vaudra 5 €(et $x_A = 1$) sinon 0
     $\rightarrow$ dans ces deux cas, le sac vaut $5x_A$ €.
   - Idem, avec uniquement l'objet B, le sac vaudra $8x_B$ €
     $\rightarrow$ La contribution des objets A et B dans le poids du sac à dos est additive : $5x_A + 8x_B$ €.
     ...

Formulation du sac à dos en optimisation linéaire

Objet	A	B	C	D	E	F	G	H
Masse	2	3	5	2	4	6	3	1
Prix	5	8	14	6	13	17	10	4

Contenance du sac à dos : 12kg

2. Formulation de la partie "prix".
   - Si on prend uniquement l'objet A, le sac vaudra 5 €(et $x_A = 1$) sinon 0
     $\rightarrow$ dans ces deux cas, le sac vaut $5x_A$ €.
   - Idem, avec uniquement l'objet B, le sac vaudra $8x_B$ €
     $\rightarrow$ La contribution des objets A et B dans le poids du sac à dos est additive : $5x_A + 8x_B$ €.
     ...
   - Au final, le sac à dos vaut :
     $5x_A + 8x_B + 14x_C + 6x_D + 13x_E + 17x_F + 10x_G + 4x_H$

Formulation du sac à dos en optimisation linéaire

Objet	A	B	C	D	E	F	G	H
Masse	2	3	5	2	4	6	3	1
Prix	5	8	14	6	13	17	10	4

Contenance du sac à dos : 12kg

Formellement, on peut écrire ce problème comme un problème d'optimisation sous contrainte :

$$
\begin{array}{ll}
\max & 5x_A + 8x_B + 14x_C + 6x_D + 13x_E + 17x_F + 10x_G + 4x_H \\
s.c : & 2x_A + 3x_B + 5x_C + 2x_D + 4x_E + 6x_F + 3x_G + x_H \leqslant 12 \\
\forall i \in \{A, \ldots, H\} & x_i \in \{0, 1\}
\end{array}
$$

Quel sens mathématique ?

Quand on écrit

$$\begin{array}{ll} \max & 5x_A + 8x_B + 14x_C + 6x_D + 13x_E + 17x_F + 10x_G + 4x_H = f(x) \\ s.c : & 2x_A + 3x_B + 5x_C + 2x_D + 4x_E + 6x_F + 3x_G + x_H \leqslant 12 \\ \forall i \in \{A, \ldots, H\} & x_i \in \{0, 1\} \end{array}$$

C'est un raccourci pour dire que l'on cherche à calculer le maximum de l'application définie comme suit :

$$\varphi : x \in K = \{x \in \{0,1\}^8 \mid g(x) \leqslant 12\} \subset \mathbb{R}^8 \longmapsto f(x) \in \mathbb{R}$$
avec $g(x) = 2x_A + 3x_B + 5x_C + 2x_D + 4x_E + 6x_F + 3x_G + x_H$

C'est bien défini : le maximum d'une fonction continue définie sur compacte, pas de problème d'existence mathématique !

En prenant $K' = \{x \in [0,1]^8 \mid g(x) \leqslant 12\}$, cela définit également un problème d'optimisation valide par cet argument.

Remarque : K est un ensemble fini, ça assure l'existence. La compacité est l'argument nécessaire pour prouver qu'on a un problème bien définira vec K' à la place de K, K' est infini et non dénombrable

Formulation PLNE du problème de sac à dos

De manière générale :

- On dispose de N objets, de masses en kg (m_i) et de valeur financière (c_i).
- On dispose d'un sac à dos pouvant porter jusqu'à M kg.

Remplir le sac à dos en maximisant la valeur financière contenue dans le sac à dos se formule comme le problème d'optimisation sous contrainte :

$$\begin{aligned} & \max_{x_i} \sum_{i=1}^{N} c_i x_i \\ s.c : \quad & \sum_{i=1}^{N} m_i x_i \leqslant M \\ \forall i \in [\![1, N]\!], \quad & x_i \in \{0, 1\} \end{aligned} \tag{5}$$

Cad, on cherche à calculer le maximum de l'application :

$$\varphi : K = \{x \in \{0,1\}^N \mid \sum_{i=1}^{N} m_i x_i \leqslant M\} \subset \mathbb{R}^N \to \varphi(x) = \sum_{i=1}^{N} c_i x_i \in \mathbb{R}$$

Réciproquement

Réciproquement, est dit problème de sac à dos tout problème d'optimisation qui s'écrit sous la forme :

$$\begin{aligned} & \max_{x_i} \sum_{i=1}^{N} c_i x_i \\ s.c : \quad & \sum_{i=1}^{N} m_i x_i \leqslant M \\ \forall i \in [\![1, N]\!], \quad & x_i \in \{0, 1\} \end{aligned} \tag{6}$$

avec $m_i \geqslant 0$ pour tout $i \in [\![1, N]\!]$

Réciproquement

Réciproquement, est dit problème de sac à dos tout problème d'optimisation qui s'écrit sous la forme :

$$\begin{aligned} & \max_{x_i} \sum_{i=1}^{N} c_i x_i \\ s.c : \quad & \sum_{i=1}^{N} m_i x_i \leqslant M \\ \forall i \in [\![1, N]\!], \quad & x_i \in \{0, 1\} \end{aligned} \tag{6}$$

avec $m_i \geqslant 0$ pour tout $i \in [\![1, N]\!]$

Remarque 1 : Pour que des solutions existent et que le problème soit réalisable, il faut que $M \geqslant 0$

Remarque 2 : si on a des indices avec $c_i \leqslant 0$, on ne prendra jamais l'objet, on peut retirer de tels objets et avoir $c_i \geqslant 0$ pour tout $i \in [\![1, N]\!]$.

Problème de sac à dos continu

Est dit problème de *sac à dos continu* tout problème d'optimisation qui s'écrit sous la forme :

$$\begin{aligned} & \max_{x_i} \sum_{i=1}^{N} c_i x_i \\ s.c : \quad & \sum_{i=1}^{N} m_i x_i \leqslant M \\ \forall i \in [\![1, N]\!], \quad & x_i \in [0, 1] \end{aligned} \tag{7}$$

avec $m_i \geqslant 0$ pour tout $i \in [\![1, N]\!]$

Cad, on cherche à calculer le maximum de l'application :

$$\varphi : x \in K = \left\{ x \in [0,1]^N \;\middle|\; \sum_{i=1}^{N} m_i x_i \leqslant M \right\} \subset \mathbb{R}^N \longmapsto \varphi(x) = \sum_{i=1}^{N} c_i x_i \in \mathbb{R}$$

Sac à dos continu

- Problème de sac à dos continu : correspond au cas où on a des objets sécables (ex : beurre, quantités vendues au poids au détail)
- Problème d'optimisation sur un ensemble infini, argument de compacité permet d'en garantir l'existence
- Problèmes se calculant en temps $O(N \log N)$ par un algorithme glouton, plus faciles que les sacs à dos discrets, NP-complet. (on le verra)

Autre situation

Une usine automatisée de production de masques FFP2 dispose de 4 lignes de productions A, B, C, D, qui ont chacune un coût de fonctionnement journalier et une capacité de production journalière définis ci dessous

Ligne de prod	A	B	C	D
Coût de prod	2	6	7	2
Capacité de prod	3	4	5	2

L'usine reçoit une commande de 10 containers de masques pour un jour donné.

Comment répartir la production pour minimiser les coûts ? ie comment décider quelle(s) ligne(s) mettre en service, et quelle(s) autre(s) resteront arrêtée(s) ?

Autre situation

Ligne de prod	A	B	C	D
Coût de prod	2	6	7	2
Capacité de prod	3	4	5	2

A produire : 10 containers de masques.

On définit les variables : $x_A, x_B, x_C, x_D \in \{0,1\}$, $x_A = 1$ si la ligne A est mise en service, 0 sinon, ...

Cela s'écrit comme un problème d'optimisation sous contrainte :

$$\begin{array}{ll} OPT = \min & 2x_A + 6x_B + 7x_C + 2x_D \\ s.c : & 3x_A + 4x_B + 5x_C + 2x_D \geqslant 10 \\ & x_A, x_B, x_C, x_D \in \{0,1\} \end{array}$$

Est-ce un problème de sac à dos ?

Knapsack or not knapsack, that's the question

Stricto sensu, non, le sac à dos, c'est maximiser une fonction, avec une contrainte d'inégalité de budget maximal.

Si on définit les variables : $y_A, y_B, y_C, y_D \in \{0,1\}$, $y_A = 0$ si la ligne A est mise en service, 1 sinon, ...
On a en fait défini $y_A = 1 - x_A$. On a le problème d'optimisation sous contrainte :

$$\begin{array}{ll} OPT = \min & 2(1-y_A) + 6(1-y_B) + 7(1-y_C) + 2(1-y_D) \\ s.c : & 3(1-y_A) + 4(1-y_B) + 5(1-y_C) + 2(1-y_D) \geqslant 10 \\ & y_A, y_B, y_C, y_D \in \{0,1\} \end{array}$$

On remplace, et on a

$$\begin{array}{lll} OPT = \min & 17 - 2y_A - 6y_B - 7y_C - 2y_D \\ s.c : & -3y_A - 4y_B - 5y_C - 2y_D \geqslant 10 - 14 \\ & y_A, y_B, y_C, y_D \in \{0, 1\} \end{array}$$

ce qui équivaut à résoudre le problème de sac à dos suivant :

$$\begin{array}{lll} OPT_2 = 17 - OPT = \max & 2y_A + 6y_B + 7y_C + 2y_D \\ s.c : & 3y_A + 4y_B + 5y_C + 2y_D \leqslant 4 \\ & y_A, y_B, y_C, y_D \in \{0, 1\} \end{array}$$

On récupère le résultat du problème initial avec
$OPT = 17 - OPT_2$ et on renvoie comme affectation $x_i = 1 - y_i$

Plus généralement

$$\begin{aligned} & \min_{x_i} \sum_{i=1}^{N} c_i x_i \\ s.c : \quad & \sum_{i=1}^{N} m_i x_i \geqslant M \\ \forall i \in [\![1, N]\!], \quad & x_i \in \{0, 1\} \end{aligned} \tag{8}$$

avec $m_i \geqslant 0$ pour tout $i \in [\![1, N]\!]$, revient à résoudre le problème de sac à dos suivant :

$$\begin{aligned} & \max_{y_i} \sum_{i=1}^{N} c_i y_i \\ s.c : \quad & \sum_{i=1}^{N} m_i y_i \leqslant \sum_{i=1}^{N} m_i - M \\ \forall i \in [\![1, N]\!], \quad & y_i \in \{0, 1\} \end{aligned} \tag{9}$$

On récupère le résultat du problème initial en retirant $\sum_{i=1}^{N} c_i$ à l'objectif et on renvoie comme affectation $x_i = 1 - y_i$

Retour du directeur d'usine

Notre directeur d'usine revient, il a un problème, avec des solutions fournies par l'optimisation précédente, ses lignes d'assemblages consomment parfois trop d'électricité !

Il avait oublié de nous dire que ses 4 lignes de production ont une consommation électrique donnée, et qu'il ne peut pas dépasser un certain seuil sous peine d'avoir de grosses pénalités financières ou un black-out.

Ses 4 lignes de productions A, B, C, D, ont également une consommation électrique associée ci dessous, et on ne peut pas dépasser un seuil de 1200 kWh.

Ligne de prod	A	B	C	D
Coût de prod	2	6	7	2
Capacité de prod	3	4	5	2
Conso énergétique	450	450	500	200

A produire : 10 containers de masques, sans dépasser 1200 kWh de consommation électrique.

Comment peut on s'en sortir ?

Retour du directeur d'usine

Ligne de prod	A	B	C	D
Coût de prod	2	6	7	2
Capacité de prod	3	4	5	2
Conso énergétique	450	450	500	200

A produire : 10 containers de masques, sans dépasser 1100 kWh.

On définit les variables : $x_A, x_B, x_C, x_D \in \{0,1\}$, $x_A = 1$ si la ligne A est mise en service, 0 sinon, ...

Cela s'écrit comme un problème d'optimisation sous contrainte :

$$\begin{array}{ll} \min & 2x_A + 6x_B + 7x_C + 2x_D \\ s.c: & 3x_A + 4x_B + 5x_C + 2x_D \geqslant 10 \\ & 450x_A + 450x_B + 500x_C + 200x_D \leqslant 1100 \\ & x_A, x_B, x_C, x_D \in \{0,1\} \end{array}$$

C'est juste rajouter une contrainte de consommation d'électricité maximale, une nouvelle contrainte de type sac-à-dos

Quel sens mathématique ?

Quand on écrit

$$\begin{array}{ll} \min & 2x_A + 6x_B + 7x_C + 2x_D \\ s.c: & 3x_A + 4x_B + 5x_C + 2x_D \geqslant 10 \\ & 450x_A + 450x_B + 500x_C + 200x_D \leqslant 1100 \\ & x_A, x_B, x_C, x_D \in \{0,1\} \end{array}$$

C'est un raccourci pour dire que l'on cherche à calculer le maximum de l'application définie comme suit :

$\varphi : x \in K \subset \mathbb{R}^4 \longmapsto f(x) = 2x_A + 6x_B + 7x_C + 2x_D \in \mathbb{R}$
avec
$K = \{x \in \{0,1\}^4 \mid g_1(x) \geqslant 10\} \bigcap \{x \in \{0,1\}^4 \mid g_2(x) \leqslant 1100\}$ avec
$g_1(x) = 3x_A + 4x_B + 5x_C + 2x_D$ et $g_2(x) = 450x_A + 450x_B + 500x_C + 200x_D$

C'est bien défini : le maximum d'une fonction continue définie sur compacte, pas de problème d'existence mathématique !

Remarque $K = \{x \in \{0,1\}^4 \mid g_1(x) \geqslant 10 \text{ et } g_2(x) \leqslant 1100\}$

En ce sens, les contraintes s'ajoutent indépendamment !

Quel sens mathématique ? (2)

Quand on écrit

$$\begin{array}{ll} \min & 2x_A + 6x_B + 7x_C + 2x_D \\ s.c: & 3x_A + 4x_B + 5x_C + 2x_D \geqslant 10 \\ & 450x_A + 450x_B + 500x_C + 200x_D \leqslant 1100 \\ & x_A, x_B, x_C, x_D \in \{0,1\} \end{array}$$

C'est équivalent à calculer le maximum de l'application définie comme suit :

$\varphi : x \in K = \{x \in \{0,1\}^4 \mid g(x) \leqslant 0\} \longmapsto f(x) = 2x_A + 6x_B + 7x_C + 2x_D \in \mathbb{R}$
avec

$$g(x) = \begin{pmatrix} 10 - (3x_A + 4x_B + 5x_C + 2x_D) \\ 450x_A + 450x_B + 500x_C + 200x_D - 1100 \end{pmatrix} \leqslant \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

L'écriture s'étend avec une fonction de contraintes de dimension 2, chaque contrainte correspond à une coordonnée

En ce sens, les contraintes s'ajoutent indépendamment !

Extension du problème de sac à dos (2)

Sac à dos multiple : le sac à dos peut porter jusqu'à M kg et a un volume maximal de V cm^3.

$$\begin{aligned} & \max_{x_i} \sum_{i=1}^{N} c_i x_i \\ s.c : \quad & \sum_{i=1}^{N} m_i x_i \leqslant M \\ s.c : \quad & \sum_{i=1}^{N} v_i x_i \leqslant V \\ \forall i \in [\![1, N]\!], \quad & x_i \in \{0, 1\} \end{aligned} \tag{10}$$

$\Longrightarrow$ On peut considérer simultanément plusieurs contraintes de dimensionnement, les contraintes s'ajoutent indépendamment

Retour du directeur d'usine (2)

Notre directeur d'usine, ayant bien maximisé ses profits, a acheté d'autres machines et ouverts d'autres lignes de productions plus modernes.

Problème : Pour des contraintes d'installation physique et électrique, la ligne E ne peut fonctionner que si la ligne A fonctionne, et les lignes D et F ne peuvent pas fonctionner simultanément

Ligne de prod	A	B	C	D	E	F
Coût de prod	2	6	7	2	3	4
Capacité de prod	3	4	5	2	6	6
Conso énergétique	450	450	500	200	350	250

A produire : 17 containers de masques, sans dépasser 1500 kWh de consommation électrique.

Peut on modéliser un tel problème ?

Retour du directeur d'usine (2)

Deux contraintes additionnelles :

- les lignes D et F ne peuvent pas fonctionner simultanément
on ne peut pas avoir $x_D = 1$ et $x_F = 1$
Ne serait-ce pas exactement la relation $x_D + x_F \leqslant 1$?

- la ligne E ne peut fonctionner que si la ligne A fonctionne
cad $x_E = 1$ doit impliquer $x_A = 1$
on ne peut pas avoir $x_E = 1$ et $x_A = 0$
Ne serait-ce pas exactement la relation $x_E \leqslant x_A$ ou $x_E - x_A \leqslant 0$?

Pour s'en convaincre, on peut vérifier les 4 cas comme une table de vérité

Retour du directeur d'usine

Ligne de prod	A	B	C	D	E	F
Coût de prod	2	6	7	2	3	4
Capacité de prod	3	4	5	2	6	6
Conso énergétique	450	450	500	200	350	250

A produire : 17 containers de masques, sans dépasser 1500 kWh de consommation électrique.

$$
\begin{array}{ll}
\min & 2x_A + 6x_B + 7x_C + 2x_D + 3x_E + 4x_F \\
s.c : & 3x_A + 4x_B + 5x_C + 2x_D + 6x_E + 6x_F \geqslant 17 \\
& 450x_A + 450x_B + 500x_C + 200x_D + 6x_E + 6x_F \leqslant 1500 \\
& x_E - x_A \leqslant 0 \\
& x_D + x_F \leqslant 1 \\
& x_A, x_B, x_C, x_D \in \{0, 1\}
\end{array}
$$

Plan

Bilan

- Un cadre général pour des problèmes d'optimisation : l'optimisation sous contrainte, avec un cas particulier notable : la programmation linéaire.
- On a fait apparaître naturellement la PLNE avec des problèmes de sac à dos.
- Enjeu : savoir calculer l'optimum parmi un nombre très grand, non énumérable de solutions.
- Théorie de la complexité : problèmes $\mathcal{NP}$-complet vs polynomiaux, une distinction nette. Enjeu à trouver des algorithmes de résolution polynomiaux
- Théorie de la complexité : un cadre primordial pour comprendre l'intérêt des développements de ce cours, on cherchera à avoir des méthodes les plus efficaces possibles pour des problèmes $\mathcal{NP}$-complet.

Pour la suite

Dans la suite du cours :

- Le cadre de la PLNE va être particulièrement étudié, comprendre sa richesse de modélisation.
- Les questions de complexité d'algorithmes va intervenir en fil rouge lorsqu'il s'agira de résoudre des problèmes d'optimisation.

Approfondissements en dehors du cadre de ce cours :

- théorie de la complexité : option algo et MPRI.
- algorithmes de gradients : un cas particulier en IA, la théorie générale en master de maths appliquées.
- Problèmes généraux de commande optimale et d'optimisation de forme : master maths appliquées.