Dans ce chapitre, on découvre les problèmes d'optimisation ainsi qu'une famille d'algorithmes parfois utilisée pour les résoudre : les algorithmes gloutons.

1 Problèmes d'optimisation

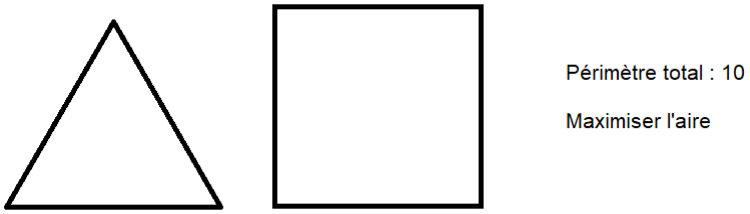
On appelle **problème d'optimisation** tout problème consistant à rechercher, parmi un certain ensemble de données E, la donnée $x^* \in E$ minimisant ou maximisant la valeur d'une certaine fonction $f(x)$.

On peut parler d'**optimisation sous contraintes** lorsqu'on impose en plus à x^* de vérifier des conditions particulière, autrement dit, on impose $x^* \in K \subseteq E$.

Exemples, en maths :

— Dans une usine, le coût de production par unité d'un certain produit en fonction du nombre total de produits fabriqués est donné pour $x \geq 0$ par $f(x) = -0.01(x-7)^2 + 12$. On recherche la quantité x^* à produire afin de minimiser le coût unitaire de production.

— On se donne deux figures, un carré et un triangle équilatéral, dont la somme des périmètres vaut 10. Trouver les valeurs des côtés des deux figures maximisant la somme des aires.



Exemples, en informatique :

— Problème du rendue de monnaie : on veut rendre une certaine somme d'argent S en n'utilisant que des pièces de valeurs fixées $\{p_0, ..., p_{n-1}\}$. Comment effectuer ce rendu en utilisant le moins de pièces possible ?

— Problème du sac à dos : on dispose d'un sac à dos dont le chargement ne peut pas dépasser un certain poids P ainsi que d'une collection d'objets à emporter, chacun des objet ayant un poids p et une valeur v. Quels objets doit-on mettre dans le sac pour en maximiser la valeur totale ?

— Problème du voyageur de commerce : un voyageur doit au cours d'un voyage visiter un certain nombre de villes, puis rentrer chez lui. Dans quel ordre doit il les visiter pour minimiser sa distance parcourue ?

2 Algorithmes glouton

Un algorithme glouton est un algorithme cherchant à construire, étape par étape, une solution à un problème d'optimisation. Il est conçu de sorte que chaque étape de la construction soit localement (c'est à dire pour l'étape considérée) le meilleur choix possible.

Remarque essentielle Les algorithmes gloutons n'offrent en général aucune garantie de résoudre le problème d'optimisation puisque l'optimisation y est faite de manière locale et non globale. Cependant :

— on peut espérer d'un résultat obtenu de la sorte qu'il ne soit pas trop éloigné d'une solution optimale ;

— la complexité d'un algorithme glouton est souvent meilleure qu'un algorithme de recherche exhaustive.

Dans la suite du cours, nous allons donner des exemples de traitements de ces problèmes par des algorithmes gloutons.

2.1 Problème du voyageur de commerce

On résout ce problème en partant de la ville de départ et en déterminant, tant qu'il reste des villes à visiter, la ville suivante comme la ville la plus proche de la ville dans laquelle on se situe. On revient finalement à la ville de départ.

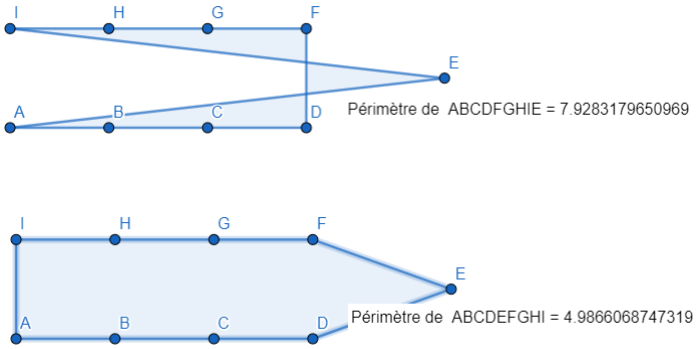
On ne détaille pas comment trouver cette ville suivante mais il suffit d'un algorithme de recherche de minimum, l'enjeu ici étant ici de :

— de minimiser la distance à la ville actuelle ;

— ne pas prendre en compte les villes déjà visitées.

Données : V la liste des villes à visiter, de longueur n, $v_0 \in V$ la ville de départ
Résultat : C la liste des villes à visiter, dans l'ordre
début
  $C \leftarrow [v_0]$;
  pour $i = 0$ **à** $n-1$ **faire**
    $u \leftarrow C[-1]$;
    $v \leftarrow$la ville non-encore visitée la plus proche de u ;
    ajouter v à la fin de C
  fin
  ajouter v_0 à la fin de C
fin

Cet algorithme ne donne pas toujours une solution optimale pour le problème, on peut le voir sur les exemples suivants (on part du point A), le premier étant un chemin obtenu par l'algorithme glouton et le second étant un meilleur chemin :



Cet algorithme, bien que non-optimal, présente toutefois l'intérêt de terminer en temps polynomial. Ce n'est le cas d'aucun algorithme exact connu aujourd'hui.

3 Problème du sac a dos

Le problème du sac à dos peut se traiter selon plusieurs stratégies gloutonne :
— chercher à emmener les objets de plus grande valeur ;
— chercher à emmener les objets les moins lourds ;
— chercher à emmener les objets dont le ratio valeur/poids est le plus élevé.

En pratique, on commence par trier la liste des objets selon le critère choisi puis on parcourt celle-ci dans l'ordre en ajoutant les objets chaque fois qu'on le peut. Par exemple, pour emmener les objets de plus grande valeur en priorité on applique l'algorithme suivant :

Données : L la liste des objets sous la forme (p, v) de taille n, P_{max} le poids maximum du sac-à-dos
Résultat : S le contenu du sac à dos
début
  trier L par valeur décroissante;
  $S \leftarrow []$;
  $P \leftarrow 0$;
  pour $i = 0$ **à** $n-1$ **faire**
    $(p, v) \leftarrow L[i]$;
    si $P + p \leq P_{max}$ **alors**
      ajouter (p, v) à la fin de S $P \leftarrow P + p$
    fin
  fin
fin

Remarque : Comme pour le problème du voyageur de commerce, on ne connaît par aujourd'hui d'algorithme polynomial donnant un solution exacte à ce problème. En particulier, notre algorithme glouton ne donne pas toujours le meilleur chargement.

4 Problème du rendu de monnaie

L'algorithme glouton de rendu de monnaie est très simple : pour rendre la somme voulue, on rend la plus grande pièce possible tant qu'il reste quelque chose à rendre.

Données : v la somme à rendre , S la liste des pièces triée par valeur croissante et de taille n contenant au moins 1
Résultat : T la liste des pièces rendues
début
    $reste \leftarrow v$;
    $T \leftarrow []$;
    $k \leftarrow n - 1$;
    tant que $reste > 0$ **faire**
        si $reste \geq S[k]$ **alors**
            $reste \leftarrow reste - S[k]$;
            ajouter $S[k]$ à la fin de T
        sinon
            $k \leftarrow k - 1$
        fin
    fin
fin

Cette méthode est optimale ou non selon l'ensemble de pièces à disposition :
— pour le système monétaire de la zone euro $S = [1, 2, 5, 10, 20, 50, 100, 200]$, il fonctionne ;
— pour le système monétaire $S = [1, 3, 6, 12, 24, 30]$, le rendu glouton de 49 donne

$$49 = 30 + 12 + 6 + 1$$

soit 4 pièces, alors qu'on pourrait rendre avec 3 pièces

$$49 = 24 + 24 + 1$$

En terminale, on verra une méthode permettant de résoudre en temps polynomial et de manière exacte ce problème, et ce quelque soit le système monaitaire choisi.